T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ASENKRON MOTORUN SAYISAL İŞARET İŞLEYİCİ TABANLI VEKTÖR KONTROLÜ

İbrahim ERTÜRK

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ A.B.D.

Konya, 2006

T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ASENKRON MOTORUN SAYISAL İŞARET İŞLEYİCİ TABANLI VEKTÖR KONTROLÜ

İbrahim ERTÜRK

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez .. / .. / 2006 tarihinde aşağıdaki jüri tarafından oy birliği / oy çokluğu ile kabul edilmiştir.

Yrd. Doç. Dr. Yüksel ÖZBAY
(Danışman)

Yrd. Doç. Dr. Abdullah ÜRKMEZ
(Üye)

Yrd. Doç. Dr. Nihat YILMAZ
(Üye)

ÖZET

Yüksek Lisans Tezi

ASENKRON MOTORUN SAYISAL İŞARET İŞLEYİCİ TABANLI VEKTÖR KONTROLÜ

İbrahim ERTÜRK

Selçuk Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği A.B.D.
2006, 130 Sayfa
Danışman : Yrd. Doç. Dr. Yüksel ÖZBAY
Jüri : Yrd. Doç. Dr. Abdullah ÜRKMEZ
Yrd. Doç. Dr. Yüksel ÖZBAY
Yrd. Doç. Dr. Nihat YILMAZ

Bu çalışma asenkron motorun sayısal işaret işleyici tabanlı hız algılayıcısız vektör kontrolünü inceleme amacıyla yapılmıştır. Öncelikle bu konuyla ilgili daha önce yapılmış çalışmalar araştırılmıştır. Çalışmamızda uygun bir vektör kontrol yöntemi bulunması için incelenen kaynaklarda kullanılan vektör kontrol yöntemleri çeşitli simülasyon yazılımları üzerinde test edilmiştir.

Daha sonra, simülasyon çalışmalarında, farklı simülasyon yazılımları ve bu yazılımların sağladığı simülasyon yöntemleri incelenmiştir ve simülasyon çalışmasında kullanılacak simülasyon yöntemi ve uygun yazılım araştırılmıştır. Simülasyon yazılımı ve yöntemi seçilirken simülasyon hızına, programın kararlılığına ve simülasyon çalışması sonucunda elde edilen verilerin incelenmesi için kullanılan araçlara dikkat edilmiştir. Böylece simülasyon için MATLAB/SIMULINK yazılım paketi ve C S-Function kullanılmasına karar verilmiştir.

Simülasyon ortamında test edilen vektör kontrol yöntemlerinin pratikte de test edilebilmesi için asenkron motora ve DSP tabanlı sürücüye ihtiyaç duyulmuştur. Sürücü sistemden beklenen en önemli özellik düşük-maliyettir. Sürücü sistem için test amaçlı kullanılabilecek bir DSP kartı araştırılmıştır ve TMS320F243 sabit-noktalı DSP içeren düşük-maliyetli bir kart satın alınmıştır. Bu DSP kartla uyumlu bir motor kontrolör kartı piyasada mevcut olmasına rağmen, oldukça pahalı olduğundan daha düşük-maliyetli benzer bir kart tasarlanmış ve imal edilmiştir. Böylece, araştırılan kaynaklarda verilen vektör kontrol yöntemlerini hem simülasyon ortamında hem de pratikte test etme imkanı sağlanmıştır.

Kaynak araştırması sırasında, hız algılayıcısız vektör kontrolünde rotor hızı ve durum değişkenleri ile makine parametrelerinin kestirilmesi için genellikle Model Referans Adaptif Sistem (MRAS) ve gözetleyici tabanlı yöntemlerle karşılaşılmıştır.

Çalışmamızda, öncelikle diğerlerinden daha basit algoritması olan MRAS üzerinde durulmuştur. Rotor hız kestirimi için MRAS kullanan vektör kontrol yöntemleri simülasyon ortamında düşük rotor hızları haricinde iyi sonuçlar vermiştir. Fakat pratikte anlamlı sonuçlar elde edilememiştir.

İkinci olarak Genişletilmiş Kalman Filtresi (EKF) içeren gözetleyici tabanlı kestirim yöntemi kullanılmıştır. Bu yöntem karmaşık bir algoritma gerektirir. Fakat parametre değişimlerine ve ölçüm gürültülerine karşı bağışıklığı nedeniyle hem simülasyon ortamında hem de pratikte sıfırdan nominal değerine kadar tüm rotor hızlarında uygun sonuçlar vermiştir.

Denemelerden sonra, vektör kontrol için gerekli tüm verilerin stator akımları ile inverter DC besleme voltajı ölçümleri yardımıyla EKF tabanlı gözetleyici tarafından kestirilen hız algılayıcısız doğrudan vektör kontrol yöntemi kullanılmasına karar verilmiştir. Önerilen yöntem simülasyon ortamında ve pratikte test edilmiştir. Önerilen yöntem her rotor hızında ve değişik yük şartları altında kararlı olmuştur. Ayrıca önerilen yöntem referans rotor hızının ve yükünün değişimlerine karşı iyi bir dinamik performans sergilemiştir.

Hız kestiriminde ortalama yaklaşık % 2.5 olarak bulunmuştur ve referans rotor hızının ve yükünün değişimlerine karşı ortalama cevap zamanı yaklaşık 300 ms dir. Bundan başka, kestirilen hız referans hızı ile tam olarak aynıdır. İlaveten, önerilen EKF tabanlı kestirim yöntemi sabit v/f orantısına dayanan altı-adım inverter ile test edilmiştir ve rotor hızının kestirimi önerilen yöntem tarafından doğru bir şekilde yapılmıştır ve bu uygulamalarda arzulanan sonuçlar elde edilmiştir.

Anahtar Kelimeler : Asenkron motor, alan yönlendirme kontrolü, algılayıcısız kontrol, vektör kontrol, DSP, Kalman Filtresi, MRAS, C S-Funtion.

ABSTRACT

MS Thesis

DIGITAL SIGNAL PROCESSOR BASED VECTOR CONTROL OF INDUCTON MOTOR

İbrahim ERTÜRK

Selcuk University
Graduate School of Natural and Applied Scienes
Department of Electrical and Electronics Engineering
2006, 130 Pages
Supervisor : Asst. Prof. Dr. Yüksel ÖZBAY
Jury : Asst. Prof. Dr. Abdullah ÜRKMEZ
Asst. Prof. Dr. Yüksel ÖZBAY
Asst. Prof. Dr. Nihat YILMAZ

This study was conducted to examine Digital Signal Processor (DSP) based speed sensorless vector control of induction motor. Firstly, previously conducted studies related with this subject were investigated. Used vector control methods in examined resources were tested on various simulation softwares to find suitable vector control method for our application.

Nextly, in simulation studies, different simulation softwares and simulation methods that provided by these softwares were examined and suitable software and simulation method that will be used in simulation study were investigated. While selecting simulation software and method, simulation speed, stability of program and examining tools used for obtained data from simulation study were considered. Thus, use of MATLAB/SIMULINK software package and C S-Function was decided for simulation.

To test vector control methods that were tested on simulation environment in practice, induction motor and DSP based driver were required. Most important property that expected from driver system is low-cost. A DSP card that can be used as test purpose was investigated for drive system and low-cost card that included TMS320F243 fixed-point DSP was bought. Additionally, a motor controller card that is suitable with this DSP card is available on the market. But, since it is extremely expensive, low-cost card that is similar to this card was designed and built in this study. Thus, the possibility of testing given vector control methods in investigated resources was provided in simulation environment and practice.

While examining literature, generally Model Reference Adaptive System (MRAS) and observer based methods were encountered to estimating rotor speed, state variables and machine parameters in speed sensorless vector control. In our study, firstly MRAS whose algorithm is simpler than others was considered. Vector

control methods that use MRAS to rotor speed estimation gave good results in simulation environment except for lower rotor speed. But any meaningful results were not obtained in practice.

The secondly, observer based estimating method that included Extended Kalman Filter (EKF) was used. This method requires a complex algorithm. But, because of its immunity to parameter variations and measuring noise, it gave suitable results in simulation and practice for every rotor speed from zero to nominal value.

After the experimentations, the use of speed sensorless direct vector control methods that requires information estimated by EKF based observer that uses stator current and inverter supply voltage measurement was decided. Proposed method was tested in simulation environment and practice. It was stable on every rotor speed and under different load condititions. Furthermore, it was exhibited good dynamic performance against the variations of reference rotor speed and load.

Average error on speed estimation was found to be about 2.5% and average response time against to variations of reference rotor speed and load was about 300 ms. Furthermore, estimated speed was exactly the same with reference rotor speed. In addition, proposed EKF based estimating method was tested with six-step inverter based on fixed v/f ratio and estimate of rotor speed was accurately made by proposed method and desired results were obtained in these applications.

Key Words : Induction motor, field oriented control, sensorless control, vector control, DSP, Kalman Filter, MRAS, C S-Funtion.

ÖNSÖZ

Asenkron makinenin alan yönlendirme kontrolü hakkında Türkçe kaynağa pek rastlanmamıştır. Ancak, yabancı dergi ve internet üzerinde İngilizce kaynaklar bulunmuş, ayrıca konu ile ilgli kitaplar edinilmiştir. Yurdumuzda, bu tür sürücüler üretilmeyip genellikle ithal edilmektedir. Türkiye'de daha çok voltaj/frekans tekniğine dayanan sürücüler üretilmektedir. Bu nedenlerle bu konuda çalışma yapılmaya karar verilmiştir. Tez çalışması hem uygulama hem de simülasyon ortamında yürütülmüş ve bu çalışma ile önerilen sistemin geçerliliği kanıtlanmıştır.

Uzun süren çalışmalar sırasında bana her zaman destek olan eşim Fatma ERTÜRK'e ve her aşamada önerilerini hiçbir zaman esirgemeyen hocam Sayın Yrd. Doç. Dr. Yüksel ÖZBAY'a teşekkür ederim.

Yapılan çalışmanın ileride yapılacak olan çalışmalara ışık tutmasını dilerim.

Konya, 2006 İbrahim ERTÜRK

İÇİNDEKİLER

Sayfa No :

ÖZET iii
ABSTRACT v
ÖNSÖZ vi
İÇİNDEKİLER vii
SİMGELER VE KISALTMALAR ix

1. GİRİŞ 1

2. KAYNAK ARAŞTIRMASI 5
2.1. Vektör Kontrol Yöntemleri 6
2.2. Asenkron Makine için Parametre ve Durum Kestirimi Yöntemleri 7

3. ASENKRON MAKİNE KONTROL ELEMANLARI 16
3.1. Genel Asenkron Makine Kontrol Sistemi 16
3.2. Asenkron Motor 17
3.2.1. Dinamik Makine Modeli 19
3.2.2. Birim Sürekli-Zaman Dinamik Makine Modeli 23
3.2.3. Sürekli ve Ayrık Zaman Birim Durum Uzayı Makine Modeli 26
3.3. Alan Yönlendirme Kontrolü 28
3.4. İnverter Modeli ve Anahtarlanması 34
3.5. Genişletilmiş Kalman Filtresi 36

4. DSP TABANLI VEKTÖR KONTROLÖR VE
ASENKRON MOTOR SİMÜLASYON TASARIMI 41

4.1. Simülasyon Dosyaları ve Fonksiyonlar 42

5. TMS320F243 TABANLI ASENKRON
MOTOR SÜRÜSÜCÜ TASARIMI..... 46

5.1. DSP Kontrolör (F243 DSK – DSP STARTER KIT)..... 46
5.2. Sayısal Motor Kontrolör (DMC) 48
5.3. TMS320F243 için Yazılım Tasarımı 49

6. DENEYSEL SONUÇLAR 55

6.1. Asenkron Makinenin DSP Tabanlı Vektör Kontrolü Simülasyonu 55
6.1.1. Vektör Kontrollü Asenkron Makinenin Boşta Çalışma Simülasyonu 55
6.1.2. Vektör Kontrollü Asenkron Makinenin Yükte Çalışma Simülasyonu 58
6.2. TMS320F243 Tabanlı Asenkron Motor Sürücüsü ile Yapılan Uygulama Sonuçları 62

7. SONUÇLAR VE ÖNERİLER 67
7.1. Sonuçlar 67
7.2. Öneriler 69

8. KAYNAKLAR 71

EKLER 75

EK-A SİMÜLASYON DOSYALARI 76
EK-B SAYISAL MOTOR KONTROLÖR KARTI ORCAD ŞEMALARI 84
EK-C DSP YAZILIMI 97
EK-D SİMÜLASYON ÇALIŞMASINDAN ELDE EDİLEN GRAFİKLER 123

SİMGELER ve KISALTMALAR

d/dt	Türev operatörü
f	Frekans [Hz]
i_r	Rotor akımı [A]
i_s	Stator akımı [A]
Ll_r	Rotor kaçak indüktans [H]
Ll_s	Stator kaçak indüktans [H]
L_m	Karşılıklı indüktans [H]
n_s	Senkron hız [d/dk]
p	Çift kutup sayısı
R_r	Rotor direnci [Ω]
R_s	Stator direnci [Ω]
s	Birim kayma
T_e	Üretilen moment [N.m]
T_L	Yük momenti [N.m]
V_s	Stator voltajı [V]
ω	Açısal frekans [rd/s]
λ	Akı [wb]
σ	Stator indüktansı için dönüşüm sabiti
θ	Rotor akısı uzaysal pozisyonu [rd]

(abc)	Üç-fazlı sistem
AC	Alternatif akım
ADC	Analolog/Dijital Dönüştürücü
ASM	Asenkron motor
Cosϕ	Güç katsayısı
DC	Doğru akım
(d-q)	Durgun koordinat sistemi
(D-Q)	Hareket eden koordinat sistemi
DSP	Sayısal işaret işleyici
EKF	Genişletilmiş Kalman filtresi

IGBT	Beyzi izole edilmiş bipolar transistör
EKF	Genişletilmiş Kalman filtresi
KF	Kalman filtresi
PI	Oransal-İntegral regülatör
PWM	Darbe genişlik modülasyonu
SVPWM	Uzay vektör dalga genişlik modülasyonu
YSA	Yapay sinir ağı

1. GİRİŞ

Asenkron makineler basit yapıları, ucuz olmaları, fazla bakıma ihtiyaç duymamaları ve yüksek verimleri nedeniyle endüstride yaygın olarak kullanılırlar. Sıfırdan tam-yüke kadar neredeyse sabit hızda çalışırlar. Ancak, doğrusal olmayan yapıları nedeniyle daha karmaşık kontrol algoritmaları gerektirirler ve değişken hızlı uygulamalara kolaylıkla uydurulamazlar. Bu nedenle çok uzun bir süre yalnızca sabit hızlı uygulamalarda kullanılmıştır. Değişken hızın gerekli olduğu uygulamalarda DC makineler öne çıkmıştır. Ancak, DC makineler dönen kısımdaki sargının beslenmesi için fırçaya ihtiyaç duyduklarından, verimleri düşüktür ve asenkron makinelere göre oldukça sık bakıma ihtiyaç duyarlar.

Yarı-iletken teknolojisindeki gelişmeler sayesinde çeşitli güç elektroniği elemanları ve mikro-denetleyiciler geliştirilmiştir. Bu gelişmelerden sonra asenkron makineleri değişken hızlı uygulamalarda kullanmak mümkün olmuştur. Bilindiği gibi asenkron makinenin hızı statora uygulanan kaynağın frekansına ve kutup sayısına bağlıdır. Kutup sayısı değiştirilerek yapılan hız ayarı zordur ve her biri aynı anda kullanılamayan sargılara ihtiyaç vardır. Bu da fazladan bakır kullanımı anlamına gelir ve hem maliyeti, hem de makine boyutlarını artırır. Yine bu şekilde yapılan hız ayarı sürekli biçimde olmayıp, kademeli olarak sağlanabilir. Bundan başka, asenkron makinenin statoruna uygulanan gerilimin büyüklüğü ve frekansı değiştirilerek de hızı ayarlanabilir. Bu günümüzde çeşitli açık çevrim frekans konvertörleriyle yapılmaktadır.

Bir frekans konverteri şebeke gerilimini doğrultarak, istenilen frekans ve büyüklükte voltaj sağlar. Böylece asenkron makinenin hızı frekansa bağlı olarak ayarlanabilir. Frekans değiştirilerek yapılan hız kontrol tekniklerinde hız geri beslemesi olmadığından asenkron makinenin doğrusal olmayan tork-hız karakteristiği nedeniyle kararlı bir çalışma her zaman sağlanamayabilir. Bu şekilde yapılan kontrol biçimine skaler kontrol denir. Çünkü, değişkenlerin uzaysal durumları dikkate alınmayıp yalnızca büyüklüğüne bakılır.

DC makinelerde birbirinden tamamen ayrı alan ve armatür sargısı mevcuttur. Bu sargılara istenilen DC gerilim ayrı ayrı uygulanabilir. Makinenin momenti armatür akımı ile manyetik akının çarpımı ile ve manyetik akı da alan sargısı

akımıyla doğru orantılıdır. Böylece, elde edilmek istenilen manyetik akı ve moment bu sargılara uygulanan DC gerilimin basitçe ayarlanmasıyla elde edilebilir. Buradan yola çıkılarak asenkron makinenin de çeşitli dönüşüm teknikleriyle alan ve armatür sargısına ayrı gerilim uygulanabilen DC makineye benzetilebileceği ispatlanmıştır. Asenkron makinenin stator akım vektörü DC makineye benzer şekilde alanı ve momenti oluşturan iki dik bileşene ayrılabilir. Bu vektörlerden manyetik akı vektörüne paralel olan akım vektörü; manyetik akıyı oluşturan akımı, buna dik olan akım vektörü de momenti oluşturan akımı temsil ederler. Böylece, bu iki stator akım vektörü bileşeni ayrı ayrı kontrol edilerek asenkron makine de DC makine gibi kontrol imkanına kavuşur. Bu şekilde yapılan kontrol şekli; kontrol değişkenlerinin hem büyüklükleri hem de uzaysal durumları dikkate alındığından "vektör kontrol" veya bu vektör değişkenler stator veya rotor alanını referans kabul eden yeni bir koordinat sistemine dönüştürüldüklerinden "alan yönlendirme kontrolü" olarak adlandırılır.

Yukarıda bahsedilen asenkron makinenin hız kontrol teknikleri karmaşık kontrol algoritmaları ve elektronik güç anahtarları gerektirdiklerinden, bunların uygulanabilir hale gelmeleri için yarı-iletken teknolojisindeki gelişmeler önemli olmuştur. Özellikle, vektör kontrol tekniği çok hızlı güç elektroniği anahtarları ve makineden aldığı verileri hızlı bir şekilde işleyip tekrar kontrol sinyaline dönüştürerek bu güç elektroniği anahtarlarına gönderebilecek çok hızlı işlemcilere ihtiyaç duyar. Bu amaçla yeterince güçlü ve çok hızlı yarı-iletken anahtar olan IGBTler (Isolated Gate Bipolar Transistor) geliştirilmiştir.

Vektör kontrol tekniğinin gerektirdiği işlemleri çok hızlı bir şekilde yerine getirebilecek işlemciler olan DSPlerin (Digital Signal Processor) geliştirilmesi ile artık vektör kontrol tekniğinin ticari olarak da uygulanması mümkün hale gelmiştir. DSPlerin geliştirilmesinin amacı motor kontrolü olmadığından uzun bir süre ses ve görüntü işleme gibi diğer işlemler için kullanılmıştır. Fakat günümüzde motor kontrolüne yönelik olarak üretilen DSPler mevcuttur. Bunlar, özellikle motor kontrolü için üretildiklerinden, ADC (Analog to Digital Converter), PWM (Pulse Width Modulation) sinyallerini üretebilen birimleri, hız ve durum algılayıcı enkoder sinyallerini tutabilecek birimleri içerirler.

Vektör kontrol tekniklerinde, rotor ve manyetik alanının hızları ve uzaysal durumları algılayıcı cihazlarla algılanabileceği gibi, makinenin matematiksel modelinden yararlanılan çeşitli hız kestirme yöntemleriyle de bulunabilir. Bu yöntemlerde, istenilen veriler doğrudan hesaplanmayıp, hız değişkeni içeren bir makine modeliyle hız değişkeni içermeyen bir makine modelinin karşılaştırılmasından elde edilen hata sinyali hız değişkeni içeren makine modeline geri beslenerek hız kestirme işlemi yapılır. Hız kestirme yöntemleri, makine miline bir algılayıcı bağlamanın mümkün olmadığı durumlarda yararlı olabilir. Ayrıca maliyetin de azaltılmasını sağlar.

Vektör kontrol tekniğinde belirli bir anda istenilen hız veya moment için gerekli kontrol değişkenleri olan manyetik akı veya akım ne ise makineye uygulananlar da onlar olacaktır. Böylece, makinenin yüksek verimli olarak çalışmasını sağlar. Günümüzde enerji maliyetleri arttıkça verimli çalışma öne çıkmaktadır. Ayrıca, çevrenin de korunması için mevcut enerji kaynaklarının ekonomik bir biçimde kullanılmaları gereklidir. Bu durumda, mevcut teknikler daha verimli hale getirilmeye çalışılmaktadır. Vektör kontrol tekniği bu bakımdan önem kazanmaktadır.

Bu tez çalışmasında asenkron makinenin algılayıcısız vektör kontrolü incelenecektir. Öncelikle, incelemeler için asenkron makinenin ve kontrol sisteminin matematiksel modellerinin oluşturulabildiği MATLAB/SIMULINK yazılım paketinden yararlanılacaktır. Bu yazılım ile yapılan simülasyon çalışmalarından sonra, özellikle motor kontrolüne yönelik olarak Texas Instruments firması tarafından geliştirilmiş olan TMS320F243 DSP kontrolörü ile tasarlanacak olan kontrol birimi ve güç elektroniği birimi ile 750 W gücündeki bir asenkron makinenin vektör kontrolü incelenecektir.

