T.C.
MİLLÎ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN GÜÇLENDİRİLMESİ PROJESİ)

ENDÜSTRİYEL OTOMASYON TEKNOLOJİLERİ

SAYISAL İŞARET İŞLEME 2

ANKARA 2008

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).
- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.
- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.
- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**523EO0383**
ALAN	**Endüstriyel Otomasyon Teknolojileri**
DAL/MESLEK	**Alan Ortak**
MODÜLÜN ADI	**Sayısal İşaret İşleme 2**
MODÜLÜN TANIMI	Sayısal işaret devresini teknik özelliklerini tanıyarak programlama becerisinin kazandırıldığı bir öğrenme materyalidir.
SÜRE	40/32 Saat
ÖN KOŞUL	Sayısal İşaret İşleme 1 modülünü almış olmak
YETERLİK	Sayısal işaret işleme uygulamaları yapmak.
MODÜLÜN AMACI	**Genel Amaç** Sayısal işaret devresini teknik özelliklerine ve programlama tekniklerine uygun olarak kullanabileceksiniz. **Amaçlar** 1. Sinüs dalga üreteci uygulaması istenen değerleri standartlara uygun olarak yapabileceksiniz. 2. Ses değiştirici uygulamasını, istenen frekans değerlerinde yapabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam** Mikrobilgisayar laboratuvarı **Donanım** Sayısal işaret işleme deney setleri, mikrofonlar, hoparlörler, bilgisayarlar.
ÖLÇME VE DEĞERLENDİRME	Her faaliyetin sonunda ölçme soruları ile öğrenme düzeyinizi ölçeceksiniz. Araştırmalarla grup çalışmaları ve bireysel çalışmalarla öğretmen rehberliğinde ölçme ve değerlendirmeyi gerçekleştirebileceksiniz.

GİRİŞ

Sevgili Öğrenci,

Sayısal Sinyal İşleme 2 modülü ile endüstriyel otomasyon teknolojileri alanında gerekli olan sayısal işaret işlemlerine yönelik bilgi ve teknolojiye ait temel yeterlilikleri kazanacaksınız.

Bu modülü başarılı bir şekilde tamamladığınızda sinüs dalga üreteci ve ses değiştirme teknolojisini kavrayarak detaylı bilgiye sahip olacaksınız. Ayrıca C programlama dilinde kendinizi geliştireceksiniz.

ÖĞRENME FAALİYETİ–1

AMAÇ

Sinüs dalga üreteci uygulaması istenen değerleri standartlara uygun olarak yapabileceksiniz.

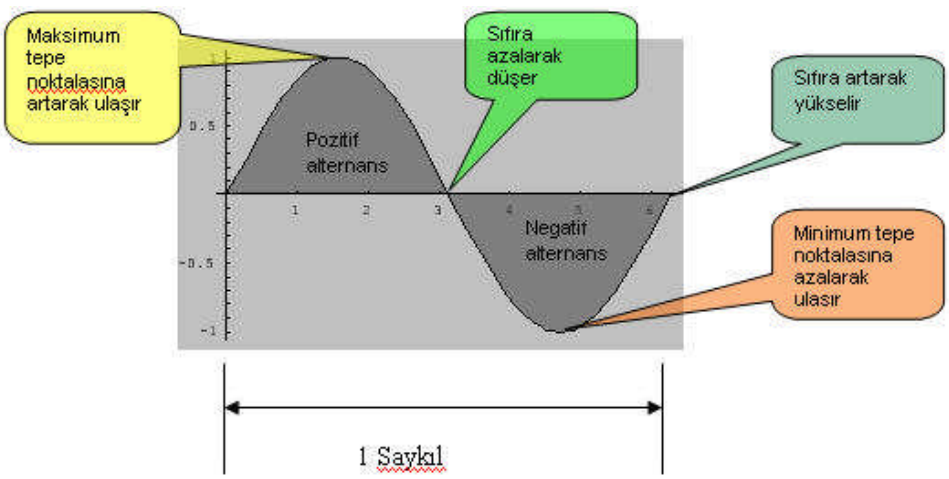
ARAŞTIRMA

Sinüs dalga üreteci hakkında bir araştırma yaparak rapor haline getiriniz.

1. SİNÜS DALGA ÜRETECİ NEDİR

Ayarlanan frekans aralığında sinüsoidal dalga üreten bir fonksiyon üretecidir.Bir periyodun ilk yarısında sinüsoidal dalga sıfırdan başlar, maksimum tepe noktasına kadar artarak gider ve sonra tekrar azalarak sıfıra düşer. Buna sinüsoidal dalganın pozitif alternansı denir. Artma ve azalma süreleri birbirine eşittir. Periyodun diğer yarısında ise sinüsoidal dalga sıfırdan başlar minimum tepe noktasına kadar azalarak gider ve sonra tekrar artarak sıfıra yükselir. Buna da sinüsoidal dalganın negatif alternansı denir. Yine azalma ve artma süreleri aynıdır. Şekil 1.1'deki sinüsoidal dalganın pozitif ve negatif alternansının oluşturduğu bu dalga şekline 1 saykıl (periyot) denir.

Sinüs dalga üretecinde frekansı değiştirdikçe sinüsoidal dalganın periyodu değişir.Yani maksimum ve minimum tepe noktalarına gidiş süreleri değişir. Sinüs dalga üretecinde genliği değiştirdikçe maksimum ve minimum tepe noktalarının genlik değerleri değişir.



Şekil 1.1: Sinüsoidal dalga

1.1. Sayısal İşaret İşleme Deney Seti Nasıl Kullanılır

- **CCS programı kullanılarak mikroişlemci nasıl programlanır**

İlk olarak "Hello, DSP !" cümlesini görüntüleyen program hakkında düşünelim. Kullanacağımız işlemci DSP olmasına rağmen yazacağımız program klasik C language programlama dilinde olacaktır.Program aşağıdaki gibi olacaktır :

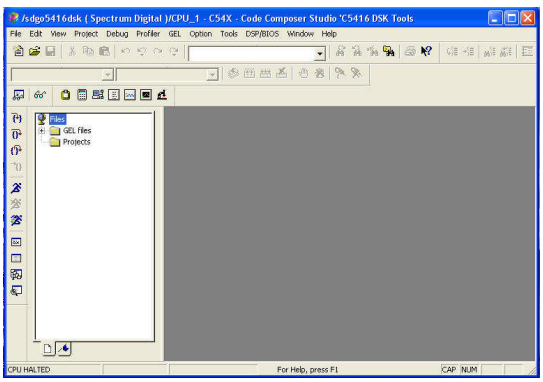
```
#include <stdio.h>

void main()
{
  printf("Hello,DSP !\n");
}
```

Programı çalıştırma yolu kullandığınız sisteme bağlıdır. Program seçtiğiniz herhangi bir sistem üzerinde yazılır, kaydedilir ve makine diline derlenir.Örneğin Windows'ta, dosya isminin uzantısı ".c" olur.Yani "hello.c" gibi. Programı derledikten sonra, çalıştırılabilir dosya tipi "hello.exe" oluşturulur. DOS'un komut satırından bu program çalıştırıldığı zaman ekranda "Hello,DSP!" gözükür.

Şimdi, DSP sistemi içinde bunun nasıl yapıldığına bakalım. DSP sistemi Windows gibi bir ekrana sahip değildir. Ancak DSP bir hata ayıklayıcı (geliştirilmiş çevre üniteleri programlayıcısı) ile birlikte çalışır. C5000 çevre ünitesinin programı olan "Code Composer Studio" ile DSP kontrol edilir. Kısacası, "Hello,DSP!" cümlesi "Code Composer Studio (CCS) " program editörü üzerinde görüntülenir. Bu kitapta, Texas çevre ünitesinin programı olan "Code Composer Studio 'C5416 Version 2.10.05"(biz ona kısaca CCS diyeceğiz) kullanılmaktadır.

- **CCS'nin çalıştırılması**

Eğer CCS bilgisayarınızda kurulu ise, bilgisayarınızın masa üstünde CCS ikonu gözükecektir. Bu ikonu çift tıklayın ve CCS programını başlatın veya Windows- Start Menu-programlardan da CCS seçilerek program çalıştırılır.



Şekil 1.2: Code composer studio program ekranının açılması

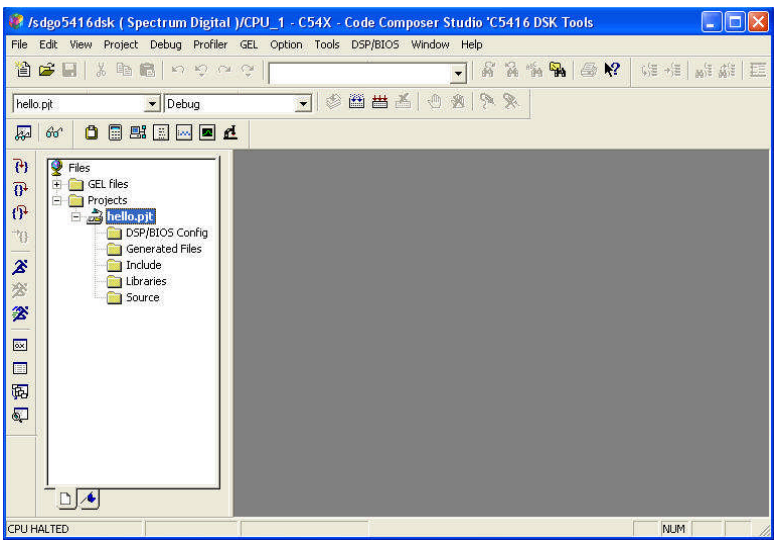
- **Yeni bir proje yapmak**
- **Hazırlık (Dosyaların düzenlenmesi)**

CCS programının kurulumu sırasında "C:" sürücüsünün kök dizininde "ti" dosyası oluşturulur. "ti" dosyası içinde de "myprojects" dosyası oluşturulur.Yazdığımız programların dosyalarını bu dosyanın içine kaydedip proje oluşturacağız.Şimdi "hello" dosyası için projemizi yapalım.

- **"Hello.pjt"'nin yapılışı**

"Project" menüsünden "New" komutunu seçin. "Project Creation" isimli pencere açılır. Sonra, projenin ismi ve yeri belirtilir (şimdi proje ismi olarak, "hello" girin). Projenin ismi girilince yeri otomatik olarak "Location" kutucuğunda belirir. Eğer proje yeri otomatik belirtilmemişse doğru adresi girin (bu durumda doğru adres "c:\ti\myproject\hello\" olacaktır). Projenin uzantısı otomatik olarak ".pjt" olacaktır. Böylece, projemiz "hello.pjt" yapılmış olur.

CCS program editörünün sol tarafında "Project View" penceresi gözükür. Bu pencerede oluşturduğunuz "hello.pjt" proje dosyasının yanındaki + işaretine tıklayarak proje dosyasının içerdiği 5 tane alt dosyayı görüntüleyin.Yeni bir proje oluşturulduğunda bu dosyalar otomatik oluşturulur ve başlangıçta içlerinde hiçbir şey yoktur.Bu dosyaların isimleri sırasıyla şöyledir: "DSP/BIOS Config", "Generated Files", "Include", "Libraries" ve "Source".



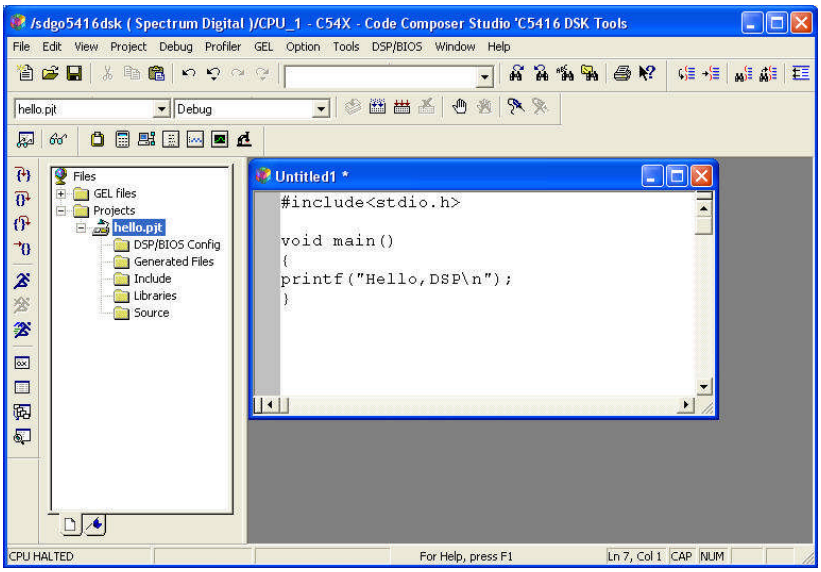
Şekil 1.3: "Hello.pjt" 'nin yapısı

- **Projeye bazı dosyaların eklenmesi**

Projenin kaynak dosya, konfigürasyon dosyası gibi bazı dosyalara ihtiyacı vardır.

➢ **Kaynak dosya (Source file)**

Yeni bir kaynak dosya yapmak için (örneğimizdeki "hello.c" gibi) menüden "File" -> "New" -> "Source file" seçildiği zaman "Untitled1" program penceresi açılır,buraya Liste 6-1'deki C dilindeki programı yazın. Bu programı kaydetmek için File menüden "Save As" seçeneğini seçin ve dosyanın ismini "hello.c" olarak girin ve Save butonunu tıklayın.



Şekil 1.4: Kaynak programın girilmesi

➢ **Projeye kaynak dosyanın eklenmesi**

Projeye kaynak program "hello.c" 'nin eklenmesi için Project menüden "Add Files to Project" seçeneğini seçin. Sonra, ekrana gelen pencerenin içinden "hello.c" seçin ve open butonunu tıklayın.Hello.pjt dizini altındaki source dosyasının yanındaki ⊞ işaretine tıklayarak "hello.c" kaynak dosyasının eklendiğini görebilirsiniz.

➢ **Bir konfigürasyon dosyasının yapılması ve eklenmesi**

Yeni bir konfigürasyon dosyası oluşturmak için menüden "File" -> "New" -> "DSP/BIOS Configuration..." seçin. Kullandığımız DSP'nin konfigürasyon dosyası olan "dsk5416.cdb" dosyasını seçin ve "OK" tıklayın. Ekranda "Config1" isimli yeni bir pencere gözükecektir. Bu dosyayı "hello.cdb" olarak kaydettiğimizde oluşturulan konfigürasyon dosyasını projemize eklemiş oluruz. File menüden "Save As" seçeneğini seçin ve konfigürasyon dosyasının ismini "hello.cdb" olarak girin ve Save butonunu tıklayın. Şimdi oluşturulan bu konfigürasyon dosyasını (.cdb uzantılı dosyayı) projeye eklemeliyiz. Menüden "Project" -> "Add Files to Project" seçeneğini tıklayın.Ekrana gelen pencerede "File of type" kutucuğundan konfigürasyon dosyası seçeneğini "Configuration File (*.cdb)" seçin. Konfigürasyon dosyası gözükecektir (örneğimizdeki "hello.cdb" dosyası gözükür).

Dosyayı seçin ve "open" butonunu tıklayın, "hello.cdb" dosyasının projeye eklendiğini "Project Creation" penceresinden "DSP/BIOS Configuration..." dosyasının yanındaki + işaretine tıklayarak görebilirsiniz.

- **Compiler ve linker seçenekleri**

Konfigürasyon için kullandığımız "far call" seçeneğine değiştirmemiz gerekir. Derleyici programı derlerken eğer "Use Far Calls" seçeneği seçili olursa geniş bir program yapılabilir.Bu seçeneği seçmek için menüden "Project" -> "Build option" seçeneğini tıklayın gelen menüde "Compiler" sekmesini tıklayın ve Category 'nin içinden "Advanced" seçin.Buradan "Use Far Calls" seçeneğini tıklayın. Yine aynı menüden bu sefer "Linker" sekmesini seçin ve Category 'nin içinden "Basic" seçin.Buradaki "Include Libraries" kutucuğuna "dsk5416f.lib" yazın ve "OK" tıklayın.

- **Linker komut dosyası**

Konfigürasyon dosyası yapıldığı zaman linker komut dosyası otomatik olarak yapılır. Burada, sadece linker dosyasını (.cmd uzantılı dosyayı) projeye eklemeliyiz. Menüden "Project" -> "Add Files to Project" seçeneğini tıklayın.Ekrana gelen pencerede "File of type" kutucuğundan linker komut dosyası seçeneğini "Linker Command File(*.cmd)" seçin. Komut dosyası gözükecektir (örneğimizdeki "hellocfg" dosyası gözükür). Dosyayı seçin ve "open" butonunu tıklayın, "hellocfg.cmd" dosyası projeye eklendiğini "Project Creation" penceresinden de görebilirsiniz.

- **Otomatik olarak yüklenen seçenekler**

Proje derlendikten sonra otomatik olarak programın DSP borduna yüklenmesi için CCS programında bazı seçenekler aktif yapılır.Bunun için "Option" menüden -> "customize" -> seçeneğini seçin ve ekrana gelen menüde "Program Load Options" sekmesini tıklayın.Burada "Load Program After Build" seçeneğinin aktif olması için yanındaki kutucuğu tıklayın.

