ELEKTRİK MAKİNALARI DENETİM SİSTEMLERİNDE TMS320F2812 DSP KULLANIMI

Selami KESLER

Pamukkale Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Eğitimi Böl., DENİZLİ
skesler@pau.edu.tr

ÖZET

Mikroişlemcili denetleyicilerin performansı ve kapasitelerinin gelişime bağlı olarak elektrik makinalarının hız, konum ve moment gibi dinamiklerin denetiminde de daha karmaşık yöntemler başarıyla uygulanabilmektedir. Özellikle vektör tabanlı denetim sistemler yüksek hızlı ve doğruluklu işlem başarımına ihtiyaç duymaktadır. Bu nedenle günümüzde sayısal işaret işleyiciler(DSP's) elektrik makinalarının vektör tabanlı denetim sistemlerinde yaygın olarak kullanılmaya başlamıştır.

Bu çalışmada elektrik makinalarının denetimi için son yıllarda tercih edilen eZdsp TMS320F2812 DSK'in programlama ilkeleri ve açık kaynak kodları verilmiştir. Temel olarak; DSP kurulumu, sistem başlangıç ayarları, portların yönlendirilmesi, kayıtçıların kullanılması, hız, akım vb. analog bilgilerin okunması, zaman kesmelerinin ayarlanması, sayısal hız kodlayıcı bilgilerin okunması ve normalizasyonu, bağımsız PWM kanallarının etkinleştirilmesi ve bağımsız PWM çiftlerinin üretilmesinde ölü zamanların ayarlanması gibi belli başlı işlemlerin nasıl gerçekleştirileceği açık kaynak kodları ile incelenmiştir. Örnek bir model için sonuçlar verilmiştir.

***Anahtar Kelimeler**: Sayısal İşaret İşleyici (DSP), eZdsp TMS320F2812 DSK, Denetim Sistemleri, Elektrik Makinaları*

USING TMS320F2812 DSP IN CONTROL SYSTEMS FOR ELECTRICAL MACHINES

ABSTRACT

The complicated methods used in controlling electrical machines' dynamics such as speed, position and induced torque can be implemented successfully in conjunction with developing performance and capacity of controllers using microprocessor. Specially, the vector based control systems entail performance of process with high accuracy and speedy. Therefore, digital signal processors are commonly used in vector based control systems for electrical machines.

In this paper, the open source codes and programming principal of eZdsp TMS320F2812 DSK, which has held in high regard recently for controlling electrical machines, are given in order. Mainly, how to be realized some fundamentals such as installing DSP, settings of system initialization, directing of ports, using of registers, converting of analog signals to digital, settings of time interrupts, reading digital information from shaft encoder and its normalization, enabling of independent PWM channels and setting of dead band are described by open source codes. The results for exemplary model are included.

Keywords*: Digital Signal Processor (DSP), eZdsp TMS320F2812 DSK, Control Systems, Electrical Machines*

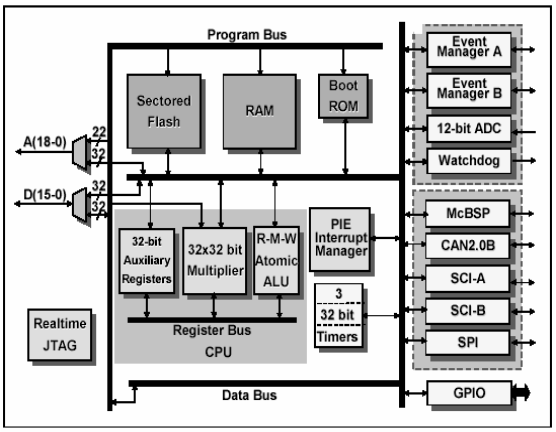
1.GİRİŞ

Günümüzde DSP'ler video, müzik, haberleşme ve ölçme tekniğinde yaygın olarak kullanılmaktadır. Elektrik makinalarının hız, moment ya da konum denetiminde yüksek hızlı ve doğruluklu işlem başarımının elde edilmesinde ve daha karmaşık denetim yöntemlerinin uygulanmasında sayısal işaret işleyiciler önemli yer tutmaktadır. Güç elektroniği sürücü düzeneklerinde kullanılan yarı iletken anahtarların daha hızlı ve daha karmaşık denetimleri, sürekli gelişen mikroişlemci teknolojisiyle daha kolay yapılabilmektedir.

Sayısal işaret işleyiciler DSP olarak bilinmektedirler. Her türlü konvertör ve motor denetimi için üretilmiş hızlı ve yetenekli DSP'lerden birisi de TMS320F2812'dir. Spectrum Digital tarafından deneysel kullanımlar için geliştirilmiş kartı olan eZdsp TMS320F2812 DSK, bilgisayarların paralel portu üzerinden Code Composer Studio (CCS) arayüzü ile programlanabilmektedir[1-3]. Bir kontrol sisteminin DSP ille gerçekleştirilmesinde, DSP'nin kullanımı ve programlama ilkelerinin öğrenilmesi oldukça uzun zaman alabilmektedir. Bu amaçla bu çalışma bir rehber niteliğinde sunulmuştur.

2. TMS320F2812 DSP YAPISI

TMS320F2812 DSP, sabit noktalı işlem yapan 32-bit 150Mhz bir işlemci olup, 16-kanal 6.67ns çözünürlüklü ölü zaman ayarlı programlanabilir PWM çıkışı, 16 kanal 12-bit 80ns dönüşüm zamanlı A/D çevirici, 4 adet sayısal yakalama girişi ve 4 adet kare dalga kodlayıcı girişi, 16-bit 7 port programlanabilir sayısal giriş-çıkış, 18K word RAM ve 128K word Flash EEROM ve C/C++ programlama desteğine sahiptir. İşlemci sabit noktalı aritmetik işlem yapmasına rağmen IQmath kütüphane desteği ile kayan noktalı aritmetik işlem kolaylığında ve ona yakın doğrulukta işlem yapabilmektedir. İşlemciye ilişkin ilkesel bir model Şekil 1'de verilmiştir[1-3].



Şekil 1. TMS320F2812 DSP Blok Çizgesi

3. CCS PROGRAMLAMA ARAYÜZÜ

Texas Instruments firmasının ürettiği bu sayısal işaret işlemcisi, Code Composer Studio (CCS) ile birlikte gelmektedir. İşlemcinin programlanması C++ ve ASM ile yapılabildiği gibi MATLAB Simulink, VISSIM gibi özel paket programlar da kullanılabilmektedir. Ancak oluşturulan program kodları yine CCS aracılığıyla işlemciye yüklenmektedir. Bu işlem için CCS kullanılan program tarafından çağrılmaktadır. Paket içeriği olarak gelen CD ile sistem kurulumu standart PC donanım kurulumu gibidir. CCS 'in yeni sürümlerinde kart bağlantısı çalışma anında kesilebilmekte ve tekrar bağlanabilmektedir. Ancak önceki sürümlerinde DSP program yüklemesi ve koşturulması bitene kadar kart ile bağlantının kesilmesine izin vermemektedir.