Tez içerisindeki bölümler kısaca şöyle özetlenebilir. Birinci bölüm olan bu bölümde asenkron makine ve kontrol yöntemleri kısaca açıklanmaya çalışılmıştır. İkinci bölümde ise daha önce bu tezin konusu ile ilgili olarak yayınlanmış çalışmalara yer verilmiştir. Üçüncü bölümde asenkron motor ve kontrolünde kullanılan yöntemler incelenmiştir. Bu amaçla otomatik kontrolde kullanılan asenkron motor modelleri, vektör kontrol teorisi ve Kalman Filtresi üzerinde durulmuştur. Dördüncü bölümde MATLAB/SIMULINK yazılım paketi ile DSP

kontrolörün ayrık zaman simülasyon modeli ve asenkron motorun sürekli zaman simülasyon modeli tasarımı üzerinde durulmuştur. Bu amaçla tez çalışmasında kullanılan simülasyon yönteminin temelini oluşturan SIMULINK C S-Function incelenmiştir. Beşinci bölümde tez projesinde kullanılan TMS320F243 tabanlı asenkron motor sürücüsü donanım ve yazılım tasarımı üzerinde durulmuştur. Altıncı bölümde, önerilen DSP tabanlı asenkron motor vektör kontrol yönteminin pratikte uygulanabilirliğini araştırmak amacıyla simülasyon ortamında ve pratikte yapılan deneysel çalışmalara yer verilmiştir. Yedinci bölümde tez çalışması boyunca varılan kannatler sunulmuş, ayrıca bu tez çalışmasında yapılmayıp ileride yapılabilecek olan çalışmalar ile ilgili önerilerde bulunulmuştur. Sekizinci bölümde ise kullanılan kaynaklara yer verilmiştir. Son olarak tezin bütünlüğünü korumak amacıyla tez metni içerisinde bulunmayan, ancak burada yapılan çalışmaları daha anlaşılır hale getirebileceği düşünülen çeşitli grafikler, şemalar ve dosyalar ek olarak sunulmuştur.

2. KAYNAK ARAŞTIRMASI

Bu bölümde asenkron makinenin vektör kontrolü ile ilgili literatürde yer alan, özellikle bu tez çalışmasının konusuyla ilgili yayınlanmış olan çalışmalara yer verilecektir.

Asenkron makinenin yüksek performanslı kontrolünün temeli vektör kontrol tekniklerine dayanır. Vektör veya alana yönelik kontrol yöntemlerinde makinenin dinamik modeli gerekli olur. Makine dinamik modeli durum denklemleriyle temsil edilir. Durum değişkeni olarak genellikle stator akımı ve rotor akısı kullanılır. Vektör kontrolünden beklenen performansı elde edebilmek için öncelikle durum değişkenlerinin vektörel büyüklüklerinin ve yönlerinin kesin olarak bilinmesi gereklidir. Stator akım ve voltajı makine terminallerinden doğrudan ölçülebilir. Rotor ve stator akısının ölçülebilmesi için ise standart makinelerde bulunmayan yarı-iletken Hall etkisi algılayıcısı veya yalnızca alan algılayıcı olarak kullanılan ikincil sargılar gereklidir. Tabii ki, bu durumda sürücü maliyeti artacaktır. Bu nedenle, genellikle literatürde sürücü maliyetinin azaltılması amacıyla, rotor veya stator alan vektörlerinin çevrim-içi olarak kestirilmesi üzerinde durulmaktadır. Ancak, bu yöntemlerden elde edilen sonuçların doğruluğu makine parametrelerinin kesin olarak bilinmesine bağlıdır. Dinamik model için gerekli parametrelerin makine üzerinde yapılan standart testler ile bulunabilmesine rağmen, normal çalışma süresince sargı sıcaklığı ve demirin doyumu gibi etkenlerle değişmeleri mümkündür. Bu durumda, hesaplanan durum değişkenleri hatalı olacaktır ve değişen parametrelere göre makine modelinin güncellenmesi gerekecektir. Literatürde sıkça karşılaşılan çalışma konularından biri de değişen parametrelerin çevrim-içi olarak kestirilmesi ve buna göre makine modelinin güncellenmesidir.

Bölüm içerisinde öncelikle konunun daha iyi anlaşılabilmesi açısından, literatürde bahsi geçen vektör kontrol yöntemlerine yer verilecektir. Daha sonra makine parametreleri kestirimi ve durum kestirimi yöntemleri ile ilgili yayınlar özetlenmeye çalışılacaktır. Parametre ve durum kestirimi için genellikle aynı yöntemler kullanılmaktadır. Bu nedenle ilgili yöntemler kestirim yöntemleri olarak bir başlık altında verileceklerdir.

2.1. Vektör Kontrol Yöntemleri

Vektör kontrol yöntemlerinin temelinde vektör dönüşümleri vardır. Statora göre durgun koordinat sisteminde ifade edilen stator voltaj ve akım vektörleri yönelimin yapıldığı alana ait akı vektörünü referans kabul eden yeni bir dönen koordinat sisteminde ifade edilebilecek şekilde dönüşüm işlemine tabi tutulurlar. Bu nedenle öncelikle yeni koordinat sisteminin ani uzaysal durumunun bilinmesi gereklidir. Literatürde genellikle yönelimin yapıldığı alan ve bu alanın uzaysal durumunun bulunma yöntemine göre sınıflandırmalara rastlanmaktadır.

Boussak ve Jarray'e (2006) göre vektör kontrol teknikleri, doğrudan ve dolaylı alan vektör yönlendirme kontrolü olarak iki sınıfa ayrılabilir. Aynı sınıflandırma Bose (1998) tarafından "dolaylı (ileri-beslemeli) vektör kontrol" ve "doğrudan (geri-beslemeli) vektör kontrol" biçiminde yapılmıştır. Doğrudan kontrol yönteminde alan vektörü stator terminal büyüklüklerinden elde edilir. Dolaylı kontrolde ise alan yönlendirmenin sağlanması için makine kayma frekansı kullanılır. Doğrudan vektör kontrolde mevcut kayma frekansının hesaplanması gerekirken, dolaylı vektör kontrolde kayma hızı sisteme referans kayma hızı olarak verilir. Bunun anlamı, doğrudan vektör kontrolde yönlendirmenin yapıldığı alanının uzaysal durumu kesin olarak bilinmelidir. Dolaylı vektör kontrolde ise yönelimin yapıldığı alanının uzaysal durumu aslında bilinmeyip, alanın istenen uzaysal durumuna göre vektör dönüşümleri yapılır.

Bose (1998) tarafından yönlendirmenin yapıldığı alana göre "stator alan yönlendirme kontrolü" ve "rotor alan yönlendirme kontrolü" şeklinde bir sınıflandırma yapılmıştır. Ancak literatürde alan yönlendirme kontrolü veya vektör kontrol terimleri genel olarak rotor alanına yönlendirme kontrolü anlamında kullanılmaktadır. Holtz'e (2001) göre vektör kontrol için kullanılan asenkron makine dinamik modelini oluşturan durum denklemlerindeki durum değişkenleri stator alan yönlendirme kontrolü için stator akımı ve stator akısı, rotor alan yönlendirme kontrolü için stator akımı ve rotor akısı olarak seçilmelidir. Rotor alanına yönelik vektör kontrolü, makine dinamiklerinden bağımsız olarak, hava aralığında istenilen herhangi bölgede ve yoğunlukta stator manyeto-motor kuvvet (mmk) dağılımı

istenildiğinde uygulanır. Stator alanına-yönelik vektör kontrolü ise, hızlı bir stator akım kontrolü yapılmak istenildiğinde uygulanır.

Li ve ark. (2006) göre vektör kontrol yöntemleri, rotor hızının veya alan akısı konumunun belirlenmesi için algılayıcı kullanılıyor ise "algılayıcılı vektör kontrol" aksi halde "algılayıcısız vektör kontrol" şeklinde iki sınıfa ayrılabilir. Burada kullanılan algılayıcı terimi stator akım ve voltajlarının ölçülmesi için kullanılan cihazlar anlamında olmayıp, rotor hız veya konumunu algılayan cihaz anlamında kullanılmaktadır. Tabii ki, tüm vektör kontrol yöntemlerimde stator akımlarının ve voltajlarının ölçülmesi için algılayıcılar bulunmalıdır. Gerçekte değişken hızlı asenkron motor sürücülerinde rotor hız veya konum algılayıcı bulunmaz ise sistem açık-çevrim kontrol sistemi olur. Hassas konumlandırma sistemlerinde vektör kontrollü asenkron motor kullanılacak ise rotor hız veya konum algılayıcı bulunması daha uygun olacaktır (Li, Chiasson, Bosson ve Tolbert 2006).

Yukarıda bahsedilen vektör kontrol teknikleri birbirlerinden ayrı olarak kullanılmayıp, algılayıcılı/algılayıcısız, dolaylı/dolaysız ve rotor/stator alan yönlendirme kontrolü şeklinde birlikte kullanılırlar.

2.2. Asenkron Makine için Parametre ve Durum Kestirimi Yöntemleri

Sistem teorisinde, "kestirim" terimi genel olarak sistemin modelini oluşturan durum denklemlerindeki ölçülebilen büyüklükler yoluyla ölçülebilen büyüklüklerin tahmin edilmesi anlamında kullanılmaktadır. Durum denklemlerindeki büyüklükler ise durum değişkenleri ve parametrelerdir. Teoride parametreler sabit olarak kabul edilirken, gerçekte değişkendirler. Bu nedenle durum değişkenlerinin kestirimi için kullanılan yöntemler parametre kestirimi için kullanılabilir. Aynı şekilde, parametre kestirimi için kullanılan yöntemler durum değişkeni kestirimi amacıyla kullanılabilir.

Genel olarak kestirim yöntemleri, "spektral analiz tabanlı", "gözetleyici tabanlı" ve "Model Reference Adaptive System (MRAS) tabanlı" yöntemler olarak üç gruba ayrılabilir. Bunların dışında parametrelerin ve durum değişkenlerinin doğrudan makine durum denklemlerinden hesaplanabildiği yöntemler de geliştirilmiştir. Ancak doğrudan hesaplama yöntemleri genellikle sistemin kararlılığını kötüleştirmektedir. Ayrıca dolaylı olarak makine parametrelerinin

kestirimi için yöntemlerde geliştirilmiştir. Örnek olarak sargı sıcaklılığı ölçülerek sargı direnci dolaylı hesaplanabilmektedir.

Toliyat ve ark. (2003) tarafından asenkron motor parametre kestirim teknikleri için yukarıdakine benzer bir sınıflandırma yapılmıştır. Ayrıca çevrim-dışı kestirim tekniklerine de değinmişleridir. Ancak çevrim-dışı teknikler makine parametrelerinin kestirilmesi amacıyla değil, belirlenmesi amacıyla uygulanırlar. Kestirim teknikleri genellikle çalışmalarda birbirlerinden ayrı olarak kullanılmayıp, birlikte kullanılarak vektör kontrolün performansı artırılmaya çalışılmaktadır. Bu nedenle burada parametre ve durum kestirimi konusunda incelenen kaynaklar bir başlık altında verilecektir. Ancak öncelikle yayınlarda sıkça bahsedilen yöntemleri kısaca açıklanmaya çalışılacaktır.

Tüm kestirim yöntemleri genellikle vektör kontrol için gerekli veriler makine terminallerinden doğrudan ölçülebilen stator akımlarından ve voltajlarının analizi ile elde edilir. Ancak stator voltajı kontrolör tarafından belirlendiğinden stator voltajının ölçülmesi yerine inverter DC bus voltajının ölçülmesi yeterlidir. Stator voltajı böylece inverter anahtarlama sinyalleri ve DC bus voltajının bilinmesiyle hesaplanabilir. Bu durumda vektör kontrol için DC bus voltajının ve stator akımlarının ölçülmesi yeterli olacaktır.

Spektral analiz tabanlı yöntemlerde veri olarak stator akımları kullanılır. Bu yöntemlerde genellikle Fourier Dönüşümü ile zaman domeninden frekans domenine dönüştürülür. Çevrim-içi tekniklerde hesaplama süresi önemli olduğundan genellikle Fast Fourier Transformation (FFT) yöntemi ile dönüşüm yapılır. Böylece elde edilen frekans domeni verisinden rotor oluklarının stator akımında oluşturduğu parazitler aranmaya çalışılır. Yöntemin yalnızca rotor hızı ve akısının kestirimi için kullanılabileceği açıktır. Parametre kestirimi için uygun değildir. Bu yöntemi diğerlerinden ayıran tarafı makine modeline ihtiyaç duymamasıdır. Spektral analiz tabanlı yöntemler parametre değişimlerden etkilenmez (Toliyat, Levi ve Raina 2003).

Gözetleyici tabanlı yöntemlerde makine parametrelerinin ve dinamik modelinin tam olarak bilinmesi gereklidir. Bu yöntem makine modeli kullanarak ölçülebilen makine giriş ve çıkışları yardımıyla ölçülemeyen durum değişkenlerinin hesaplanmasına dayanır. Her kontrol çevriminde makine giriş ve çıkışları ölçülerek durum değişkenleri güncellenir. Bu yöntem ile makine terminallerinden doğrudan

ölçülemeyen parametrelerin kestirimi de yapılabilir. Yöntemin performansı makine modelinin ve parametrelerinin doğruluğuna bağlıdır.

Gözetleyici algoritma olarak en sık kullanılan yöntem Kalman Filtresinin (KF) doğrusal olmayan sistemler için değiştirilmiş biçimi olan Genişletilmiş Kalman Filtresi (Extended Kalman Filter – EKF) kullanılır. KF ilk olarak 1960 yılında R. E. Kalman tarafından ayrık verilerin filtrelenmesine tekrarlamalı bir çözüm olarak ortaya atılmıştır ve bir çok uygulama alanı bulmuştur (Welch ve Bishop 2002). Meybeck'e (1979) göre ise KF bir optimal tekrarlamalı veri işleme algoritması olup sistem ve ölçüm elemanı gürültüleri ile birlikte ölçüm elemanından alınan verilerden ölçülemeyen verileri elde etmek amacıyla kullanılır, yani KF kestirim amacıyla kullanılabilir. KF ile parametre ve ölçüm gürültüleri filtre edilebilir. Filtrenin ayrık veriler için önerilmesinden de anlaşılacağı gibi sayısal işaret işlemeye yöneliktir. Bu nedenle, DSPlerin de gelişmesiyle birlikte, filtre pratikte uygulanabilir hale gelmiş, ve vektör kontrollü asenkron makine sürücülerinde de parametre düzeltilmesi ve rotor hızı kestirimi amacıyla kullanılmıştır.

Model Reference Adaptive System (MRAS) tabanlı kestirim yöntemi gözetleyici tabanlı yönteme benzer. Ancak MRAS tabanlı yöntem için iki model kullanılır. Modellerden biri referans model olarak adlandırılır. Bu model giriş olarak ölçülebilen durum değişkenleri ve makine girişleri kullanılır. Diğer model adaptif model olarak adlandırılır. Bu model ise giriş olarak ölçülebilen ve ölçülemeyen durum değişkenleri veya parametreleri giriş olarak kullanılır. Her iki modelin çıkışı aynı durum değişkeni olmalıdır. Aynı durum değişkeni için her iki modelden elde edilen değerler arasındaki fark bir adaptasyon mekanizması yoluyla adaptif modele giriş olarak verilen, ölçülemeyen durum değişkeni veya parametrenin güncellenmesi amacıyla kullanılır.

MRAS kestirim yöntemi parametre değişimlerine karşı oldukça hassastır. Özellikle düşük ve sıfıra yakın rotor hızlarında sistemi kararsızlığa götürür (Bose ve Patel 1998). Ayrıca referans model açık-çevrim integrasyonlar içerir. Açık-çevrim integrasyonlar ise sürüklenme eğilimindedirler. Sürüklenme genellikle ölçülen değerlerdeki DC bileşenler ve diğer ölçüm gürültülerinden kaynaklanır. Bunun anlamı, sürüklenme ile simülasyon çalışmalarında değil, pratik uygulamalarda karşılaşılabilir, ve düzeltilmez ise kontrol sistemini karasızlığa götürür.

Yukarıdaki yöntemlerin dışında Yapay Sinir Ağı (YSA) ve bulanık mantık (Fuzzy Logic – FL) gibi yapay zeka (Artificial Inteligence – AI) yöntemleri ve Genetik Algoritmalar (GA) kestirim amacıyla kullanılmaktadır. Ancak bu yöntemler genellikle yukarıda bahsedilen diğer yöntemlerin performansını artırmak amacıyla kullanılmaktadırlar. Aşağıda bahsedilen kestirim yöntemleri ile ilgili literatürde yer alan kaynaklar özetlenmeye çalışacaktır.

Bose ve Patel'e (1998) göre yüksek rotor hızlarında parametre değişimleri sonuca ihmal edilebilir bir etki yaparken, çok düşük rotor hızlarında zıt emk'nın stator sargı direncinde düşen gerilime göre küçük olmasından dolayı bu etki ihmal edilemez düzeyde olur. Bu nedenle özellikle, stator sargı direncinin çevrim-içi olarak düzeltilmesi gerekli olur. Diğer makine parametrelerindeki değişimlerin etkileri ise ihmal edilebilecek düzeydedir. Bu soruna çözüm olarak stator sargı direncinin çevrim-içi olarak kestirimini sağlayan yarı-bulanık bir yöntem önermişledir. Önerilen yöntemde doğrudan direnç değeri kestirimi yapılmayıp, stator akımı ve frekansına dayanılarak bulanık bir yöntemle önce stator sıcaklığı kestirilerek, elde edilen sıcaklık değeri yardımıyla stator direncinin yeni değeri hesaplanmaktadır. Statoruna sıcaklık algılayıcıları bağlanmış 5 hp gücündeki bir asenkron makine üzerinde yapılan simülasyon ve deney çalışmalarında oldukça iyi sonuçlar elde edilmiştir.

Faiz ve ark. (1999) tarafından elektrikli ulaşım sistemleri için önerilen stator akım ve gerilimlerinden yararlanan başka bir yöntemde stator ve rotor dirençleri doğrudan hesaplanabilmektedir. Ancak, Bose ve Patel (1998) tarafından önerilen yarı-bulanık yönteme göre daha fazla hesap gerektirmektedir. Kubota ve ark. (1994) tarafından önerilen başka bir yöntemde yine stator direnci doğrudan hesaplanabilmektedir (Holtz 2001). Ancak bu yöntemle stator direncinin belirlenmesi 1.4 s gibi bir gecikmeyle olmaktadır.

Chao ve Liaw (2000) tarafından Model Reference Adaptive System (MRAS) rotor hız kestirme yöntemine makine parametrelerinin etkisi geniş bir şekilde incelenmiştir. Bu incelemede hız kestirme işleminin doğruluğuna en çok etki eden makine parametresinin stator etkin direnci olduğu görülmüş ve MRAS hız kestirmenin doğruluğunu artırmak için stator etkin direncini düzelterek makine modelini güncelleyen ek bir sistem önerilmiştir. Variable Structure System (VSS) adı verilen bu sistem PI regülatör çıkışından almış olduğu kestirilen hız sinyalini giriş

olarak alıp, buna göre, stator direncini düzelterek referans sisteme vermektedir. Çeşitli rotor hızlarında yapılan simülasyon çalışmalarından oldukça iyi sonuçlar alınmıştır.

Bharadwaj ve ark. (1999) tarafından asenkron makinenin hız kestirimi için Yapay Sinir Ağı (YSA) içeren bir yöntem önerilmiştir. Bu yöntem rotor oluklarının stator akımı üzerinde oluşturduğu harmoniklerin aranmasına dayanmaktadır. Stator akımı işareti YSA tabanlı bir adaptif durum filtresiyle işlenerek istenilen rotor hızı elde edilmeye çalışılmaktadır. Durum filtresi Kalman Filtresi prensiplerine dayanmaktadır. Filtrenin ayarlanması, yani YSAnın eğitilmesi için durgun durumda 12-13 veri çevrimi gerekmektedir. 3 hp gücündeki asenkron makine üzerinde, dengeli ve dengesiz güç kaynağı, normal ve kırık rotor çubukları, ve değişik yük durumları için denemeler yapılmıştır. Bu değişik durumlar için yaklaşık %0-10 arasında değişen hatalarla hız kestirimi yapılabildiği görülmektedir.

Zidan ve ark. (2000) tarafından önerilen bir yöntem ile rotor hızı doğrudan hesaplanabilmektedir. Yöntem özellikle, DSP uygulamalarına yönelik olup, makine modelini oluşturan durum denklemleri ayrık biçimde düzenlenmiştir. 0.75 kW gücündeki bir asenkron makine üzerinde yapılan denemeler ve simülasyon çalışmaları, hem 1.0 ms hem de 0.25 ms örnekleme periyotları için yapılmıştır. Bu örnekleme periyotlarında rotor akısı ve dönme hızının değişmediği kabul edilerek eşitlikler düzenlenmiştir. Çok küçük rotor hızlarında, rotor alanı açısının doğrudan hesaplanması sırasında sıfıra bölme oluşmaktadır ve önerilen yöntem ile bu durum aşılmaktadır. Hem yüksek rotor hızlarında, hem de küçük rotor hızlarında yapılan denemeler ve simülasyon çalışmaları sonucunda, önerilen yöntemin pratik uygulamalar için uygun olduğunu bildirmişlerdir.

Rotor hızının MRAS yöntemi ile kestiriminde genellikle model çıkışları olarak rotor akısı kullanılmaktadır. Mohan (2001) tarafından rotor akısının küçük değerlerde olması ve kestirme işleminin doğruluğunun az nedeniyle, rotor akısı yerine daha yüksek değerlerde olan zıt emk'nın kullanılması önerilmiştir. 3 hp gücündeki bir asenkron makine üzerinde, yol alma, yük momentinin değişmesi ve rotor bir yönde dönerken diğer yöne doğru verilen hız komutuna olan cevabı için simülasyonlar yapılmış ve pratikte uygulanabilir olduğu görülmüştür. Ancak pratik

uygulama verilmemiştir. Gerçekte düşük hızlarda zıt emk sıfıra yaklaşır. Bu yöntemin anma hızına yakın hızlarda çalışabileceği açıktır.

Ouhrouce (2002) tarafından EKF tabanlı bir kestirim yöntemi önerilmiştir. Makinenin temel frekansta üç fazlı sinüsoidal bir kaynak ile beslenmesi durumunda rotor hızının ve direncinin eş zamanlı olarak kestirilmesi mümkün değildir. Ancak, önerilen yöntem ile 5. dereceden bir makine modeli yardımıyla, her iki değişken eş zamanlı kestirilebilmektedir. Bu yöntemde asenkron makinenin modeli ayrık biçimdedir ve gürültüyü temsil eden rasgele vektörler içerir. Yapılan simülasyon çalışmalarından, yol alma süreci dışında oldukça iyi sonuçlar alınmıştır. Ayrıca, düşük hızlarda da çalışabildiği görülmüştür. Uygulama verilmemiştir.

EKF tabanlı kestirim yönteminde kullanılan makine modelinde durum değişkeni olarak genellikle stator akımları ve rotor akısı kullanılır. Her bir durum değişkeni reel ve imajiner bileşenlerden oluşur. Böylece her durum değişkeni için iki durum denklemi gerekli olur. Rotor hızının kestirimi amacıyla bir durum denklemine daha ihtiyaç duyulur. Bu nedenle 5. dereceden makine modeli kullanılmalıdır. Bunun anlamı EKF algoritmaları için *5x5* boyutunda matrisler arasında işlemlerin yerine getirilmesi gerektiğidir. Bu ise kontrolörün oldukça fazla hesaplama kapasitesine sahip olması gerektiğini gösterir.