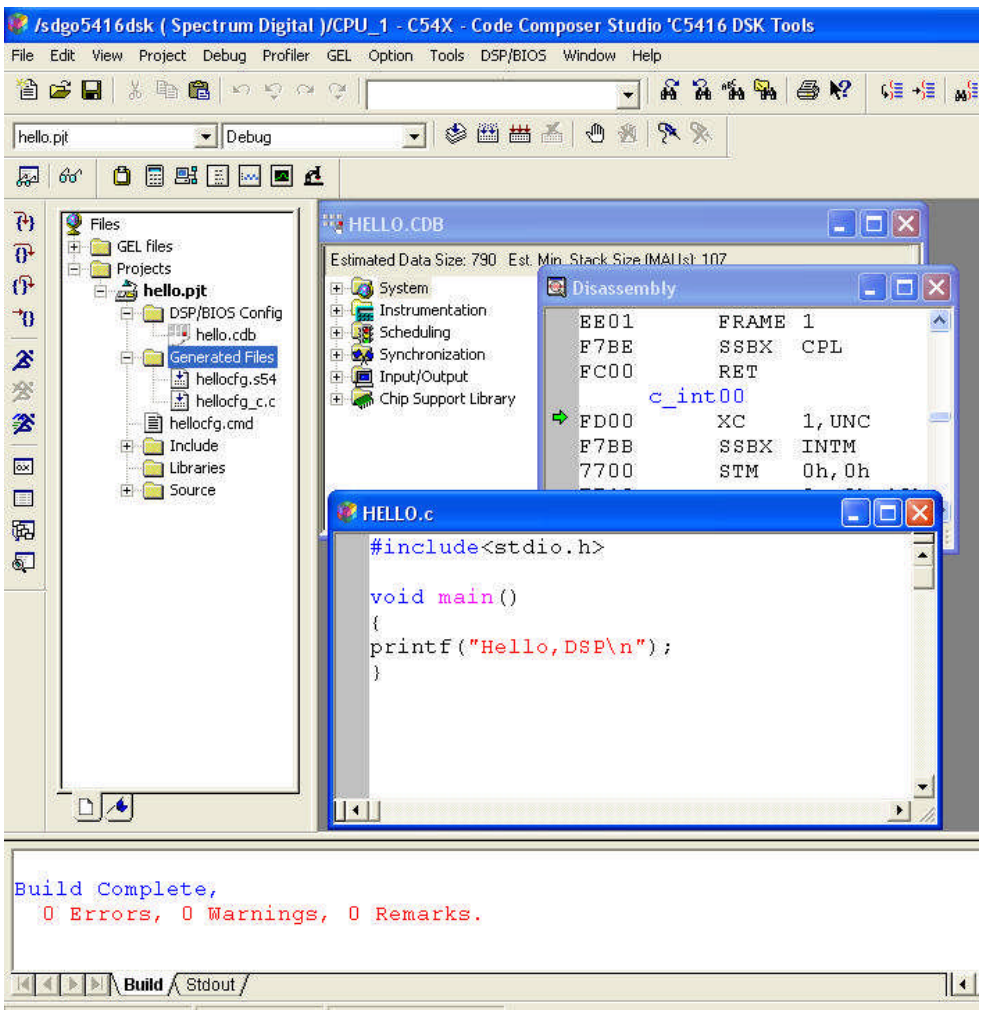
- **Programın derlenmesi**

Buraya kadar yukarıdaki işlemlerin hepsi yapılarak bütün gerekli dosyalar tamamlanmış olur. Projeyi tamamlamak için ise kaydetmek zorundayız. Project -> menüden save seçeneğini tıklayın. Çalışabilir bir dosya yapmak için derleme "Build" yapılması gerekir. CCS program editöründe, sırasıyla "Projects" -> menüden "rebuild All" seçeneğini seçin veya araç çubuğu üzerindeki  (Rebuild All) ikonunu tıklayın. Bir süre sonra, eğer bir hatanız yoksa çalışabilir dosya (hello.out) oluşturulur. Hem de çalışabilir dosya (hello.out) DSP borduna gönderilmiş olur. Şekil 6.4'te gösterildiği gibi CCS editörünün en altındaki "Build" ekranında :

"Build Complete,

0 Errors,0 Warnings,0 Remarks."

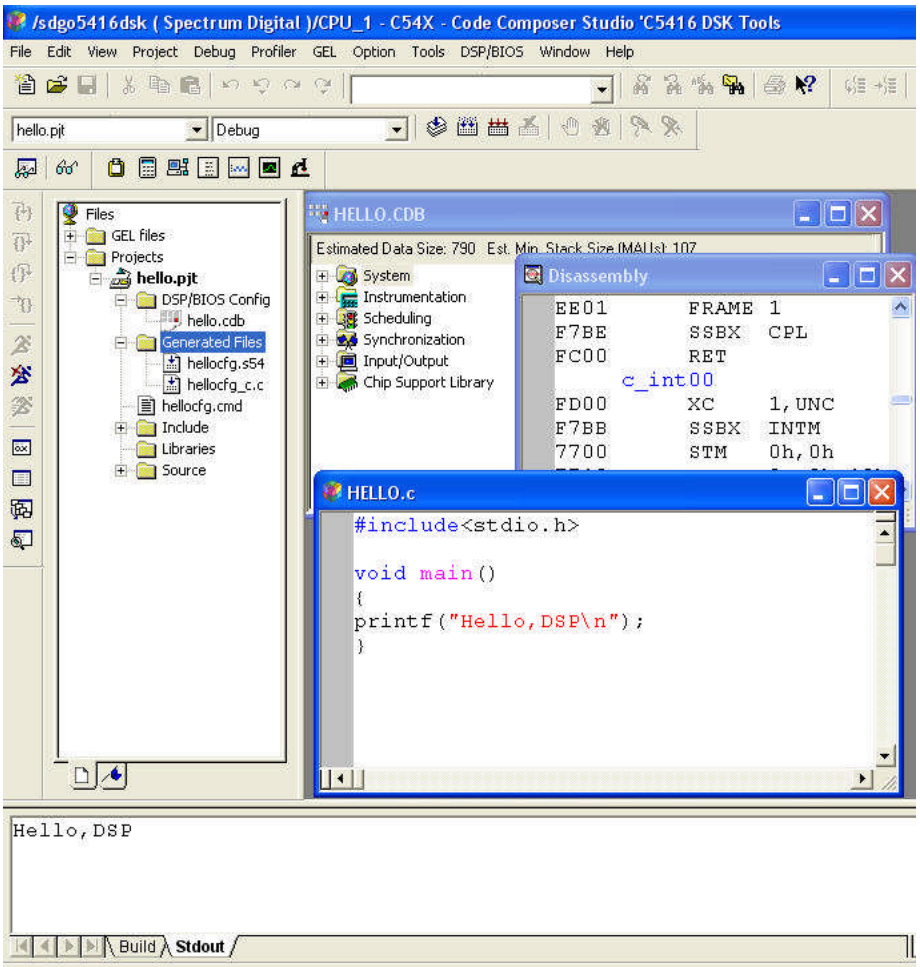
mesajını görürsünüz. Eğer programınızda hata varsa derleme yapıldıktan sonra program DSP borduna gönderilmez ve hatalar yine "Build" ekranında sırasıyla yazar.



Şekil 1.5: Derlemenin hatasız olarak tamamlandığı durum

- **Programın çalıştırılması**

Programı çalıştırmak için araç çubuğu üzerindeki (Run) butonu tıklanır. CCS editörünün en altındaki "Stdout" ekranında "Hello ,DSP! " mesajı gözükecektir.



Şekil 1.6: Uygulamanın sonucu

Programın görüntüsü bitmiş gibi olsa da otomatik olarak programın çalışması durmaz. Programın çalışmasını sonlandırmak için araç çubuğu üzerindeki (Halt) butonunu tıklayın.

Eğer programı tekrar baştan çalıştırmak isterseniz, "Debug" menüden "Restart" seçeneğini tıklayın (burada "Reset CPU" seçeneğini sakın seçmeyin). Restart seçeneği ile Program Counter içeriği programın başlangıç adresi "c_int00" olur ve program tekrar baştan çalışmaya başlar. CCS editörünün en altındaki "Stdout" ekranında "Hello ,DSP! " mesajının diğer satıra tekrar yazıldığı görülerek, programın tekrar çalıştırıldığı daha iyi anlaşılır.

1.2. Sinüs Dalga Üreteci Programı Nasıl Yazılır

Yazılan programla farklı periyotlara sahip iki sinüs dalgası oluşturulur ve DSP tarafından bu iki sinüs dalgasının birleşiminden oluşan bir dalga oluşturulmaktadır. Sinüs dalga üreteci programı aşağıdaki gibidir:

```
#include <math.h>
#define BUFF_SIZE    96

double sinbuff1[BUFF_SIZE];
double sinbuff2[BUFF_SIZE];
double sinbuff1_2[BUFF_SIZE];

void main()
{
   int i;
   double radius1=0,0;
   double radius2=0,0;
   for (i=0; i<BUFF_SIZE; i++)
   {
       radius1 +=0.2;
       radius2 +=1.0;

       sinbuff1[i]=sin(radius1);
       sinbuff2[i]=sin(radius2);
       sinbuff1_2[i]□sinbuff1[i]*sinbuff2[i];
   }
}
```

Bu program "Hello DSP" programından daha uzundur, fakat çok fazla karışık değildir. Bu örnekte bazı yeni fonksiyonlar var. Şimdi bunları açıklayalım:

- **sin() fonksiyonu hakkında**

Programın çalışma süresince bir destek kütüphanesi (DSPLIB olarak bilinen) kullanılır. Bu örnekte, sin() fonksiyonu bu destek kütüphanesinden sağlanır.Destek kütüphanesini programın içermesi için programa aşağıdaki komut ve kütüphanenin ismi eklenmelidir.

#include <math.h>

Çünkü bir sin() fonksiyonunun ilk örneğinin tanımı (sin fonksiyonunu nasıl kullanılacağı) bu "math.h" kütüphane dosyasında tanımlanmıştır. Derleyici, bu kütüphane dosyasının açılmasıyla sin() fonksiyonunun kullanım yolunu yorumlar. Bu kütüphane dosyasının içinde, sin() fonksiyonu aşağıdaki gibi tanımlanmıştır:

double sin(double x);

Derleyici, benzer tipteki argümentlerden birinin değişiminin alındığı ve fonksiyonun değeri olarak bu benzer tiplerin geri döndüğü bir sin() fonksiyonunu yorumlar. Bu nedenle, sin() fonksiyonu için çift tipli olarak tanımlanan "radius1" ve "radius2" değişkenleri girilir. Tabii ki bu çift tiplerin içinde bir dizi de (sonuçların saklanması) tanımlı olacaktır.

- **Ses değişiminin prensibi**

Eğer iki farklı frekanstaki sinüs dalgası çarpılırsa aşağıdaki olay meydana gelecektir:

$$\sin(2\pi f_1 t)\times\sin(2\pi f_2 t)=\frac{1}{2}\cos\{2\pi(f_1+f_2)t\}+\frac{1}{2}\cos\{2\pi(f_1-f_2)t\}$$

Bu sonuç, iki cos() fonksiyonunda iki frekans olduğunu gösterir. Bunun anlamı bu iki frekans sentezlenmiştir (karıştırılmıştır). Bu frekanslarda birisi "f1+f2", diğeri ise "f1-f2" ‘dir. O nedenle frekans analizi yapılırken (FFT), "f1+f2" ve "f1-f2" frekanslarının tepe noktası gözlenebilir. Bunun anlamı " f1 Orjinal Sinyali " alçak frekans tarafına (f1-f2) ve yüksek frekans tarafına (f1+f2) kaydırılmıştır.

Bir f1 orjinal sinyali, bir speech sinyal olduğu zaman, alçak ve yüksek sesin sentezlenmiş sesi ancak "f2" ‘nin miktarında yapılabilir.

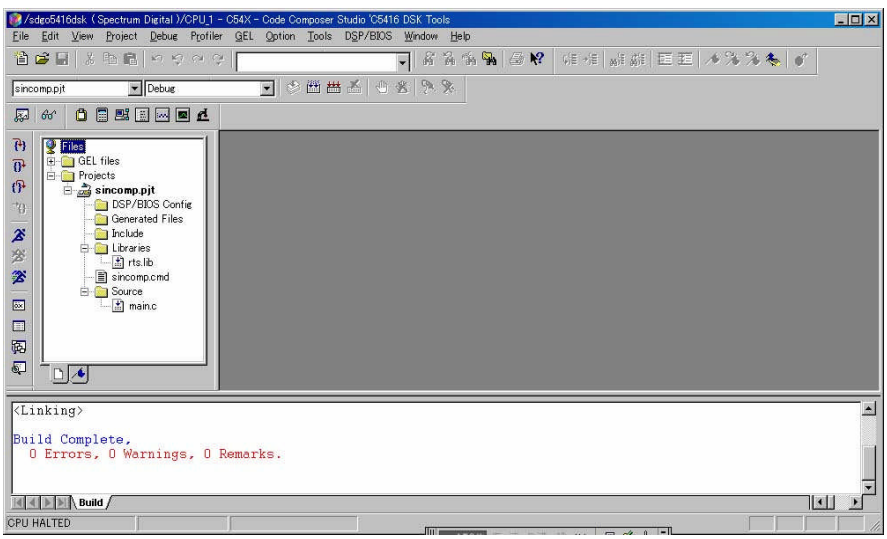
- **CCS kullanımının doğruluğunun kesinleştirilmesi**

CCS ile bu ifadenin uygulamasını kesinleştirelim. Bu bölümde, sinüs dalgası hesaplanarak grafikleri ekranda gösterilir. Böylece, FFT’nin sonuçları grafiklerle görüntülenir.

- **CCS ile çalıştırılabilir bir dosya yapımı**

"sincomp.pjt" isimli yeni bir proje açın ve Sinüs dalga üreteci programını kaynak dosya olarak girin.Daha sonra, konfigürasyon dosyasını ve linker komut dosyasını da oluşturun. Bu projeyi de daha önce yaptığınız "Hello DSP" projesindeki yöntemleri kullanarak Şekil. 2-7’deki gibi tamamlayın.

Çalışabilir bir dosya yapmak için projeyi derleyin. Eğer hata yoksa, çalışabilir bir dosya "sincomp.out" DSP borduna gönderilecektir.



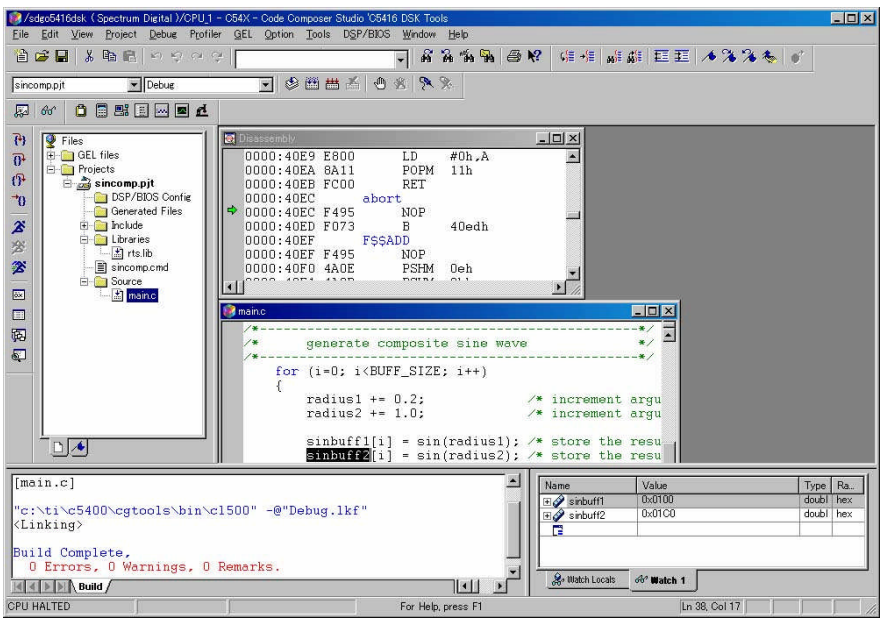
Şekil 1.7: "sincomp" projesinin tamamlanması

- **Sincomp programının çalıştırılması**

Program (Run) butonu tıklanarak çalıştırılır. Veya programı adım adım çalıştırmak için (step over) butonu tıklanır. Bu buton programların hatalarının analizi için çok önemlidir. Program her zaman (Run) butonuna basılması sonucu çalıştırılabilir.

- **Watch window**

Sonuçlar "sinbuff1[ ]" ve "sinbuff2[ ]" isimli iki dizide saklanır. Genellikle yapılan işin doğrulanması için bu değişkenlerin gösterilmesinde "Watch window" kullanılır. Bunu göstermek için, C dilinde yazılan kaynak dosyanın üzerinde istediğiniz diziyi seçin.Örneğin "sinbuff1[ ]" dizisi için sinbuff1 ifadesi farenin sol tuşu ile taranarak seçilir ve üzerinde farenin sağ tuşu tıklanır.