CCS arayüzünde standart C/C++ proje oluşturma işlemleriyle, CSS örnek dosyalarından her hangi birinde ana program çatısı oluşturulabilir.

CCS'de açılan her yeni proje için kullanılan işlemciye ilişkin "*f2812.gel*" dosyası eklenmelidir. F2812 işlemcisinin bütün birimlerinin adresleri ve adları C++ desteği ile neyse yönelimli olarak tanımlanmıştır. Bu nedenle standart olarak işlemci birimlerine ilişkin başlık dosyaları, kütüphane dosyaları ve kullanıcı tarafından değiştirilebilen kaynak dosyaları açılan projeye eklenmelidir. Bu dosyalar, kullanıcı tarafından oluşturulacak yeni denetim yazılımları gibi kaynak dosyaları ile bağlanarak çıkış dosyası üretilir ve işlemciye yüklenir.

İşlemcinin kullanılan birimlerine göre giriş/çıkış kapıları, ADC girişleri, PWM çıkışları, sayısal veri çıkışları, işlemci ve kullanılan birimlerin uygun çalışma hızları mevcut kaynak dosyalarında değiştirilir. Özellikle kesme (interrupt) vektörleri program zamanlaması açısından doğru ayarlanmalıdır. Örneğin 2.5kHz'lik bir PWM işareti üretimi için 400µs'lik bir zaman kesme vektör yazılımı yapılmalıdır. Denetimi yürütecek program ya da program grubu bu süre içinde bir çevrimini tamamlamalıdır. Her giriş/çıkış kapısı; giriş, çıkış, işaret yakalama ya da PWM için bağımsız olarak ayarlanabilir. Ancak işlemci bilgi sayfalarında verilen bacak bağlantı yapılarına dikkat edilmelidir. İşlemcinin temel birim kayıtçıları tam korumalı olduğundan gerekli ayarlar yapılırken koruma kaldırılır ve ayarlama sonunda koruma kodları yeniden etkin yapılır. Bundan sonraki bölümlerde DSP'nin programlanmasında takip edilecek işlem sırası aynı kod sırasıyla verilmiştir. Kodlar verildiği sırada yazılmalıdır.

3.1. DSP Sistem Başlangıç Ayarları

Kullandığımız işlemcinin bütün başlık dosyalarının bulunduğu ana başlık dosyası olarak aşağıdaki dosyalar programa eklenir.

```
DSP281x_Device.h"
DSP281x_Examples.h"
```

DSP yongasında bulunan kendi çevresel birimlerinin (PLL, WatchDog Timer vb.) çalışma hızları ve etkin edilip edilmeme durumları "*DSP281x_SysCtrl.c*" kaynak

dosyasında ayarlanmıştır. Programa bu kaynak dosya;

```
InıtSysCtrl( );
```

koduyla dahil edilir. Kaynak kod dosyalarının bulunduğu (source files) bölüme de "*DSP281x_SysCtrl.c"* dosyası eklenmiştir. Bu dosya da sistem saat hızı ayarı HSPCLK hızına oranla PLL çıkışı olarak 150MHz çalışma durumu için aşağıdaki kod, koruma kaldırılarak, eklenerek kayıtçılar tekrar korumaya alınır. Kullanılan kayıtçı nesne adları:

```
EALLOW;
SysCtrlRegs.HISPCP.all = 0x0000;
EDIS;
```

biçimindedir.

Genel amaçlı giriş/çıkış kayıtçılarının ayarları kaynak dosyasında olduğundan, sayısal giriş ve çıkış olarak kullanılacak kapıların kayıtçıları yine korumalı olarak bu dosyada ayarlanır ve dosya ana program içine;

```
InıtGpio( );
```

koduyla dahil edilir. "*DSP281x_Gpio.c*" kaynak dosyası da projeye eklenir. GPIO PortB'nin ilk sekiz bitinin çıkış, son sekiz bitinin giriş olarak ayarlandığı kaynak kodları aşağıdaki gibidir. Genel tanımlama kapı(port) yönlendirme kayıtçısında (GPBDIR) yapılmaktadır.

```
EALLOW;
GpioMuxRegs.GPBMUX.all = 0x0;
GpioMuxRegs.GPBDIR.all = 0x00FF;
GpioMuxRegs.GPBQUAL.all = 0x0;
EDIS;
```

Çevresel birimlerin kesme vektör tablosu işlenirken işlemci temel kesmelerinin yetkisiz kılınması gerekir. Bunun için CPU kesmeleri yetkisiz kılınır ve kesme bayrakları (interrupt flag) temizlenir. Gerekli kod sırası aşağıdaki gibidir:

```
DINT;
InıtPieCtrl( );
IER=0x0000;
IFR=0x0000;
InitPieVectTable();
```

Çevresel birimlerin kesme vektörleri "*DSP281x_PieCtrl.c"* kaynak dosyasında ayarlanır. Bu dosya kaynak dosyalar bölümüne dahil edilir. Bu dosyada eğer ADC örnekleme zamanına göre bir kesme vektörü kullanılacaksa;

```
InitAdc();
EALLOW;
PieVectTable.ADCINT = &adc_isr;
EDIS;
PieCtrlRegs.PIEIER1.bit.INTx6 = 1;
IER |= M_INT1;
EINT;
ERTM;
```

kaynak kodları girilmelidir. Ancak ADC'nin kesme vektörleri CPU zamanlayıcılarından bağımsız değildir. Bu yüzden CPU'nun gerçek zamanlı kesmeleri de yetkilendirilmiştir. Eğer bir bağımsız zaman kesmesi kullanılacak ve ADC'nin buna uyması istenirse, zamanlayıcılardan birisine ilişkin kesme vektörü ayarlanmalıdır. Ayrıca zamanlayıcının sayaç durumuna göre kesmenin zamanı da belirlenebilmektedir. Program döngüsü sonunda da kullanılan kesme vektörüne uygun bayraklar ayarlanır. TIMER1'in setlenen sayıcı değerine göre, sayıcı sıfırlama durumuna göre, sayıcı sonlanma durumuna göre ya da sayıcı karşılaştırma değerine (CMPR VALUE) kesme üretebilen bir kesme vektörüne sahiptir. TIMER1 periyot değerine göre kesme üreten TIMER1 zamanlayıcısı kesme vektörü kaynak kodları aşağıdaki gibi oluşturulmuştur.

```
EALLOW;
PieVectTable.T1PINT = &t1pint_isr;
EDIS;
PieCtrlRegs.PIEIER2.bit.INTx4 = 1;
        IER =0x0002;
        EINT;
        ERTM;
```