Ge ve Feng'e (2000) göre EKF için kullanılan makine modelinin derecesi düşürülebilir. Gerçekte makine modelini oluşturan durum denklemlerinde bulunan stator akımı ölçülebilir durumdadır. Bu nedenle kestirilmesine gerek yoktur. Böylece durum denklemleri modelden çıkarılabilir. Ancak bunun sonucunda stator akımı ölçüm gürültüleri EKF tarafından filtre edilmemiş olacaktır. Sistemin uygulanabilir olduğu simülasyon ve deneysel olarak ispatlanmıştır. Uygulama tasarımında EKF algoritması için bir DSP ve vektör kontrolör için bir mikroişlemci kullanıldığı görülmektedir. İki işlemci kullanılması durumunda vektör kontrollü sürücünün maliyetinin artacağı açıktır.

Hilairet ve ark. (2000) tarafından önerilen başka bir EKF tabanlı kestirim yönteminde EKF için kullanılan makine modelinin derecesini düşürmeden hesaplama süresini kısaltacak bir yol önerilmiştir. Yöntem makine modelini oluşturan matrisin simetrik olmasına dayanır. Bu durumda matris işlemlerinde simetriden dolayı tekrarlanmaktadır. Önerilen yöntemde tekrarlanan işlemler tespit

edilerek tekrar hesaplanması önlenmektedir. Böylece işlemci hesaplama kapasitesinin verimli olarak kullanılmaktadır. Bu şekilde 503 çarpma ve 428 toplama işleminden oluşan bir EKF algoritması 205 çarpma ve 137 toplama işlemi ile yerine getirilebilmektedir.

EKF algoritmasının performansı gürültü matrislerinin uygun olarak seçilmesine bağlıdır. Gürültü matrisleri ölçümlere ve kestirimlere ne kadar güvenileceğini belirler. Sistem gürültü seviyesi ile bilgiler mevcut ise matrislerin belirlenmesi kolaydır. Ancak deneme yöntemiyle uygun matrisler seçilebilir.

Shi ve ark. (2002) tarafından rotor hızının kestirilmesi amacıyla kullanılacak bir EKF kestirim algoritmasının GA ile optimize sağlayan bir yöntem önerilmiştir. GA algoritması çevrim dışı olarak kullanılmaktadır. Önce makine test için çalıştırılarak stator akımı, stator voltajı ve rotor hızı saklanmaktadır. Daha sonra bu veriler yardımıyla EKF gürültü matrisleri oluşturulmaktadır. Elde edilen gürültü matrisleri vektör kontrolörde kullanılmakta ve EKFnin performansı incelenmektedir. Bu şekilde V/Hz ve vektör kontrol için yapılan testlerden uygun sonuçlar elde edilmiştir.

MRAS rotor hız kestirim yönteminde genellikle referans model girişleri stator akım ve voltajı, adaptif modelin girişleri ise stator akımı ve rotor hızı kullanılır. Model çıkışları ise rotor akısıdır. Adaptasyon mekanizması ile gerçekleşen akı ile kestirilen akı arasındaki fark sıfır olacak şekilde adaptif model girişi olan rotor hızını günceller. Böylece rotor hızı kestirilmiş olur.

Kojabadi ve Chang (2002) tarafından rotor hızı ve akısı ile stator direncinin kestirilmesi için farklı bir MRAS kestirim yöntemi önerilmiştir. Yöntem MRAS Adaptive Pseudo-reduced Flux Observer (APFO) olarak adlandırılmaktadır. APFO ile akı doğrudan kestirilmemektedir. Bunun yerine stator akımı kestirilmektedir. Stator akımı doğrudan ölçülebildiğinden referans modele ihtiyaç duyulmamaktadır. Daha sonra kestirilen stator akımı kullanılarak rotor akısı hesaplanmaktadır. Hesaplanmada kestirilen stator akımı kullanıldığından hesaplanan değerde rotor akısının kestirilmiş değeri olacaktır. Ancak bu kestirimler için rotor hızının bilinmesi gerekmektedir. Rotor hızının kestirimi için ayrı bir MRAS kullanılmaktadır. Bu MRAS için APFO ile kestirilmiş rotor akısı ve kestirilen stator akımı ile gerçekleşen stator akımı arasındaki hata kullanılmaktadır. Ayrıca reaktif güç tabanlı başka bir

sistem stator direncinin kestirimi amacıyla kullanılmaktadır. Sistemin kararlılığını sağlamak için Popov ve Lyapunov kriterleri kullanılmıştır. Önerilen sistem simülasyon ortamında ve deneysel olarak test edilmiş, sıfıra hızdan anma hızına kadar tüm hızlarda çalışabilir olduğu kanıtlanmıştır.

Bharadwaj ve ark. (2003) tarafından önerilen spektral analiz yönteminde EKF tabanlı bir hız filtresi YSA ile oluşturulmaya çalışılmıştır. Yöntem rotor akımlarının stator sargılarında oluşturduğu harmonik akımların filtrelenmesine dayanmaktadır. YSA ile oluşturulmak istenen filtre Parlos ve ark. (1999) tarafından EKF prensipleri kullanılarak geliştirilen adaptif durum filtresidir. YSAnın eğitilmesi için gerekli veri motor akımının 10 çevrimi kaydedilmektedir. Bu veriler daha sonra ± 0.5 aralığında ölçeklenmektedir. Filtre üç adet YSAdan oluşmaktadır. İlk YSA kestirilen hız ile stator voltajını kullanarak akım kestirimi yapar. İkinci YSA da gerçekleşen akım, kestirilen hız ve stator voltajı yardımıyla hız kestirimi yapar. Üçüncü YSA ise gerçekleşen akım ile kestirilen akım arasında oluşan hataya göre kestirilen hızı düzeltip güncellemektedir. Önerilen sistem sağlam ve kırık rotorlu motor ile test edilmiş ve %1.1 – 2.9 hata ile rotor hızının kestirilebildiği rapor edilmiştir.

Toqeer ve Bayındır (2003) tarafından önerilen başka bir YSA tabanlı sistemde Jordan-Elman Neural Network (ENN) kullanılmıştır. Sistem makine modeline dayanmaktadır. ENN hem gerçekleşen ve istenen akımlar ile birlikte stator voltajları da veri olarak kullanmaktadır. Sistemin geçerliliği yalnızca simülasyon ile test edilmiştir. Gerçekleşen hız ile kestirilen hız için verilen grafikten kestirimde yapılan hatanın küçük olduğu anlaşılmaktadır.

Guzinski ve ark. (2003) algılayıcısız vektör kontrol için oldukça farklı ve düşük maliyetli bir yöntem önermişlerdir. Makinenin üretmiş olduğu tork stator akımı ve rotor akısı arasındaki açıyla orantılıdır. Bu açının kontrolü için bir faz kilitlrme çevrimi kullanılmaktadır. Genellikle makine modelini oluşturan durum denklemlerindeki durum değişkenleri stator akımı ve rotor akısı olurken, burada faklı durum denklemleri kullanmışladır. Bu yöntem de ise sırasıyla rotor açısal hızı, stator akımı ile rotor akısının skaler ve vektörel çarpımları ve rotor akısının karesidir. Rotor hız kestirimi gözetleyici tabanlıdır. Referans olarak hızı değil tork ile orantılı olan stator akımı ile rotor akısı arasındaki açı kullanılmaktadır. Uygulanabilirliği simülasyon ortamında ve deneysel olarak kanıtlanmıştır.

Koubaa (2004) tarafından makine parametrelerinin kestirimi amacıyla kullanılacak bir yöntemde MRAS ve tekrarlamalı en az kareler (Recursive Least Square – RLS) yöntemi birlikte kullanılmaktadır. Geleneksel MRAS yöntemde adaptasyon mekanizması olarak oransal-integral (Proportional-Integral – PI) regülatör kullanılırken burada RLS kullanılmaktadır. Tekrarlamalı olması açısından EKFye benzer. Adaptif makine modeli ile referans makine modelinin çıkışlarının karşılaştırılmasından elde edilen hata yardımıyla RLS tabanlı sistem kullanılarak parametreler kestirilmekte ve adaptif model güncellenmektedir. Önerilen yöntemin uygulanabilirliği simülasyon ortamında ve deneysel olarak kanıtlanmıştır.

Beguenane ve ark. (2006) tarafından önerilen doğrudan vektör kontrol sisteminde akı kestirimi için EKF ve hız kestirimi için MRAS kullanılmaktadır. Önerilen sistem düşük rotor hızlarında sürücü performansını artırmayı amaçlar. Bunun için kestirimin doğruluğunda en büyük etkiye sahip makine parametresi olan stator sargı direnci de MRAS ile kestirilmeye çalışılmaktadır. Önerilen sistem simülasyon ortamında test edilmiştir. Ağır yük şartları altında 0-150 d/dk. arasındaki hızlarda yapılan simülasyonlarda maksimum hata %10 olmaktadır. 30 d/dk. gibi düşük hızlar civarında kestirim hatası %5'i aşmamaktadır.

Vaclavec ve Blaha (2006) tarafından algılayıcısız vektör kontrolünde hız kestirimi amacıyla kullanılmak üzere Lyapunov-Fonksiyonu tabanlı EKF durum gözetleyicisi kullanılan bir yöntem önerilmiştir. Ayrıca aynı gözetleyici ile makine parametrelerinin kestirimi amacıyla da kullanılmaktadır. Lyapunov-Fonksiyonu yardımıyla EKF kazançları güncellenmektedir. DSP tabanlı sürücü üzerinde yapılan deneysel çalışmalar sonucunda 20 d/dk gibi küçük hızlarda sistemin kararlılığını koruduğu rapor edilmiştir.

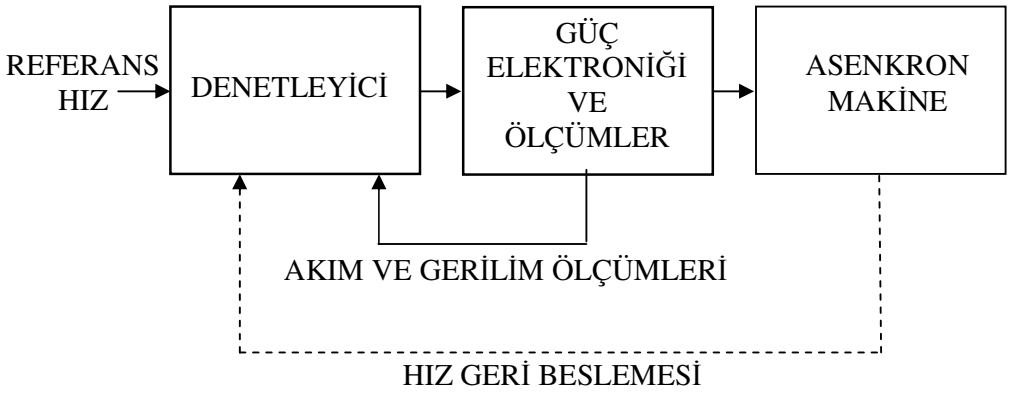
Li ve ark. (2006) tarafından asenkron makine hız kestirimi için diferansiyel-aritmetik yaklaşım kullanılmaktadır. Makine modelini oluşturan durum denklemlerinin tekrar türevi alınarak durum değişkenlerinin ikinci dereceden türevleri de kullanılmaktadır. Kararlılığın sağlanması amacıyla durum gözetleyici önerilmiştir. Yöntemin oldukça karmaşık hesaplamalar gerektirdiği anlaşılmaktadır. Yalnızca simülasyon ortamında testler yapılmıştır. 3 rad/s civarındaki rotor hızlarında sistemin çalışabildiği rapor edilmiştir. Gerçek ortamda uygulama yapılmamıştır.

3. ASENKRON MAKİNE VE KONTROL ELEMANLARI

3.1. Genel Asenkron Makine Kontrol Sistemi

Bu kısımda kontrol sistemi işlevsel bloklar şeklinde genel olarak anlatılmaya çalışılacaktır. Şekil 3.1'de görüldüğü gibi kontrol sistemi DSP denetleyici, güç elektroniği, ölçümlerin yapıldığı blok ve asenkron makineden oluşmaktadır. Rotor hızı ve konumu veya alan konumu kestirim yöntemleri ile bulunmaya çalışılacağından makineden herhangi bir mekaniksel algılayıcı yardımıyla denetleyiciye geri besleme bulunmamaktadır. Bu nedenle bu döngü kesikli çizgi ile gösterilmiştir. Asenkron makinenin referans çalışma hızı sisteme komut olarak girilmektedir.

Denetleyici bloğu DSP denetleyici içermektedir. Bu kısımda geri besleme sinyalleri ve referans hız sinyali değerlendirilerek, güç elektroniği bloğu için gerekli sinyaller üretilmektedir. Hız algılayıcısı bulunması durumunda algılayıcıdan gelen sinyaller yardımıyla, bulunmaması durumunda ise, ölçülen akım ve gerilimler yardımıyla makinenin belirli bir anda hızı bulunmaya çalışılmaktadır. Ayrıca, akım ve gerilim değerlerinin makineye zarar verecek şekilde aşılmaması için gerekli kontroller de bu blok içerisinde yapılmaktadır. Bu bloğun çıkışı PWM sinyalleri şeklindedir.



Şekil 3.1 Genel asenkron makine kontrol sistemi

Güç elektroniği ve ölçümlerin yapıldığı blok güç anahtarları ve ölçüm yükselteçleri içermektedir. Makine akım ve gerilim sinyalleri, içerdiği bilgilere zarar vermeyecek şekilde filtreleme işlemine ve denetleyici bloğundaki analog dijital dönüştürücülere uygun voltaj değerlerine koşullama işlemine tabi tutulur. Bu işlem ölçüm yükselteçleri tarafından yapılmaktadır. Güç elektroniği, IGBT içeren üç-fazlı tam-köprü PWM inverter devresinden oluşmaktadır. IGBTler yüksek anahtarlama frekansına uygun olarak seçilmişlerdir. Ayrıca, gerekli DC voltaj, bu blok içerisinde basit bir şekilde köprü doğrultucu devresi yardımıyla elde edilir. Yine AC kaynakta oluşabilecek pik şeklindeki ani yükselmelerden sistemi koruyacak bir filtre de mevcuttur. Bu bloğun çıkışı asenkron makine için gerekli değerlerdeki üç-fazlı voltaj sinyalidir.

Hız algılayıcısının bulunmadığı durumda, asenkron makinenin yalnızca önceki blokla bağlantısı vardır. Bu durumda kontrol için hem voltaj, hem de akımlar gerekli olmaktadır. Diğer durumda ise kontrol için hız sinyali ve makinenin çektiği akımlar yeterli olmaktadır.

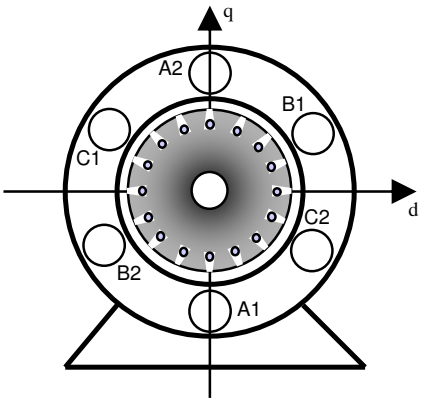
3.2. Asenkron Makine

Bir asenkron makine alternatif akım makinesidir. Sincap-kafesli rotorlu ve sargılı rotorlu olmak üzere iki asenkron makine tipi vardır. En ucuz ve en yaygın kullanılan türü sincap-kafesli asenkron makinedir. Bu tez çalışmasında kullanılan makine de üç-fazlı bir sincap-kafes asenkron makinedir. Şekil 3.2'de sincap-kafes asenkron makinenin yapısı basitleştirilmiş biçimde gösterilmiştir. Makine Şekilde görüldüğü gibi, durgun kısım olan stator ve hareketli kısım olan rotor olmak üzere iki temel parçadan oluşur. Stator, silindirik makine gövdesi içerisindeki oluklara birbirlerine göre 120° aralıklarla yerleştirilmiş üç adet sargıdan oluşur. Rotor ise uçları birer iletken halka ile birleştirilmiş sincap kafesine benzeyen bakır çubuklardan oluşur.

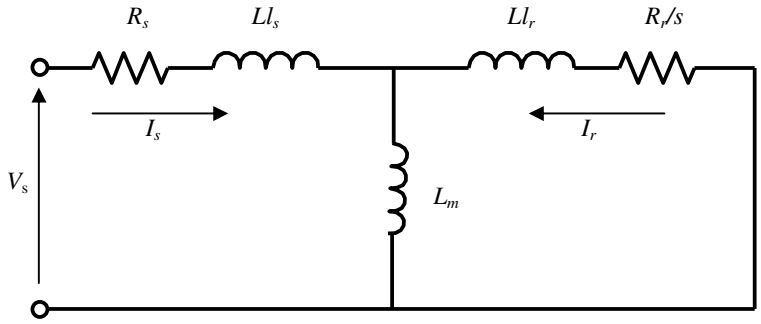
Stator sargılarına üç-fazlı sinüzoidal voltaj uygulandığında makine içerisinde döner bir manyetik alan meydana gelir. Döner alanın hızı statora uygulanan kaynağın frekansına bağlıdır ve senkron hız veya elektriksel hız olarak adlandırılır. Bu döner manyetik alan rotor çubuklarında bir gerilim indüklenmesine sebep olur. Bu durumda

rotor da yine dönen bir manyetik alan meydana getirir. Meydana gelen iki manyetik alan birbirleriyle etkileşerek rotoru döndürecek şekilde bir moment meydana getirirler. Rotor hızı senkron hıza yaklaştıkça, stator manyetik alanının rotor çubuklarını kesme hızı azalacağından, çubuklarda indüklenen gerilim de azalacaktır. Bu durumda döndürme momenti de azalır. Pratikte hiçbir zaman rotor hızı senkron hıza ulaşamaz. Makine, asenkron ismini de buradan alır. Rotor, yük ve diğer rotor kayıplarını karşılayacak bir kayma ile dönmelidir. Burada kayma senkron hız ile rotor mekanik hızı arasındaki fark anlamındadır.

Asenkron makine hem elektriksel, hem de mekanik bir cihazdır. Analiz ve tasarım için pratikte, Şekil 3.3'te verilen bir-faz eşdeğer devreleri kullanılır. Bu devreler, makinenin durgun durumdaki özelliklerini içerirler. Ancak vektör kontrolüne uygun değillerdir. Vektör kontrol için uygun bir makine modeli sonraki kısımlarda incelenecektir. Ayrıca bu tez çalışmasında kullanılan makinenin değerleri Tablo 3.1'de verilmiştir.



Şekil 3.2 Sincap-kafes asenkron makinenin şematik gösterimi



Şekil 3.3 Sincap-kafes asenkron makine durgun-durum bir-faz eşdeğer devresi

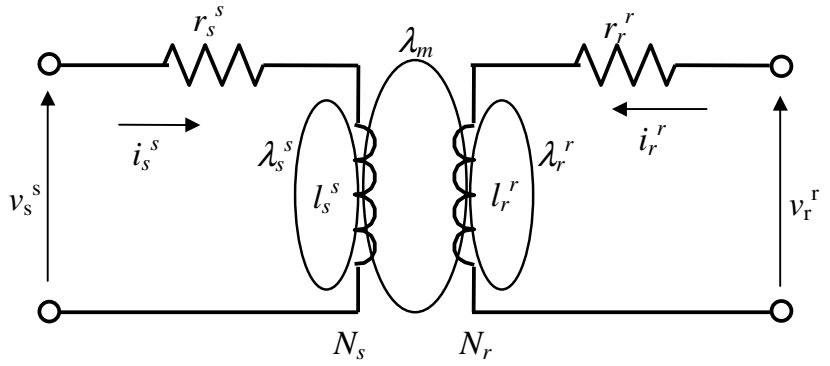
Tablo 3.1 Tez projesinde kullanılan asenkron makinenin değerleri

Makine Anma Değerleri	
Anma gücü	P_n = 750 W
Anma voltajı	V_n = 220 Vrms (faz)
Anma Akımı	I_n = 1.9 Arms
Anma frekansı	fn = 50 Hz
Anma hızı	n = 1395 d/dak.
Kutup sayısı	p = 4
Kayma	s = 0.07
Anma Momenti	T_n = 5.134 N·m
Eşdeğer Devre Parametreleri	
Stator direnci	r_s = 9.9 Ω
Rotor direnci	r_r = 12.345 Ω
Ortak indüktans	l_m = 0.590 H
Stator indüktansı	l_s = 0.661 H
Rotor indüktansı	l_r = 0.661 H

3.2.1 Dinamik Makine Modeli

Daha önce vektör kontrol için asenkron makinenin dinamik modelinin gerekli olduğundan bahsedilmişti. Bu bölümde asenkron makine dinamik modeli oluşturulmaya çalışılacaktır.