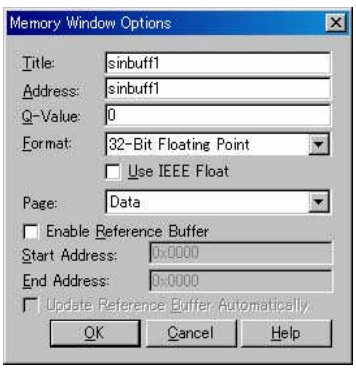


Şekil 1.8: Watch window

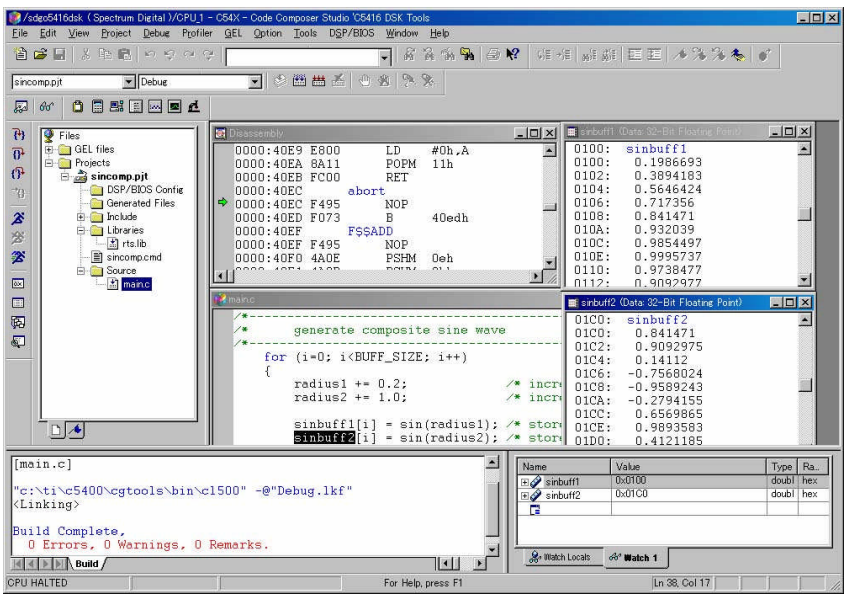
Sonra ekrana gelen menüden, "Add to Watch Window" seçeneği seçilir. Watch window, Şekil.2.8'deki gibi CCS program editörünün sağ alt bölümünde gözükür. Watch Window penceresinde Name kutusunun altında seçilen dizinin ismi görüntülenir.Bu dizinin (sinbuff1) yanındaki + işaretine tıklanarak 96 tane değişkenin değeri ekrana listelenir. Value kutucuğunun altındaki "0x0100" hexadecimal adres ilk dizinin saklandığı adresi gösterir.DSP bordu 32 bitlik adresleme yapmaktadır.Burada double fonksiyonu kullanıldığı için her değişken 64 bit olduğundan her bir değişken DSP de 2 adreslik yer tutmaktadır.Yani sinbuff1 dizinindeki ilk değişkenin adresi "0x0100" iken ikinci değişkenin adresi "0x0102" olur.

- **Memory window**

Hafızada saklanan değerleri ve adreslerini doğrudan kontrol etmek için "Memory window" kullanılır. "View" menüsünden "Memory" seçeneği seçilince ekrana "Memory Window Options" penceresi açılır. Bu pencerede title kutucuğuna etiket ismi ve Address kutucuğuna hangi dizi hafızasının adresi kontrol etmek isteniyorsa onun ismi (Şekil .2.9'da gösterildiği gibi burada sinbuff1 dizi ismi) girilir. Format kutucuğuna "32-Bit Floating Point" seçilir ve OK tıklanınca hafızanın adresleri ve verileri bir pencerede görüntülenir (Şekil 1.10).



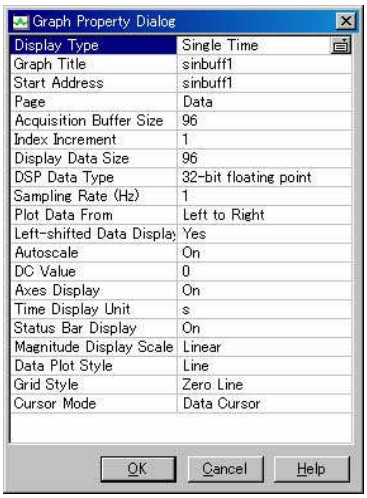
Şekil 1.9: Etiketin adresin ve biçimin belirtilmesi



Şekil 1.10: Memory window (sağ tarafta gösterilmektedir)

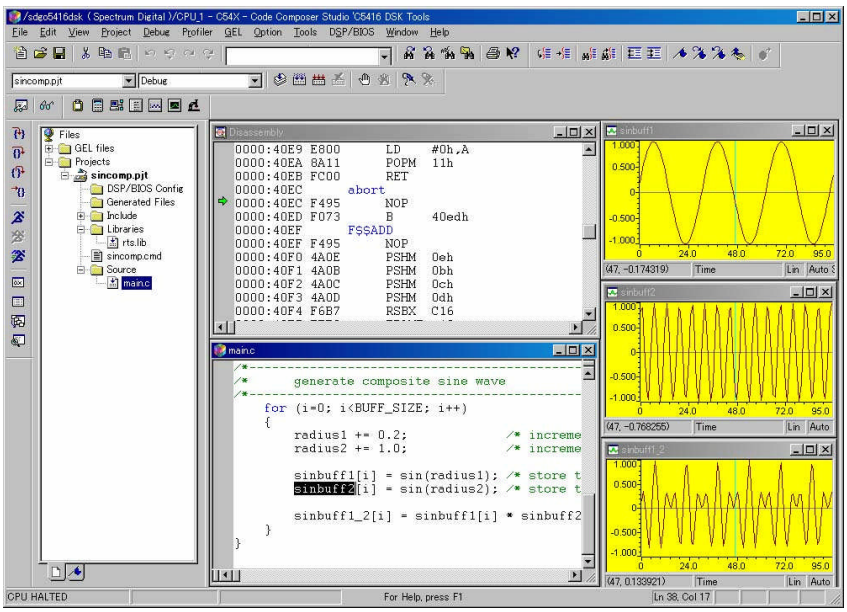
- **Hafızadaki değerlerin grafiksel görüntülenmesi**

"Watch window" ve "Memory window" kullanılarak, uygulama sonuçlarının doğruluğu görülebilir. Ancak CCS programı hafızadaki değerlerin görüntüsünü grafiksel olarak gösterebilir. Bu fonksiyonu kullanmak için, View menüden -> Graph -> Time/Frequency seçeneğini seçin. Bu seçenek ile "Graph Property Dialog" penceresi ekranda gözükür. Şekil 2.11'deki gibi değerleri girin ve OK tıklayın.



Şekil 1.11: "Graph property dialog" kutusu

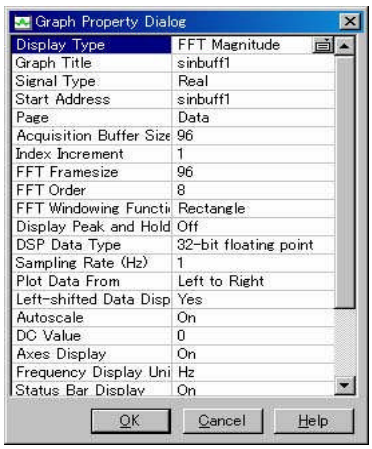
Ekranda sinbuff1[] dizisinin değişkenlerinin oluşturduğu noktalarla çizilen grafik görüntülenir. Aynı işlemleri sinbuff2[] ve sinbuff1_2[] için de yaparak Şekil 1.12'deki gibi üç sinüs dalgasını da ekranda görüntüleyin.



Şekil 1.12: Hafızadaki değerlerin grafiksel görüntülenmesi

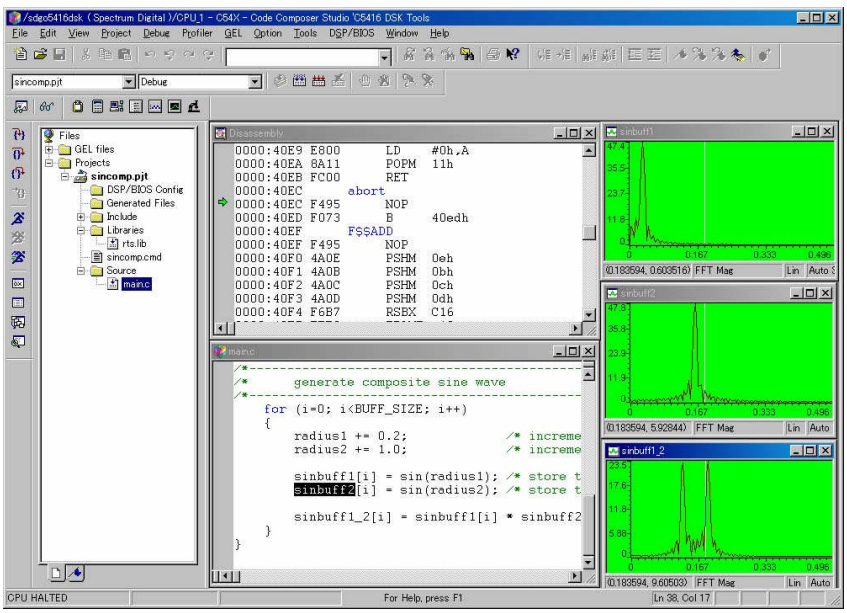
- **FFT ‘nin grafiksel görüntülenmesi**

Bir sinüs dalgası görüntülendiğinde FFT’de ,ani yükselen bir tepe (sharp peak) gözlenir. FFT grafiğinin içinde sinüs dalga grafiğini değiştirmek için, daha önce görüntülenen grafik ekranının üzerinde (sinbuff1 grafiği) farenin sağ tuşunu tıklayın ve gelen menüden "Properties" seçeneğini seçin. Sonra ekrana gelen "Graph Property Dialog" penceresi içindeki verileri Şekil 1.13’teki gibi değiştirildiğinde sinüs dalga grafiği bir FFT grafiği olur.



Şekil 1.13: Temel frekans grafiği için ayarlar

Diğer iki dizi için benzer işlemleri yaparak Şekil 1.14’teki gibi üç dizinin de FFT grafiğini ekranda görüntüleyin.

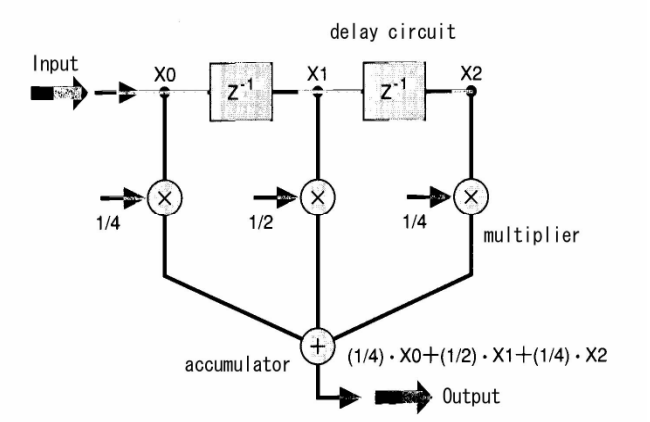


Şekil 1.14: FFT grafikleri

- **Dijital filtre**

Dijital işlemlerde kullanılan filtreye dijital filtre denir. Dijital filtre, fazla olan dijital sinyalleri ortadan kaldırır (yok eder). Değişik dijital filtre çeşitleri vardır. Bir dijital filtre üzerindeki araştırma hızlı bir şekilde gelişmektedir ve çeşitli alanlarda uygulanmaktadır. Bu teknoloji olmasaydı, bir dijital iletişim (cep telefonu vb.) gerçekleşemezdi. Bu bölümde, tipik bir dijital filtre olarak FIR (Finite Impulse Response) filtresi ele alınacaktır.

FIR filtresinin diyagramı Şekil 1.15'te gösterilmektedir.



Şekil 1.15: FIR filtresinin diyagramı

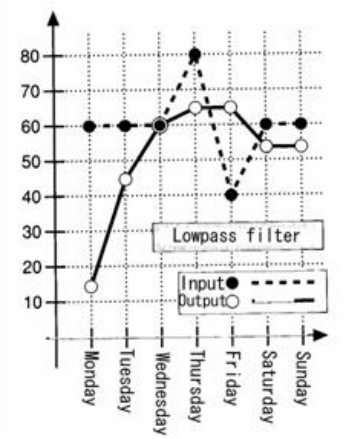
Bu filtrenin matematiksel bir filtre olduğu anlaşılmaktadır, çünkü şekilde toplama ve çarpma işaretleri vardır. Bu işaretlerin dışında bir de "Z^{-1}" işareti mevcuttur. Buna "Unit delay circuit (gecikme devresi)" denir. Bu "Unit delay circuit (gecikme devresi)" bir zaman gecikmesinde rol oynar. Kısacası, "Z^{-1}" ,yeni bir değer gelene kadar X0 değerini saklar. Tabii ki, X1 ve X2 arasındaki "Z^{-1}" devresinin rolü de aynıdır. Ancak yeni veri X0 noktasından akar ve aşağıdaki hesaplama sonucu çıkışa aktarılır:

- **Çıkış (Output): = (1/4) X0 + (1/2) X1 + (1/4) X2**

Sonunda, X0 giriş verisi (impulse) gecikme devresi ile sağa doğru kaydırılır, X2 işlendikten sonra dışarı atılır ve bu sistemde artık gösterilmez. Bu sebepten dolayı ,FIR (FINITE impulse response (sonlu puls cavaplayan)) filtre denir.

FIR filtredeki veri geçişi nasıldır? Bunu Şekil 1.16'da verilen bir örnekle anlatalım.

A day of the week (Sampling period)	Share value (Input)	Filter (Output)
Monday	60	15
Tuesday	60	45
Wednesday	60	60
Thursday	80	65
Friday	40	65
Saturday	60	55
Sunday	60	55

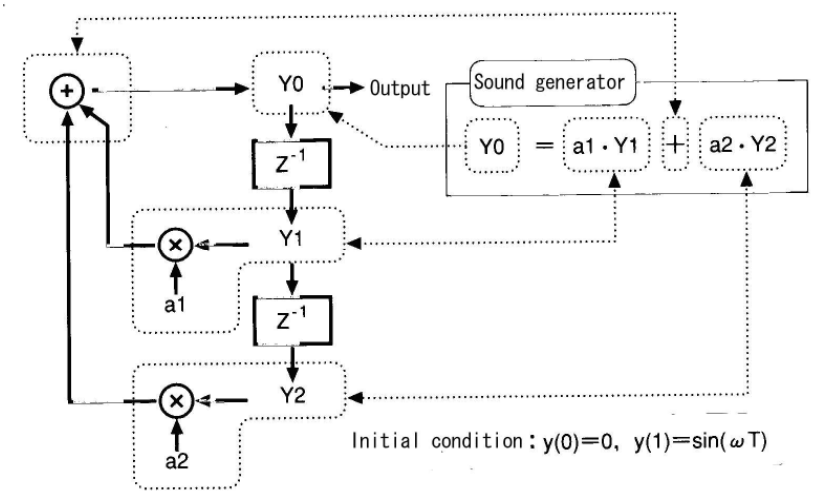


Şekil 1.16: FIR filtrenin giriş ve çıkışı

Örneğimizdeki tabloda pazartesiden pazara kadar girilen parça değerleri ve filtre çıkışları gösterilmektedir. Burada giriş ve çıkış arasındaki bağıntı gözlenir. Bu filtrenin perşembe ve cuma günlerdeki hızlı değişen giriş değerlerini geçirmediği görülür. Başka ifadeyle bu bölüm yüksek frekans bileşenidir. Bu durumda FIR filtresi yüksek frekans bileşenlerini geçirmiyor. Yani bu filtre alçak geçiren bir filtredir.

- **IIR filtresinin diyagramı**

Şekil 1.17'de sinüs dalga jeneratör diyagramı planı kullanan IIR filtresi gösterilmektedir.



Şekil 1.17: IIR filtrenin diyagramı

IIR filtrenin özelliği, girişte çıkıştan gelen bir geri beslemenin bulunmasıdır. FIR filtresinde, sabit periyod bitiminde giriş palsı gözükmez. IIR filtresi karşılaştırma yapar. Çıkış aşağıdaki formülle hesaplanır:

$$y(n) = a_1 y(n-1) + a_2 y(n-2) \quad \text{.....(6-1)}$$

Burada sırasıyla y(n), y(n-1) ve y(n-2) ifadeleri Y0, Y1 ve Y2, değerlerine karşılık gelir. Burada Y0 şimdiki çıkışa, Y1 önceki çıkışa ve Y2 bir önceki çıkışa karşılık gelir. Birinci şart

$$y(0) = 0, y(1) = \sin(\omega T_s)$$

olarak verilir. Ayrıca sabitler aşağıdaki gibi verilir:

$$a_1 = 2\cos(\omega Ts)$$
$$a_2 = -1$$

Burada "ω" genel sinüs dalgasının açısal frekansıdır ve "Ts" örnek bir periyottur. Bu ifadeden geçirilen değerlerinin alınarak yeni çıkışın sonucunun hesaplandığı anlaşılır. Kısacası, IIR filtresi doğrusal olmayan kararsız sistemler için kullanılır ve FIR filtreden yaklaşık 10-100 kat kadar daha verimlidir.

- **Sinüs dalga jeneratörü (Floating point ile uygulama)**
  - **Sabit ve birinci koşul**

Burada, floating point ile sinüs dalga jeneratörünün IIR filtreyle yapılması anlatılacaktır. Örneğin bir piyanonun standart ayarları yapılırken ses frekansının (örneğin C majör anahtarı Ra) f=440Hz olması istenir. Eğer bu örneğimizdeki gibi sabitler ve birinci koşul verilmiş ise çıkış aşağıdaki gibi hesaplanır.

440Hz -> $\omega = 2\pi f = 2\pi \times 440 = 880\pi$

Örnek frekans fs=8000Hz -> $Ts = \frac{1}{fs} = \frac{1}{8000}$.