Genel kesme hizmet yordamları (Interrupt Service Routine) "*DSP281x_DefaultIsr.c"* kaynak dosyasında olup, bu dosya kaynak dosyaları bölümüne dahil edilmelidir. Kaynak kodlarını yazdığımız kesme vektörleri ayarlanmadan önce;

```
InıtPieVectTable();
```

kaynak kodu ana programa dahil edilmelidir. Ayrıca çevresel birimler başlangıç değerlerine ayarlanması gerektiğinden,

```
DSP281x_InitPeripherals.c
```

dosyası kaynak dosyaları bölümüne dahil edilip ana programa;

```
InıtPeripherals();
```

kaynak kodu eklenmelidir. Denetim sistemleri uygulamasında en çok kullanılan birimlerden birisi de ADC dir. Bu yüzden gerekli ayarlamalar sonra yapılmak üzere başlangıç ayar kodları bölümünde;

```
InıtAdc();
```

kodu yazılarak, kaynak dosyaları bölümüne "*DSP281x_Adc.c*" dosyası da eklenir. F2812 işlemcisine ilişkin genel değişken ve nesne tanımlarının yapıldığı;

```
DSP281x_GlobalVariableDefs.c
DSP281x_CodeStartBranch.asm
```

dosyaları yine kaynak dosyaları bölümüne eklenmelidir.

Denetim amaçlı oluşturacağımız ana programa başlamadan önce kütüphane ve başlık dosyalarının kaynak dosyaların derlenmesinden sonra çıkış dosyasının işlemci belleğinde yerleşeceği haritalama, bağlayıcı ve yükleyici dosyalar eklenmelidir. Bunlar ;

```
DSP281x_Header_nonBIOS.cmd
F2812_EzDSP_RAM_lnk.cmd
```

dosyalarıdır. Bütün bu sözü edilen dosyalar CCS kurulumunda mevcuttur. Sadece açılan projeye eklenmesi ve ayarlanması gerekir. Örnek projelerden yola çıkarak, gerekli ekleme ve düzenleme işlemi daha kolaydır. Ancak F2812 işlemcisine ilişkin çevresel birimlerin başlık dosyalarının üretici firma web destek sitesinden indirilip kurulması gerekmektedir.

3.2. DSP Kayıtçılarının Kullanımı

F2812 işlemcisinin bütün çevresel birimleri C++ desteğinde nesne olarak bit düzeyine kadar tanımlıdır. Programlamaya başlamadan önce başlık dosyaları tek tek incelenip kayıtçı nesne adları bellenmelidir. Program yazımı sırasında CCS de yardımcı olmaktadır. Açılan TAB menülerden kayıtçının ilgili bitleri ya da tamamı seçilebilir. Yapılacak işleme göre işlemci bilgi sayfalarından uygun kayıtçılar belirlenip sırasıyla setlenmelidir. Aşağıda genel amaçlı giriş/çıkış seçisi kayıtçılarından GPIO_ PORT_A'nın bit düzeyinde PWM çıkışı yetkilendirmesi için bir örnek verilmiştir.

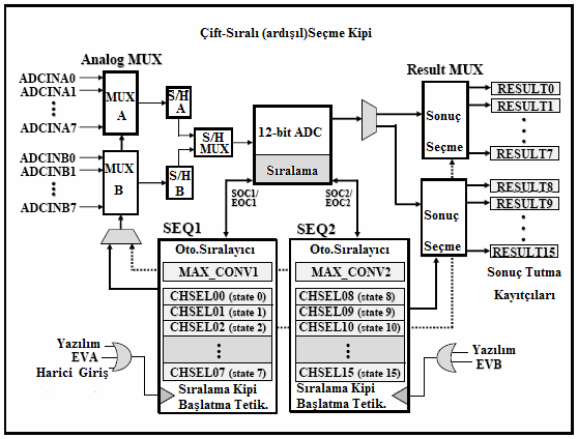
```
GpioMuxRegs.GPAMUX.bit.PWM1_GPIOA0=1;
```

Burada;

"GpioMuxRegs: Genel Amaçlı Giriş/Çıkış Seçici Kayıtçısı, ***GPAMUX:*** GpioMuxRegs kayıtçısı altında bulunan A Kapısı Seçicisi, ***bit:*** A Kapısı Seçme Kayıtçısında "bit" düzeyinde işlem yapılacağı, ***PWM1_GPIOA0=1:*** A Kapısının A0 biti seçileceği ve PWM çıkışı olduğu" yazılan kodla belirtilmiştir.

3.3. Analog Sayısal Dönüştürücü Ayarları

F2812 işlemcisi 12-bit çözünürlüklü 16 adet (2x8) 0-3V DC analog girişli bir ADC'ye sahiptir. Programın işlemcide koşturulması sırasında okunacak analog girişlerin sayısı ve analog işaretin hangi bacağa bağlandığı, dönüştürme sırası ve biçimi ayarlanmalıdır. Örnekleme zamanı ve dönüştürülen işaretin sayısal değerinin ana program tarafından hangi sıklıkla alınacağı kesme vektörleriyle belirlenir. ADC'nin örnekleme hızı ise sistem saat hızına göre bağımsız olarak ayarlanabilmektedir (Şekil 2 ve Şekil 3). Bu çalaışmada ADC **25Mhz** hızında ve dönüştürme biçimi **sıralı kip** olarak seçildi. Böylece okunacak akım ya da diğer analog girdiler öncelik sırasına göre dönüştürülmüştür.



Şekil 2. ADC birimimin sıralı dönüşüm kipi

Okunacak analog kanal sayısı (ADCMAXCONV), dönüşüm yapılınca hangi kanalın hangi sonuç kayıtçısında tutulacağı (ADCSELSEQx ve ADCRESULTx) , hangi olay yöneticisinin dönüşümü yeniden başlatacağı (EVA_SOC_SEQx) ve kesme vektörünün yeniden yetkilendirilmesi (INT_ENA_SEQx) her TIMER1 periyoduna bağlı gerçekleşen kesmeler için ayarlanır. İlgili kayıtçılar (ACTRLx,

ADCMAXCONVxx, ADCCHSELSEQx) **AdcRegs** içinde bulunur. Üç analog giriş için kullanılan ayarlar örnek olarak aşağıda verilmiştir.

```
AdcRegs.ADCMAXCONV.all = 0x0002;
    // üç adet kanal dönüştürülecek
AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x7;
    // ADCINA7 bilgisi ADCRESULT0'a
AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x6;
    // ADCINA6 bilgisi ADCRESULT1'e
AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x0;
    //ADCINA0 bilgisi ADCRESULT2'e
AdcRegs.ADCTRL2.bit.EVA_SOC_SEQ1 = 1;
    // Sıralı mod EVA olay yön.yetkili
AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 1;
    // Kesmeler dönüşüm sonunda yetkili
AdcRegs.ADCTRL3.bit.ADCCLKPS=3;
    // ADC örnekleme hızı 150/6=25Mhz
AdcRegs.ADCTRL1.bit.CPS=0;
    // Ardışıl dönüşüm hızı (pipeline)
AdcRegs.ADCTRL1.bit.ACQ_PS=7;
    // ADC örnekleme penceresi
```

Sayısal değere dönüştürülen analog işaretler ADCRESULTx kayıtçılarında tutulur. Eğer dörtten fazla kanal dönüşümü yapılacaksa, SEQ1 ve SEQ2 kipleri ADCTRL1 ve ADCTRL2 kayıtçılarında setlenir. Bunun için Tablo 1'den faydalanılabilir.