Şekil 3.4'te asenkron makinenin transformatöre benzeyen eşdeğer devresi verilmiştir. Üst indisler elemanın hangi referans yapıya göre değerlendirildiğini göstermektedir. Şekil 3.4'e göre stator sargılarından geçen toplam akı stator referans yapıya göre,



Şekil 3.4 Sincap-kafes asenkron makine bir-faz dinamik eşdeğer devresi

$$\lambda_s^s = \lambda_{ls}^s + \lambda_m^s \tag{3.1}$$

şeklinde yazılabilir. Burada üst indis "*s*" stator referans yapıyı, eşitliğin sol tarafındaki terim stator akısını, eşitliğin sağ tarafındaki birinci terim stator kaçak akısını, ikinci terim ortak akıyı temsil etmektedir. Benzer şekilde rotor sargılarından geçen toplam akı da,

$$\lambda_r^s = \lambda_{lr}^s + \lambda_m^s \tag{3.2}$$

şeklinde yazılabilir. Burada eşitliğin sol tarafındaki terim rotor akısını, eşitliğin sağ tarafındaki birinci terim ise rotor kaçak akısını temsil eder. Karşılıklı indüktansın l_m^s olması halinde ortak akı,

$$\lambda_m^s = l_m^s \cdot i_s^s + l_m^s \cdot i_r^s \tag{3.3}$$

burada i_s^s ve ve i_r^s sırasıyla stator referans yapıya göre stator ve rotor sargılarından geçen akımlardır. Kaçak akılar ise sırasıyla,

$$\left.\begin{aligned} \lambda_{ls}^s &= l_{ls}^s \cdot i_s^s \\ \lambda_{ir}^s &= l_{lr}^s \cdot i_r^s \end{aligned}\right\} \tag{3.4}$$

burada l_{ls} ve l_{lr} sırasıyla stator referans yapıya göre stator ve rotor kaçak indüktanslarıdır. Denklem 3.3 ve denklem 3.4 denklem 3.1 ve denklem 3.2'de yerlerine yazılırsa stator akısı,

$$\begin{aligned} \lambda_s^s &= l_{ls}^s \cdot i_s^s + l_m^s \cdot i_s^s + l_m^s \cdot i_r^s \\ \lambda_s^s &= l_s^s \cdot i_s^s + l_m^s \cdot i_r^s \end{aligned} \tag{3.5}$$

benzer şekilde rotor akısı

$$\lambda_r^s = l_m^s \cdot i_s^s + l_r^s \cdot i_r^s \tag{3.6}$$

biçiminde elde edilir. Burada $l_s^s = l_{ls}^s + l_m^s$ ve $l_r^s = l_{lr}^s + l_m^s$ olduğu açıktır. Basitlik açısından tüm elemanlar statora göre ifade edildiğinden devre parametrelerindeki üst indis "*s*" denklemlerde gösterilmezse son iki eşitlik,

$$\left.\begin{aligned} \lambda_s^s &= l_s \cdot i_s^s + l_m \cdot i_r^s \\ \lambda_r^s &= l_m \cdot i_s^s + l_r \cdot i_r^s \end{aligned}\right\} \tag{3.7}$$

şeklinde gösterilebilir. Böylece kendi referans yapılarına göre Şekil 3.4'deki devrenin voltaj denklemleri,

$$\left.\begin{aligned} v_s^s &= r_s \cdot i_s^s + \frac{d}{dt}\lambda_s^s \\ v_r^r &= r_r^r \cdot i_r^r + \frac{d}{dt}\lambda_r^r \end{aligned}\right\} \tag{3.8}$$

şeklinde yazılabilir. Burada üst indis '*r*' rotor referans yapıyı temsil eder. Öncelikle stator referans yapı denklemleri oluşturulursa denklem 3.8'deki birinci eşitlikle uğraşmaya gerek kalmaz. Ancak ikinci denklemin stator referans yapıya aktarılması gerekir. Şekil 3.4'te stator ve rotor sarım sayıları sırasıyla N_s ve N_r için sarım oranı,

$$\nu = \frac{N_s}{N_r} \tag{3.9}$$

olsun. Transformatörlerde bir empedansın iki farklı referans yapıdaki değerlerinin oranı dönüşüm oranının karesi ile aynıdır. Buna göre rotor direnci için,

$$r_r^r = \frac{1}{\nu^2} r_r \tag{3.10}$$

yazılabilir. Rotorun belirli bir anda statordan θ_r kadar ileride olduğunu kabul edelim. Ayrıca,

$$\frac{d\theta_r}{dt} = \omega_r \tag{3.11}$$

olsun. Burada ω_r rotor açısal hızıdır. Statordan θ_r kadar ileride olan bir referans yapıdaki vektörleri statora aktarmak için bu vektörlerin $e^{j\theta_r}$ ile çarpılması yeterlidir. Buna göre sarım oranı ilişkisi de kullanılarak akım, voltaj ve akı vektörleri rotor referans yapıdan stator referans yapıya,

$$\left.\begin{aligned} i_r^s &= \frac{e^{j\theta_r}}{\nu} i_r^r \\ v_r^s &= \nu \cdot e^{j\theta_r} \cdot v_r^r \\ \lambda_r^s &= \nu \cdot e^{j\theta_r} \cdot \lambda_r^r \end{aligned}\right\} \tag{3.12}$$

biçiminde aktarılabilir. Denklem 3.10 ve denklem 3.12, denklem 3.8'de yerine yazılırsa,

$$v_r^s = r_r \cdot i_r^s + \frac{d}{dt}\lambda_r^s - j\omega_r \lambda_r^s \tag{3.13}$$

elde edilir. Böylece voltaj eşitlikleri,

$$\left.\begin{aligned} v_s^s &= r_s \cdot i_s^s + \frac{d}{dt}\lambda_s^s \\ v_r^r &= r_r \cdot i_r^s + \frac{d}{dt}\lambda_r^s - j\omega_r \lambda_r^s \end{aligned}\right\} \tag{3.14}$$

biçiminde tüm değerler stator referans yapıda olmak üzere yeniden yazılabilir. Şimdi işlemleri bir adım daha ileriye götürüp eşitlikleri herhangi bir referans yapıda ifade etmeye çalışalım. Herhangi bir referans yapının herhangi bir anda durgun stator ile yaptığı açı θ_k olsun, ve açısal hızı,

$$\omega_k = \frac{d\theta_k}{dt} \tag{3.15}$$

olsun. Daha önce rotor referans yapıdan stator referans yapıya yapılan vektör dönüşümlerine benzer şekilde statordaki bir A^s vektörü için,

$$A^s = A^k \cdot e^{j\theta_k} \tag{3.16}$$

yazılabilir. Böylece stator referans yapıdaki stator ve rotor voltaj eşitlikleri statordan θ_k kadar ileride olan herhangi bir referans yapıda,

$$\left.\begin{aligned} v_s^k &= r_s \cdot i_s^k + \frac{d}{dt}\lambda_s^k + j\omega_k \lambda_s^k \\ v_r^k &= r_r \cdot i_r^k + \frac{d}{dt}\lambda_r^k + j(\omega_k - \omega_r)\lambda_r^k \end{aligned}\right\} \tag{3.17}$$

biçiminde ifade edilebilir. Ayrıca denklem 3.7 ile verilen akı eşitlikleri de benzer şekilde statordan θ_k kadar ileride olan herhangi bir referans yapıda,

$$\left.\begin{aligned} \lambda_s^k &= l_s \cdot i_s^k + l_m \cdot i_r^k \\ \lambda_r^k &= l_m \cdot i_s^k + l_r \cdot i_r^k \end{aligned}\right\} \tag{3.18}$$

şeklinde ifade edilebilir. Denklem 3.18'den i_s^k ve i_r^k çözülürse,

$$\left.\begin{aligned} i_s^k &= \left(l_s l_r - l_m^2\right)^{-1}\left(l_r \lambda_s^k - l_m \cdot \lambda_r^k\right) \\ i_r^k &= \left(l_m^2 - l_s l_r\right)^{-1}\left(l_m \lambda_s^k - l_s \cdot \lambda_r^k\right) \end{aligned}\right\} \tag{3.19}$$

eşitlikleri elde edilir. Denklem 3.17, 3.18 ve 3.19 makinenin elektriksel modelini oluşturur. Ayrıca sincap kafes durumunda rotor kısa devre olduğundan $v_r^k = 0$ alınmalıdır.

Şimdi tam makine modelini elde etmek için tork denkleminden ω_r'nin bulunması gereklidir. Genel olarak makinenin üretmiş olduğu tork,

$$T_e = \frac{3}{2}\frac{p}{2}\frac{l_m}{l_r}\,\mathrm{Im}\left(i_s^k \cdot \lambda_r^{k^*}\right) \tag{3.20}$$

ile verilir. Burada p kutup sayısı, ve üst indis "*" vektörün kompleks eşleniğini temsil eder. Rotor mekaniksel hızı ise,

$$J\frac{d}{dt}\omega_{rm} = T_e - T_L \tag{3.21}$$

eşitliğinden bulunabilir. Burada J atalet momentini, T_L yük torkunu, ω_{rm} rotor mekaniksel açısal hızını temsil eder. $\omega_{rm} = (2/p)\cdot \omega_r$ yazılırsa,

$$\frac{d}{dt}\omega_r = \tfrac{1}{J}\tfrac{p}{2}\left(T_e - T_L\right) \tag{3.22}$$

şeklinde rotor elektriksel açısal hızı bulunabilir. Denklem 3.20,21,22 makinenin mekaniksel modelini oluşturur.

Bu denklemler makinenin herhangi bir referans yapıda modellenebilmesi için yeterlidir. Stator referans yapı için $\omega_k = 0$, rotor referans yapı için $\omega_k = \omega_r$ konulması yeterli olacaktır. Ayrıca dönüşüm sırasında stator referans yapı için $s \rightarrow k$ ve rotor referans yapı için $r \rightarrow k$ yazılmalıdır. Akım, gerilim ve akıların dönüşümü için gerekli eşitlikler denklem 3.12'de verilmiştir.

3.2.2 Birim Sürekli-Zaman Dinamik Makine Modeli

Bu kısımda makinenin birim dinamik modeli oluşturulmaya çalışılacaktır. Önceki kısımda elde edilen makine modeli eşitlikleri simülasyon yazılımı için uygun olsalar bile, DSP yazılımında kullanılmaları mümkün değildir. DSP içerisinde ondalık sayılar doğrudan gösterilememektedir. Bu projede kullanılan DSP sabit-noktalı olarak isimlendirilmektedir. Sayılar basitçe 4.12 formatında gösterilmekte olup, en soldaki ilk bit sayının işaretini, sonraki 3 bit sayının tam kısmını, ve kalan 12 bit ondalık kısmını göstermek amacıyla kullanılmaktadır. Bu durumda yazılabilecek sayılar (8 -8] aralığında kalmaktadır. Böylece bu aralık dışında değer alan makine değişkenlerini bu sistemde ifade edilmeleri mümkün olmayacaktır. Bu durumu aşabilmek için birim makine modelinden yararlanılacaktır. Birim makine modelin basitçe her bir makine değişkeni kendi anma değerine bölünerek ölçeklendirilmektedir, ve böylece değişkenin alacağı değerler 1.0 civarında olacaktır.

Öncelikle her bir makine değişkenini uzay vektörü olarak kabul edelim, ve her bir değişkeni Şekil 3.5'te gösterildiği gibi "*d*" ve "*q*" bileşenlerine ayıralım. Burada "*d*" reel eksen "*q*" imajiner ekseni temsil etmektedir. Bu durumda bir **A** vektörü dik koordinatlar sisteminde,

$$A^k = A_d^k + j \cdot A_q^k \tag{3.23}$$

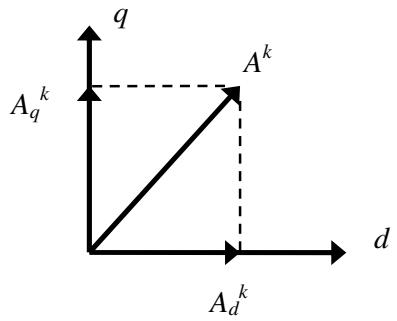
biçiminde gösterilebilir. Verilen eşitlikte alt indis "*d*" ve "*q*" sırasıyla vektörün reel ve imajiner bileşenini, üst indis "*k*" referans yapıyı temsil eder.

Bu açıklamalara göre denklem 3.17-3.18-3.20-3.22'de verilen eşitliklerden stator akımı, rotor akısı, ve rotor elektriksel açısal hızı çözülecek olursa stator referans yapı için,

$$\left.\begin{aligned}
\frac{d}{dt} i_s^s &= \left(\frac{r_s}{\sigma l_s} - \frac{l_m^2}{\sigma l_s l_r \tau_r} \right) i_s^s + \frac{l_m}{\sigma l_s l_r \tau_r} \lambda_r^s - j \frac{l_m}{\sigma l_s l_r \tau_r} \omega_r \lambda_r^s \\
\frac{d}{dt} \lambda_r^s &= \frac{lm}{\tau_r} i_s^s - \frac{1}{\tau_r} \lambda_r + j \omega_r \lambda_r^s \\
\frac{d}{dt} \omega_r &= \frac{p}{2} \frac{1}{j} \left(\frac{3}{2} \frac{p}{2} \frac{l_m}{l_r} \mathrm{Im}\left(i_s^s \cdot \lambda_r^{s*} \right) - T_L \right)
\end{aligned}\right\} \tag{3.24}$$

biçiminde sürekli-zaman dinamik denklemleri elde edilir. Burada $\sigma = 1 - l_m^2/(l_s l_r)$, ve $\tau_r = l_r/r_r$ rotor zaman sabitidir. Bu denklemler simülasyon yazılımı için yeterlidir. Ancak denetleyici için birim eşitlikler bulunmalıdır. Tasarlanan sistemin hem simülasyon hem de gerçekleştirme aşamalarında birim sistem kullanılacaktır.

Şimdi birim sistemi oluşturacak olan temel değerler belirlenecektir. Temel değerler,



Şekil 3.5 "*k*" referans yapısındaki bir vektörün bu referans yapıya göre bileşenleri

$$\left.\begin{aligned} V_b &= \sqrt{2}\cdot V_n \\ I_b &= \sqrt{2}\cdot I_n \\ \omega_b &= 2\pi\cdot f_n \end{aligned}\right\} \tag{3.25}$$

biçiminde tanımlanır. Bu temel değerlerden,

$$\left.\begin{aligned} \lambda_b &= \frac{V_b}{\omega_b} \\ Z_b &= \frac{V_b}{I_b} \\ T_b &= \frac{3}{2}\frac{V_b I_b}{\omega_b} \end{aligned}\right\} \tag{3.26}$$

biçiminde türetilmiş temel değerler elde edilir. Buna göre değişkenler birim olarak,

$$\left.\begin{aligned} i &= \frac{I}{I_b} \\ v &= \frac{V}{V_b} \\ \lambda &= \frac{\lambda}{\lambda_b} \\ \omega &= \frac{\omega}{\omega_b} \\ T &= \frac{T}{T_b} \end{aligned}\right\} \tag{3.27}$$

biçiminde birim olarak ifade edilebilir. Ayrıca denklemlerde bulunan sabitlerin de birim sisteme göre,

$$\left.\begin{aligned} r_s &= \frac{r_s}{Z_b} \\ r_r &= \frac{r_r}{Z_b} \\ x_s &= \frac{\omega_b l_s}{Z_b} \\ x_r &= \frac{\omega_b l_r}{Z_b} \\ x_m &= \frac{\omega_b l_m}{Z_b} \\ x_{sig} &= \frac{\omega_b \sigma l_s}{Z_b} \end{aligned}\right\} \tag{3.28}$$

biçiminde güncellenmelidir. Böylece, denklem 3.24 birim sistemde stator referans yapıda "*d*" ve "*q*" bileşenlerine ayrıldığında,

$$\left.\begin{aligned}
\frac{di_{sd}}{dt} &= -\frac{i_{sd}}{T_{S2}} + \frac{x_m}{x_{sig}x_r}\frac{\lambda_{rd}}{T_r} + \frac{x_m}{x_{sig}x_r}\omega_b\omega\lambda_{rq} + \frac{\omega_b}{x_{sig}}v_{sd} \\
\frac{di_{sq}}{dt} &= -\frac{i_{sq}}{T_{S2}} + \frac{x_m}{x_{sig}x_r}\omega_b\omega\lambda_{rd} + \frac{x_m}{x_{sig}x_r}\frac{\lambda_{rq}}{T_r} + \frac{\omega_b}{x_{sig}}v_{sq} \\
\frac{d\lambda_{rd}}{dt} &= \frac{x_m}{T_r}i_{sd} - \frac{\lambda_{rd}}{T_r} - \omega_b\omega\lambda_{rq} \\
\frac{d\lambda_{rq}}{dt} &= \frac{x_m}{T_r}i_{sq} + \omega_b\omega\lambda_{rd} - \frac{\lambda_{rq}}{T_r} \\
\frac{d\omega}{dt} &= {}^{p}\!/_{2}\,\frac{x_m}{T_m x_r}\left(i_{sq}\lambda_{rd} - i_{sd}\lambda_{rq}\right) - \frac{T_L}{T_m}
\end{aligned}\right\} \qquad (3.29)$$

biçiminde elde edilir. Burada, basitlik açısından, bütün değişkenlerin stator referans yapıda olması nedeniyle üst indisler yazılmamıştır. Ayrıca, burada gösterimi basitleştirmek amacıyla,

$$\left.\begin{aligned}
T_{S2} &= \left(\frac{r_s\omega_b}{x_{sig}} + \frac{x_m{}^2}{x_{sig}x_rT_r}\right)^{-1} \\
T_r &= \frac{x_r}{r_r\omega_b} \\
T_m &= \frac{J\omega_b}{{}^{p}\!/_{2}\,T_b}
\end{aligned}\right\} \qquad (3.30)$$

şeklinde kısaltmalar yapılmıştır.

3.2.3 Sürekli ve Ayrık Zaman Birim Durum Uzayı Makine Modeli

Bilindiği gibi sistem teorisinde, sistemin karakteristiklerinin belirlenebilmesi için sistem durum denklemleri gerekli olur. Bunun için, sistemdeki bazı değişkenler durum değişkeni olarak kabul edilir ve sistem bir matris eşitliği şeklinde ifade edilir. Bu kısımda makine için birim sürekli-zaman, denetleyici için birim ayrık zaman durum uzayı oluşturulmaya çalışılacaktır.

Denklem 3.29'daki asenkron makine modelini oluşturan eşitlikler incelendiğinde sistem değişkenlerinin çarpımı şeklinde terimler içerdiği görülür. Bu durumda sisteme non-lineer sistem denir. Değişkenlerin çarpımı şeklinde terimler yok ise lineer sistem olur. Bir non-lineer durum uzayı,

$$\left.\begin{aligned}\frac{dx}{dt} &= g\left[x(t),u(t),t\right]\\ y(t) &= Cx(t)\end{aligned}\right\} \tag{3.31}$$

biçiminde ifade edilebilir. Burada, *x* durum matrisi, *y* çıkış matrisi ve *u* giriştir. *C* ise katsayı matrisidir. Asenkron makine için bu matrisler,

$$\left.\begin{aligned}u &= \begin{pmatrix} v_{sd} & v_{sq}\end{pmatrix}^T\\ x &= \begin{pmatrix} i_{sd} & i_{sq} & \lambda_{rd} & \lambda_{rq} & \omega\end{pmatrix}^T\\ y &= \begin{pmatrix} x_1 & x_2\end{pmatrix}\end{aligned}\right\} \tag{3.32}$$

şeklinde seçilebilir. Böylece, makine sürekli-zaman birim durum uzayı eşitlikleri,

$$\left.\begin{aligned}\frac{dx_1}{dt} &= -\frac{x_1}{T_{S2}} + \frac{x_m}{x_{sig}x_r}\frac{x_3}{T_r} + \frac{x_m}{x_{sig}x_r}\omega_b x_4 x_5 + \frac{\omega_b}{x_{sig}}u_1\\ \frac{dx_2}{dt} &= -\frac{x_2}{T_{S2}} + \frac{x_m}{x_{sig}x_r}\omega_b x_3 x_5 + \frac{x_m}{x_{sig}x_r}\frac{x_4}{T_r} + \frac{\omega_b}{x_{sig}}u_2\\ \frac{dx_3}{dt} &= \frac{x_m}{T_r}x_1 - \frac{x_3}{T_r} - \omega_b x_4 x_5\\ \frac{dx_4}{dt} &= \frac{x_m}{T_r}x_2 + \omega_b x_3 x_5 - \frac{x_4}{T_r}\\ \frac{dx_5}{dt} &= p/2\,\frac{x_m}{T_m x_r}\left(x_2 x_3 - x_1 x_4\right) - \frac{T_L}{T_m}\\ y_1 &= x_1\\ y_2 &= x_2\end{aligned}\right\} \tag{3.33}$$

şeklinde yazılabilir.

Ayrık-zaman birim durum uzayını elde etmek için türevli terimlerin belirli bir örnekleme zamanında integrasyona tabi tutulmaları gerekir. Buna göre, trapezoidal integrasyon metodu için,

$$\frac{dx(t)}{dt} = \frac{x[(k+1)T] - x[kT]}{T} \tag{3.34}$$

eşitliği kullanılabilir. Burada, *T* örnekleme zamanını, (*k*+1) şimdiki örnekleme zamanını ve *k* ise bir önceki örnekleme zamanını temsil eder. Böylece ayrık-zaman durum uzayı,

$$\left.\begin{aligned}x(k+1) &= f\left[x(k),u(k),k\right]\\ y(k) &= Cx(k)\end{aligned}\right\} \tag{3.34}$$

şeklinde yazılabilir. Asenkron makine için ayrık-zaman birim durum uzayı değişkenleri,

$$\left.\begin{aligned} i_{sd}(k+1) &= f_1[x(k),u(k),k] \\ i_{sq}(k+1) &= f_2[x(k),u(k),k] \\ \lambda_{rd}(k+1) &= f_3[x(k),u(k),k] \\ \lambda_{rq}(k+1) &= f_4[x(k),u(k),k] \\ \omega(k+1) &= f_5[x(k),u(k),k] \\ i_{sd}(k) &= i_{sd}(k) \\ i_{sq}(k) &= i_{sq}(k) \end{aligned}\right\} \tag{3.35}$$

şeklinde seçilebilir. Böylece, ayrık-zaman birim durum uzayı eşitlikleri,

$$\begin{cases} f_1 = (1-\dfrac{T}{T_{S2}})x_1(k) + \dfrac{x_m}{x_{sig}x_r}\dfrac{T}{T_r}x_3(k) + \dfrac{x_m T\omega_b}{x_{sig}x_r}x_4(k)x_5(k) + \dfrac{\omega_b T}{x_{sig}}u_1(k) \\ f_2 = (1-\dfrac{T}{T_{S2}})x_2(k) - \dfrac{x_m T\omega_b}{x_{sig}x_r}x_3(k)x_5(k) + \dfrac{x_m}{x_{sig}x_r}\dfrac{T}{T_r}x_4(k) + \dfrac{\omega_b T}{x_{sig}}u_2(k) \\ f_3 = \dfrac{x_m T}{T_r}x_1(k) + (1-\dfrac{T}{T_r})x_3(k) - \omega_b T x_4(k)x_5(k) \\ f_3 = \dfrac{x_m T}{T_r}x_2(k) + \omega_b T x_3(k)x_5(k) + (1-\dfrac{T}{T_r})x_4(k) \\ f_5 = x_5(k) \\ y_1(k) = x_1(k) \\ y_2(k) = x_2(k) \end{cases} \tag{3.36}$$

şeklinde yazılır.

Bu kısımda oluşturulan denklem 3.33 simülasyon aşamasında makine modellenmesinde ve denklem 3.36 hem simülasyon aşamasında, hem de DSP yazılımında kontrolörün modellenmesi ve gerçeklenmesi amacıyla kullanılacaktır.

3.3. Alan Yönlendirme Kontrolü

Alan yönlendirme kontrolü asenkron makine için yabancı uyartımlı DC makine yaklaşımıdır. Bu kısımda öncelikle kısaca yabancı uyartımlı DC makinenin kontrolünden bahsedilecek ve aynı tür bir kontrolün asenkron makineye nasıl uygulanacağı açıklanmaya çalışılacaktır.