Böylece, Constant

$$a_1 = 2\cos(\omega Ts) = 2\cos(880\pi \times \frac{1}{8000}) = 1.8817615$$
$$a_2 = -1.$$

Birinci koşul
$$y(0) = 0$$
$$y(1) = \sin(\omega Ts) = \sin(880\pi \times \frac{1}{8000}) = 0.33873792.$$

- **Sinüs dalga jeneratörü fonksiyonu**

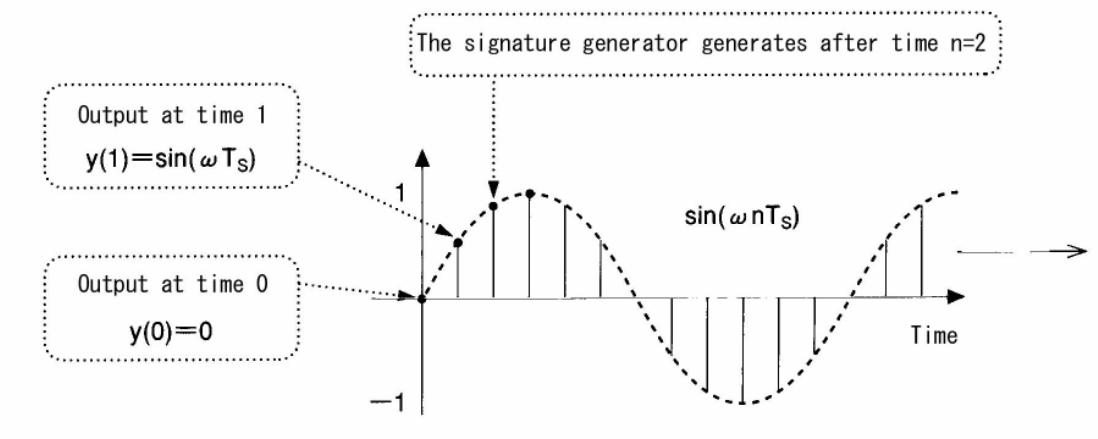
y[3] dizisi y(n), y(n-1) ve y(n-2) dizilerine içine alan bir dizi olarak tanımlanır. Programda "y(n)" şimdiki çıkış olan y[0] dizisinin, "y(n-1)" önceki çıkış y[1] dizisinin ve "y(n-2)" bir önceki çıkış y[2] dizisinin içeriğini korur. Ayrıca, her hesaplama sonucunda, gecikme işlemi "Delay processing" gereklidir. Bu gecikme işlemi y[1] 'in içeriğinin y[2]'ye, y[0] 'ın içeriğinin de y[1]'e kopyalanması ile yapılır.

Böylece, "function Singen()" (sinüs fonksiyonu) aşağıdaki gibi yapılır:

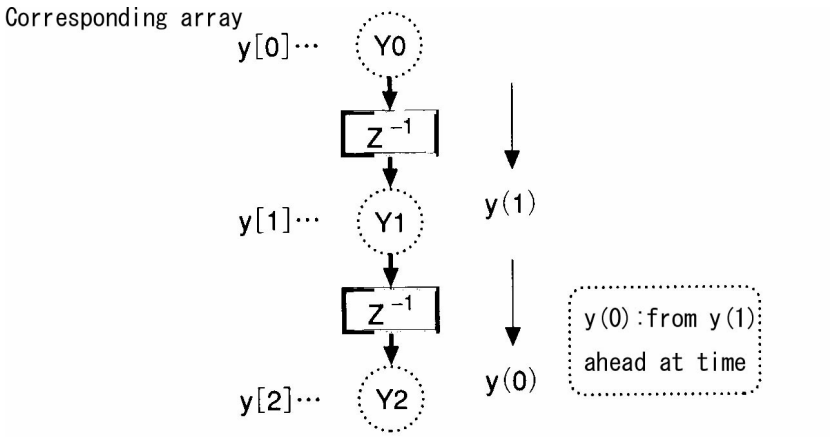
```
float y[3];                          // y(n),y(n-1),y(n-2)
float a1=1.8817615;                  // 2cos(wT)

float singen()
{
     y[0] = a1*y[1] + (-1.0)*y[2];   // Get y(n)
     y[2] = y[1];                    // delay
        y[1] = y[0];                 // delay

        return(y[0]);
}
```

- **İlk değerin belirlenmesi**

İlk değerin belirlenmesi gereklidir. Birinci koşulda, ilk değer olarak "y(0)" ve "y(1)" çıkışlarının değerleri verildi. "y(2), y(3),…" çıkışları ise IIR filtre ile hesaplanacaktır (Şekil 1.18).



Şekil 1.18: IIR filtrenin çıkışı



Şekil 1.19: İlk değer hakkında düşünür

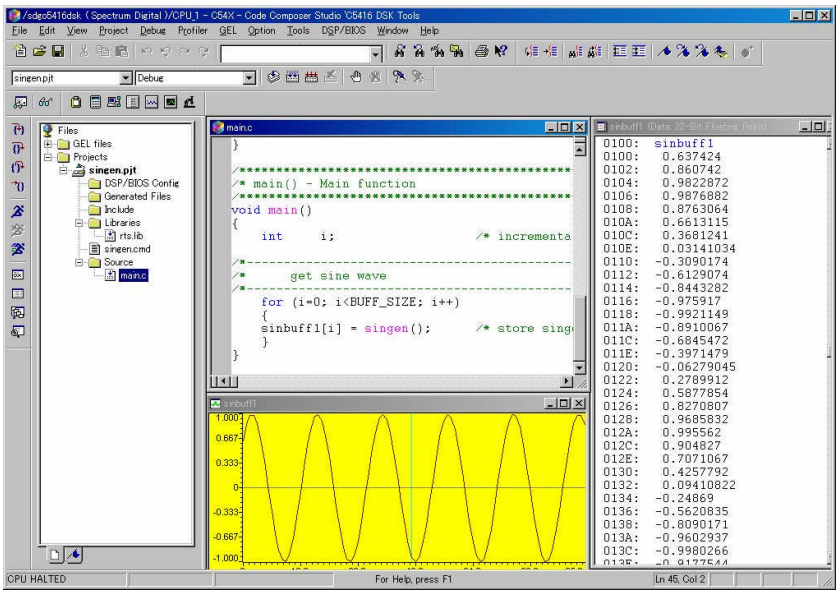
Burada ilk değeri ne olarak belirtmeliyiz? y(0) değeri y(1)'in ilerde alacağı değerdir. Bundan dolayı, y(0) yeni veri, Y[2] ise y(1)'den daha sonradır. Böylece ilk koşul:

y[0] = Her şey olabilir,
y[1] = y(1) = 0.33873792,
y[2] = y(0) = 0.

Bu programa kontrol için bir main() fonksiyonu eklenerek tamamlanan Sinüs dalgası üreten program aşağıdaki gibi olur:

```
#define BUFF_SIZE    96

float sinbuff1[BUFF_SIZE];

float y[3]={0, 0.33873792, 0};        //  y(n),y(n-1),y(n-2)
float a1 = 1.8817615;                 //  2cos(wT)
float a2 = -1.0;

float singen()
{
    y[0] = a1*y[1] + a2*y[2]; //Get y(n)
    y[2] = y[1];                      //delay
    y[1] = y[0];                      //delay

    return(y[0]);
}

void main()
{
       int i;
       for (i=0; i<BUFF_SIZE; i++)
           sinbuff1[i] = singen();
}
```

Sonucu görmek için Sinüs dalgası üreten programı uygulayın. Sonra, sinbuff1 dizisinin değerlerinin grafiğini görüntüleyin. Sonuç Şekil 1.20'deki gibi olacaktır. Yani sinüs dalgası meydana gelecektir.



Şekil 1.20: Floating point ile IIR filtrenin uygulama sonuçları

- **Sinüs dalga jeneratörü (Fixed point ile uygulama)**
  - **Floating point uygulamasının dsp bordunda doğrudan yapılamaması**

Fixed point ile IIR filtresinde sinüs dalga jeneratörünün yapılması için kullanılan algoritma floating point uygulaması ile tamamen aynıdır. O nedenle, fixed point uygulaması, floating point uygulamasına bakılarak yapılacaktır.

Fixed point uygulaması ile floating point uygulaması arasındaki fark sayısal değerlerin ifadesidir. Kısacası, aşağıdaki floating point uygulamasındaki programda yalnız koyu renkli değerlerin yerine ne yazılması gerektiği düşünülür.

```
float y[3]={0, 0.33873792, 0};        //  y(n),y(n-1),y(n-2)
float a1 = 1.8817615;                 //  2cos(wT)
float a2 = -1.0;

float singen()
{
    y[0] = a1*y[1] + a2*y[2]; //Get y(n)
```

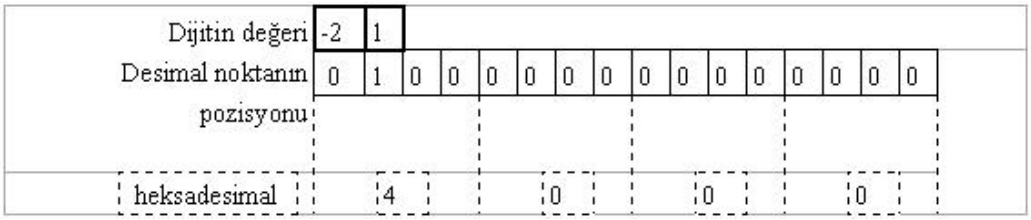
- **Q Format**

Öncelikle, değişkenleri fixed point tipi olan "int" tipine değiştirmeliyiz. Burada "int" tipinin 16 bit olduğu kabul edilir.

Ayrıca, "0", "0.33873792", "1.8817615" sayılarını ifade etmek için Q14 format kullanılır. Bu +2 ‘den -2’ye kadar olan sayılar arasında ifade edilebilir.

- **Q14 format sayısal değerinin ifadesi**

Q14 format, "1", "0", "-1", ve "-2" değerleri sırasıyla "0x4000", "0x0000", "0xc00", ve "0x8000", olarak ifade edilir. "0x" ifadesi bir heksadesimal sayıyı ifade eder. Ayrıca, "0x4000" heksadesimal sayısının desimal değeri 16384 ‘dür (Şekil 1.21).



Şekil 1.21: Q14 formatın "1" değeri

Diğer değerler, örneğimizdeki, y(1) =0.33873792 aşağıda hesaplanır:

"1" of Q14 format x 0.33873792= 16384×0.33873792 = 5549.882081

= 5550(şeklinde ayarlanır)
= 0x15AE.

Benzer şekilde, a1=1.8817615 değeri 0x786F olur. Sinüs dalga jeneratörünün fixed point uygulamasında DSP ‘de kullanılacak sayısal değerler aşağıdaki gibi alınır.

Piyanonun C major key Ra için ses frekansı 440Hz -> $\omega = 2\pi \times 440 = 880\pi$
Örnek frekansı fs = 8,000Hz -> Ts = 1/8000.
Sayısal değer 16bit fixed point ve Q14 formatı ile anlatılır.
[Sabitler]
$a_1 = 2\cos(\omega Ts) = 2\cos(880\pi/8000) = 1.8817615 = 0x786F$
$a_2 = -1 = 0xC000$.
[İlk koşul]
$y(0) = 0$
$y(1) = \sin(\omega Ts) = \sin(880\pi/8000) = 0.33873792 = 0x15AE$.

Yukarıda hesapladığımız bu değerleri programda yerine yazarsak fixed point uygulaması ile yapılan program aşağıdaki gibi olur:

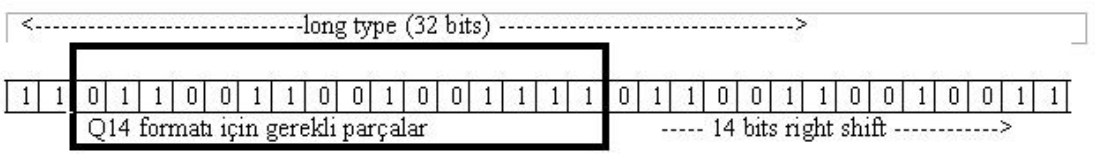
```
int y[3]={0, 0x15AE, 0};            //  y(n),y(n-1),y(n-2)
int a1 = 0x786F;                    //  2cos(wT)
int a2 = 0xC000;

int singen()
{
    y[0] = a1*y[1] + a2*y[2]; //Get y(n)
    y[2] = y[1];                    //delay
    y[1] = y[0];                    //delay
```

- **Hesaplamalar**

Fixed point değerleri ile gerçek hesaplamaları yapılırken bu değerler işlemler esnasında "long" tip kalıbına sahip olurlar. Hesaplamalar sonunda, sonuçların 14 bitin üzerinde alınması gerekecektir.

y[0] = ((long)a1*(long)y[1] + (long)a2*(long)y[2]) >> 14;

Çarpım sonucu "Q14×Q14=Q28 format" olur. Desimal sayıların hesaplama şekli de aynıdır. Örneğin bir desimal sayının çarpımında "5.19×10.19=52.8861" örneğimizdeki gibi çarpılan sayıların virgülden sonra 2 dijit sayısı varken çarpım sonunda 4 dijit olur. Şekil. 2-22'de gösterildiği gibi 14 bit sağa kaydırma yapılarak, Q14 format gerekli bölümü "int" tipin alanına kaydırılabilir.



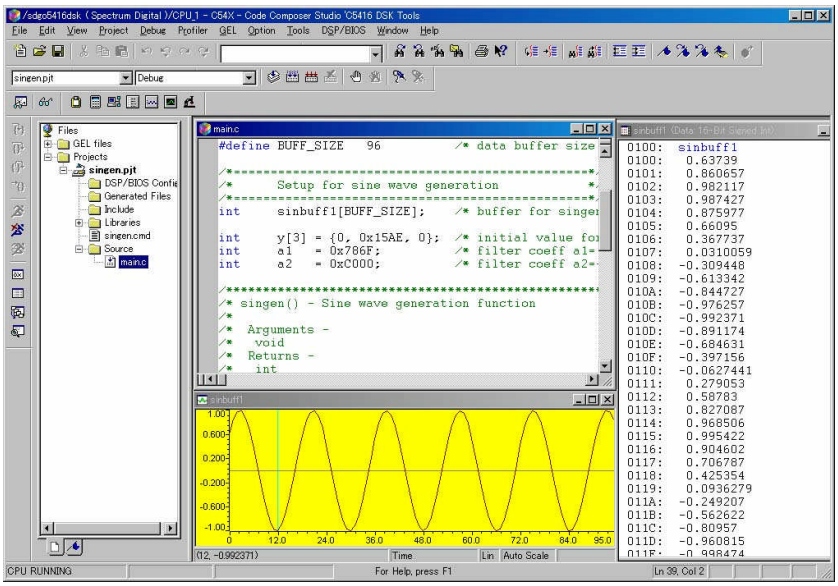
Şekil 1.22: Long tipinden Q14'ün alınması

Şimdi, singen()fonksiyonunun fixed point uygulaması tamamlanmıştır. Programın doğrulanması için main() fonksiyonu yazılır ve program aşağıdaki gibi olur:

```
#define BUFF_SIZE    96

int sinbuff1[BUFF_SIZE];

int y[3]={0,0x15AE,0};          // y(n),y(n-1),y(n-2)
int a1=0x786F;                  // 2cos(wT)
int a2=0xC000;

int singen()
{
     y[0]=((long)a1*(long)y[1] + (long)a2*(long)y[2]) >> 14;
     y[2]=y[1];                 // delay
     y[1]=y[0];                 // delay

     return(y[0]);
}

void main()
{
      int i;
      for (i=0; i<BUFF_SIZE; i++)
           sinbuff[i]=singen()
}
```

- **Programın uygulanması**

Bu programı derleyin ve çalıştırın. Sonucu görmek için sinbuff1[] dizisinin değerlerini grafiksel olarak floating point uygulamasında olduğu gibi görüntüleyin. Fakat bu durumda, "DSP Data Type(DSP veri tipi)" "16bit-signed integer" olarak belirtilmiş olacaktır. Sonuç Şekil 1.23'teki gibi olur. Mükemmel bir sinüs dalga üretimi tamamlanmış olacaktır.



Şekil 1.23: Fixed point uygulama sonuçları

Bu arada,

y[0]=((long)a1*(long)y[1] + (long)a2*(long)y[2]) >> 14;

işlemi yerine aşağıdaki işlemler de kullanılabilir:

a) y[0] = ((long)a1*y[1] + (long)a2*y[2]) >> 14;

b) y[0] = ((long)a1*y[1] + a2*y[2]) >> 14;

c) y[0] = ((long)al*y[1] >> 14) - y[2];

C dilinde, değişkenlerin tipi bir işlem sonunda ihtiyaç duyulan "long" tipe otomatik olarak denkleştirilir. O nedenle, y[1]'in tipi otomatik olarak "long" tipe atanır. Benzer şekilde, b) ve c) 'deki işlemlerde sorunsuz hesaplanır.

Örnek:

Bilindiği gibi IIR filtresi doğrusal olmayan kararsız sistemler için kullanılır ve FIR filtreden yaklaşık 10-100 kat kadar daha verimlidir. Sinüs dalga jeneratörü (floating point ile uygulama) programı için örneğin bir piyanonun standart ayarları yapılırken ses frekansının (örneğin C majör anahtarı Ra) f=440Hz olması istenir. Eğer bu örneğimizdeki gibi sabitler ve birinci koşul verilmiş ise, çıkış aşağıdaki gibi hesaplanır.

440Hz -> $\omega = 2\pi f = 2\pi \times 440 = 880\pi$

Örnek frekans fs=8000Hz -> $Ts = \frac{1}{fs} = \frac{1}{8000}$.