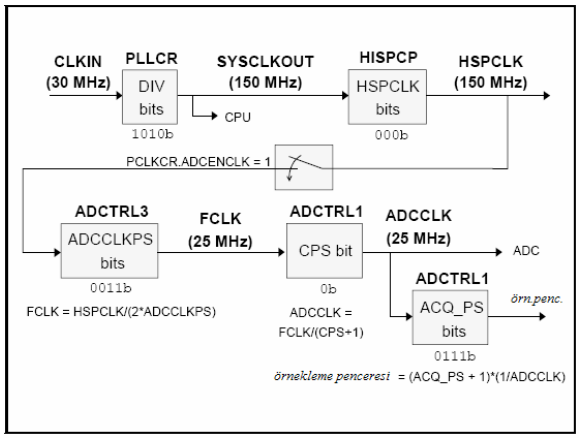
Tablo 1. ADC kanal seçim kayıtçı değerleri

	Bits 15-12	Bits 11-8	Bits 7-4	Bits 3-0	
0x007103	CONV03	CONV02	CONV01	CONV00	ADCCHSELSEQ1
0x007104	CONV07	CONV06	CONV05	CONV04	ADCCHSELSEQ2
0x007105	CONV11	CONV10	CONV09	CONV08	ADCCHSELSEQ3
0x007106	CONV15	CONV14	CONV13	CONV12	ADCCHSELSEQ4

Bu kayıtçılar 16-bit olmasına rağmen ADC 12-bit olduğundan tutulan sayısal bilgi normal analog değerine dönüştürülmeden önce 4-bit sağa kaydırılarak kullanılır. Örnek kod aşağıdaki gibidir.

```
RefVoltDigital=(AdcRegs.ADCRESULT2 >>4);
RefVoltAnalog= RefVoltDigital *3.0/4095;
```

F2812 işlemcisi ADC'si 0.732mV/bit duyarlılıktadır. Çok salınım yapan 12-bit yerine 11-bit kullanılabilir. Bu durumda beş bit sağa kaydırılabilir. ADC çalışma zamanı ayarlaması Şekil 3'de verilmiştir.



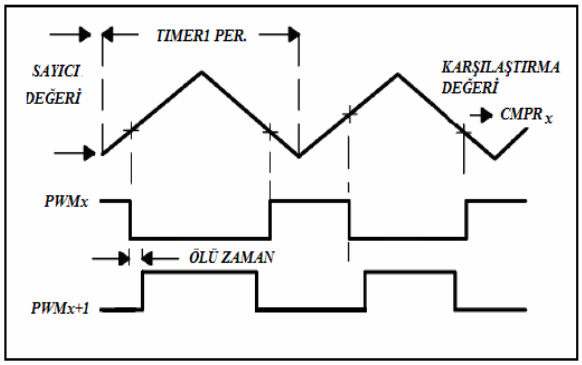
Şekil 3. ADC çalışma hızının ayarlanması

3.4. PWM işaretlerinin Üretimi

İşlemcinin daha önce sözü edilen zamanlayıcıları ve karşılaştırma mantık birimi kullanılarak üç-çift bağımsız tümleyenli PWM işaretleri üretilir. Aynı hat üzerindeki anahtarların kısa devre olmasını önlemek amacıyla ölü zaman mantık birimi de ayarlanır. Çalışmamızda TIMER1 ve EVA olay yöneticisi kullanılarak farklı frekanslarda PWM işareti üretilmiştir. Sayıcı yukarı-aşağı (up-down mode) çalışma kipine ayarlanarak, örnek olarak,

$$f_{pwm} = \frac{f_{pcu}}{T1_{per}.TPS_{T1}.HISCP} = \frac{150Mhz}{(2x30000).1.1} = 2500\,Hz$$

anahtarlama frekansı elde edilir.



Şekil 4. Zamanlayıcılar ve simetrik PWM çiftlerinin üretimi

TIMER1_PERIOD değeri 30000 setlendiğinde, toplam zaman sayacı önce yukarı sonra aşağı 60000 sayacaktır. Ayrıca her zamanlayıcının

(TIMERx) bağımsız olarak üç ayrı karşılaştırma değeri girilebilmektedir. Bunlar CMPRx kayıtçılarında tutulur. İlgili kayıtçılar EvaRegs ve EvbRegs içinde bulunur. Simetrik bir PWM işaretinin zamanlayıcılarla üretim ilkesi Şekil 4'de gösterilmiştir. EVA olay yöneticisinde sayıcıyı sıfırdan başlatan, TIMER1 periyodunu setleyen ve periyot sonunda ADC için kesme üretecek kodlar aşağıdaki gibidir:

```
EvaRegs.T1PR =30000;
        //2.5kHz için periyod değeri
EvaRegs.GPTCONA.bit.T1TOADC = 2;
        //Periyot sonunda kesme yetkisi
EvaRegs.T1CNT=0x0000;
        //Sayıcı başlangıç değeri
```

Her periyot sonunda kesme bayrakları ve vektörleri yeniden ayarlanır.