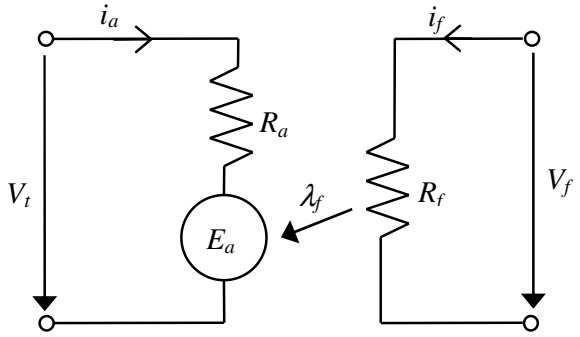
Bilindiği gibi yabancı uyartımlı DC makinenin hız ve tork kontrolü asenkron makineye göre oldukça basittir. Makinenin üretmiş olduğu tork,

$$T_e = K_t \cdot i_a \cdot \lambda_f \tag{3.37}$$

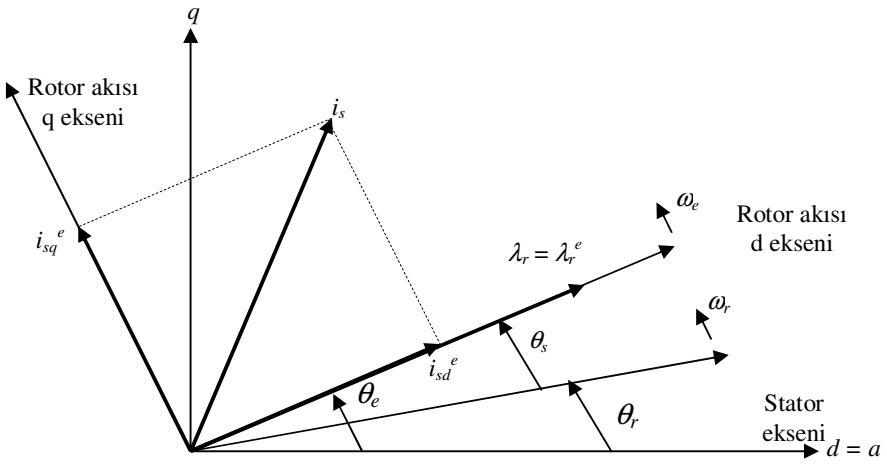
şeklinde bulunur. Burada K_t tork sabiti, i_a armatür akımı, ve λ_f alan akısıdır. Şekil 3.6'dan anlaşılabileceği gibi armatür akımı armatüre bağlı kaynak ile ve alan akısı da alan kaynağı ile doğrudan ve birbirinden bağımsız olarak değiştirilebilir. Böylece, yabancı uyartımlı DC makine istenilen herhangi bir hızda istenilen bir yükü sürebilir. Bunun anlamı, istenilen herhangi bir torkun her hızda üretilmesinin mümkün olacağıdır.

Asenkron makinede üretilen tork ise Denklem 3.20'den de anlaşılabileceği gibi DC makineye benzer şekilde rotor akısı ve stator akımının çarpımı ile orantılıdır. Ancak asenkron makine durumunda bu iki değişkeni birbirinden bağımsız olarak kontrol edebilecek iki ayrı sargı mevcut değildir. Bu durumda, Şekil 3.7'de olduğu gibi, stator akımı, bir tanesi rotor akısı doğrultusunda ve diğeri rotor akısına dik doğrultuda olmak üzere iki bileşene ayrılır. Tarif edilen bu doğrultular senkron referans yapıyı oluşturur ve bu referans yapı sürekli olarak rotor akısı ile birlikte döner. Stator akımının rotor akısı doğrultusundaki bileşenine alan oluşturan bileşen denir ve DC makinede alan sargısı akımına karşılık gelir. Stator akımının diğer bileşenine tork oluşturan bileşen denir ve DC makinede armatür sargısı akımına karşılık gelir.

Alan yönlendirme kontrolü veya vektör kontrol bu iki bileşenin kontrolüne dayanır. Tabii ki, bunların kontrolü için büyüklüklerinin ve yönlerinin kesin olarak bilinmesi gereklidir. Bu değişkenlerden stator akımının ölçülmesi, rotor akısının ölçülmesine göre oldukça basittir. Stator akımı makine terminallerinden ölçülebilir. Ancak, rotor akısını ölçmek için makine içerisine alan sensörleri yerleştirmek gerekli olur. Bu da standart makinelerde mevcut değildir ve sürücü maliyetini artırır.



Şekil 3.6 Yabancı uyartımlı DC makine eşdeğer devresi



Şekil 3.7 Alan yönlendirme kontrolü için koordinat sistemleri ve vektörler

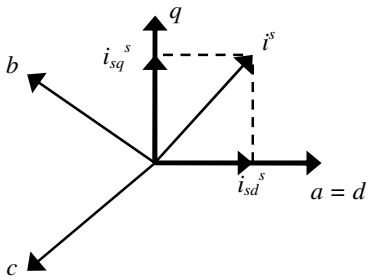
Rotor akısının büyüklüğü ve yönü önceki kısımda geniş bir şekilde açıklanan dinamik makine modelinden hesaplanabilir. Ancak, bu durumda rotor hızının bilinmesi gerekli olur. Rotor hızı iki yöntemle belirlenebilir. Birinci yöntem, rotor hızının doğrudan makine milinden ölçülmesidir. Bu yöntemde hız algılayan ek bir eleman gerektiğinden sürücü maliyeti artacaktır. İkinci yöntem, rotor hızının ölçülebilen diğer değişkenler yardımıyla hesaplanması veya kestirilmesidir. Bu durumda ek bir eleman gerekli olmadığından sürücü maliyetinde bu nedenle bir artış olmayacaktır. Ancak birinci yönteme göre, güvenilirliği daha az olacaktır. Bu tez çalışmasında ikinci yöntem kullanılacak ve işleyişi kısım 3.5'de açıklanmaya çalışılacaktır. Bu bölümün sonraki kısımlarında alan yönlendirme kontrolünün işleyişinden bahsedilecektir.

Öncelikle, alan yönlendirme kontrolünde daha önce elde edilmiş olan eşitliklerden yararlanabilmek için, üç-faz için ölçülen akım ve gerilimin tek bir vektör ile temsil edilmeleri gereklidir. Şekil 3.8 dikkate alındığında birbirlerinden 120°'lik açılarla ayrılmış "*a*", "*b*" ve "*c*" doğrultularında yönlenmiş olan üç adet stator akım vektörü stator referans yapıda,

$$i_s^s = i_{sa}^s + i_{sb}^s \cdot e^{j2\pi/3} + i_{sc}^s \cdot e^{j-2\pi/3} \tag{3.38}$$

biçiminde gösterilebilir. Elde edilen bu stator akım vektörü Şekil 3.8'deki gibi,

$$i_s^s = i_{sd}^s + j \cdot i_{sq}^s \tag{3.39}$$



Şekil 3.8 Stator akımının stator referans yapıya göre bileşenleri

"*d*" ve "*q*" bileşenlerine ayrılabilir. Bu bileşenler denklem 3.38, ve 3.39 eşitliklerinden elde edilen,

$$\left.\begin{aligned} i_{sd}^{s} &= i_{sa}^{s} \\ i_{sq}^{s} &= \left(i_{sa}^{s} + 2\cdot i_{sb}^{s}\right)\Big/\sqrt{3} \end{aligned}\right\} \tag{3.40}$$

eşitlikleri ile basitçe hesaplanabilir. Sonraki aşamada, elde edilen akım vektörünün rotor alanı yönündeki ve bu alana dik bileşenlerinin bulunmasıdır. Bunu yapmak için stator referans yapıdaki vektör senkron referans yapıya aktarılmalıdır. Senkron referans yapı rotor alanı doğrultusunu yatay eksen kabul eden referans yapı anlamında kullanılmıştır. Belirli bir anda rotor alanı durgun stator referans yapıdan θ_e kadar ilerde ise stator referans yapıdaki akım vektörü senkron referans yapıya,

$$i_s^e = i_s^s \cdot e^{j\theta_e} \tag{3.41}$$

eşitliği yardımıyla aktarılabilir. Denklem 3.41 eşitliğine Euler formülünün uygulanması ile stator referans yapıdaki akım vektörünün senkron referans yapıdaki "*d*" ve "*q*" bileşenleri,

$$\left.\begin{aligned} i_{sd}^{e} &= i_{sd}^{s}\cdot\cos\theta_e + i_{sq}^{s}\cdot\sin\theta_e \\ i_{sq}^{e} &= -i_{sd}^{s}\cdot\sin\theta_e + i_{sq}^{s}\cdot\cos\theta_e \end{aligned}\right\} \tag{3.42}$$

eşitlikleri ile bulunabilir. Denklem 3.41'deki birinci eşitlik alan üreten bileşeni, ikinci eşitlik tork üreten bileşeni temsil eder. Denklem 3.40 ve denklem 3.42 ile verilen dönüşümlerin tersleri de mevcuttur. Ayrıca, burada yalnızca akım vektörü için verilen bu eşitlikler gerilim ve akı vektörleri içinde uygulabilir.

Buraya kadar stator akım vektörünün senkron referans yapıdaki bileşenleri bulunmuş oldu. Bundan sonra, şimdiye kadar yapılanların tersine bir yol izlenecektir.

Senkron referans yapı üzerinde, referans bir akıya göre seçilmiş bir stator akımı "*d*" bileşeni ve referans bir torka göre seçilmiş bir stator akımı "*q*" bileşeni var ise, öncelikle bu referans tork ve akı için seçilen stator akım bileşenlerinin stator referans yapıya aktarılmaları gereklidir. Bu seçimlerin yapılışı bu kısmın içerisinde daha sonra açıklanacaktır. Denklem 3.41 eşitliklerinden bu akımlar,

$$\left.\begin{aligned} i_{sd}^{s*} &= i_{sd}^{e*}\cdot\cos\theta_e - i_{sq}^{e*}\cdot\sin\theta_e \\ i_{sq}^{s*} &= i_{sd}^{e*}\cdot\sin\theta_e + i_{sq}^{e*}\cdot\cos\theta_e \end{aligned}\right\} \tag{3.43}$$

biçiminde hesaplanabilir. Burada, üst indis "*" referans büyüklüğü temsil eder. Elde edilen bu referans akımlar, denklem 3.40 eşitlikleri yardımıyla üç-fazlı sisteme,

$$\left.\begin{aligned} i_{sa}^{s*} &= i_{sd}^{s*} \\ i_{sb}^{s*} &= \left(-i_{sd}^{s*} + \sqrt{3}\cdot i_{sq}^{s*}\right)\Big/ 2 \\ i_{sc}^{s*} &= -i_{sa}^{s*} - i_{sb}^{s*} \end{aligned}\right\} \tag{3.44}$$

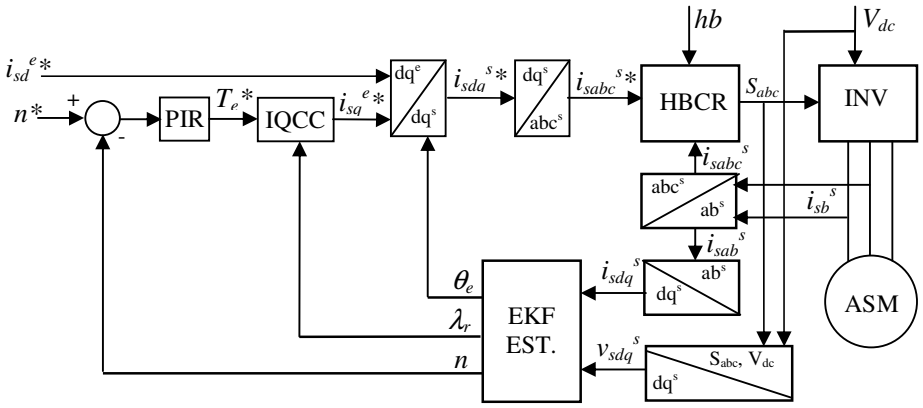
biçiminde aktarılabilir. Böylece, üç-faz için belirlenen referans akımlar, gerçek akımlarla karşılaştırılarak, makineye uygulanması gereken voltaj belirlenebilir. Bu tez çalışmasında, voltajlar histerisiz band akım regülatörü ile belirlenmektedir.

Buraya kadar, alan yönlendirme kontrolü konusunda yapılan açıklamalar Şekil 3.9 ile özetlenebilir. Şekilde görülen koordinat sistemi dönüşümleri yukarıda açıklanmıştır. Diğer bloklardan PIR ve IQCC bloğunun işleyişinden bu kısım içerisinde HBCR ve INVERTER bloklarının işleyişinden kısım 3.4'te EKF ESTIMATOR bloğunun işleyişinden kısım 3.5'te bahsedilecektir.

PIR bloğu oransal-integral regülatör içerir. Bu blok referans torkun belirlenmesi amacıyla kullanılmaktadır. Sistemin çalışması,

$$\left.\begin{aligned} e_n &= n - n^* \\ T_e^* &= k_p e_n + k_i \int e_n \cdot dt \end{aligned}\right\} \tag{3.45}$$

eşitlikleri ile açıklanabilir. Denklem 3.45'deki birinci eşitlik referans hız ile gerçekleşen hız arasındaki farktır. Bu hız hatasının düzeltilmesi için makinenin üretmesi gereken referans tork ikinci eşitlik ile belirlenir. Hatanın oransal sabit ile çarpımından elde edilen değer ile hatanın integralinin integral sabit ile çarpımından elde edilen değerin toplamı referans torku verir. Ancak bazı durumlarda makinenin ve kontrol sisteminin aşırı yüklenmesini önlemek amacıyla referans tork belli sınırlar



Şekil 3.9 Önerilen alan yönlendirme kontrolü blok şeması

arasında tutulabilir. Ayrıca, denklem 3.45'te görüldüğü gibi integral sürekli-zaman integralidir. Bu integrasyon kontrolörde,

$$I_{e_n}[k+1] = I_{e_n}[k] + T \cdot e_n[k+1] \tag{3.46}$$

eşitliği ile ayrık-zamanda gerçekleştirilebilir. Burada T örnekleme zamanıdır. Bu eşitlikler yardımıyla elde edilen referans torktan IQCC bloğunda referans stator akısının d bileşeni hesaplanır. Önceki kısımda verilen denklem 3.20 ile verilen tork eşitliği senkron referans yapı için tekrar düzenlenirse,

$$T_e = \frac{3}{2}\frac{p}{2}\frac{l_m}{l_r} i_{sq}^{e} \cdot \lambda_r^{e} \tag{3.47}$$

eşitliği elde edilebilir. Buradan stator akımı çekilirse,

$$i_{sq}^{e*} = \frac{2}{3}\frac{2}{p}\frac{l_r}{l_m}\frac{1}{\lambda_r^{e}} T_e^{*} \tag{3.48}$$

elde edilebilir. Böylece, senkron referans yapıda, referans stator akımının "*q*" bileşeni elde edilmiş olur. Aynı akımın "*d*" bileşeni ise doğrudan sisteme uygulanmıştır. Akımın "*d*" bileşeni rotor akısını üretmektedir. Rotor akısı sabit olduğundan, akımın bu bileşeni de sabit olacaktır. Denklem 3.24'teki rotor akısını veren eşitlik senkron referans yapıda, durgun-durum için çözülürse,

$$i_{sd}^{e*} = \frac{l_m}{\lambda_r^{e*}} \tag{3.49}$$

eşitliği bulunur. Burada, λ_r^{e*} referans rotor akısıdır. Akının referans değeri simülasyon ortamında makine nominal şartlarda çalışırken elde edilmiştir. Durgun-

durum için çözülmesinin sebebi, bu akımın örnekleme periyodu içerisinde sabit olarak kabul edilebilir olmasından kaynaklanmaktadır. Rotor akısının zamana göre değişimi örnekleme periyoduna göre yeterince yavaştır.

3.4 İnverter Modeli ve Anahtarlanması

Önceki kısımda Şekil 3.9 ile ilgili olarak HBCR, ve INVERTER bloklarından bahsedilmişti. Bunlardan birincisi histerisiz band akım regülatörünü, ikincisi inverteri içermektedir. Bu kısımda bu bloklar yardımıyla makinenin sürülmesi açıklanmaya çalışılacaktır.

Histerisiz band akım regülatörünün çalışması basitçe, referans akım ile gerçekleşen akım arasındaki hatanın belirli bir band içerisinde tutulması ilkesine dayanır. Makinenin üç-fazından her biri birbirlerinden bağımsız olarak kontrol edilir. Referans akım ile gerçekleşen akım arasındaki farkın belirlenen bandı pozitif yönde aşması gerçekleşen akımın yetersiz olduğu anlamına gelir ve ilgili faza ait anahtar kapatılarak akımın artması sağlanır. Diğer taraftan, referans akım ile gerçekleşen akım arasındaki farkın belirlenen bandı negatif yönde aşması gerçekleşen akımın gereğinden fazla olduğu anlamına gelir ve ilgili faza ait anahtar açılarak akımın azalması sağlanır. Histerisiz bandı gereksiz anahtarlamaların ve böylece, meydana gelecek anahtarlama kayıplarının önlenmesini sağlar. Bloğun çalışması ayrıntılı olarak simülasyon tasarımında açıklanmaya çalışılacaktır.

Şekil 3.9'daki inverter bloğu için Şekil 3.10'u dikkate alalım. Şekle göre sistemde altı adet anahtar mevcuttur. Ancak, bu anahtarlardan aynı kola bağlı olanlar birlikte çalışırlar. Bunlardan biri diğerinin mantıksal değili ile ifade edilir ve hiçbir zaman iki anahtar aynı anda açık veya kapalı durumda olmazlar. İki anahtarın aynı anda kapalı olması durumunda DC kaynak kısa-devre olur. Böylece, sistem üç adet anahtar olduğu kabul edilebilir. Uygulamada veya simülasyon çalışmasında, makinenin her bir fazının voltajları ölçülmek yerine, yalnızca DC kaynak voltajı ölçülmekte ve anahtarlama durumları yardımıyla faz voltajları hesaplanabilmektedir. Anahtarlama durumlarına göre elde edilen fazlar arası voltajlar ve faz-nötr voltajları Tablo 3.2'de verilmiştir. Ancak fazlar arası voltajlar;

$$\left.\begin{aligned} V_{ab} &= V_{dc}\left(S_a - S_b\right) \\ V_{bc} &= V_{dc}\left(S_b - S_c\right) \\ V_{ca} &= V_{dc}\left(S_c - S_a\right) \end{aligned}\right\} \tag{3.50}$$

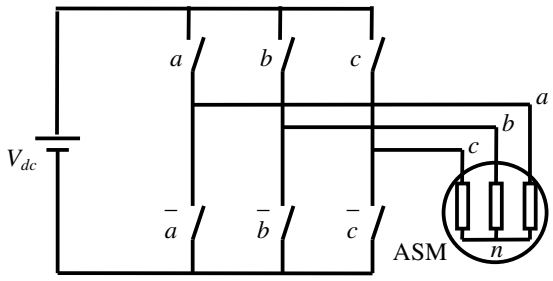
eşitlikleri ile de hesaplanabilir. Ayrıca faz-nötr voltajları da,

$$\left.\begin{aligned} V_{an} &= \left(V_{ab} - V_{ca}\right)/3 \\ V_{bn} &= \left(V_{bc} - V_{ab}\right)/3 \\ V_{cn} &= \left(V_{ca} - V_{bc}\right)/3 \end{aligned}\right\} \tag{3.51}$$

eşitlikleri ile hesaplanabilir. Ancak, sistem içerisinde bu voltajlar değil "*d*", "*q*" bileşenleri gereklidir. Önceki kısımda denklem 3.40 ile akım için verilen koordinat dönüşüm eşitliği burada voltaj için uygulanıp düzenlenirse statora uygulanan voltajın "*d*", "*q*" bileşenleri doğrudan,

$$\left.\begin{aligned} V_{sd}^{s} &= V_{dc}\left(2S_a - S_b - S_c\right) \\ V_{sq}^{s} &= V_{dc}\left(S_b - S_c\right)/\sqrt{3} \end{aligned}\right\} \tag{3.52}$$

eşitlikleri ile hesaplanabilir.



Şekil 3.10 Üç-faz köprü inverter

Tablo 3.2 İnverter anahtar konumlarına makine terminal voltajları

S_c	S_b	S_a	V_{an}	V_{bn}	V_{cn}	V_{ab}	V_{bc}	V_{ca}
0	0	0	0	0	0	0	0	0
0	0	1	$2V_{dc}/3$	$-V_{dc}/3$	$-V_{dc}/3$	V_{dc}	0	$-V_{dc}$
0	1	0	$-V_{dc}/3$	$2V_{dc}/3$	$-V_{dc}/3$	$-V_{dc}$	V_{dc}	0
0	1	1	$V_{dc}/3$	$V_{dc}/3$	$-2V_{dc}/3$	0	V_{dc}	$-V_{dc}$
1	0	0	$-V_{dc}/3$	$-V_{dc}/3$	$2V_{dc}/3$	0	$-V_{dc}$	V_{dc}
1	0	1	$V_{dc}/3$	$-2V_{dc}/3$	$V_{dc}/3$	V_{dc}	$-V_{dc}$	0
1	1	0	$-2V_{dc}/3$	$V_{dc}/3$	$V_{dc}/3$	$-V_{dc}$	0	V_{dc}
1	1	1	0	0	0	0	0	0

İnverterin modellenmesi için denklem 3.52 ile verilen eşitlikler yeterlidir. Ancak, uygulama aşamasında; aynı kola bağlanmış anahtarlardan birinin açılıp diğerinin kapatılması sırasında, anahtarların kesime gitmesindeki gecikmeler nedeniyle, kısa-devre durumlarının oluşmaması için bir anahtar kapatılmadan önce diğerinin açılması için birkaç mikro saniye civarında beklenebilir. Bu durumda, çıkış voltajı, burada hesaplanandan bir miktar küçük olabilir. Bu durum uygulama aşamasında değerlendirilerek hesaplanan voltajın yeniden düzeltilmesi gereklidir.

Bundan başka; uygulama aşamasında, akım örneklemesi, inverterin alt kısmındaki anahtarlara seri bağlı dirençler üzerinde yapılmaktadır. Bu durumda, akım örneklemesinin yapılabilmesi için bu anahtarların mutlaka kapalı olması gerekli olur. Her örnekleme periyodu başlangıcında, akım örneklemesi için yeterli bir süre bu anahtarların üçü de aynı anda kapatılmakta ve makine sargılarına sıfır voltaj uygulanmaktadır. Uygulamada statora uygulanan voltaj hesaplanırken bu durumda dikkate alınmalıdır.