Böylece, Constant

$$a_1 = 2\cos(\omega Ts) = 2\cos(880\pi \times \frac{1}{8000}) = 1.8817615$$
$$a_2 = -1.$$

Birinci koşul

$$y(0) = 0$$
$$y(1) = \sin(\omega Ts) = \sin(880\pi \times \frac{1}{8000}) = 0.33873792.$$

UYGULAMA FAALİYETİ

SİNÜS DALGA ÜRETECİ PROGRAMI	
İşlem basamakları	**Öneriler**
➢ CCS programını açınız. ➢ “Project” menüden “New” komutunu seçiniz. ➢ “Project Creation” isimli açılan pencereye proje ismi olarak “sinüs” yazınız. ➢ “File” menüden “New” -> “Source File” seçerek ekrana gelen “Untitled1” program penceresine C dilinde sinüs dalga üreteci programı yazınız. ➢ “File” menüden “Save As” seçeneğini seçerek dosyanın ismini “sinüs.c” olarak yazın ve “Save” butonunu tıklayınız. ➢ “Project” menüden “Ad Files to Project” seçeneğini seçiniz. Ekrana gelen pencerenin içinden “sinüs.c” tıklayarak kaynak dosyayı projenize ekleyiniz.	➢ “Project Creation” penceresinde projenin ismi girilince otomatik olarak “Location” kutucuğunda belirir.Eğer proje yeri otomatik belirtilmemişse doğru adresi aşağıdaki gibi giriniz. “c:\ti\myproject\sinüs\” ➢ CCS program editörünün sol yanındaki “Project view” penceresinde “sinüs.pjt” proje dosyasının yanındaki + işaretine tıklayarak proje dosyasının içerdiği "DSP/BIOS Config", "Generated Files", "Include", "Libraries" ve "Source" isimli içi boş 5 alt dosyanın varlığını teyit ediniz. ➢ “sinüs.c” kaynak dosyasının projenize eklendiğini “sinüs.pjt” dizini altındaki “source” dosyasının yanındaki + işaretine tıklayarak gözleyiniz.

SİNÜS DALGA ÜRETECİ PROGRAMI	
İşlem basamakları	**Öneriler**
➢ Yeni bir konfigürasyon dosyası oluşturmak için menüden "File" -> "New" -> "DSP/BIOS Configuration..." seçiniz. Kullandığımız DSP'nin konfigürasyon dosyası olan "dsk5416.cdb" dosyasını seçin ve "OK" tıklayınız. Ekranda "Config1" isimli yeni bir pencere gözükecektir. Bu dosyayı "sinüs.cdb" olarak kaydettiğimizde oluşturulan konfigürasyon dosyasını projemize eklemiş oluruz. File menüden "Save As" seçeneğini seçin ve konfigürasyon dosyasının ismini "sinüs.cdb" olarak girin ve Save butonunu tıklayınız. ➢ Menüden "Project" -> "Add Files to Project" seçeneğini tıklayın.Ekrana gelen pencerede "File of type" kutucuğundan konfigürasyon dosyası seçeneğini "Configuration File (*.cdb)" seçin. Konfigürasyon dosyası "sinüs.cdb" gözükecektir Dosyayı seçiniz ve "open" butonunu tıklayınız. ➢ "Project" -> "Build option" seçene- ğini tıklayın gelen menüde "Compiler" sekmesini tıklayınız ve Category 'nin içinden "Advanced" seçin.Buradan "Use Far Calls" seçeneğini tıklayın. Yine aynı menüden bu sefer "Linker" sekme- sinde Category 'nin içinden "Basic" seçiniz.Burada "Include Libraries" kutucuğuna "dsk5416f.lib" yazarak "OK" tıklayınız.	➢ "sinüs.cdb" dosyasının projeye eklendiğini "Project Creation" penceresinden "DSP/BIOS Configuration..." dosyasının yanındaki ⊞ işaretine tıklayarak görebilirsiniz. ➢ Konfigürasyon dosyası yapıldığı zaman linker komut dosyası otomatik olarak yapılır. Burada, sadece linker dosyasını (.cmd uzantılı dosyayı) projeye eklemeliyiz.

SİNÜS DALGA ÜRETECİ PROGRAMI	
İşlem basamakları	**Öneriler**
➢ Menüden "Project" -> "Add Files to Project" seçeneğini tıklayın.Ekrana gelen pencerede "File of type" kutucuğundan linker komut dosyası seçeneğini "Linker Command File(*.cmd)" seçin. Komut dosyası gözükecektir (örneğimizdeki "sinüscfg" dosyası gözükür). Dosyayı seçin ve "open" butonunu tıklayınız. ➢ "Option" menüden -> "customize" -> seçeneğini seçin ve ekrana gelen menüde "Program Load Options" sekmesini tıklayın.Burada "Load Program After Build" seçeneğinin aktif olması için yanındaki kutucuğu tıklayınız. ➢ Project -> menüden save seçeneğini tıklayınız. ➢ “Projects" -> menüden "rebuild All" seçeneğini seçin veya araç çubuğu üzerindeki  (Rebuild All) ikonunu tıklayınız. ➢ Programı çalıştırmak için, araç çubuğu üzerindeki  (Run) butonu tıklayınız.	➢ "sinüscfg.cmd" dosyası projeye eklendiğini "Project Creation" penceresinden de görebilirsiniz. ➢ Proje derlendikten sonra otomatik olarak programın DSP borduna yüklenmesi için CCS programında bazı seçenekler aktif yapmalısınız. ➢ Projeyi tamamlamak için ise kaydetmek zorundasınız. ➢ Çalışabilir bir dosya yapmak için derleme "Build" yapmalısınız. ➢ Bir hatanız yoksa CCS editörünün en altındaki "Build" ekranında : “Build Complete, 0 Errors,0 Warnings,0 Remarks.” mesajını görürsünüz. ➢ Eğer programınızda hata varsa derleme yapıldıktan sonra program DSP borduna gönderilmez ve hatalar yine build ekranında sırasıyla yazar.

SİNÜS DALGA ÜRETECİ PROGRAMI	
İşlem basamakları	**Öneriler**
➢ View menüden -> Graph -> Time/Frequency seçeneğini seçiniz. ➢ Graph Property Dialog kutusunda; Graph Title kutucuğuna sinbuff1,Start Address kutucuğuna sinbuff1, Acquisition Buffer Size kutucuğuna 96, DSP Data Type kutucuğuna 32-bit floating point girip OK butonunu tıklayınız. ➢ Ekranda görüntülenen sinbuff1[] dizisinin değişkenlerinin oluşturduğu noktalarla çizilen sinüs grafiğini gözleyiniz. ➢ Sinbuff2[] ve Sinbuff1_2[] için de aynı işlemleri tekrarlayarak ekrana 2 sinüs dalga grafiğini de getiriniz.	➢ Bu seçenek ile "Graph Property Dialog" penceresi ekrana geldiğine dikkat ediniz. ➢ Graph Property Dialog kutusunda; grafiğin etiket ismini ve başlangıç adresini sunbuff1 seçtiğinize, 96 tane değişken kullanarak 32 bitlik adresleme yapılarak sinüs dalgasının oluşturulacağına dikkat ediniz. ➢ Ekranda görüntülenen 3 sinüs dalgasının grafiklerini inceleyiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları doğru “D” veya yanlış “Y” olarak cevaplayınız.

1. Sinüs dalga üreteci : Ayarlanan frekans aralığında sinüsoidal dalga üreten bir fonksiyon üretecidir.

2. Sinüs dalga üretecinde frekansı değiştirdikçe sinüsoidal dalganın periyodu da değişir.

3. CCS programının kurulumu sırasında "C:" sürücüsünün kök dizininde "ti" dosyası oluşturulur. "ti" dosyası içinde de "myprojects" dosyası oluşturulur.

4. CCS program editöründe,  (Rebuild All) ikonunu tıklanarak program derlendiğinde eğer bir hatanız yoksa çalışabilir dosya (örneğin sinüs.out) oluşturulur. Hem de çalışabilir dosya (sinüs.out) DSP borduna gönderilmiş olur.CCS editörünün en altındaki "Build" ekranında :

“Build Complete,
0 Errors,0 Warnings,0 Remarks.” mesajı yazar.

5. DSP borduna yüklenen programı çalıştırmak için araç çubuğu üzerindeki  (Run) butonu tıklanır.

6. DSP borduna yüklenen programın çalışmasını sonlandırmak için araç çubuğu üzerindeki  (Halt) butonu tıklanır.

7. Sinüs daga üreteci programının çalışma süresince bir destek kütüphanesi (DSPLIB kütüphanesi) kullanılır. Bu örnekte, sin() fonksiyonu bu destek kütüphanesinden sağlanır. Destek ktüphanesini programın içermesi için programa #include <math.h> eklenmelidir.

8. Sinüs daga üreteci programında sonuçlar "sinbuff1[ ]" ve "sinbuff2[ ]" isimli iki dizide saklanır. Genellikle yapılan işin doğrulanması için bu değişkenlerin gösterilmesinde "Watch window" kullanılır..

9. Hafızada saklanan değerleri ve adreslerini doğrudan kontrol etmek için "Memory window" kullanılır.

10. "Watch window" ve "Memory window" kullanılarak, uygulama sonuçlarının doğruluğu görülebilir. Ayrıca, CCS programı hafızadaki değerlerin görüntüsünü grafiksel olarak da gösterebilir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları faaliyete geri dönerek tekrar inceleyiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Ses değiştirici uygulamasını istenen frekans değerinde yapabileceksiniz.

ARAŞTIRMA

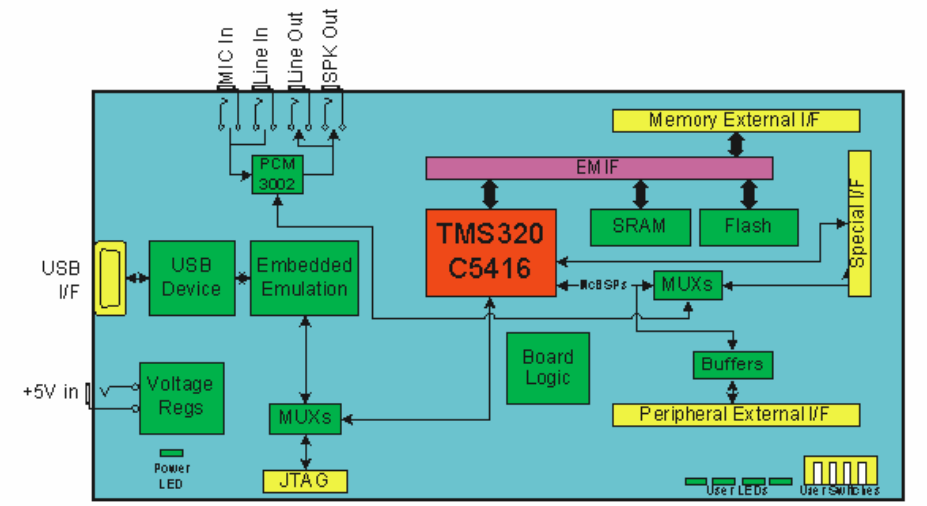
- Ses değiştirici hakkında bir araştırma yaparak, rapor haline getiriniz.

2. SES DEĞİŞTİRİCİ NEDİR

Burada, DSP kartının ses fonksiyonunun nasıl kullanılacağı anlatılacaktır. Örnek olarak “Sinüs dalga üreteci”, “Echo”, “Ses Değiştirici(i)” ve “Ses Değiştirici(ii)” programları verilmiştir. Bu programlar bir çok teknik faktörleri içermektedir.

2.1. Ses Sisteminin Tanıtımı

TMS320C5416DSK kartının blok diyagramı aşağıdaki gibi gösterilmektedir:



Şekil 2.1: TMS320C5416DSK nin blok diyagramı

Line-in konnektörü ve line-out konnektorü A/D dönüştürücü entegresine bağlıdır (Texas Instruments firması tarafından bu ses entegresi ya da ses CODEC entegresi PCM3002 olarak adlandırılmıştır). PCM3002 DSP entegresine Çok-kanallı tamponlanmış seri portların (Multi-channel Buffered Serial Ports (McBSP)) ikinci portu vasıtasıyla bağlanmaktadır. Bu fonksiyonları kullanabilmek için C5416(DSP) entegresinin, McBSP’ nin ve PCM3002’nin tanımlanması gereklidir.

2.1.1. Dsp Entegresinin Tanımlanması

İlk olarak, DSK5416 kartının tanımlanması gerekir. Bu da aşağıdaki şekilde yapılır:

DSK5416_init();

Bu fonksiyonun da kullanılabilmesi için "dsk5416.h" başlık dosyası (header file) programımıza eklenmelidir.

2.1.2. MCBSP*'nin Tanımlanması

PCM3002 McBSP2 portuna bağlıdır. Bu portu tanımlamak için, özelliklerinde değişiklikler yapmak gerekmektedir (Tablo 1-1).

*(Çok-kanallı tamponlanmış seri portlar(Multi-channel Buffered Serial Ports)

Kategori	Özellik	Değer
Transmit (Gönderim)	Clock Polarity(CLKXP)	Falling Edge
	Word Length Phase1(XWDLEN1)	32-bits
	Words/Frame Phase1(XFRLEN1)	2
Receive (Alım)	Clock Polarity(CLKRP)	Rising Edge
	Word Length Phase1(RWDLEN1)	32-bits
	Words/Frame Phase1(RFRLEN1)	2
Sample-Rate Gen.	Generator Clock Source(CLKSM/SCLKME)	BCLKR Pin
	Frame Width(FWID)	32

Tablo 2.1: McBSP için yapılması gereken tanımlama değerleri

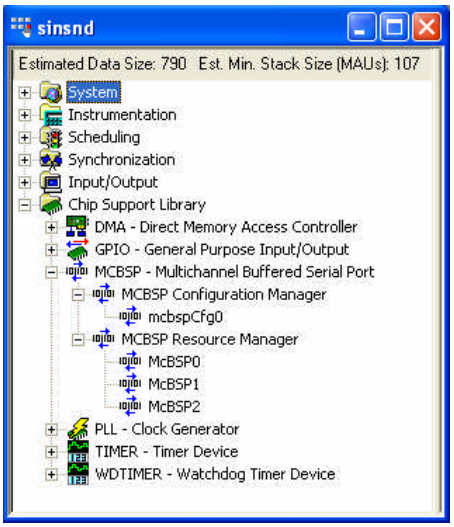
Bu özelliklerin değiştirilmesi için gerekli olan işlem adımları aşağıda verilmektedir:

"DSP/BIOS Conf" un + sını tıklayın-> ?????.cdb dosyasını çift tıklayın->
Chip Support Library nin + sını tıklayın-> MCBSP nin + sını tıklayın - >
MCBSP Configuration Manager üzerinde sağ butona tıklayın-> mcbspCfg ekleyin.

Bunun ardından,

MCBSP configuration Manager +-> mcbspCfg0 üzerinde sağ tıklayınız ->

Properties sekmesini tıklayınız.

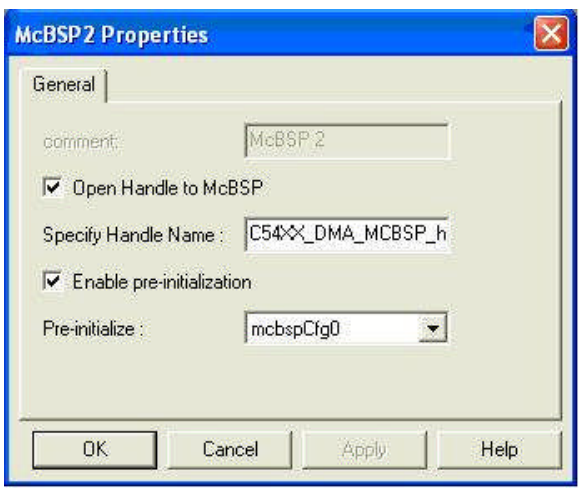


Şekil 2.2: McBSP2' nin konfigürasyonu

Pencere açıldığında Tablo 2.1'de gösterildiği şekilde gereken özellikleri değiştiriniz. Değişiklikler yapıldığında McBSP2 nin konfigürasyonunu onaylamak için aşağıdaki işlemleri yapmamız gereklidir:

MCBSP Resource Manager [+]-> McBSP2 üzerinde Sağ tuşu tıklayınız-> Properties sekmesini işaretleyiniz.

Açılan pencerede (Şekil 2.3), "Open Handle to McBSP" işaretlenmiş olmalıdır ve "Specify Handle Name" alanına değeri "C54XX_DMA_MCBSP_hMcbsp" olarak girilmelidir. Bir de "Enable pre-initialization" işaretlenmeli ve "Pre-initialization" alanınıdan "mcbspCfg0" seçilmelidir. Ardından OK tıklanır ve McBSP nin konfigürasyonu tamamlanmış olur.



Şekil 2.3: McBSP2 özellikleri

Bu konfigürasyonun kullanılabilmesi için de eklenti dosyasına “(kaynak programın dosya adı) cfg.h” ihtiyaç vardır.

2.1.3. PCM3002’nin Tanıtımı ve Bağlantı Kurulması

Bu kartta yer alan PCM3002 ses entegresi giriş yapılan ses sinyallerini 48 kHz,16 bit, stereo kanal (tabii ki, entegrenin başka ayarları da olmakla birlikte) ile örneklemektedir. Bu ses entegresini kullanabilmek için “dsk5416_pcm3002.h” başlık dosyası programa eklenmelidir. Aşağıda konfigürasyon bilgisi ve ses entegresinin tanıtımı gösterilmektedir :

```
DSK5416_PCM3002_Config setup = {
        0x1ff,      // Set-Up Register 0 - Left channel DAC attenuation volume(Sol kanal DAC
zayıflaması)
        0x1ff,      // Set-Up Register 1 - Right channel DAC attenuation(Sol kanal DAC
zayıflaması)

        0x0,        // Set-Up Register 2 - Çeşitli kontroller eklenmektedir(örn: güç-düşürme
modu)
        0x0         // Set-Up Register 3 - Codec veri biçim kontrolü.
};
void main()
{
        DSK5416_PCM3002_CodecHandle hCodec;  // Ses entegresi için Tutamaç
        ...
        hCodec = DSK5416_PCM3002_openCodec(0, &setup);
                                            // ses entegresini aç ve Tutamacı geri döndür..
```

Şekil 2.4: Ses entegresinin tanımlanması

Register 0 ve 1 aynı değere sahiptir. 0x100(ses yok) den 0x1ff (en yüksek ses) e kadar değerler alırlar. Register 3 de yer alan sıfırın değeri ise:

DAC -> 16-bit, MSB ilk, 32-bit word içinde sola yanaşık.
ADC -> 16-bit, MSB ilk, 32-bit word içinde sağa yanaşık.