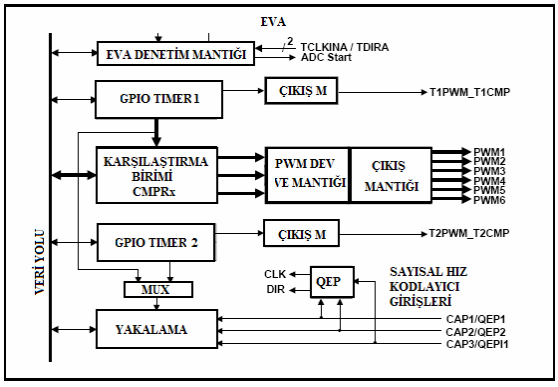
```
EvaRegs.EVAIMRA.bit.T1PINT=0;
EvaRegs.EVAIFRA.bit.T1PINT=1;
EvaRegs.EVAIMRA.bit.T1PINT=1;
```

EVA olay yöneticisinin TIMER1 için yukarı-aşağı sayma kipi ve karşılaştırma mantık biriminin etkin edilmesi için aşağıdaki setleme yapılır.

```
EvaRegs.T1CON.all = 0x0842;
```

Her karşılaştırma işleminde PWM_X *ve* PWM_{x+1} işaret çiftinin, sayıcı değerinin karşılaştırma değerine ulaştığında PWM_X in yükselmesi ve PWM_{x+1}'in düşmesi için aşağıdaki setleme yapılmalıdır.

```
EvaRegs.ACTRA.all=0x0666
```



Şekil 5. EVA Olay yöneticisine bağlı işlemci birimleri

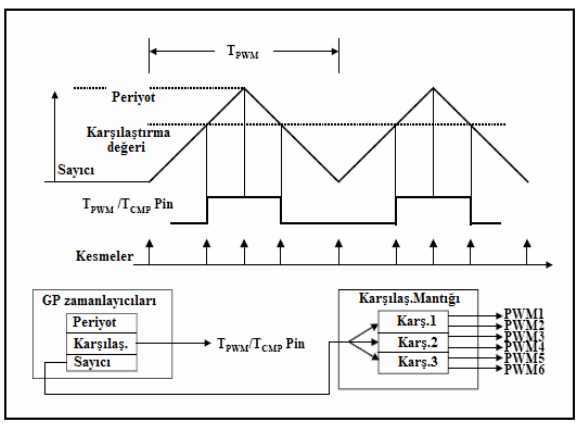
Sabit darbeleme oranlı PWM üretimi için karşılaştırma değerleri (CMPRx) sabit girilir. SVPWM üretiminde anahtarlama süreleri her bir IGBT anahtar grubu için farklı ve her adımda değiştiği için hesaplanan anahtarlama süreleri CMPR değeri olarak alınmıştır. Örnek kodlar aşağıda verilmiştir. S1, S2, S3 hesaplanan anahtarlama süreleridir. Bu süreler zamanlayıcıların periyoduna oranlıdır. Saniye cinsinden değer değildir.

```
EvaRegs.CMPR1= EvaRegs.T1PR -S1; //PWM1-2
EvaRegs.CMPR2= EvaRegs.T1PR -S3; //PWM3-4
EvaRegs.CMPR3= EvaRegs.T1PR -S5; //PWM5-6
```

F2812 sayısal işaret işlemcisi daha önce de sözü edildiği gibi genel amaçlı giriş/çıkış kayıtçı ve kapılarına sahiptir. Bu nedenle üretilen PWM işaretlerinin görülebilmesi için bit düzeyinde PWM çıkışları ayarlanmalıdır. EVA kayıtçı ve mantıksal birimleri Şekil 5'de gösterilmiş olup, PWM çıkışlarını ayarlayan kod yazılımı aşağıdaki gibidir.

```
EALLOW;
GpioMuxRegs.GPAMUX.bit.PWM1_GPIOA0=1;
GpioMuxRegs.GPAMUX.bit.PWM2_GPIOA1=1;
GpioMuxRegs.GPAMUX.bit.PWM3_GPIOA2=1;
GpioMuxRegs.GPAMUX.bit.PWM4_GPIOA3=1;
GpioMuxRegs.GPAMUX.bit.PWM5_GPIOA4=1;
GpioMuxRegs.GPAMUX.bit.PWM6_GPIOA5=1;
EDIS;
```

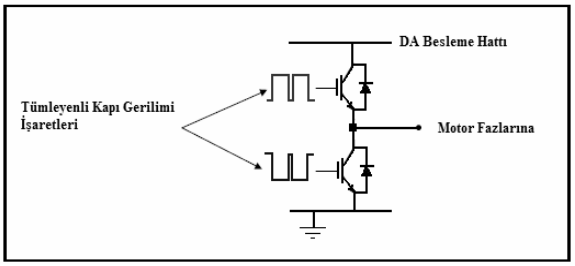
Ayrıca GP zamanlayıcılarının da bağımsız olarak periyot ve karşılaştırma değerleri ayarlanarak, iki-çift tümleyenli PWM işareti üretilebilmektedir. Bunlar T1PWM_T1CMP ve T2PWM_T2CMP'dir. Sabit darbe genişlikli simetrik bir PWM işaretinin üretim süreci blok çizgesi Şekil 6'da verilmiştir. GPIO_A kapısından alınan PWM işaretleri güç ve sürücü devresinden yalıtılarak kullanılmalıdır.



Şekil 6. PWM işareti üretim süreci blok çizgesi

3.5. PWM İşaretlerinin Ölü Zaman Ayarı

IGBT eviricilerde aynı hatta bağlı anahtarlar, biri diğerinden tümleyenli üretilmiş PWM işareti ile sürülürse sürücü işaretin düşen ve yükselen kenarlarında anahtar hızları yavaş kaldığı için üstteki anahtar henüz tıkamaya gitmeden alttaki anahtar tetiklenmiş ve iletime sokulmaya zorlanmıştır (Şekil 7). Bu durumda DA besleme hattı bu anahtarı grubu tarafından kısa devre olmakla birlikte IGBT anahtarlar da kısa devre akımından etkilenerek zarar görmektedir.

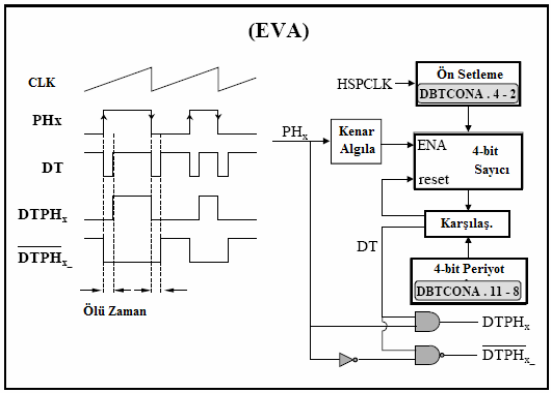


Şekil 7. Aynı Besleme hattına bağlı anahtarların sürülmesi

Bu durum basit elektronik devreler ya da entegre devrelerle çözülebilse de haricen kullanılan devre elemanlarının hızı sayısal işaret işlemciye yetişemediği ve devre elemanlarının toleransları farklı olduğu için yine sorun çıkmaktadır. En doğru çözüm tümleyenli PWM işaretlerinin sayısal işaret işlemcisinde ölü zamanlı olarak üretilmesidir. F2812 işlemcisinde bunun için özel kayıtçılar ve mantıksal birimler vardır. Bu mantıksal birimler Şekil 8'de gösterilmiştir. İşlemcide ölü zaman üretimi için öncelikle ölü zaman mantık birimi etkinleştirilmelidir. Daha sonra güç devresindeki anahtar elemanların iletime geçme ve tıkamaya gitme süreleri göz önüne alınarak uygun bit kombinasyonu DBTCONA ve COMCONA kayıtçılarında ayarlanır. PWM tümleyen çiftleri arasında 3µs'lik bir ölü zaman ayar kodları aşağıdaki gibidir.

```
EvaRegs.DBTCONA.bit.EDBT1=1;//PWM1-2
EvaRegs.DBTCONA.bit.EDBT2=1;// PWM3-4
EvaRegs.DBTCONA.bit.EDBT3=1;// PWM5-6
EvaRegs.COMCONA.all=0xA600;//SVPWM
0xB600
EvaRegs.DBTCONA.bit.DBTPS=6;//Ölü zaman
EvaRegs.DBTCONA.bit.DBT=12;//C->3µs
```