3.5 Genişletilmiş Kalman Filtresi

Bu kısımda Bölüm 3.3'te bahsedilen EKF ESTIMATOR bloğundan bahsedilecektir. Şekil 3.9'da gösterilen bu blok kontrol sisteminin temelini oluşturmaktadır ve hesaplamaların tamamı neredeyse bu blok içerisinde gerçekleşmektedir.

Kalman filtresi lineer sistemler için kullanılan stokastik gözlemcidir. Bu algoritmanın amacı model ve ölçüm belirsizliklerinde durum vektör bileşenlerini kestirmektir. Diğer gözlemci-denetleyici yapılarda durum kestirimi model tarafından sağlanan tahmin terimi ile ölçülen çıkış ve tahmin edilen durumun karşılaştırılmasıyla elde edilen bir düzeltme terimine bağlıdır. Düzeltme terimi genellikle bir kazanç ile ağırlıklandırılır. Kalman filtresi durumunda, kazanç değişkendir ve her örnekleme periyodunda sistem gürültüsüne bağlı olarak yeniden hesap edilir.

EKF (Extented Kalman Filter - Genişletilmiş Kalman Filtresi) yukarıda bahsedilen Kalman filtresinin non-lineer sistemler için düzeltilmiş halidir. Non-lineer bir sistemin modelini oluşturan matematiksel eşitlikleri durum değişkenlerinin

çarpımı olan terimler içerir. Bu durumda sistemin karakteristik matrisi lineer sistemlerde olduğu gibi kolaylıkla oluşturulamaz.

Şekil 3.11'de asenkron makine için bir EKF sistemi gösterilmiştir. Bir gözlemci sisteminin amacı ölçülebilen değişkenler yardımıyla ölçülemeyen değişkenlerin tahminidir. Bir EKF gözlemci sistemde kestirim yapılırken sistemdeki parametre değişimleri ile ölçüm elemanlarından kaynaklanan gürültüler de dikkate alınmaktadır. Kalman filtresi istatistik bir sistem olup, sistemden elde edilen veriler istatistik olarak değerlendirilip ölçülemeyen değişkenlerin tahmin edilmesine çalışılır.

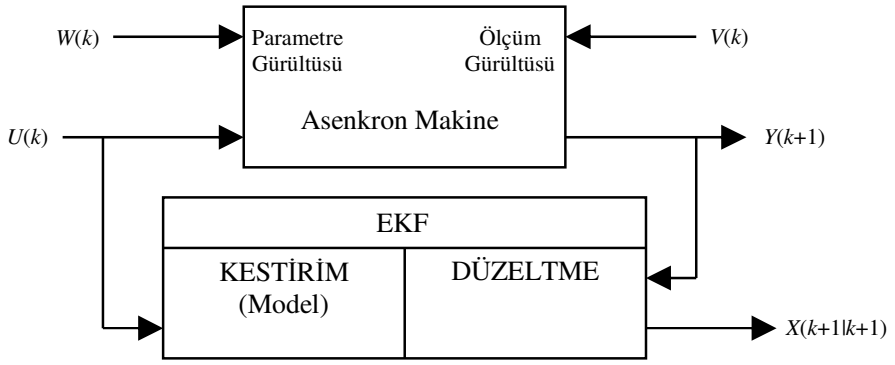
EKF algoritması asenkron makine modeli üzerinde açıklanabilir. Öncelikle, denklem 3.36 ile verilen birim asenkron makine ayrık-zaman eşitliklerini dikkate alalım. Burada ayrık-zaman eşitliklerinin kullanılma nedeni, bu algoritmanın DSP üzerinde çalıştırılacak olmasıdır. Ayrıca simülasyon aşaması da ayrık-zaman için tasarlanmıştır. İlk olarak, sistem durum değişkenlerinin tüm parametrelere göre türevi alınarak parametre değişimlerinden nasıl etkilendikleri belirlenir. Bu ise denklem 3.36 ile verilen f_n fonksiyonlarının her bir değişkene göre,

$$A(k)=\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} & \frac{\partial f_1}{\partial x_5} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} & \frac{\partial f_2}{\partial x_5} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} & \frac{\partial f_3}{\partial x_5} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} & \frac{\partial f_4}{\partial x_5} \\ \frac{\partial f_5}{\partial x_1} & \frac{\partial f_5}{\partial x_2} & \frac{\partial f_5}{\partial x_3} & \frac{\partial f_5}{\partial x_4} & \frac{\partial f_5}{\partial x_5} \end{bmatrix} \tag{3.53}$$

biçiminde diferansiyeli alınırsa,

$$A(k)=\begin{bmatrix} 1-\frac{T}{T_{S2}} & 0 & \frac{x_m}{x_{sig}x_r}\frac{T}{T_r} & \frac{x_m T\omega_b}{x_{sig}x_r}x_5(k|k) & \frac{x_m T\omega_b}{x_{sig}x_r}x_4(k|k) \\ 0 & 1-\frac{T}{T_{S2}} & -\frac{x_m T\omega_b}{x_{sig}x_r}x_5(k|k) & \frac{x_m}{x_{sig}x_r}\frac{T}{T_r} & -\frac{x_m T\omega_b}{x_{sig}x_r}x_3(k|k) \\ \frac{x_m T}{T_r} & 0 & 1-\frac{T}{T_r} & -\omega_b T x_5(k|k) & -\omega_b T x_4(k|k) \\ 0 & \frac{x_m T}{T_r} & \omega_b T x_5(k|k) & 1-\frac{T}{T_r} & \omega_b T x_3(k|k) \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.54}$$

biçiminde EKF için kullanılacak yeni sistem matrisi belirlenir.



Şekil 3.11 Asenkron makine için EKF algoritması

EKF algoritması aşağıda açıklandığı gibi beş aşamada tamamlanır. Şimdi sırasıyla bu beş aşama açıklanmaya çalışılacaktır.

1. $A(k)$ matrisindeki x_3, x_4, x_5 terimlerini içeren terimler güncellenerek,

$$\begin{bmatrix} \hat{x}(k+1|k)_{[1]} \\ \hat{x}(k+1|k)_{[2]} \\ \hat{x}(k+1|k)_{[3]} \\ \hat{x}(k+1|k)_{[4]} \\ \hat{x}(k+1|k)_{[5]} \end{bmatrix} = \begin{bmatrix} A(k)_{[1,1]} & A(k)_{[1,2]} & A(k)_{[1,3]} & A(k)_{[1,4]} & 0 \\ A(k)_{[2,1]} & A(k)_{[2,2]} & A(k)_{[2,3]} & A(k)_{[2,4]} & 0 \\ A(k)_{[3,1]} & A(k)_{[3,2]} & A(k)_{[3,3]} & A(k)_{[3,4]} & 0 \\ A(k)_{[4,1]} & A(k)_{[4,2]} & A(k)_{[4,3]} & A(k)_{[4,4]} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \hat{x}(k|k)_{[1]} \\ \hat{x}(k|k)_{[2]} \\ \hat{x}(k|k)_{[3]} \\ \hat{x}(k|k)_{[4]} \\ \hat{x}(k|k)_{[5]} \end{bmatrix} + \begin{bmatrix} \frac{\omega_b T}{x_{sig}} u_{[1]} \\ \frac{\omega_b T}{x_{sig}} u_{[2]} \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.55)$$

bir sonraki durum vektörü tahmini yapılır. Denklem 3.55'te görülen üst indis "^" kestirilen durum değişkenini temsil eder. Ayrıca k şimdiki örnek k+1 sonraki örnek anlamındadır.

2. Tahmin hatası kovaryans matrisi hesaplanır.

$$P(k+1|k) = A(k) \cdot P(k|k) \cdot A(k)^T + Q \quad (3.56)$$

Burada Q matrisi yapılan tahmine ne kadar güvenileceğini gösterir. Q matrisi ne kadar büyük olursa, önceki tahmin ile yeni tahmin arasındaki fark o kadar büyük olabilir. Böylece, Q matrisi yeni tahminin eski tahminden en fazla ne kadar sapabileceğini gösterir. Bu durumda, Q matrisi sistemdeki gürültü seviyesi ile ters orantılı olarak seçilmelidir.

3. Kalman kazanç matrisi hesaplanır.

$$K(k+1) = P(k+1|k) \cdot C^T \left\{ C \cdot P(k+1|k) \cdot C^T + R \right\}^{-1} \quad (3.57)$$

Burada R matrisi sistemden alınan ölçümlere ne kadar güvenileceğini gösterir. Denklem 3.57'den görülebileceği gibi R matrisi ile K matrisi ters orantılıdır. K

matrisinin büyüklüğü ise yeni tahminin eski tahminden ne kadar sapabileceğinin bir ölçüsüdür. Bunun anlamı, *R* matrisi büyük seçilmesi ölçümlere az güvenildiğini, ve bunun sonucunda ölçüm gürültüsünün yüksek seviyede olduğu, küçük seçilmesi ölçümlerin güvenilir olduğu, ve bunun sonucunda ölçüm gürültüsünün düşük seviyede olduğudur.

4. Yeni durum vektörü kestirimi yapılır.

$$\hat{x}(k+1 \mid k+1) = \hat{x}(k+1 \mid k) + K(k+1)\left\{ y(k+1) - C \cdot \hat{x}(k+1 \mid k) \right\} \quad (3.58)$$

Önceki aşamada bahsedildiği gibi *K* matrisinin büyüklüğü yeni tahminin eski tahminden ne kadar sapabileceğinin bir ölçüsüdür. Burada basitçe, ölçülen sistem çıkışı ile tahmin edilen sistem çıkışı arasındaki fark *K* matrisi ile ağırlıklandırılarak bir sonraki tahmin düzeltilmektedir.

5. Filtre kovaryans matrisi hesaplanır.

$$P(k+1 \mid k+1) = P(k+1 \mid k) - K(k+1) \cdot C \cdot P(k+1 \mid k) \quad (3.59)$$

Son aşamada bir sonraki örnekleme zamanında kullanılacak olan *P* kovaryans matrisi hesaplanmaktadır. Bundan sonra, bu örnekleme sonunda tahmin edilen durum değişkenlerine göre *A(k)* matrisi güncellenerek 1. aşamaya dönülür. Tabii ki, 1. aşamadan itibaren tüm algoritma sonraki örnekleme periyodu boyunca işletilecektir.

Tüm algoritma Şekil 3.12'deki gibi özetlenebilir. Bu projede kullanılacak olan EKF kestirimci sistemin çıkışları,

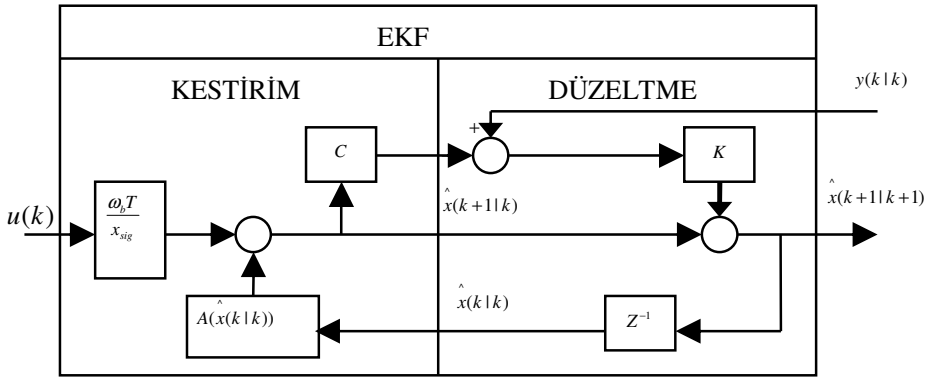
$$\left.\begin{aligned} \lambda_r^s &= \sqrt{\lambda_{rd}^{s2} + \lambda_{rq}^{s2}} \\ \cos\theta_e &= \frac{\lambda_{rd}^s}{\lambda_r^s} \\ \sin\theta_e &= \frac{\lambda_{rq}^s}{\lambda_r^s} \\ \omega &= x_5 \end{aligned}\right\} \quad (3.60)$$

eşitliğinde verilmiştir. Denklem 3.60 ile verilen ilk eşitlikte karekök işlemi görülmektedir. Simülasyon yazılımında karekök işlemi için kullanılacak bir komutun bulunmasın karşın, DSP yazılımında böyle bir komut yoktur. Bu işlemin DSP ortamında yapılabilmesi için Newton iterasyon yöntemi kullanılmıştır. Bu yöntem,

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{c}{x_n} \right) \quad (3.61)$$

eşitliği ile tanımlanır. Burada c karekökü alınacak sayı, x_{n+1} karekök için (n+1). iterasyon sonucu x_n karekök için n. iterasyon sonucudur. Rotor akısı birim sistem durumunda yaklaşık 1.0 birim civarında olduğundan $x_0 = 1$ alınabilir. Bu durumda, 4. iterasyon sonucunda yaklaşık 1/1000 civarında bir hata ile karekök hesaplanabilmektedir.

Buraya kadar verilen eşitlikler yardımıyla alan yönlendirme kontrolü için gerekli olan, denklem 3.61 ile verilen çıkışlar elde edilebilir.



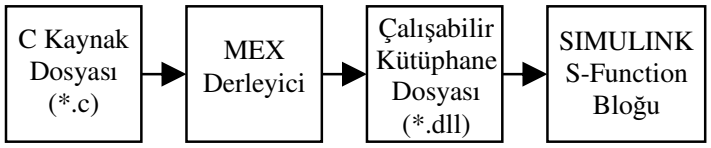
Şekil 3.12 EKF akış şeması

4. DSP TABANLI VEKTÖR KONTROLÖR VE ASENKRON MOTOR SİMÜLASYON TASARIMI

Bu bölümde önceki bölümlerde açıklanan alan yönlendirme kontrolünün pratikte uygulanabilir olup olmadığı, eğer uygun ise uygulama aşamasına yardımcı olabilecek verilerin elde edilmesine çalışılacaktır. Simülasyon ortamı olarak MATLAB/SIMULINK V7.0.1 yazılım paketi üzerinde çalışılmıştır. Başlangıçta aynı paketin eski sürümleri üzerinde de çalışmalar yapılmıştır. Ayrıca, The AEgis Technologies Group, Inc. tarafından geliştirilen ticari bir yazılım olan acslXtreme sürekli simülasyon dili ve INRIA Scilab Consortium tarafından geliştirilen ve serbest olarak dağıtılan MATLAB/SIMULINK (M/S) benzeri SCILAB/SCICOS 4.0 üzerinde de çalışmalar yapılmıştır. Yapılan çalışmalar sonucunda M/S paketinin performans, yazılım, doküman, simülasyon hızı gibi açılardan en uygun simülasyon ortamı olduğuna karar verilmiştir.

Başlangıçta, simülasyon çalışmaları yapılırken yalnızca M/S paketinde hazır olarak bulunan Simulink blokları yardımıyla tasarımlar yapılmış, ancak, simülasyon hızı çok düşük olmaktadır. Matlab m-file ile simülasyondan kaçınılmıştır. Bunun nedeni sürekli-simülasyonu desteklememesidir. Simulink ise potansiyel olarak sürekli-simülasyonu desteklemektedir. Ancak, yukarıda bahsedildiği gibi, tasarımda yalnızca paket içerisinde önceden hazır olan Simulink bloklarının kullanılması simülasyon hızını düşürmektedir. Daha sonra, simülasyon hızını artırmak amacı ile, Simulink blokları yerine yine Simulink'e ait C S-Function'dan yararlanılmıştır. Böylece, hem doğrudan C programları kullanıldığından simülasyon hızı artırılmış, hem de sürekli-simülasyon ortamından vazgeçilmemiştir.

M/S C S-Function kullanımı ve işlemesi basitçe şöyledir. Öncelikle, program normal C ortamında olduğu gibi yazılmaktadır. Tabii ki, bu fonksiyon içinde önceden hazır bazı fonksiyonlar mevcuttur. Bunlar, girişlerin tanımlanması, sürekli ve ayrık-zaman değişkenlerinin tanımlanması, çıkışların tanımlanması, başlangıç değerlerinin atanması, çıkışların atanması gibi fonksiyonlardır. Bu tez projesi için yazılan C S-Function dosyaları EK-A'da verilmiştir.



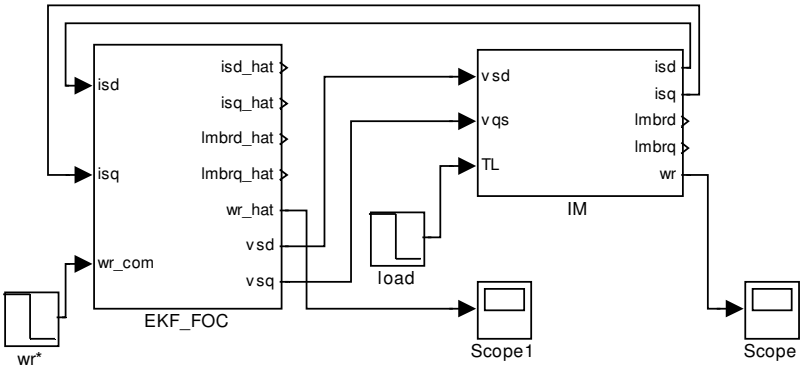
Şekil 4.1 SIMULINK S-Function oluşturulması

Simulink C S-Function bloğu Şekil 4.1'de gösterildiği gibi oluşturulur. Öncelikle EK-A'da verilen yazılıma benzer bir C-kaynak dosyası oluşturulmalıdır. Daha sonra C-kaynak dosyası MATLAB komut satırında "*mex*" komutu ile derlenerek çalışabilir kütüphane dosyası elde edilir. Elde edilen çalışabilir dosya Matlab-Executable olarak adlandırılır. Bu dosyanın Simulink model dosyası ile aynı klasörde olması gereklidir. Bu dosyayı kullanabilmek için Simulink dosyasına bir C S-Function bloğu eklenir ve dosyanın ismi bu blok için açılan diyalogda parametre olarak tanımlanır. Blokta tek giriş ve tek çıkış bulunur. Bu giriş ve çıkışlar vektör değişken olup fonksiyon içerisinde tanımlanan tüm giriş ve çıkışları içerirler. Bu bloğa giriş yapmak için, giriş vektörünün boyutuyla aynı boyutta bir "*mux*" bloğu, ve bloktan istenen çıkışları elde etmek için, çıkış vektörü ile aynı boyutta bir "*demux*" bloğu kullanılır. Bunun dışında, C S-Function bloğunun kullanımı normal Simulink bloklarının kullanımından farklı değildir.

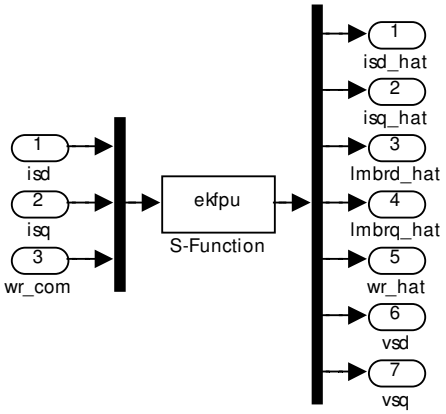
C S-Function kullanımı simülasyon hızını neredeyse gerçek hıza yakın hızlara çıkarmaktadır. Ancak, yinede EKF algoritması oldukça zaman almaktadır. Simulink jenerik blokları ile EKFnin tasarlanması mümkün olsa bile, hız ve model dosyası büyüklüğü açısından oldukça zordur.

4.1. Simülasyon Dosyaları ve Fonksiyonlar

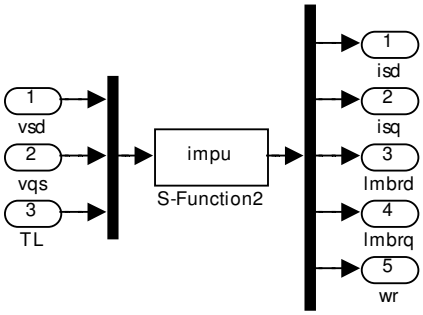
Bu kısım içerisinde Bölüm 3'te ayrıntıları açıklanan alan yönlendirme kontrolü sistemin simülasyon yazılımı ortamına aktarılması açıklanmaya çalışılacaktır. Öncelikle, Şekil 4.2'de gösterilen M/S model dosyası oluşturulmalıdır. Şekilde görülen EKF_FOC bloğu denetleyici sistemi IM bloğu ise asenkron makineyi temsil etmektedir. Her iki bloğun içerisinde Şekil 4.3 ve Şekil 4.4'te gösterildiği gibi birer adet C S-Function bloğu ve yukarıda bahsedilen şekilde giriş ve çıkışların düzenlenmesi amacıyla kullanılan "*mux*" ve "*demux*" blokları mevcuttur.



Şekil 4.2 DSP tabanlı vektör kontrolör ve asenkron motor Simulink modeli



Şekil 4.3 EKF_FOC Bloğu içeriği



Şekil 4.4 IM Bloğu içeriği

Bunun dışında model dosyasında herhangi bir blok yoktur. Yalnızca, yine aynı şekilde görülen çıkışların gözlenebilmesi için osiloskop blokları ve sistem girişlerini temsil eden sabit atama blokları vardır.

Asenkron makine bloğuna ait C S-Function "*impu.c*" olarak, Denetleyici bloğa ait C S-Function "*ekfpu.c*" olarak, EK-A'da verilmiştir. Bunun dışında her iki C S-Function tarafından da kullanılan dosyalar da vardır. Bunlardan birincisi, asenkron makine parametrelerini içeren C başlık dosyasıdır ve EK-A'da "*im_par.h*" olarak verilmiştir. İkinci dosya ise matris operasyonlarını yerine getirmek için kullanılan EK-A'da "*matopt.c*" olarak verilen dosyadır. Her iki dosya C diline ait "*#include*" anahtarı ile ana dosyalara dahil edilmişlerdir.

Asenkron makine sürekli-zaman sistem olarak simüle edilmiştir. Bunun için Bölüm 3'te denklem 3.32-3.33 ile verilen birim sürekli-zaman eşitlikleri kullanılmıştır. Makine parametrelerini içeren "*im_par.h*" dosyası,

```
#include "im_par.h"
```

biçiminde C S-Function dosyası "*impu.c*" dosyasına dahil edilmiştir. Örnekleme zamanı ve sürekli-zaman ayrık-zaman seçimi,

```
mdlInitializeSampleTimes(SimStruct *S)
```

fonksiyonu ile seçilmektedir. Durum değişkenlerinin ilk değerleri,

```
mdlInitializeConditions(SimStruct *S)
```

fonksiyonu ile atanmaktadır. Makine parametrelerine göre birim parametrelerin hesaplanması, ve bunlara göre sistem matrislerinin belirlenmesi,

```
mdlStart(SimStruct *S)
```

fonksiyonu içerisinde yapılmaktadır. Hesaplanan değerlerin çıkışa aktarılması,

```
mdlOutputs(SimStruct *S, int_T tid)
```

fonksiyonu tarafından sağlanmaktadır. Son olarak, diferansiyel denklemlerin tanımlanması,

```
mdlDerivatives(SimStruct *S)
```

fonksiyonu içerisinde yapılmaktadır. Son fonksiyon içerisinde tanımlanan diferansiyel denklemler her çevrimde bir kere integre edilerek durum değişkenlerinin değeri güncellenmektedir.