Bu yaslama işlemi ile sol ve sağ kanallardan elde edilen ses sinyali tek bir word içinde ifade edilebilir.

Ses entegresini programın içerisinde kullanabilmek için, configürasyon değerleri kullanılarak açılmış olması gerekir. “open” fonksiyonu geriye ses entegresini kullanmak için gerekli olan “handle” (tutamaç) değerini döndürür.

hCodec = DSK5416_PCM3002_openCodec(0, &setup);
Sinyali çıkışa yollamak için, aşağıdaki fonksiyon kullanılır :
DSK5416_PCM3002_write16(hCodec, val);

Sol ve sağ kanallar için aynı write fonksiyonu kullanılır. İşlem ses kartına bir değer gönderilene kadar durmalıdır. Bu nedenle program aşağıdaki gibi yazılır:

while (!DSK5416_PCM3002_write16(hCodec, l_val);
while (!DSK5416_PCM3002_write16(hCodec, r_val);

burada “l_val” ve “r_val” in anlamı sırasıyla sağ kanal ve sol kanal değerleridir. Bunların değerleri 16-bit integer (Int16) olarak tanımlanmıştır.

Ses entegresinden okuma işlemi yapmak için gereken fonksiyon aşağıda gösterilmektedir:

DSK5416_PCM3002_read16(hCodec,&val);

Burada “val” giriş değerinin girileceği değişkeni, “&” bu değişkenin adresini göstermektedir. Benzer şekilde, bu fonksiyon sadece bir kanaldan okuma yapmaktadır, dolayısıyla sol ve sağ kanallar için iki kez bu fonksiyonun kullanımına ihtiyaç vardır.

2.1.4. “LED” “DIP Switch”ler (Çoklu Anahtar)

Her ne kadar “LED”lerin ve “DIP Switch”lerin ses sistemi ile ilişkisi olmasa da bu konuların anlatımına burada yer verilmiştir.

DSP kartında programlarda durumlarına yer verilebilecek dört LED bulunmaktadır. Bu LED leri kullanabilmek için, ilk olarak, “dsk5416_led.h” eklenti dosyasına programda yer verilmesi gerekmektedir. Ardından, aşağıdaki fonksiyonlar LED denetleme programımızda kullanılabilir hale gelmektedir.

DSK5416_LED_on (LED No.)
DSK5416_LED_off (LED No.)
DSK5416_LED_toggle (LED No.)

Burada “LED No.” 0 dan 3e kadar değerler alabilir. LED 0 kartta “D9” olarak gösterilmektedir. “on” fonksiyonu LED I yakmakta, “off” söndürmektedir ve “toggle” LED in o anki durumunu terslemektedir. Ayrıca, DSP kartında dört DIP switch yer almaktadır. Bu anahtarlar kullanıcının basit geri besleme yapmasına imkan verir. Bu anahtarların durumu program ile okutulabilir. Buna ilişkin fonksiyon aşağıda verilmektedir:

DSK_5416_DIP_get(DIP sw No.)

Burada “DIP sw No.” 0 dan 3 e kadar değerler almaktadır. Switch 0 programda “1” olarak belirtilirken kartın üzerinde “S2” olarak işaretlenmiştir. Geriye dönen değer bir(anahtar açık) ya da sıfır(anahtar kapalı) dır.

2.2. Sinüs Dalga Üreteci

Bu bölümde, sinüs dalgasının sesi hoparlörlerinizden(veya kulaklıklarınızdan) duyulabilecektir.

2.2.1. Matematiksel Fonksiyon Kullanarak Sinüs Dalgası Oluşturulması

İlk olarak, sinüs dalga tablosunun oluşturulması için önceki bölümde anlatılandan farklı bir yöntem anlatılmaktadır. Bu fonksiyonun kullanılabilmesi için programa "math.h" başlık dosyasının eklenmesine gerek duyulmaktadır.

```
void InitSineTablo(void)
{
        int i;

        for (i = 0; i < SINE_TABLO_SIZE; i++)
            sineTablo[i] = (int)(sin(2.0*PI*(i/SINE_TABLO_SIZE)) * SINE_MAX);
}
```

Şekil 2.5: Sinüs dalga üreteci

Dalga SINE_TABLO_SIZE(=48, bu durum için) a bölünmektedir. Çünkü sin() fonksiyonu radyan cinsinden değer döndürmektedir (sinüs dalgasının bir çevrimi 2π [radian] dır), sin() ün bağımsız değişkeni şöyle olacaktır:

2.0 * PI * (i / SINE_TABLO_SIZE).

i değişkeni sıfır(0) dan kırk sekize(48) kadar değerler almaktadır, bu nedenle, (i/SINE_TABLO_SIZE) sıfır(0) dan bir(1) e kadar değer döndürecektir.Bu nedenle yukarıdaki anlatımın değeri sıfır(0) la 2π arasında olacaktır.SINE_MAX sinüs dalgasının en yüksek değeridir.

2.2.2. "sinsnd.c" Programı

"sinüs dalga üreteci" programı ("sinsnd" olarak isimlendirilmiştir) aşağıdaki gibi olur:

```
// sine wave generator. Output for 30 seconds.
#include "sinsndcfg.h"                          // konfigürasyon için
#include "dsk5416.h"                            // dsk kart fonksiyonları
#include "dsk5416_pcm3002.h"                    // ses fonksiyonları
#include <math.h>                               // sin()

#define SINE_TABLO_SIZE   48                    // sinüs tablosu
#define PI               ((double)3.1415927)    // pi
#define SINE_MAX         0x7FFE                 // entegreye maksimum değer

int sineTablo[SINE_TABLO_SIZE];                 // sinüs tablosu

DSK5416_PCM3002_Config setup = {
    0x1ff,                          // Set-Up Reg 0 - Sol kanal DAC attenuation
    0x1ff,                          // Set-Up Reg 1 - Sağ kanal DAC attenuation
    0x0,                            // Set-Up Reg 2 - Çeşitli ctl v.b. power-down modu
    0x0                             // Set-Up Reg 3 - Codec verisi biçim kontrolü
};

void InitSineTablo(void)            // sine tablosu tanıtımı
{
       int i;

       for (i = 0; i < SINE_TABLO_SIZE; i++)
           sineTablo[i] = (int)(sin(PI*(i/SINE_TABLO_SIZE*360.0)/180.0) * SINE_MAX);
}

void main()
{
    DSK5416_PCM3002_CodecHandle hCodec;         // pcm3002 için tutamaç
    int i, j;

    DSK5416_init();                             // DSK5416 nın tanıtımı
    InitSineTablo();                            // sinüs tablosu yapımı

    hCodec = DSK5416_PCM3002_openCodec(0, &setup); // codec in açılması

    for (i = 0; i < 30000; i++) {               // 30 saniye
        for (j = 0; j < SINE_TABLO_SIZE; j++) {     // bir çevrim
            while (!DSK5416_PCM3002_write16(hCodec, sineTablo[j]));      // sol kanal
            while (!DSK5416_PCM3002_write16(hCodec, sineTablo[j]));      // sağ kanal
        }
    }
    DSK5416_PCM3002_closeCodec(hCodec);         // pcm3002 yi kapat
}
```

Bir projenin daha nasıl tanımlanacağının bilgisi daha önce verilmişti. Böylelikle proje oluşturulur, daha sonra, kaynak program, konfigürasyon dosyası ve link komutu dosyaları hazırlanır ve projeye eklenir. Konfigürasyon dosyası McBSP yi kullanabilmek için yeniden yazılır.

2.2.3. Yürütme

"build-all" komutu işletildiğinde eğer hata yoksa yürütülebilir obje DSP kartına transfer edilir. Program yürütüldüğünde, 30 saniye boyunca ses duyulacaktır.

Soru:

Bu sesin frekansı nedir? Niçin 30 saniye sürmektedir?

Örnek:

DSP kartında programlarda durumlarına yer verilebilecek dört LED bulunmaktadır. Bu LED' leri kullanabilmek için ilk olarak, "dsk5416_led.h" eklenti dosyasına programda yer verilmesi gerekmektedir. Ardından, aşağıdaki fonksiyonlar LED denetleme programımızda kullanılabilir hale gelmektedir.

DSK5416_LED_on(LED No.)
DSK5416_LED_off(LED No.)
DSK5416_LED_toggle(LED No.)

Burada “LED No.” 0 dan 3e kadar değerler alabilir. LED 0 kartta “D9” olarak gösterilmektedir. “on” fonksiyonu LED I yakmakta, “off” söndürmektedir ve “toggle” LED in o anki durumunu terslemektedir.Ayrıca, DSP kartında dört DIP switch yer almaktadır. Bu anahtarlar kullanıcının basit geri besleme yapmasına imkan verir. Bu anahtarların durumu program ile okutulabilir. Buna ilişkin fonksiyon aşağıda verilmektedir:

DSK_5416_DIP_get(DIP sw No.)

Burada “DIP sw No.” 0 dan 3 e kadar değerler almaktadır. Switch 0 programda “1” olarak belirtilirken kartın üzerinde “S2” olarak işaretlenmiştir. Geriye dönen değer bir(anahtar açık) ya da sıfır(anahtar kapalı) dır.

DİKKAT: Aşağıdaki adımları gerçekleştiriniz.
1) DSP kartında programlarda durumlarına yer verilebilecek dört LED bulunduğuna dikkat ediniz.
2) “dsk5416_led.h” eklenti dosyasını programa ekleyiniz.
3) Aşağıdaki fonksiyonları LED denetleme programınızda kullanılabilir hale getiriniz. DSK5416_LED_on(LED No.) DSK5416_LED_off(LED No.) DSK5416_LED_toggle(LED No.)
4) DSP kartında dört DIP switch yer almaktadır. Bu anahtarlar kullanıcının basit geri besleme yapmasına imkan verir.
5) Bu anahtarların durumu program ile okutulabilir. Buna ilişkin fonksiyonu aşağıdaki gibi yazınız. DSK_5416_DIP_get(DIP sw No.)

2.3. Echo

“Echo” programı sesin DSP kartının line-input girişinden girip line-output çıkışından hiçbir işleme tabi tutulmaksızın yollandığı bir programdır. Aşağıdaki gösterilmekle beraber, bu programda da yapılan sadece okuma ve yazmadır:

```
DSK5416_PCM3002_read32(hCodec,&val);
while(!DSK5416_PCM3002_write32(hCodec,val));
```

Burada “hCodec” ses entegresinin tutamacıdır ve “val” de giriş ve çıkış sinyalleri için kullanılan değişkendir. Bu 32 bit integer olarak tanımlanmıştır. Sağ kanal ve sol kanal verilerini içermektedir. Program aşağıdaki şekildedir:

```
#include "echocfg.h"

#include "dsk5416.h"
#include "dsk5416_pcm3002.h"

DSK5416_PCM3002_Config setup = {0x1ff, 0x1ff, 0x0, 0x0};

void main()
{
    Int32 val;
    DSK5416_PCM3002_CodecHandle hCodec;

    DSK5416_init();
    hCodec = DSK5416_PCM3002_openCodec(0, &setup);

    while(1) {
       DSK5416_PCM3002_read32(hCodec,&val);
       while(!DSK5416_PCM3002_write32(hCodec,val));
    }
}
```

Bu programın yürütülebilmesi için, McBSP için gerekli konfigürasyon hemen hemen aynıdır. Eğer mikrofon ve kulaklık DSP kartına bağlanırsa, bir uğultu duyulabilir, çünkü hoparlörün çıkış sesi mikrofon tarafından tekrar alınacaktır.

2.4. Sayısal İşaret İşleme Deney Seti Ses Değiştirme Uygulaması İçin Nasıl Kullanılır

2.4.1. Ses Değiştirici (I)

2.4.1.1. Ses Değiştiricinin Temel Mantığı

Öğrenme faaliyeti 1'deki örnekte, f1 ve f2 frekanslarına sahip iki sinüs dalgasının çarpımı alınmıştı. Sonuç olarak, iki frekansın sentezlenmesi ve |f1+f2| [Hz] ve |f1-f2| [Hz] e taşınması ile oluşan frekanslar gözlenmişti.

Fourier dönüşümü bize şunu öğretmişti; ses sinyali farklı frekansların sentezlenmiş halidir. Bu nedenle eğer sinüs dalgasının frekansı olan F[Hz] ses sinyali ile çarpılırsa sonuçta oluşan ses sinyali frekansı |ses frekansı + F| ve |ses frekansı - F| değerlerini alacaktır. Sonuç olarak, esas ses sinyalinin niteliği değişmiş ve ses başka bir kişinin sesi halini almıştır.

2.4.1.2. Program

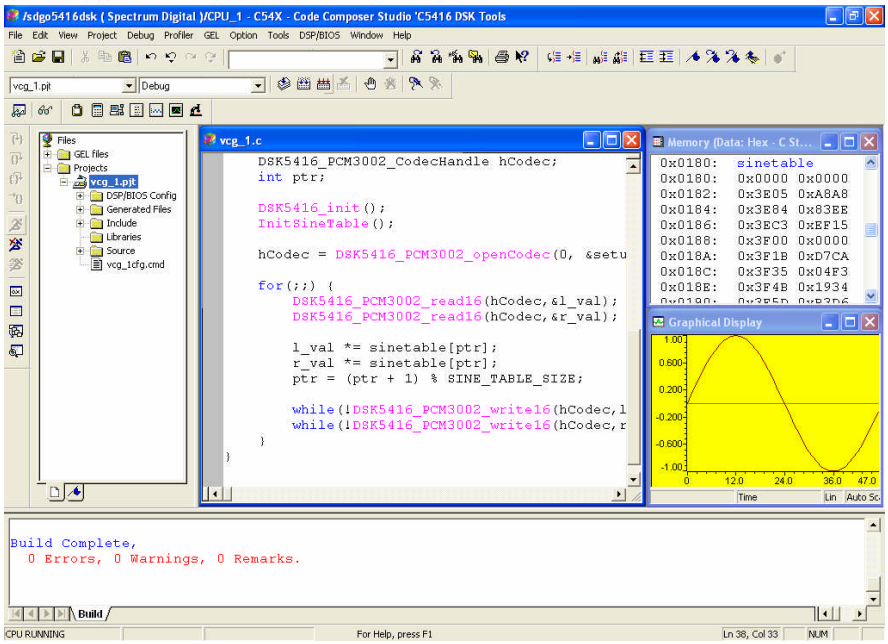
Daha önceden anlatılmış olan "sinsnd.c" programını değiştirelim ve ses sinyalimizi kolayca elde edelim.

InitSineTablo() fonksiyonu "sinsnd.c" nin aynısıdır, fakat sadece SIN_MAX kaldırılmıştır. Bu nedenle, sinTablo[] değeri sıfır(0) dan bir(1) e kadar değerler almaktadır. Ses entegresinden verinin okunması sırasında "echo.c" den sağ ve sol kanallar bölünmüştür. l_val ve r_val değişkenleri sağ ve sol kanalların bilgilerini içermektedir. Çarpma işlemleri

kanal verilerine karşı yapılmakta ve ses entegresine çalınması için geri yazılmaktadır. Bu program sadece okuma, çarpma ve yazma işlemlerini yapmaktadır.
vcg1.c program kodları:

```
// voice changer
#include "vcg_lcfg.h"
#include "dsk5416.h"
#include "dsk5416_pcm3002.h"
#include <math.h>

#define SINE_TABLO_SIZE   48
#define PI               ((double)3.1415927)

float sineTablo[SINE_TABLO_SIZE];

DSK5416_PCM3002_Config setup = {0x1ff, 0x1ff, 0x0, 0x0};

void InitSineTablo(void)
{
    int i;
    for (i = 0; i < SINE_TABLO_SIZE; i++)
       sineTablo[i] = sin(PI*(i*360.0/SINE_TABLO_SIZE)/180.0);
}
void main()
{
    Int16 l_val, r_val;
    DSK5416_PCM3002_CodecHandle hCodec;
    int ptr;

    DSK5416_init();
    InitSineTablo();

    hCodec = DSK5416_PCM3002_openCodec(0, &setup);
    for(;;) {
       DSK5416_PCM3002_read16(hCodec,&l_val);
       DSK5416_PCM3002_read16(hCodec,&r_val);

       l_val *= sineTablo[ptr];
       r_val *= sineTablo[ptr];
       ptr = (ptr + 1) % SINE_TABLO_SIZE;

       while(!DSK5416_PCM3002_write16(hCodec,l_val));
       while(!DSK5416_PCM3002_write16(hCodec,r_val));
    }
}
```

Bu programı yürütmeden önce, mikrofon ve hoparlör karta takılmalıdır. Mikrofona konuşulduğu zaman, dikkatlice dinlendiğinde, değişmiş olan sesin hoparlörlerden çıktığı görülmektedir. sinTablo[] grafik pencereden de izlenebilir(Şekil 2.29).



Şekil 2.6: "vcg_1.c" nin yürütülmesi

2.4.2. Ses Değiştirici (II)

Bu bölümde, DMA kullanan bir ses değiştirici yapılmaktadır. Bu program Ping-Pong buffer(tampon bellek ya da arabellek) teknolojisini, DMA' yı kullanımını, DMA ve McBSP' nin iletişimini ve kesmeleri içermektedir.