Burada, DBTS değer kombinasyonu 6,4,3,2,1,0 ve DBT değer kombinasyonu F,E,D,C,A,9,…0'dır. Şekil 8'de verilen EVA kayıtçılarının 16-bitlik açılımları bilgi sayfalarından bakılarak ölü zaman kombinasyonu değiştirilebilir.

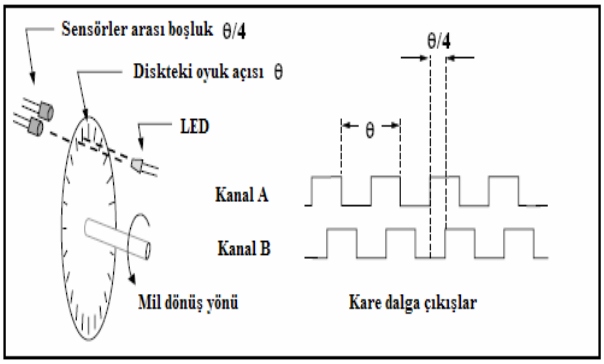


Şekil 8. EVA olay yöneticine bağlı ölü zaman birimleri

İşlemci hızı 75Mhz seçilirse yukarıdaki ayarlamalar 6µs'lik ölü zaman üretecektir. Bunun nedeni, DBTCONA zamanlayıcılarının işlemci saatini kullanmasıdır.

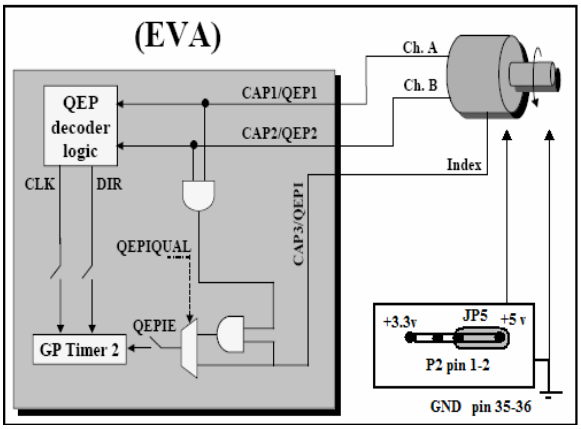
3.6. Hız Kodlayıcıdan Bilgi Okunması

Elektrik makinalarının farklı çalışma durumları göz önüne alınarak konum bilgisinin analog olarak alınmasında gürültü ve ADC etkilerini ve ortadan kaldırmak için sayısal konum kodlayıcı kullanılması daha doğru bir denetim yapılmasına yardımcı olur. (Örnek bir model: ENB-1024-3-1). Bir sayısal hız/konum kodlayıcının çalışma blok diyagramı Şekil 9'da gösterilmiştir.



Şekil 9. Sayısal konum kodlayıcının çalışma ilkesi

Hız/konum kodlayıcının beslemesi işlemciden alınarak, kodlayıcı çıkış işaretlerinin GND seviyesi F2812 DSP ile eşitlenmelidir. Kullanılan sayısal hız/konum kodlayıcı iki fazlı işaret üretmektedir. Fazlardan biri QEP1, diğeri QEP2 adlı işlemci bacağına bağlanır. Bu bağlantılar GPIOA kayıtçı ve kapısında olduğundan GPIOA8_QEP1 ve GPIOA9_QEP2 öncelikle giriş olarak ayarlanmalıdır. A ve B kanal işaretleri arasındaki konum farkının negatif olması, sayıcının büyük bir değerden başlaması anlamına gelir. Buda motor milinin ters dönmesi demektir. Bu durumda dönüş yönü önemli değilse sayıcı üst değerine göre (65535) okunan sayıcı değeri doğrulanmak üzere, bu üst değerden çıkartılır.



Şekil 10. Hız kodlayıcının F2812 işlemcisine bağlantısı

QEP1 ve QEP2'den gelen kare dalga işaretlerinin hem yükselen hem de düşen kenarlarında sayma işlemi gerçekleşir. Böyle devir başına 1024 kare dalga üreten bir kodlayıcı için sayıcı 4096 sayısına ulaşır. PWM anahtarlama periyodu 2.5kHz ise, sayısal hız bilgisi de 400µs aralıkla okunacaktır. Bu durumda en fazla devir sayısı 3000 dev/d göz önüne alınarsa 400µs'de sayıcı değeri 81,92 olmalıdır. Bu tam sayı olmadığından, hız bilgisinin her anahtarlama periyodunda okunması hataya yol açacaktır. Bu nedenle sayıcı 25 döngüde okunarak en yakın tam sayısı 2048 elde edilebilir. Bu da tam duyarlıkla hız bilgisinin normal değere çekilmesini sağlar. 5kHz anahtarlama frekansı için bu sayı 1024 olarak hesaplanır. Bu durumda gerçek hız bilgisi normal değeri çarpanı (3000/1024) olur. Sayısal hız kodlayıcı Şekil 10'daki gibi bağlanarak, GPIO_TIMER2 ve sayıcısı üzerinden hız bilgisi okunur. Bunun için aşağıdaki kodlama bit düzeyinde yapılmalıdır:

1.adım: QEP1 ve QEP2 giriş için yetkilendirilir.

```
EALLOW;
GpioMuxRegs.GPAMUX.bit.CAP1Q1_GPIOA8=1
GpioMuxRegs.GPAMUX.bit.CAP2Q2_GPIOA9=1
EDIS;
```

2.adım: Zamanlayıcı periyodu ayarlanır ve sayıcı sıfırlanır

```
EvaRegs.T2PR=0xFFFF;
EvaRegs.T2CNT=0x0000;
```