Asenkron makine simülasyonu için kullanılan C S-Function içerisinde matris operasyonları olmadığından bu dosya içerisine bu fonksiyon dahil edilmemiştir.

Denetleyici simülasyonu için kullanılan C S-Function "*ekfpu.c*" olarak EK-A'da verilmiştir. Bu dosyanın "*impu.c*" dosyasından farkı yalnızca ayrık-zaman simülasyonu olarak tasarlanmış olmasıdır. Bunun sebebi, bu dosya içerisinde verilen algoritmaların DSP üzerinde çalıştırılacak olmasıdır. Bu dosyada örnekleme zamanı, 200 μs olarak seçilmiştir. Bu süre, DSP yazılımında da aynıdır, ve uygulamada bütün algoritma bu sürenin biraz altında tamamlanabilmektedir. Bu nedenle minimum örnekleme zamanı bu değerin altında olamaz. Gerçekte sürekli-zamana yaklaşmak için bu değer mümkün olduğunca küçük tutulmalıdır. Ayrık-zaman sistem tasarlandığından, denetleyici C S-Function içerisinde diferansiyel eşitlikler için kullanılan fonksiyon yoktur. Bunun yerine,

```
mdlUpdate(SimStruct *S, int_T tid)
```

fonksiyonu kullanılmaktadır. Kullanılan diğer fonksiyonlar makine için verilen dosyadaki fonksiyonlar ile aynıdır.

Model güncelleme fonksiyonu içerisinde öncelikle EKF algoritması çalıştırılmakta ve alan yönlendirme kontrolü için gerekli veriler elde edilmektedir. EKF algoritması denklem 3.53-3.61'de verilen eşitliklerden oluşur. Ancak, denklem 3.53-3.54 yalnızca başlangıçta "*im_par.h*" dosyasında verilen parametreler yardımıyla bir sefer hesaplanırken, diğerleri her çevrimde güncellenir. Matris işlemleri, EK-A'da "*matopt.c*" olarak verilen dosyadaki fonksiyonlar ile yapılır. Bu dosya da, ana dosyaya dahil edilmiştir.

EKF algoritması tamamlandıktan sonra, bu algoritmaya göre oldukça kısa süren, denklem 3.40-3.52 eşitlikleri yardımıyla alan yönlendirme kontrolü algoritması yerine getirilir. Sonuçta, referans hızın elde edilebilmesi için inverter anahtar konumları ve buradan makineye uygulanacak voltajlar belirlenir. Böylece bir kontrol çevrimi tamamlanmış olur.

Buraya kadar, bahsi geçen dosyalar, Matlab komut penceresinde "*mex*" komutu ile derlenerek çalışabilir dosyalar elde edilir. Bunun dışında, S C-Function için ayrıntılara M/S dokümantasyonlarından, C-dili ayrıntıları için Bölüm 8'de verilen kitaplara veya başka C dili referans kitaplarına bakılabilir.

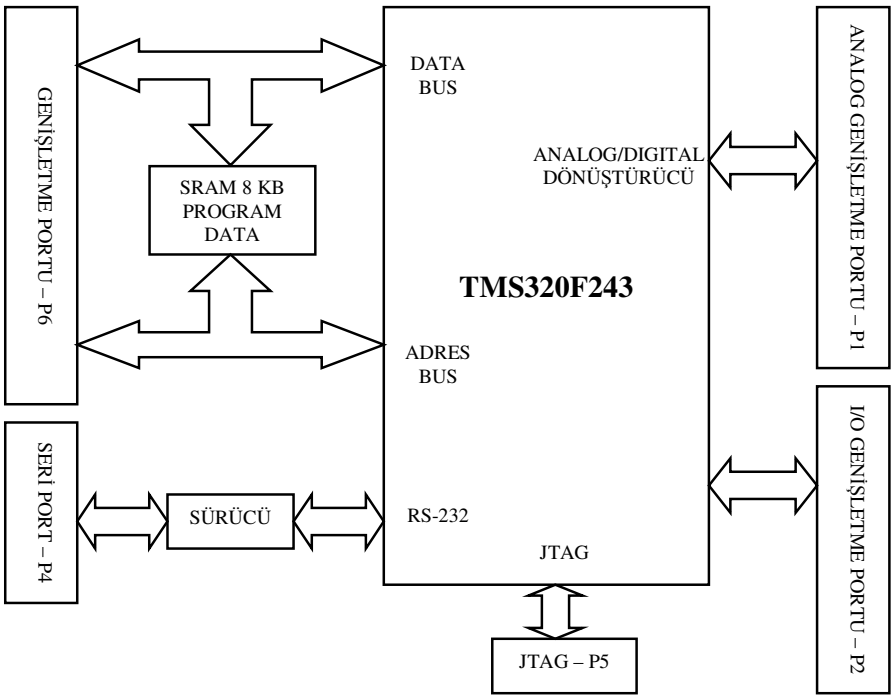
5. TMS320F243 TABANLI ASENKRON MOTOR SÜRÜSÜCÜ TASARIMI

Tez projesi için kullanılan donanım iki ayrı kısımdan oluşmaktadır. Bunlardan birincisi, gerekli kontrol yazılımının yüklenebildiği ve bir DSP kontrolör içeren kontrolör kartıdır. İkincisi ise kontrolör kartı ile doğrudan bağlantılı olarak çalışan, güç elektroniği anahtarları, sürücüler ve ölçüm elemanları da içeren motor kontrolör kartıdır. Ayrıca, motor kontrolör kartı, DSP kontrolörün çevresel birimlerle bağlantısını sağlamak için gerekli donanımları da üzerinde bulundurur ve böylece DSP kontrolör ile çevresel birimler motor kontrolör kartı yoluyla bağlanırlar.

5.1. DSP Kontrolör (F243 DSK – DSP STARTER KIT)

Bilindiği gibi vektör kontrol tekniği karmaşık matematiksel hesaplamalar içerir ve bu işlemlerin çok hızlı bir şekilde yapılması gereklidir. Çünkü motor içerisinde meydana gelen değişimlere göre, gerekli olan kontrol sinyallerinin hızlı bir şekilde üretilip tekrar motora uygulanması gereklidir. Bunun için daha önce de bahsedilen DSP kontrolörler uygun bir seçimdir. Bu tez projesinde Texas Instruments firması tarafından, özellikle motor kontrol uygulamalarına yönelik olarak geliştirilmiş olan TMS320F243 DSP kontrolör entegresi kullanılmıştır.

Tez projesinde kullanılan kontrolör kartı proje içerisinde geliştirilmemiştir. Bunun yerine yazılım geliştirme amacıyla Spectrum Digital firması tarafından geliştirilen TMS320F243 içeren hazır bir kart temin edilmiştir. Bu kart üzerinde DSP kontrolör entegresine ek olarak yalnızca okunup-yazılabilen bir bellek ve bir PC bilgisayar ile seri haberleşme sağlayan arabirim mevcuttur. Diğer giriş-çıkış portları konnektörler aracılığıyla doğrudan kart dışına alınabilmektedir. Şekil 5.1'de bu kartın yapısı bloklar halinde görülmektedir. Kart içerisinde kalıcı belleğe yerleştirilmiş bir monitör sistem mevcuttur. Bu nedenle kendi kendine çalışabilmektedir. Ayrıca, RS-232 seri haberleşme arabirimi aracılığıyla PC ile haberleşme yeteneğine sahiptir. Kart ile birlikte gelen PC yazılımı yardımıyla kart içindeki bellekler okunabilmekte, değiştirilebilmekte ve ayrıca çalışabilen yazılım geçici belleğe yüklenebilmektedir. Güç kaynağı ise kart üzerinde olmayıp ayrı bir kaynaktan karta güç sağlanmaktadır.



Şekil 5.1 F243 DSK (DSP STARTER KIT) kartı blok şeması

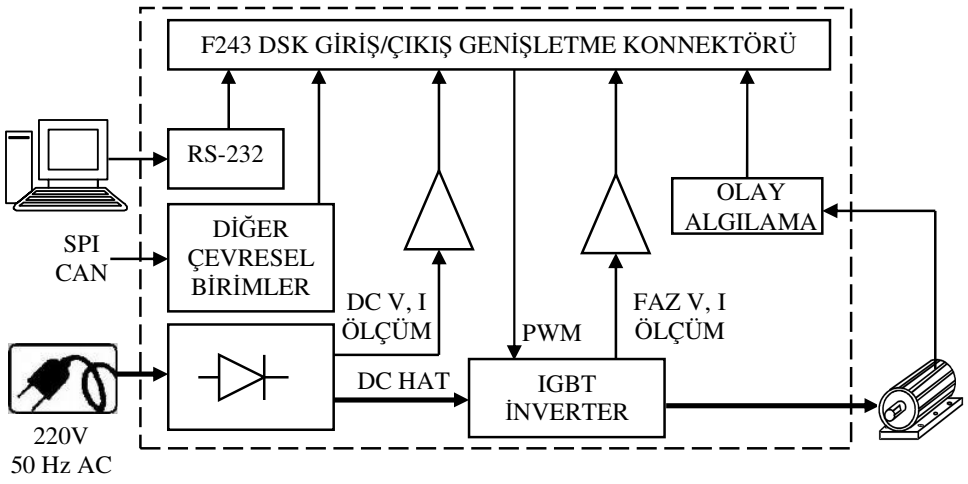
Şekil 5.1'de görüldüğü gibi kontrolör entegresi içerisindeki belleklere ek olarak, program yüklemek için 8 KB büyüklüğünde bir RAM bellek eklenmiştir. Bellek düzenlemesi, program belleği ve veri belleği olarak iki kısma ayrılmıştır. Yine seri haberleşme için bir sürücü de eklenmiştir. Diğer kontrolör portları ise doğrudan konnektörlere bağlıdırlar. Tez projesinde bu portların tümü kullanılmamaktadır. Kullanılan portlar, analog-digital dönüştürücülerin bağlı olduğu P1 analog genişletme portu, çevre birimleri için gerekli olan giriş-çıkışların bağlı olduğu P2 giriş-çıkış genişletme portu ve PC haberleşmesi için kullanılan P5 seri portudur. P1 ve P2 portları üzerinde herhangi bir arabirim olmayıp yine doğrudan DSP kontrolör entegresine bağlıdırlar. Gerekli sürücü devreler motor kontrolör kartında mevcuttur. Bunun dışında kalan portlar boş bırakılmışlardır. P1 ve P2 portu aynı biçimde motor kontrolör kartında da mevcuttur. İki kart birbirlerine doğrudan bağlanmaktadır.

F243 DSK kartı hakkında daha geniş bilgi için Spectrum Digital firması web sitesine ve TMS320F243 DSP kontrolörü hakkında daha geniş bilgi için Texas Instruments firması web sitesine bakılabilir. Bu kartın Orcad yazılımı ile çizilmiş tam devre şeması "F243 DSK Technical Reference" isimli kaynakta verilmiştir.

5.2. Sayısal Motor Kontrolör (DMC)

Motor kontrolör kartı, güç elektroniği anahtarları, ölçüm elemanları ve diğer çevre birim haberleşme sürücülerini içerir. Bu kart F243 DSK kartına uygun olarak, Spectrum Digital firması tarafından geliştirilmiş olan motor kontrolör kartı esas alınarak tasarlanmıştır. Kart işlevsel bloklar halinde Şekil 5.2'deki gibi gösterilebilir. Kesikli çizgi içerisinde kalan kısım motor kontrolör kartını temsil eder. Kalın çizgi ile gösterilen hatlar güç devresini belirtmektedir. Diğer hatlar kontrol ve ölçüm sinyallerini göstermektedir. Giriş ve çıkışların tümü F243 DSK kartına doğrudan bağlanabilecek şekilde tasarlanmıştır.

Güç kaynağı olarak tek faz 220 V AC besleme gereklidir. AC giriş basit bir filtreden geçirilerek köprü doğrultucuya verilir. Daha sonra elde edilen DC voltaj altı adet IGBT yarı-iletken anahtardan oluşan inverter devresine uygulanır. Böylece, asenkron makine PWM sinyallerine uygun olarak beslenir. IGBT anahtarlar motor akımı ve PWM frekansına uygun olarak seçilmişlerdir. Şekil 5.2'de olay algılama olarak belirtilen blok içerisinde motor miline bağlanacak olan herhangi bir hız algılayıcısından gelen sinyalleri algılanabilmesi gerekli sayıcı girişleri vardır. Ayrıca, tez projesinde kullanılmayan, ancak daha ileri uygulamaların desteklenmesi amacıyla seri çevresel arabirim haberleşmesi (SPI) ve harici cihazlarla haberleşme (CAN) için kullanılacak olan girişler de vardır.



Şekil 5.2 Sayısal motor kontrolör (DMC) blok şeması

PC ile F243 DSK kartı arasında seri haberleşme amacıyla optik olarak yalıtılmış RS-232 arabirim sürücüsü kullanılmıştır. Yine harici kullanım amacıyla optik olarak yalıtılmış iki adet giriş ve çıkış mevcuttur. Tüm faz akım ve gerilimleri ile DC hat akım ve gerilimi ölçüm yükselteçleri aracılığıyla DSP kontrolördeki ADC'lere ulaştırılmaktadır. Akım ölçümleri için yükselteçler 42 kHz, gerilim ölçümleri için ise 3.2 kHz alçak geçiren filtre görevi üstlenmektedir. DMC kartı için gerekli besleme voltajları kart üzerindeki voltaj regülatörleri tarafından üretilmektedir. DMC kartı AC girişten başka harici bir kaynağa ihtiyaç duymaz. Motor kontrolör kartının Orcad yazılımı ile çizilmiş tam devre şeması EK-B'de verilmiştir. Daha geniş bilgi için Spectrum Digital firması web sitesine bakılabilir.

5.3. TMS320F243 için Yazılım Tasarımı

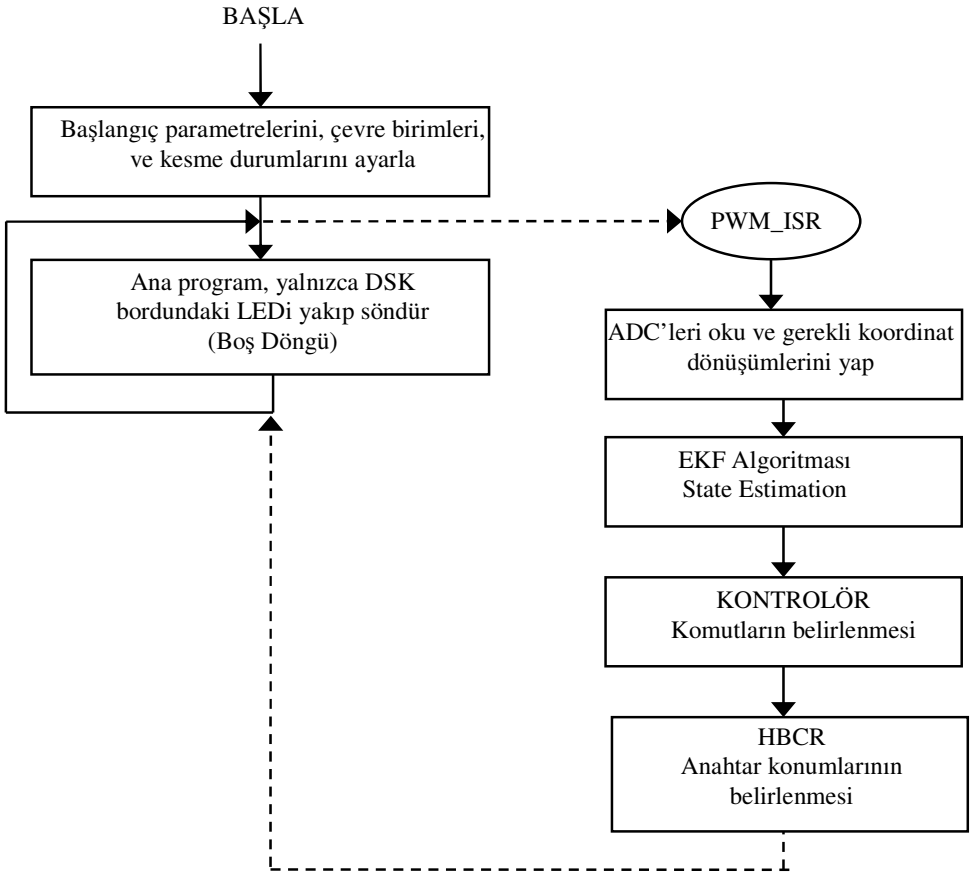
Yazılım ortamı için çeşitli seçenekler mevcuttur. Bunlardan birincisi, basit bir metin editörü ile doğrudan assembly dilinde programın yazılmasıdır. F243 DSK kiti ile birlikte verilen SD24XASM isimli bu program makine diline çevrilerek DSP kontrolöre yüklenebilir. Ancak bu şekilde programın oluşturulması zaman alıcıdır, ve daha karmaşıktır. Diğer bir seçenek ise, programın C-dilinde oluşturulmasıdır. Bunun için, kullanılan DSP kontrolöre uygun bir C-derleyicisi gereklidir. C-dilinde oluşturulan program bu derleyici yardımıyla çevirici komutlarına dönüştürülebileceği gibi doğrudan makine diline de çevrilebilir. Bu yöntemle elde edilen programda sonradan gerekebilecek değişiklikler kolaylıkla yapılabilir ve anlaşılması daha kolay olacaktır.

Başlangıçta C yazılım ortamı seçilmiştir. Ancak C kaynak dosyasından C-derleyicisi ile elde edilen programın DSP tarafından çalıştırılma süresi seçilen örnekleme zamanını aşmaktadır. Bu nedenle daha karmaşık olmasına rağmen, ürettiği programın DSP tarafından çalıştırılma süresi, belirlenen örnekleme süresini karşılayabilen doğrudan assembly kodları ile yazılım geliştirme yöntemi tercih edilmiştir.

Şekil 5.3'te DSP için yazılan alan yönlendirme kontrolü program akış şeması görülmektedir. Kontrol çevrimi senkronizasyonu inverter donanımından gelen kesme sinyali tarafından sağlanmaktadır. Alan yönlendirme kontrolü algoritmasının tamamı

bu şekilde PWM_ISR olarak belirtilen kesme hizmeti alt-programı içerisinde gerçekleşmektedir. Bu algoritma işletilmeden önce yazılım ve donanım için gerekli olan ilk ayarların yapılması gereklidir. Yazılım için gerekli ilk ayarlar, değişkenlerin ilk değerlerinin atanmasından ibarettir. Bu değerler makine parametrelerine göre hesaplanırlar. Donanım ayarları ise, birden fazla seçimlik fonksiyona sahip DSP giriş/çıkış portlarının hangi fonksiyonu yerine getireceği, port yönü, izin verilecek kesmeler, kesmelerin gerçekleşeceği zaman veya durum, çıkış portlarının ilk değerleri, PWM temel frekansı, ölü-band genişliği, ADC için akım örnekleme zamanının senkronizasyonu gibi çeşitli ayarlardır.

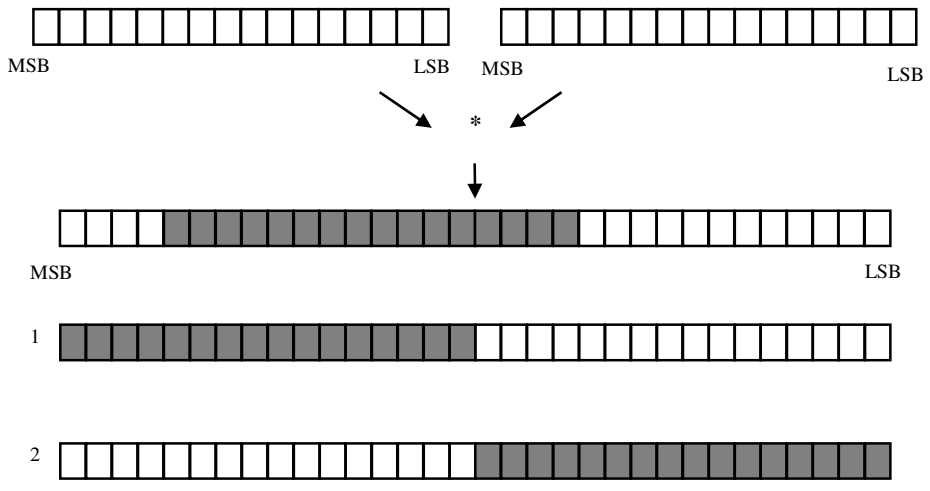
Ana program içerisinde vektör kontrol algoritması için gerekli herhangi bir yazılım yoktur. Ancak programın işleyip işlemediğinin dışarıdan bakıldığında anlaşılabilmesini sağlayan F243 DSK kartı üzerinde bulunan bir LED lambanın yakılıp söndürülmesinden sağlayan basit bir yazılım konulmuştur.



Şekil 5.3 DSP yazılımı akış şeması

Yukarıda bahsedildiği gibi bütün kontrol algoritması kesme hizmet alt-programı içerisinde gerçekleşmektedir. Bu sayede, hesaplanan ayrık zaman eşitliklerindeki örnekleme zamanı garanti altına alınmaktadır. Akış şemasında görüldüğü gibi ilk olarak ADCler tarafından okunup ADC kuyruğuna saklanmış olan akım ve gerilim örnekleri okunup birim sisteme uygun biçimde örneklenerek, stator referans yapıda temsil edilmektedirler. Bu işlemler için denklem 3.40 ve denklem 3.52 ile verilen eşitliklerden yararlanılmaktadır. Elde edilen bu değerler 4.12 formatında saklanmaktadır. Burada en soldaki ilk bit sayının işaretini, sonraki 3 bit sayının tam kısmını ve kalan 12 bit ondalık kısmını göstermektedir. Daha önce bahsedildiği gibi bu sayılar (8 -8] aralığında kalmaktadır. Birim makine modeli kullanıldığından herhangi bir taşma oluşmamaktadır.