2.4.2.1. "Buffer"(Tampon Bellek) in Gerekliliği

Pek çok durumda, DSP gerçek zamanlı sistemler tarafından kullanılır. Bu durumlarda, "giriş veri hızının yükselip azalması"ve "sistem cevap süresinin yükselip azalması" gibi pek çok problemler oluşmaktadır. Bu dalgalanmaları absorbe etmek için, buffer(tampon bellek) kavramı geliştirilmiştir.

2.4.2.2. Giriş Hızının Yükselip Alçalması

Veri düzenli aralıklarla gelmekteyken, giriş hızında herhangi bir dalgalanma gözlenmez. Bununla birlikte, veri sabit hızla gelmezse, "Tampon bellek" gereklidir. Örneğin, mağazada, başka bekleyen yoksa ilk önce siparişi sizin vermeniz mümkün olur. Ama eğer kalabalık varsa beklemek de bir zorunluluk olmaktadır.

Ardından, gerçek işlem yapılmadan önce "tampon bellek" hazırlanır. Eğer giriş hızı işlem hızını aşarsa tampon bellekte biriktirilir. Başlangıçtaki veriden başlayarak ve giriş verisi tampon belleğin kuyruğunda birikmeye devam eder. Bu tampon belleğe "Queue" ("kuyruk") denir.

2.4.2.3. Sistem Cevap Süresinin Yükselip Azalması

"Sistem cevap süresinin yükselip azalması" nın absorbe edilmesi gerektiğinde, Tampon bellek bunda etkili rol oynamaktadır.

"Sistem cevap süresinin yükselip azalması" girişten bir sinyal gelmekte olsa bile sistemin hemen cevap vermemesine neden olabilir.

Genellikle, işlemci çoklu işlemler yapmaktadır. Örneğin, işlemci uzun işlem rutinleri için çok zaman harcamakta iken gelen bir kesme isteğine cevap verilmeyebilir. "Tampon bellek" kullanımı ile bir gecikme işleminin hazırlanması ve yürütülmesi mümkündür.

2.4.2.4. Verinin Yetersizliği

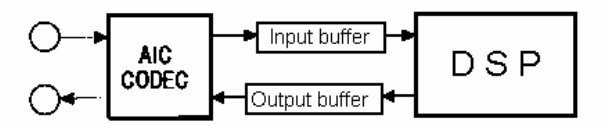
İşlemci bir işlemi veri yetersizliği dolayısıyla yürütemiyorsa, bu durumda da "Tampon Bellek" etkilidir. İşlemci buffer dolu olduğu sürece bekler, ardından, işlem yürütülür.

2.4.3. Ping-Pong Tampon Bellek

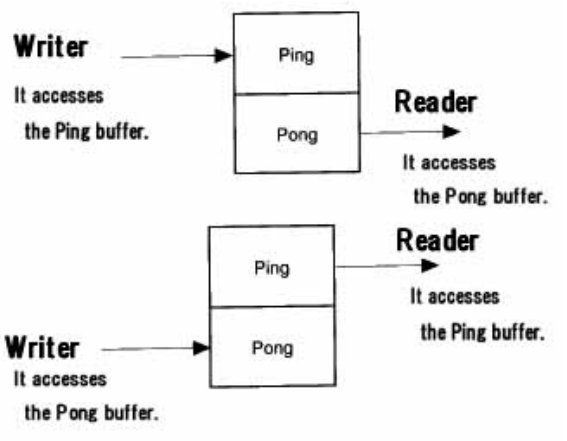
Tampon bellek sırasıyla giriş ve çıkış parçaları için hazırlanır (Şekil 2.7). Ayrıca, her tampon bellek içinde iki farklı tampon belleğe ayrılır bu parçalara "Ping-pong buffer" denir. Tampon belleğin içi iki bloğa bölünmektedir "Ping" ve "Pong". "Yazıcı" ve "Okuyucu" şeklinde çalışır ve aynı anda kullanılmaları mümkündür (Şekil 2.8).

Tamponun esası bellektir. Bu nedenle, "okuyucu" ve "yazıcı" aynı yere eş zamanlı olarak erişemezler. Tamponu "Ping" ve "Pong" adında iki bölüme ayırarak, "okuyucu" nun veriyi "Pong" dan okuması sağlanırken, bu sırada "yazıcı" nın da "Ping" e yazması sağlanır. Böylelikle bu çarpışmanın önüne geçilmiş olur.

Bu belleklerin görüntüsü masa tenisini andırdığı için "Ping-pong buffer" adı verilmiştir.



Şekil 2.7: Giriş ve çıkış tampon belleği

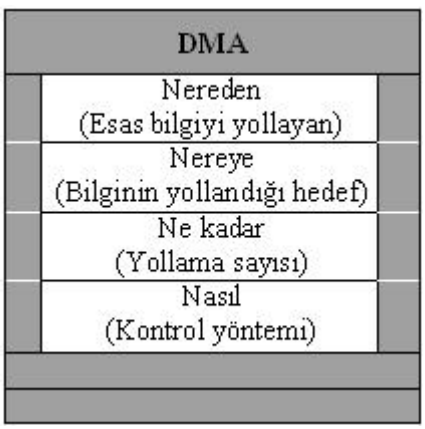


Şekil 2.8:Ping-Pong Buffer düzeneği

2.4.4. DMA (Doğrudan Hafıza Erişimi)

Sayısal işaret işleme işlemi için hafıza adresinden hafıza adresine veri transferi gereklidir. Fakat CPU bu transferi gerçekleştirebilmek için çok fazla güç harcamak zorunda kalır. Bu gibi işlemler için başka birim birim devreye sokulur. Böylece "Doğrudan Hafıza Erişimi" (DMA—Direct Memory Access) sağlanmış olur. "Doğrudan" ifadesiyle CPU dan yardım almaksızın hafızaya ulaşılması kastedilmektedir. Daha açık ifadeyle CPU nun yerini DMA almaktadır. C5426DSK kartında 6 adet DMA yer almaktadır.

DMA bu işlemi CPU'dan yardım almadan yapmaktadır ancak eğer herhangi bir bilgilendirme yapılmazsa, bu işlemin yapılması da mümkün olmayacaktır. DMA veriyi hafızadan hafızaya aktarmaktadır. Aslında, işe başlamadan evvel en azından aşağıdaki bilgileri almak gereklidir (Şekil 2.9).

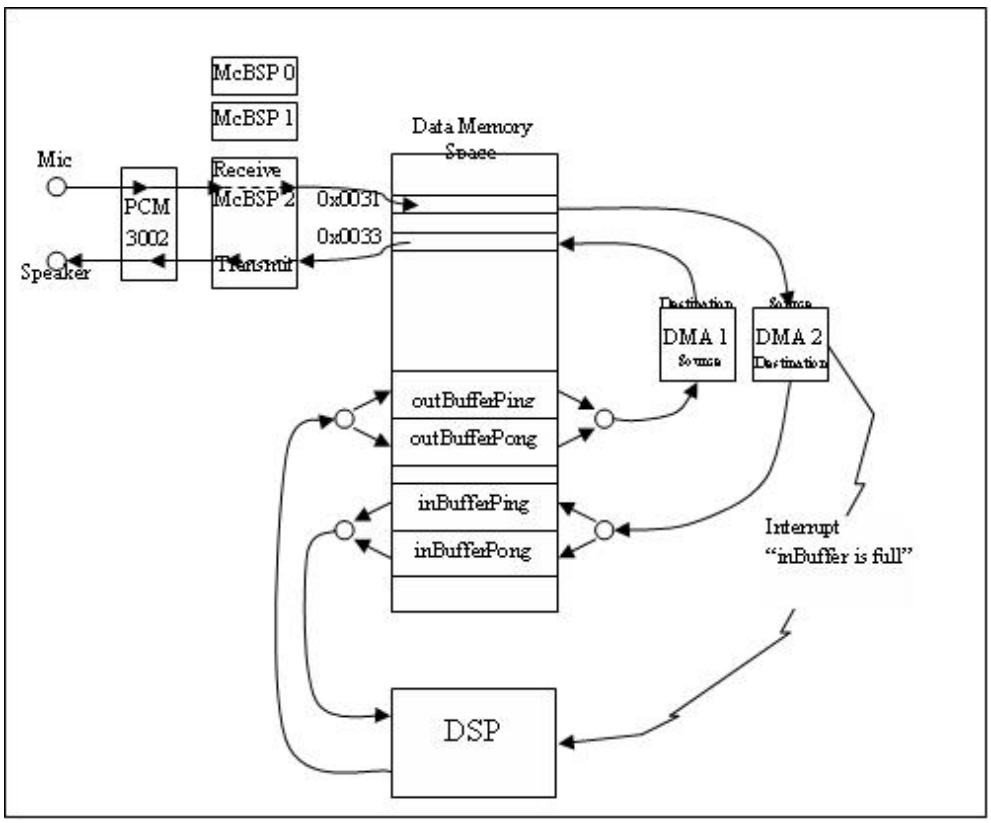


Şekil 2.9: DMA için ihtiyaç duyulan bilgiler

DMA için, bu dört bileşen temel olarak gereklidir. DMA bir registera sahiptir ve CPU işi DMA' ya bırakmadan önce bu registera uygun değeri yazmalıdır.Genelde, DMA tek yönlü bir trafikle transferi gerçekleştirir. DMA için iki yön vardır. Birincisi "Verinin McBSP' den Hafıza' ya aktarımının yapıldığı yön (Alım)", diğeri "Verinin Hafıza' dan McBSP' ye gönderildiği yön (Gönderim)" şeklindedir. Böylelikle iki DMA'nın kullanılacağını anlamış olduk.

2.4.4.1. Veri Akışı (Dataflow)

Şekil 2.10'de bu sistemin veri akışı gösterilmektedir. Ses PCM3002' deki A/D dönüştürücü ile dijitale çevrilir. McBSP PCM3002' den veriyi alır. Aktarılan veri 'veri hafıza alanı' üzerinde 0x0031 adresine konumlandırılır. DMA-2 veriyi adresinden okur ve inBuffer 'a yazar (burada, buffer büyüklüğü 0x400 dür). Eğer Ping-Pong Bufferlardan birisi doluysa DMA-2 DSP' ye kesme gerçekleştirir. DSP inBuffer' daki veriyi işlemeye başlar ve sonucu geriye outBuffer' a yazar. Eğer outBuffer dolu olacak olursa, DSP DMA-1' e başlangıç sinyali yollar. DMA-1 daha önceden belirlenen hafıza adresi olan 0x0033 e outBuffer'dan veri gönderimine başlar. McBSP veriyi adresinden okur ve PCM3002' ye gönderir. PCM3002 D/A dönüştürücüyü kullanarak dijital veriyi analog sinyale dönüştürür ve analog sinyali hoparlöre yollar. Tablo 2.2'de bu akışın özeti verilmektedir.



Şekil 2.10: Veri akış diyagramı

	DMA-1 -- gönderim		DMA-2 – alım	
	Kaynak	Hedef	Kaynak	Hedef
Hafiza Alanı	Data Memory	Data Memory	Data Memory	Data Memory
Başlangıç Adresi	outBuffer	0x0033	0x0031	inBuffer
Artım	İletim sonrası +1	Yok	yok	İletim sonrası +1
Büyüklük	0x400 (değişebilir)		0x400	
Değişim Durumu	Start signal from DSP	Clock from McBSP	Clock from McBSP	Bittiği zaman kesme
Tanımlama	Kullanıcı tarafından (program ile)		Otomatik olarak	

Tablo 2.2: DMA için başlangıç bilgisi

2.4.4.2. DMA' nın Tanımlanması (Başlangıç Durumuna Getirilmesi)

Tablo 2.2 kullanılarak, DMA' yı tanımlamak için iki DMA için gerekli değerler ayarlanmalıdır (Tablo. 2.3, 2.4). Bu işlem için gerekli değerler yukarda tanımlandığı gibidir. Konfigürasyona DMA-2 için "dmaCfgReceive" ve DMA-1 için "dmaCfgTransmit" isimleri verilebilir. Öyle ki, "hDmaRcv" ve "hDmaXmt" de "Tutamaç Adı Belirteci" olarak sırasıyla DMA-2 ve DMA-1' nin özelliklerinde tutamaç olarak kullanılmaktadır. Programlarımızda da "csl_dma.h" eklenti dosyası gerekmektedir.

Kategori	Özellik	Değer
Transfer Modes	Interrupt Generation	Based on IMOD bit
	Multiframe Interrupt At:	End of Frame and Block
	Sync Event	McBSP2 Receive Event (REVT2)
Auto Init	Enable Auto initialization	Checked (True)
	Global Reload Register Usage (AUTOIX)	Each Channel Uses its Own Reload Registers
	Global Source Address Format	Numeric
	Source Address Reload (DMGSA) - Numeric:	0x0031
	Global Destination Address Format	Symbolic
	Destination Address Reload (DMGDA) - Symbolic:	inBufferPing
	Element Count Reload (DMGCR + 1):	1024
	Frame Count Reload (DMGFR + 1):	2
Source/Desitination	Source Address Space	Data Space
	Destination Address Space	Data Space
	Source Address Format	Numeric
	Source Address - Numeric	0x0031
	Destination Address Format	Symbolic
	Destination Address - Symbolic	inBufferPing
	Source Address Transfer Index	No Modification
	Destination Address Transfer Index	Post increment
	Number of Frame (FRAME COUNT + 1)	2
	Elements per Frame (ELEMENT COUNT + 1)	1024

Tablo 2.3: Alım için DMA-2 başlangıç değerleri (dmaCfgReceive)

Kategori	Özellik	Değer
Transfer Modes	Interrupt Generation	Based on IMOD bit
	Sync Event	McBSP 2 Transmit Event (XEVT2)
Auto Init	Enable Auto initialization	Not Checked (False)
Source/Destination	Source Address Space	Data Space
	Destination Address Space	Data Space
	Source Address Format	Symbolic
	Source Address - Symbolic	outBufferPing
	Destination Address Format	Numeric
	Destination Address - Numeric	0x0033
	Source Address Transfer Index	Post Increment
	Element per Frame (ELEMENT COUNT+1)	1024

Tablo 2.4:Gönderim için DMA-1 başlangıç değerleri (dmaCfgTransmit)

Bu tabloda, "Auto Init" in anlamı otomatik-başlangıç durumuna getirmedir. Eğer DMA iletimi biterse, DMA bu değer ile ilk değere otomatik olarak dönmektedir. Sonrasında DMA iletime devam etmektedir. "Address Format" ile kaynak/hedef adreslerin sayısal ve sembolik(değişken) değerleri verilmektedir.

2.4.4.3. McBSP' nin DMA İletimi Kullanımı İçin Hazırlanması

McBSP için aşağıdaki ayarlar gerekmektedir. Nedeni, iletim metodunun DMA ile eşleşmesi gereğidir Yine, eklenti dosyası olarak "csl_mcbsp.h" ye gerek duyulmaktadır.

Kategori	Özellik	Değer
Transmit	Clock Polarity(CLKXP)	Falling Edge
	Word Length Phase 1 (XWDLEN1)	16-bits
	Words/Frame Phase1(XFRLEN1)	4
Receive	Clock Polarity(CLKRP)	Rising Edge
	Word Length Phase 1 (RWDLEN1)	16-bits
	Word/Frame Phase 1(RFRLEN1)	4
General	Breakpoint Emulation	Do Not Stop
Sample-Rate Gen	Generator Clock Source (CLKSM/SCLKME)	BCLKR Pin
	Frame Width(1-256)(FWID)	256
	Frame Period(1-4096)(FPER)	2
Receive Multi channel	RX Channel Enable	2 Blocks(Up to 16Ch./Blk.)
	Receive Channel Enable Register Partition A:	0x0005
Transmit Multi channel	TX Channel Enable	2 Blocks(Up to 16Ch./Blk. -XMCM=01)
	Transmit Channel Enable Register Partition A:	0x000a

Tablo 2.5: McBSP nin başlangıç değerleri ayarları

2.4.4.4. Kesme

Tampon bellekler dolduğu anda DSP'ye kesme yollanır. Kesme sinyali DMA-2'den "Donanım Kesme Numarası 10(HWI_SINT10)" kullanılarak DSP 'ye yollanır. Bu sinyal ile DSP kesme fonksiyonunu çalıştırır. Bu durumda, C-dili ile yazılan kesme fonksiyonuna "filterInterrupt" adı verilir. Bunu zamanlamak için, "csl_irq.h" eklenti dosyasının programda yer alması gerekir ve DSP/BIOS 'un konfigüre edilmesi gerekmektedir. Bunu ayarlamak için gerekli işlem:

- DSP/BIOS Config ->
- vcg_2.cdh ->
- Scheduling ->
- HWI ->
- HWI_SINT10 üzerinde sağ tıklama ->
- Properties
- Açılan pencereden,
- General ->
- function: _filterInterrupt
- Dispatcher ->
- Use Dispatcher işaretlendi.

"_filterInterrupt" ın alt çizgisi ("_") gereklidir çünkü bu konfigürasyonlar bağlantı zamanında kullanılmaktadır. Tüm fonksiyon isimlerine derleyici tarafından derleme esnasında alt çizgi konulmaktadır. Bu nedenle iletim katmanında, fonksiyon isimlerinin önüne alt çizgi koyma gereği doğmaktadır.