3.adım: Sayıcının iki yönlü sayışına izin veren, TIMER1'den bağımsız çalışmayı sağlayan, arıza durumunda programa durma işareti üreten, işlemci frekansından bağımsız çalışmayı sağlayan TIMERx karşılaştırıcılarını devre dışı bırakan ve TIMER2 periyodunu esas alan EvaRegs içindeki T2CON kayıtçısının FREE=0, SOFT=0, DMODE=3, T2SWT1=0, TPS=0, TCLKS10=3, TCLD10=0, TECMPR=0,SET1PR=0, TENABLE=1 bit düzeyinde kodlarına karşılık gelen:

```
EvaRegs.T2CON.all=0x1870;
```

kodu ana programa eklenir. Index işaretine ihtiyaç duyulmadığından yakalama kayıtçısını devre dışı bırakmak için CAPCONA kayıtçısının uygun değerine ayarlanması için aşağıdaki setleme yapılır:

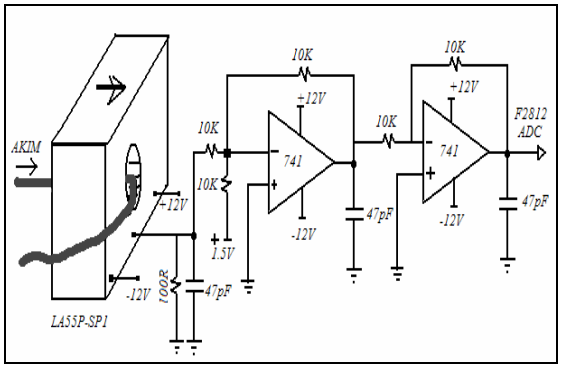
```
EvaRegs.CAPCONA.all=0xE000;
```

Her zaman kesmesi (time interrupt) sonunda okunan EvaRegs.T2CNT sayıcı değeri hız kodlayıcıdan gelen darbe sayısı olduğundan bir bölünebilir tamsayı değerine ulaşıncaya kadar döngü yapılır ve normalize edildiğinde sayıcı değeri ve sayıcı toplam darbe sayısını tutan değişken sıfırlanır.

3.7. Analog İşaretlerin Örneklenmesi

F2812 DSP 25 MHz hızında çalışan ve 0-3V doğru gerilim değerini okuyup 12-bit sayısal dönüşüm yapan bir ADC birimine sahiptir. Bu sayısal değerler 16-bit'lik kayıtçılarda tutulur. Akım, gerilim ve analog hız bilgileri okunurken kayıtçı verileri 4-bit sağa kaydırılarak kullanılır. Ancak alternatif genlikli işaretlerin bu ADC'den okunması için örneklenecek işaretler en fazla

1.5V genlikli olacak şekilde ayarlanmalı ve bir analog işaret bindirme devresinden geçirilmelidir. Böylece örneklenecek işaretin negatif genlikleri de ADC'ye verilebilir. Böyle bir analog işaret bindirme devresi Şekil 11'de ilkesel olarak gösterilmiştir. Burada kullanılan işlemsel yükselteçler ile bir toplama devresi tasarlanmıştır[6]. Ancak örneklenecek işarete bu devrede bindirilecek gürültüler için çıkış işaretleri filtre edilebilir ya da işlemsel yükselteçlerde uygun ayarlamalar yapılabilir.



Şekil 11. Analog işaret bindirme devresi

Analog işaret bindirilerek örneklenmiş olan işaretlerde +1.5V'luk ekleme olduğundan ADC sonuç kayıtçılarından okunan değerden 2048 değeri çıkartılarak normalize edilir. Bu durumda ADC için dönüştürülecek en büyük analog genlik 1.5V olacaktır.

3.8. IQmath Kütüphane Desteği

F2812 DSP sabit noktalı işlem aritmetiğine çalışır. Bu nedenle programda kullanılan bütün değerler işlemcinin tanıdığı en büyük tamsayıya ölçeklendirilir. Ondalıklı sayılar yerine tamsayılarla çalışılır. Bu durumda işlem başarım hızı azalmaz. Çarpma, toplama, çıkarma ve bölme gibi işlemler birlikte yapıldığında anlamlı bitlerin kaybedilmemesi için bit kaydırma yöntemleri kullanılsa da karekök, üstel ve trigonometrik işlemlerde genellikle okuma tabloları kullanılır. Bu değer tabloların işlemciye yüklenmesi gerekir. Bu işlem hem bellekten harcar hem de programlama da zorluklara neden olur. Aynı zamanda elde edilecek analog değerlerde yuvarlamalara neden olur[1].

Ancak bu işlemcilerin programlanmasında assembly yerine C/C++ kullanıldığında derleyici doğrudan yazılmış en karışık işlem basamaklarını, her türlü ondalıklı işlem için, işlemciye uygun olarak dönüştürse de bu oldukça zaman almaktadır.

İşlem başarım hızını azaltmadan ondalıklı sayılarla da çalışabilmek için F2812 DSP üreticisi tarafından geliştirilen "***IQmathLib.h***" başlık dosyası, işlemci programlanırken sistem başlık dosyaları içine eklenir ve bağlı kütüphane kurallarına göre değişkenler tanımlanır.Bu durumda standart kayan noktalı aritmetik işlemler doğrudan C/C++ dilinde yazılabilir. C++ için başlık dosyası ***"IQmathCPP.h"*** dir[1,2].

IQmath yaklaşımında "I" integer (tam) , "Q" Quotient (kesir) anlamında kullanılır. Buna göre farklı formatlar kullanılmakla birlikte geleneksel 32-bit IQmath yaklaşımında I8Q24 formatı kullanılmakla birlikte Q19-Q26 formatı da işlem kararlılığına sahiptir. Şekil 12'de kayan noktalı aritmektikle yapılan bir işlemin geleneksel sabit noktalı biçimi ve bu işlemin C ve C++ için IQmath yaklaşımı gösterilmiştir[1-3].

Kayan Noktalı	`float Y, M, X, B;` `Y = M * X + B;`
Geleneksel Sabit noktalı Q format	`long Y, M, X, B;` `Y = ((i64) M * (i64) X + (i64) B << Q)) >> Q;`
C için IQmath	`_iq Y, M, X, B;` `Y = _IQmpy(M, X) + B;`
C++ için IQmath	`iq Y, M, X, B;` `Y = M * X + B;`

Şekil 12. F2812 DSp için IQmath işlem yaklaşımı

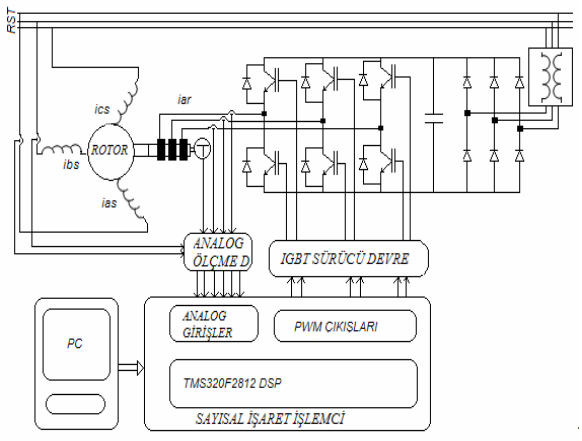
Benzer şekilde diğer fonksiyonlar içinde Tablo 2'den faydalanılabilir.

Tablo 2. IQmath yaklaşımında aritmetik işlemler ve trigonometrik fonksiyonlar

Kayan Noktalı	C'de IQmath	C++ da IQmath
float A, B;	_iq A, B;	iq A, B;
A = 1.2345	A = _IQ(1.2345)	A = IQ(1.2345)
A * B	_IQmpy(A , B)	A * B
A / B	_IQdiv (A , B)	A / B
A + B	A + B	A + B
A - B	A - B	A – B
>, >=, <, <=, ==, !=, &&, ||	>, >=, <, <=, ==, !=, &&, ||	>, >=, <, <=, ==, !=, &&, ||
sin(A),cos(A) sin(A*2pi),cos(A*2pi) atan(A),atan2(A,B) atan2(A,B)/2pi sqrt(A),1/sqrt(A) sqrt(A*A + B*B)	_IQsin(A), _IQcos(A) _IQsinPU(A), _IQcosPU(A) _IQatan(A), _IQatan2(A,B) _IQatan2PU(A,B) _IQsqrt(A), _IQisqrt(A) _IQmag(A,B)	IQsin(A),IQcos(A) IQsinPU(A),IQcosPU(A) IQatan(A),IQatan2(A,B) IQatan2PU(A,B) IQsqrt(A),IQisqrt(A) IQmag(A,B)
if(A > Pos) A = Pos if(A < Neg) A = Neg	_IQsat(A,Pos,Neg)	IQsat(A,Pos,Neg)

4. ÖRNEK MODEL VE UYGULAMA

Önceki bölümde açıklanan programlama ilkeleri kullanılarak bilezikli bir asenkron motorun rotor sargıları üzerinden bir IGBT evirici yardımıyla rotor sargılarına gerilim uygulayarak hız denetimi TMS320F2812 DSP ile gerçekleştirilmiştir. Rotor sargılarına anlık rotor gerilimi ve frekansını takip eden gerilim uygulanırsa bu gerilimin etkin değeri ve faz açısına bağlı olarak sırasıyla motorun hızı ve güç katsayısı değiştirilebilmektedir. Uygulanan gerilimle birlikte hız değişeceği için uygulanması gereken frekans da değişmektedir. Bu nedenle rotorda indüklenen gerilimin frekansı sürekli olarak takip edilmelidir[4-6]. Örnek model için ilkesel devre Şekil 13'de verilmiştir.

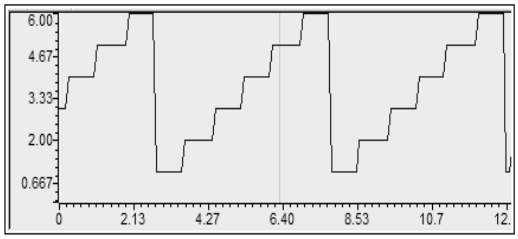


Şekil 13. Bilezikli asenkron motorun rotor sargılarına gerilim
uygulanmasına ilişkin denetim modeli

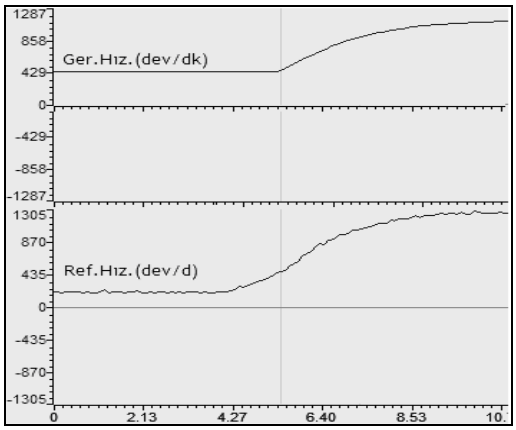
Bu modelde rotor ve stator akımları, bir analog işaret bindirme devresi yardımıyla ADC'den okunmuştur. Rotor hız/konum bilgisi için ENB1024 sayısal hız kodlayıcı kullanılmıştır. Hız/konum kodlayıcı QEP1 QEP2'den okunmuş ve normalize edilmiştir. IGBT evirici DSP'de üretilen SVPWM işaretleriyle sürülmüştür. Bunun için istenen hız denetimine uygun anahtarlama süreleri DSP'de hesaplanmıştır. Program C/C++ destekli yazılmıştır. PWM anahtarlama frekansı 2.5kHz olarak ayarlanmış ve ana program 400 µs'de sonlanacak şekilde zaman kesmeleri ayarlanmıştır.

5. SONUÇLAR

TMS320F2812 DSP'nin programlanmasında kullanılan CCS, aynı zamanda bütün kullanılan değişkenlerin ve kayıtçıların gerçek zamanlı olarak gözlenmesini sağlamaktadır. Ancak bir osiloskop gibi kullanılmasında en fazla 10Hz'lik bir örnekleme sağlamaktadır. Yine de denetim değişkenlerinin anlık durumları hakkında yeterli fikir vermektedir. Sayısal olarak hesaplanan değişkenler ve okunan işaretlerin harmonik analizleri de CCS ile kolayca yapılabilmektedir. Şekil 14 ve Şekil 15'de verilen sonuçlar DSP'den CCS vasıtasıyla alınmış örneklerdir.



Şekil 14. SVPWM anahtarlama bölgeleri



Şekil 15. Referans hız ve gerçek hız değişimi

6. KAYNAKLAR

[1] Texas Ins., TMS320f2812 Digital Signal Processor Implementation Tutorial, 2004

[2] Texas Ins., EzDSP TMS320F2812 Digital Signal Processor Technical References, 2004

[3] Texas Ins. World Wide Web site www.ti.com

[4] Tang, Y. ve Xu, L., Vector Control and Fuzzy Logic Control of Doubly Fed Variable Speed Drives with DSP Implemantation, IEEE Trans. Energy Conversion, 10, 4 (1995) 661-668

[5] Poddar, G. ve Ranganathan, V.T., Direct Torque and Frequency Control of Doubly Inverter Fed Slip-Ring Induction Motor Drive, IEEE Trans. Industrial Electronics, 51,6 (2004) 1329-1337.

[6] Kesler, S., Bilezikli Asenkron Makinaların Bulanık mantık Tabanlı Hız Denetiminin TMS3220F2812 DSP ile Gerçekleştirlmesi, Doktora Tezi, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, 2006.