Kategori	Özellik	Değer
General	function	_filterInterrupt
Dispatcher	Use Dispatcher	Checked (True)

Tablo 2.6: HWI için başlangıç değerleri

2.5. Ses Değiştirici Programı Nasıl Yazılır

2.5.1. Programlama

Tüm ayarlamalar yapıldıktan sonra, aşağıdaki program "vcg-2.c" ismi verilerek, aşağıdaki gibi yazılabilir:

```
#include "vcg_2cfg.h"
#include "dsk5416.h"
#include "dsk5416_pcm3002.h"

#include <csl.h>
#include <csl_dma.h>
```

```
#include <csl_mcbsp.h>
#include <csl_irq.h>

#include <math.h>

#define S_T_SIZE     48
#define PI       ((double)3.1415927)
#define BUFFSIZE    0x400
#define PING 0
#define PONG          1

Uint16 eventIdRcv;
int temp = 0;
float sineTablo[S_T_SIZE];
int sine_ptr = 0;
DSK5416_PCM3002_Config setup = {0x1ff,0x1ff,0x0,0x0};

#pragma DATA_SECTION(inBufferPing,"inBuffer");
int inBufferPing[BUFFSIZE];

#pragma DATA_SECTION(inBufferPong,"inBuffer");
int inBufferPong[BUFFSIZE];

#pragma DATA_SECTION(outBufferPing,"outBuffer");
int outBufferPing[BUFFSIZE];

#pragma DATA_SECTION(outBufferPong,"outBuffer");
int outBufferPong[BUFFSIZE];

void InitSineTablo(void)
{
  int i;

  for (i = 0; i < S_T_SIZE; i++)
    sineTablo[i] = sin(PI * (i * 360.0 / S_T_SIZE) / 180.0);
}

void filter(int pp)
{
  int i,j;

 float st;
```

```
  if(pp == PING) {
    for (i=0; i< BUFFSIZE/2; i++) {
      j=2*i;
      st = sineTablo[sine_ptr ++];

      outBufferPing[j ] = inBufferPing[j ] * st;
      outBufferPing[j+1] = inBufferPing[j+1] * st;
      sine_ptr %= S_T_SIZE;
    }
  } else {
    for (i=0; i< BUFFSIZE/2; i++) {
      j=2*i;
      st = sineTablo[sine_ptr ++];

      outBufferPong[j ] = inBufferPong[j ] * st;
  outBufferPong[j+1] = inBufferPong[j+1] * st;                                    sine_ptr
%= S_T_SIZE;
    }
  }
}
void filterInterrupt(void)
{
  static Uint32 pingOrPong = PING;

  if (pingOrPong==PING){
    filter(PING);
    DMA_RSETH(hDmaXmt,DMSRC,&outBufferPing);
    DMA_RSETH(hDmaXmt,DMCTR,BUFFSIZE);
    pingOrPong = PONG;
  } else {
    filter(PONG);
    DMA_RSETH(hDmaXmt,DMSRC,&outBufferPong);
    DMA_RSETH(hDmaXmt,DMCTR,BUFFSIZE);
    pingOrPong = PING;
  }

  DMA_start(hDmaXmt);

  if ( MCBSP_xrdy(C54XX_DMA_MCBSP_hMcbsp))
    MCBSP_write16(C54XX_DMA_MCBSP_hMcbsp, 0x0000);
}
```

```
void main()
{
  DSK5416_PCM3002_CodecHandle hCodec;

  DSK5416_init();
  InitSineTablo();
  hCodec = DSK5416_PCM3002_openCodec(0, &setup);

  eventIdRcv = DMA_getEventId(hDmaRcv);

  DMA_FSET(DMPREC,INTOSEL,DMA_DMPREC_INTOSEL_CH0_TO_CH5);

  IRQ_clear(eventIdRcv);
  IRQ_enable(eventIdRcv);
  DMA_start(hDmaRcv);
  DMA_start(hDmaXmt);

  if ( MCBSP_rrdy(C54XX_DMA_MCBSP_hMcbsp) )
    MCBSP_read(C54XX_DMA_MCBSP_hMcbsp);
  if ( MCBSP_xrdy(C54XX_DMA_MCBSP_hMcbsp) )
    MCBSP_write16(C54XX_DMA_MCBSP_hMcbsp, temp);

  temp = IRQ_globalEnable();

  for(;;);
```

2.5.1.1. Tanımlamalar

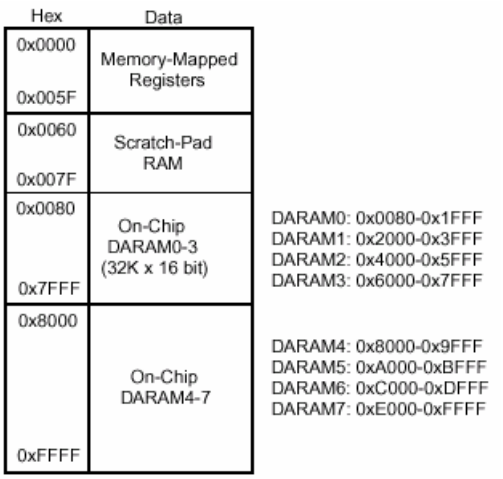
Aşağıdaki satırlar;

```
#pragma DATA_SECTION(outBufferPong,"outBuffer");
int outBufferPong[BUFFSIZE];
```

değişken tanımlamalarıdır. "#" işaretinin anlamı bu satırdaki içeriğin derleyici (compiler) ya da bağlayıcı (linker) için talimatlar taşıdığıdır. Asıl yürütme sırasında bu satırlar işletilmez.

"#pragma" nın anlamı "outBufferPong" değişkeninin hafıza alanında ayrılacak olan "DATA_SECTION" isimli bölüme konumlandırılacak olmasıdır (daha önce de yaptığımız veri hafıza alanı üzerindeki konumlandırmaya benzer şekilde).

Bu bölüm "outBuffer" olarak isimlendirilir. "outBufferPong" ve "outBufferPing" aynı bölüm adına sahiptirler. Bu nedenle bu iki değişkene de aynı DARAM numarası tahsis edilmiştir (Şekil 2.11'de gösterilmektedir).



Şekil 2.11: Veri alanının hafıza haritası

2.5.1.2. InıtSıneTable ( ) Fonksiyonu

Bu fonksiyon daha önce anlattığımız “vcg_1” programında aynı isimle yer almaktadır.

2.5.1.3. Fılter ( ) Fonksiyonu

“filter”() fonksiyonu filtreleme işini gerçekleştirmektedir. Bu fonksiyonda, ilk olarak, Ping ya da Pong olarak tampon bellek alanı seçilmektedir. SineTablo[] tampon bellekte yer alan veri ile aynen “vcg_1” programında olduğu çarpılmaktadır. McBSP den veriler, sağ kanal ve sol kanaldan sıra ile gönderilmektedir. İndeksler [j] ve [j+1] giriş çıkışın tampon belleklere aktarımını sağlarlar ve sırasıyla sol ve sağ kanallardaki bilgileri taşırlar. Sinüs tablosu için gerekli pointerlarda güncellenmektedir.

Bu fonksiyonu yeniden yazarak, diğer filtreler de kolayca geliştirilebilir.

2.5.1.4. FilterInterrupt ( ) Fonksiyonu (Donanım Kesme Sinyali ile Çağırılan Kesme Fonksiyonu)

Bu fonksiyonda, yukarıdaki filtre fonksiyonu çağırılmakta ve hesaplanan sinyali oynatmak için DMA geçişi başlatılmaktadır. Geçişi başlatmak için DMA registeri adres ve büyüklük bilgileri girilerek ayarlanmaktadır. Başlangıç fonksiyonu yürütüldükten sonra, McBSP ye yazmak için main() fonksiyonuna benzer şekilde işlem yapmak gerekmektedir. Tabii ki, alımı yapabilmek için Ping-Pong Bufferları anahtarlamak da gereklidir.

2.5.1.5. Maın( ) Fonksiyonu (Ana Fonksiyon)

Main() fonksiyonunda, tüm yapılması gereken çevresel aygıtların tanıtımı ve DMA iletiminin başlatılmasıdır.

"hDmaRcv" ve "hDmaXmt" konfigürasyon dosyaları için kullanılan tutamaçlardır. **DMA-1** ve **DMA-2** için gerekli komutlar bu tutamaçlar kullanılarak gönderilecektir.

"DMA_getEventId" fonksiyonu DMA' dan DSP' ye yapılacak kesme olayı için gerekli tanımlamaları döndürür. **Kesmelere** komutlar yollarken bu tanımlamalar kullanılır.

"DMA_FSET" fonksiyonu DMA registerlarını doğrudan ayarlamaktadır. Bu satırda, DMA nın tüm kesmeleri etkin kılınmaktadır(kanal 1'den kanal 5' e kadar). IRQ_enable fonksiyonu ile daha önce etkin kılınan kesmeler temizlendikten yani etkinliği kaldırıldıktan sonra alım işlemi için gerekli olan kesmeler tekrar etkinleştirilmektedir. Çünkü, eğer kesme hala varlığını korursa kesme etkin olur olmaz devreye sokulan kesme fonksiyonu en kısa sürede çağırılacaktır. "DMA_start" ile DMA başlatılır fakat aktarım başlatılmaz. McBSP' den aktarımın başlatılabilmesi için, bir veri okumak ve bir veri yazmak gereklidir. Bunun sonrasında, tüm kesmeler etkinleştirilir.

Tüm ayarlamalar tamamlandıktan sonra, main () fonksiyonu boşa çıkmaktadır.

2.6. Programın Yürütülmesi

Tüm dosyalar eklendikten sonra ve proje yapılandırıldıktan sonra, programın kullanımı aynen "vcg_1" programının kullanımı gibidir.

2.7. Sonuç

Bu bölümde, TMS320C5416DSK kartının kullanımı tanımlanmıştır. Kartı kullanmak için, çevre birimlerinin başlangıç durum ayarlarının yapılması gerekmektedir. Bu çevre birimi entegrelerinin kullanımları ile ilgili detaylı bilgiler kendi veri yapraklarında yer almaktadır. Bu veri yapraklarına mutlaka göz atınız.Ayrıca, örnek programlar yapılmıştır. DSP kartı için program yazmak ve onu kullanmak zor gibi görülmektedir. Fakat asıl önemli olanın; bu tarz program yazımına alışmak, tabii ki bunun da yolunun bu programlarla defalarca yapılacak tekrarlar olduğu unutulmamalıdır.

UYGULAMA FAALİYETİ

SES TEKNOLOJİLERİ	
İşlem basamakları	**Öneriler**
➢ Ses kayıt ve işleme teorisi tanımını ve kullanım alanlarını öğreniniz. **Tanım:** Ses kayıt ve işleme teknolojileri, ses dosyalarının kaydedilip bilgisayar ortamında saklanması ve bu dosyalar üzerinde çeşitli değişikliklerin yapılması amacıyla geliştirilen çözümleri içermektedir. **Kullanım Alanları:** • Bilgisayar ortamında fazla miktarda ses kaydı alınmasını gerektiren her türlü uygulama (örneğin IVR anonslarının kaydedilmesi) • Gürültülü ses kayıtlarının temizlenmesi • Farklı ses dosyası formatları arasında dönüşüm	➢ Ses kayıt ve işleme teorisinin aşağıdaki farklılıklarına dikkat ediniz. • Aynı seçeneklerle birden fazla dosya işleme özelliği • Elektro-glottograf (EGG) cihazı için stereo kayıt desteği

SES TEKNOLOJİLERİ	
İşlem basamakları	**Öneriler**
➢ Ses değiştirme teorisi tanımını ve kullanım alanlarını öğreniniz. **Tanım:** Ses değiştirmede ses ile oynanarak farklı özellikte sesler otomatik olarak üretilir. Ses değiştirme teknikleriyle yetişkin bir insanın sesinin çocuk sesine, kadın sesinin erkek sesine ya da erkek sesinin kadın sesine dönüştürülmesi mümkündür. Ayrıca çeşitli efektlerin eklenmesiyle bir sesten pek çok değişik ses üretilebilir (robot sesi, eko, fısıltılı konuşma, vs.). **Kullanım Alanları:** • Eğlence sektörü • Seslendirme (örnek animasyon karakterlerinin seslendirilmesi) • Güvenlik uygulamaları • Sağlık sektörü	➢ Ses değiştirme teorisinin aşağıdaki farklılıklarına dikkat ediniz. • Yüksek kalitede ve doğallıkta çıktı • Modem aracılığıyla telefon üzerinde kullanılabilme özelliği • Ses dosyası üzerinde ses değiştirme özelliği

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları doğru "D" veya yanlış "Y" olarak cevaplayınız.

1. TMS320C5416DSK kartında Line-in konnektörü ve line-out konnektorü A/D dönüştürücü entegresine bağlıdır.

2. DSK5416 kartının tanımlanması için " "DSK5416_init();"satırının, "dsk5416.h" başlık dosyası (header file) programımıza eklenmelidir.

3. PCM3002 ses entegresi giriş yapılan ses sinyallerini 48 kHz,16 bit, stereo kanal (tabii ki entegrenin başka ayarları da olmakla birlikte) ile örneklemektedir. Bu ses entegresini kullanabilmek için "dsk5416_pcm3002.h" başlık dosyası programa eklenmelidir.

4. Ses entegresinden okuma işlemi yapmak için gereken fonksiyon aşağıdaki gibidir:

DSK5416_PCM3002_read16(hCodec,&val);

5. DMA için iki yön vardır, birincisi "verinin McBSP' den hafızaya aktarımının yapıldığı yön (âlım)", diğeri "verinin hafızadan McBSP ye gönderildiği yön (gönderim)" şeklindedir.

6. "_filterInterrupt" ın alt çizgisi ("_") gereklidir, çünkü bu konfigürasyonlar bağlantı zamanında kullanılmaktadır. Tüm fonksiyon isimlerine derleyici tarafından derleme esnasında alt çizgi konulmaktadır.Bu nedenle, iletim katmanında, fonksiyon isimlerinin önüne alt çizgi koyma gereği doğmaktadır.

7. #pragma DATA_SECTION(outBufferPong,"outBuffer");

int outBufferPong[BUFFSIZE];

değişken tanımlamalarıdır. "#" işaretinin anlamı bu satırdaki içeriğin derleyici (compiler) ya da bağlayıcı (linker) için talimatlar taşıdığıdır. Asıl yürütme sırasında bu satırlar işletilmez.

8. "#pragma" nın anlamı "outBufferPong" değişkeninin hafıza alanında ayrılacak olan "DATA_SECTION" isimli bölüme konumlandırılacak olmasıdır.

9. "filter"() fonksiyonu filtreleme işini gerçekleştirmektedir.

10. "DMA_getEventId" fonksiyonu DMA' dan DSP' ye yapılacak kesme olayı için gerekli tanımlamaları döndürür. **Kesmelere** komutlar yollarken bu tanımlamalar kullanılır.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları faaliyete geri dönerek tekrar inceleyiniz.

MODÜL DEĞERLENDİRME

Modülde yaptığınız uygulamaları tekrar yapınız. Yaptığınız bu uygulamaları aşağıdaki tabloya göre değerlendiriniz.

DEĞERLENDİRME KRİTERLERİ	Evet	Hayır
Sinüs dalga üreticinin ne olduğunu anladınız mı?		
Sinüs dalga üretecinde, üretilen dalganın periyodunun neye bağlı olarak değiştiğini anladınız mı?		
CCS programını kurmayı başardınız mı?		
CCS programını kullanmayı öğrendiniz mi?		
Sinüs dalga üreteci programının kodlarını anladınız mı?		
Ses değişim prensibini öğrendiniz mi?		
Watch Window, Memory Window pencerelerini kullanmayı öğrendiniz mi?		
CCS programında bir sinyali grafiksel olarak görüntüleyebilir misiniz?		
FFT nin ne anlama geldiğini öğrendiniz mi?		
Dijital filtre ne demektir, öğrendiniz mi?		
Floating Point uygulamalarının DSP bodrumuz da niçin doğrudan yapılamadığını anladınız mı?		
DSP kartı üzerindeki ses sistemini anladınız mı?		
MCBSP' nin ne olduğunu ve nasıl tanımlandığını anladınız mı?		
PCM3002 entegresinin ne işe yaradığını ve nasıl tanıtıldığını anladınız mı?		
DSP kartı üzerindeki LED ve DIP Switch'lerin nasıl tanımlandığını ve kullanıldığını öğrendiniz mi?		
Matematiksel bir fonksiyon kullanarak bir sinüs dalgasının nasıl oluşturulduğunu anladınız mı?		
DSP deney seti ile bir ses uygulaması nasıl gerçekleştirilir öğrendiniz mi?		
Ses değiştirici programın temel mantığını anladınız mı?		
Tampon belleğin niçin gerektiğini anladınız mı?		
Ping-Pong tampon belleklerin ne olduğunu ve nasıl kullanıldığını anladınız mı?		
DMA kullanımının ne olduğunu anladınız mı?		

DEĞERLENDİRME

Hayır cevaplarınız var ise ilgili uygulama faaliyetini tekrar ediniz. Cevaplarınızın tümü evet ise bir sonraki modüle geçebilirsiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ 1'İN CEVAP ANAHTARI

1	**D**
2	**D**
3	**D**
4	**D**
5	**D**
6	**D**
7	**D**
8	**D**
9	**D**
10	**D**

ÖĞRENME FAALİYETİ 2'NİN CEVAP ANAHTARI

1	**D**
2	**D**
3	**D**
4	**D**
5	**D**
6	**D**
7	**D**
8	**D**
9	**D**
10	**D**

KAYNAKÇA

- NAZMAN, Mustafa, Melek TOTAN, **Sayısal İşaret İşleme,** ETOGM-JICA, İzmir, Ekim 2005.
- http://www.maxim-ic.com/appnotes.cfm/an_pk/2081
- http://www.discovercircuits.com/O/o-sine.htm
- http://www.sestek.com.tr/Products.asp?proId=3