4.12 formatındaki sayılar doğrudan toplanabilirler. Ancak, çarpma işlemi biraz farklıdır. Bunun için Şekil 5.4'ü dikkate alalım. 4.12 formatında iki 16 bit sayı çarpıldığında 8.24 formatında 32 bit bir sayı elde edilir. Sonuç DSPde 32 bit çarpım registerine kaydedilmektedir. Şekil 5.4'de koyu renk ile gösterilen bitler çarpım sonucunu temsil eder. Elde edilen bu sonuç sola doğru 4 bit kaydırılırsa çarpım üst 16 bitte 4.12 formatında tekrar elde edilir. Aynı şekilde sağa doğru 12 bit kaydırılırsa çarpım alt 16 bitte 4.12 formatında tekrar elde edilir. Birinci yol kullanılırsa üst 16 bit, ikinci yol kullanılırsa alt 16 bit çarpım sonucu olarak kaydedilir.

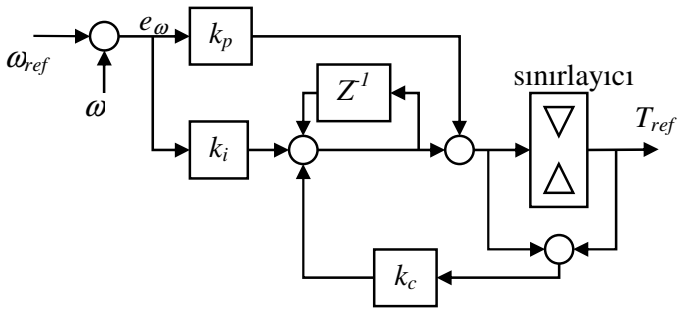


Şekil 5.4 TMS320F243 üzerinde 4.12 formatındaki iki sayının çarpılması

Daha sonra akış şemasında EKF algoritması görülmektedir. EKF algoritmasını yerine getirmek için denklem 3.53-3.61'de verilen eşitliklerden yararlanılmaktadır. Aynı eşitlikler simülasyon aşamasında da EKF algoritması için kullanılmıştır. Simülasyon aşamasında matris işlemlerini yerine getiren fonksiyonlardan oluşan bir alt-program kullanılmıştır. DSP üzerinde böyle bir alt-program kullanımı, çeşitli döngüler ve döngülerin sona erip ermediğini kontrol için sayaçlar ve karar aşamaları içerdiğinden, kontrol çevrimi süresinin uzamasına neden olmaktadır. Bu da daha önce belirtilen örnekleme zamanını sınırının aşılmasına neden olmaktadır. Bu sebeple, EKF algoritmasındaki matris işlemleri için hiçbir karar verme aşaması içermeyen düz bir mantık kullanılmıştır. EK-C'de verilen DSP programı incelendiğinde matris işlemleri için bazı döngüler görülecektir. Derleme zamanı komutları "." ile başlar. Bu döngüler çalışma zamanı döngüleri olmayıp derleme zamanı döngüleridir ve derleyici programının kısa tutulması amacıyla kullanılmışladır. Sonuç olarak çalışma zamanına etkileri yoktur. Kullanılan tüm sayılar yine 4.12 formatında saklanmaktadırlar.

EKF algoritmasında elde edilen çıkışlar, denklem 3.60'ta toplu olarak verilmişlerdir. Bu sonuçlar ile referans hız kullanılarak referans stator akımı belirlenir. Bu işlemler Şekil 5.3'de verilen akış şemasında kontrolör olarak gösterilen blokta gerçekleşir. Elde edilen bu değerler yine 4.12 formatında saklanır.

Referans stator akımının hesaplanmasında denklem 3.45-3.49 ile verilen eşitliklerden yararlanılmaktadır. Burada yalnızca PI regülatör çıkışı nominal tork ile sınırlandırılmıştır. Şekil 5.5'de PI regülatör algoritması için akış şeması verilmiştir. Şekilde görüldüğü gibi öncelikle gerçekleşen hız referans hızdan çıkartılarak hız



Şekil 5.5 Sınırlayıcı içeren PI regülatör için akış şeması

hatası elde edilir. Daha sonra integral sabit ile çarpılıp integre edilen hız hatası ile oransal sabit ile çarpılan hız hatası toplanarak referans tork elde edilir. Asenkron makinenin aşırı yüklenmesini önlemek için referans tork nominal tork ile sınırlandırılır. Bu durumda integrasyonun da sınırlandırılması gerekir. Bu amaçla sınırlama işleminden önceki referans tork ile sınırlamadan sonraki referans tork arasındaki fark bir düzeltme sabitiyle çarpılır. Bu çarpım ile integral sabit ile çarpılan hız hatası toplanarak integre edilir. Böylece integrasyon sonucu hiç bir zaman sınır torku aşmaz. İntegasyon işlemindeki bu düzeltme yapılmaz ise PI regülatör çıkışı sınırlansa bile integrasyon işleminden elde edilen sonuç artmaya devam edecektir. Bu durumda regülatör çıkışının girişteki hataya tepkisinde gecikme oluşacaktır.

Referans stator akımları belirlendikten sonra, denklem 3.43-3.44 ile verilen eşitlikler yardımıyla, stator referans yapıdaki üç-faz akımlara dönüştürülürler. Daha sonra, akış şemasında HBCR olarak gösterilen blok içerisinde, referans akım ile gerçekleşen akım karşılaştırılarak, inverter anahtar konumları belirlenir. İnverter anahtar konumları bir faz için basitçe,

```
if (ia_com-ia)>hb){sa = 1;}
if (ia_com-ia)<hb){sa = 0;}
```

biçiminde belirlenebilir. Burada `ia_com` referans akım, `ia` gerçekleşen akım `hb` histerisiz bandı ve `sa` inverter S_a anahtarının konumudur. Ayrıca, bu anahtar konumları, DSP inverter karşılaştırma registerlerine de yüklenirler. İnverter periyodu 4000 olarak seçilmiştir. Bir komut çevrimi 50 ns'dir. Sonuç olarak inverter periyodu, ve ayrıca örnekleme periyodu $4000*50*10^{-9}$ = 2000 μs olmaktadır. DSPde bulunan modülatörde karşılaştırma sayıcı vardır ve 4000'e kadar sayıp sıfıra dönmektedir. Bir faza ait karşılaştırma registerindeki değer sayaçtan küçük olursa anahtar açık, büyük olursa anahtar kapalı olmaktadır. Bu durumda, bir örnekleme periyodunda anahtarın tamamen kapalı olması için karşılaştırma registerine sıfır, tamamen açık olması için 4000'den büyük bir değer yüklenmelidir. Sonuç olarak, `sa = 1` için karşılaştırma registerine sıfır ve `sa = 0` için karşılaştırma registerine 4001 yüklenecektir. Ancak, burada akım örneklemesi için örnekleme periyodunun başlangıcı kullanılmaktadır. Bu da örnekleme periyodunun başlangıcında 5 μs civarında inverter alt anahtarlarının kapalı yani `sa = 0` olmasını gerektirmektedir. Bu ise `sa = 0` için karşılaştırma registerine 4001 ve `sa = 1` için karşılaştırma registerine 100 konularak

sağlanmakta ve `sa`'nın her durumu için örnekleme periyodu başlangıcında 5 μs'lik boşluk elde edilmektedir.

TMS320F243 DSP donanımı ve SD24XASM assembler programı ayrıntıları için Bölüm 8'de verilen kaynaklara, Texas Instruments firması web sitesine ve Spectrum Digital web sitesine bakılabilir.

6. DENEYSEL SONUÇLAR

Deneysel sonuçlar iki kısımda incelenmiştir. Bunlardan birincisi Bölüm 4'de verilen simülasyon tasarımı esas olmak üzere MATLAB/SIMULINK yazılım paketi ile yapılan simülasyon çalışmalarının incelenmesidir. İkinci kısımda ise Bölüm 5'de verilen donanım üzerinde ve yine aynı bölümde verilen, simülasyon çalışmasına uygun olarak tasarlanan yazılım yardımıyla elde edilen deney sonuçlarının incelenmesidir.

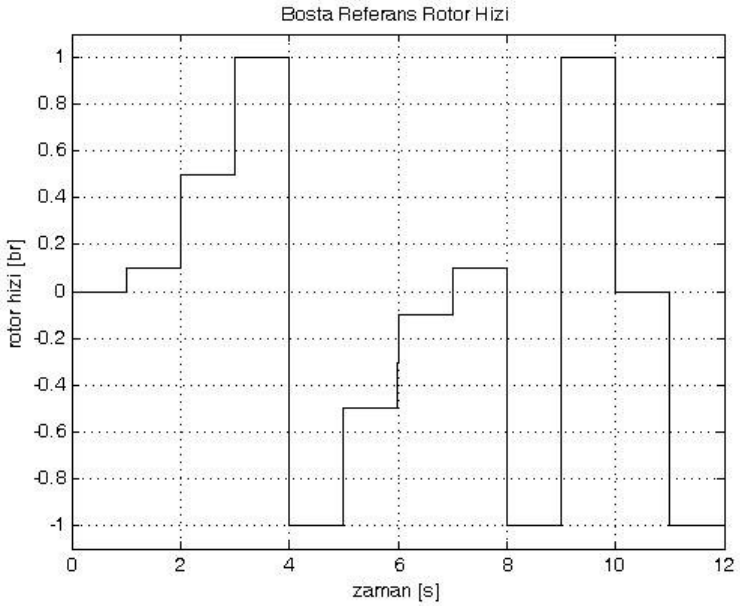
6.1. Asenkron Makinenin DSP Tabanlı Vektör Kontrolü Simülasyonu

Simülasyon ortamında değişik yük ve rotor hızlarında kısım 3.3'te önerilen vektör kontrol yönteminin dinamik performansı incelenmeye çalışılmıştır. Simülasyon sisteminin gerçek bir sistemden tek farkı, ölçüm gürültülerinin ve sistem gürültülerinin modele dahil edilmemiş olmasıdır. Bu gürültülerin gerçek sistemde EKF tarafından yok edileceği beklenmektedir. Sistem girişleri denetleyici için referans rotor hızı, ve makine için yük torkudur. Bu çalışma ile makinenin verilen referans hızlara uyup uymadığı, sistemin cevap zamanı ve hız kestiriminin doğruluğu incelenmeye çalışılmıştır. Bu kısım içerisinde yalnızca rotor hızı ile ilgili veriler sunulmuş olup, kestirilen rotor akısı ile gerçekleşen rotor akısı, stator akımı ve stator faz voltajı ile ilgili grafikler için EK-D'ye bakılabilir. Aynı değişken için elde edilen veriler birbirlerine çok yakın olduklarından aynı grafik üzerinde verildiğinde üst üste çakışmalar oluşmuştur. Bu nedenle aynı değişkene ait veriler farklı grafikler üzerinde verilmek zorunda kalınmıştır. Örneğin rotor hızının kestirilen ve gerçekleşen değerleri için farklı grafikler kullanılmıştır.

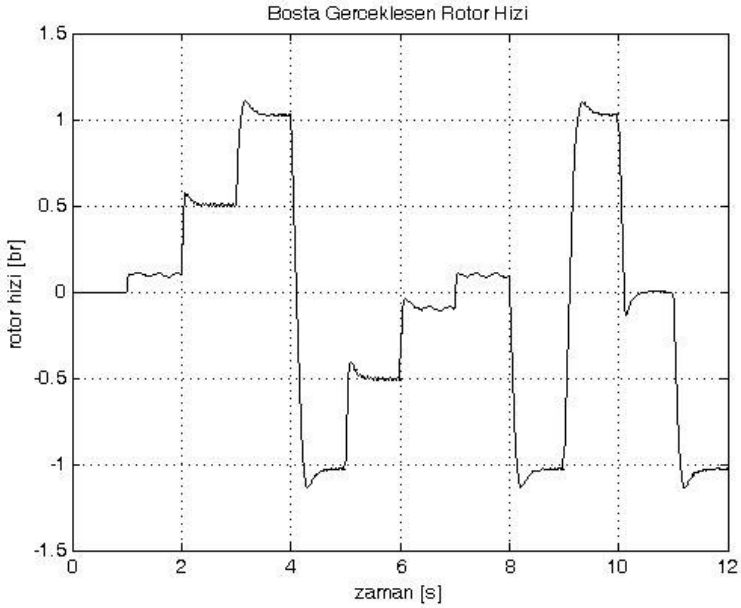
6.1.1 Vektör Kontrollü Asenkron Makinenin Boşta Çalışma Simülasyonu

Öncelikle, makine boşta çalıştırılarak, 12 s boyunca sıfır hız da dahil olmak üzere, sistemin değişik rotor hızı referanslarına cevabı incelenmiştir. Bu çalışmanın boşta yapılmasının nedeni, uygulama aşamasında yapılan deneyler de makinenin boşta çalıştırılmasıdır. Makineyi yükleyecek düzenek olmadığından, makine

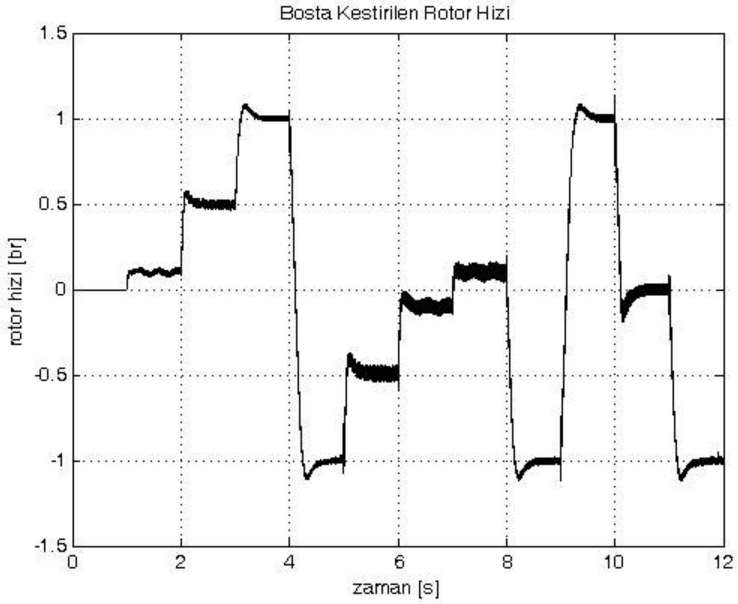
uygulamada boşta çalıştırılmıştır. Şekil 6.1, Şekil 6.2 ve Şekil 6.3'te sırasıyla asenkron makine boşta çalışırken referans rotor hızı, gerçekleşen hızı ve kestirilen rotor hızı verilmiştir.



Şekil 6.1 Boşta çalışma simülasyonunda referans rotor hızları



Şekil 6.2 Boşta çalışma simülasyonunda gerçekleşen rotor hızları



Şekil 6.3 Boşta çalışma simülasyonunda kestirilen rotor hızları

Tablo 6.1 Boşta çalışma için elde edilen simülasyon sonuçları

Zaman [s]	Referans Rotor Hızı [br]	Makine Yükü [br]	Gerçeklş. Rotor Hızı [br]	Kestirilen Rotor Hızı [br]	Referans Hıza Uyma [br]	Kestirilen Hızın Doğr. [br]	Cevap Zamanı [s]
0-1	0.0	0.0	0.0	0.0	0.0000	0.0000	0.0
1-2	0.1	0.0	0.1	0.1	0.0000	0.0000	0.29
2-3	0.5	0.0	0.505	0.5	0.0100	0.0100	0.14
3-4	1.0	0.0	1.025	1.0	0.0250	0.0250	0.19
4-5	-1.0	0.0	-1.025	-1.0	0.0250	0.0250	0.38
5-6	-0.5	0.0	-0.505	-0.5	0.0100	0.0100	0.16
6-7	-0.1	0.0	-0.1	-0.1	0.0000	0.0000	0.29
7-8	0.1	0.0	0.1	0.1	0.0000	0.0000	0.10
8-9	-1.0	0.0	-1.025	-1.0	0.0250	0.0250	0.42
9-10	1.0	0.0	1.025	1.0	0.0250	0.0250	0.53
10-11	0.0	0.0	0.0	0.0	0.0000	0.0000	0.50
11-12	-1.0	0.0	-1.025	-1.0	0.0250	0.0250	0.40

Boşta yapılan simülasyon çalışmaları göstermiştir ki, sistem sıfırdan anma hızına kadar her hızda kararlı olarak çalışabilir. Sistem tüm hızlarda, verilen referans hıza en fazla 0.025 sapmayla uymaktadır. Kestirilen rotor hızı ile gerçekleşen rotor hızı arasındaki farkın gerçekleşen rotor hızının 0.025'i civarında olduğu gözlenmiştir. Tablo 6.1'de boşta çalışma simülasyonu için elde edilen sonuçlar özetlenmiştir.

Tablo 6.1 ile Şekil 6.1, Şekil 6.2 ve Şekil 6.3'te verilen grafiklerden de görüleceği gibi önerilen vektör kontrol yöntemi rotor hız algılayıcılı asenkron makine sürücülerinin yerini alabilecek kadar yüksek doğrulukta hız kestirimi yapmakta ve aynı şekilde verilen referans hıza uymaktadır.

Önerilen vektör kontrol yönteminin cevap zamanı ilk hız ile son hız arasındaki farka bağlı olarak ortalama 0.20 s civarında olmaktadır. Bu değer, referans hız ile gerçekleşen hız arasındaki farkın, referans hızın 0.1'ine ulaştığı süre olarak alınmıştır. Elde edilen cevap zamanı oldukça iyidir.

6.1.2 Vektör Kontrollü Asenkron Makinenin Yükte Çalışma Simülasyonu

İkinci olarak yapılan simülasyon çalışmasında asenkron makinenin DSP tabanlı vektör kontrolü için önerilen yöntemin her iki yönde değişik yüklerdeki dinamik performansı incelenmeye çalışılmıştır. Bu çalışmada referans rotor hızı 0.8 br. iken ve -0.8 br. iken yük şartları değiştiğinde makine hızının korunup korunmadığı gözlenmiştir. Burada hızın 0.8 br. olarak seçilmesinin nedeni birim sistem oluşturulurken temel olarak alınan hızın senkron hız olmasıdır. Makinenin anma yükündeki anma hızı yaklaşık 0.90 br. senkron hızdır. Bu nedenle simülasyon için seçilen referans hız 1.0 br. den daha küçük bir değer olarak seçilmiştir. Şekil 6.4, Şekil 6.5, Şekil 6.6, ve Şekil 6.7'de sırasıyla makineye uygulanan yük torkları, referans rotor hızları, gerçekleşen rotor hızları ve kestirilen rotor hızları verilmiştir.

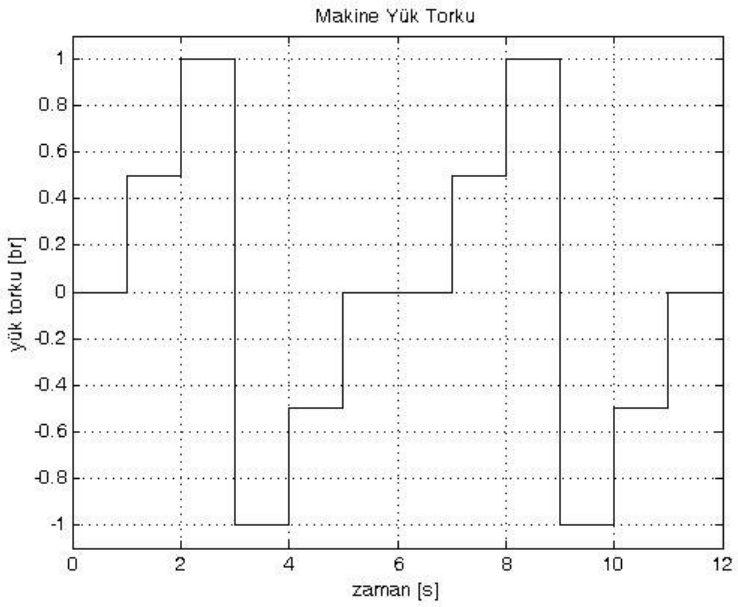
Şekil 6.6'da verilen grafikten gerçekleşen rotor hızının her iki yönde de pozitif ve negatif yük torkları için korunduğu anlaşılmaktadır. Ayrıca makine referans hıza uymaya devam etmektedir. Tablo 6.2'de önceki çalışma için oluşturulan tabloya benzer bir tablo verilmiştir. Tablo 6.2 incelendiğinde önerilen DSP tabanlı vektör yönteminin değişik yük şartları altında çalışma noktasından ayrılmayıp kararlılığını

koruduğu görülmektedir. Referans rotor hızı ile gerçekleşen rotor hızı arasındaki fark ortalama olarak referans hızın 0.025 civarında olmaktadır.

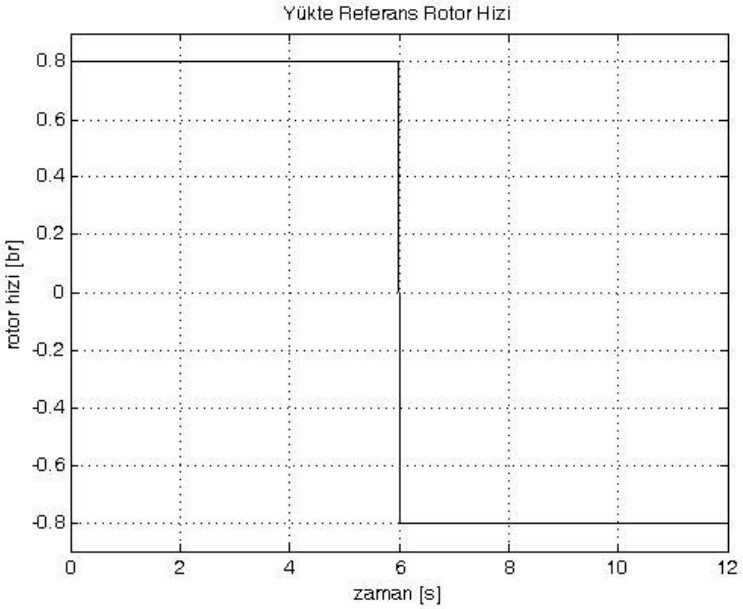
Önerilen sistemin yük değişimlerine karşı cevabının çok hızlı olduğu Tablo 6.2'den açıkça görülmektedir. Örnek olarak yük torku 1.0 br. iken -1.0 br. yapıldığında sadece 0.15 s'de rotor hızı kararlı duruma ulaşmaktadır. Daha küçük değişimler için sıfır cevap zamanı gözlenmiştir. Hız referansı değiştiğinde daha büyük cevap zamanları gözlenmiştir.

Diğer taraftan, hem Tablo 6.1 hem de Tablo 6.2'de kestirilen hız ile referans hızın tamamen aynı olduğu görülmektedir. Sonuç olarak, referans rotor hızının kestirilen hıza uyumunda hata sıfırdır. Bunun anlamı, hız algılayıcı ile birlikte sistemin sürekli hız hatası sıfır yapılabilir.

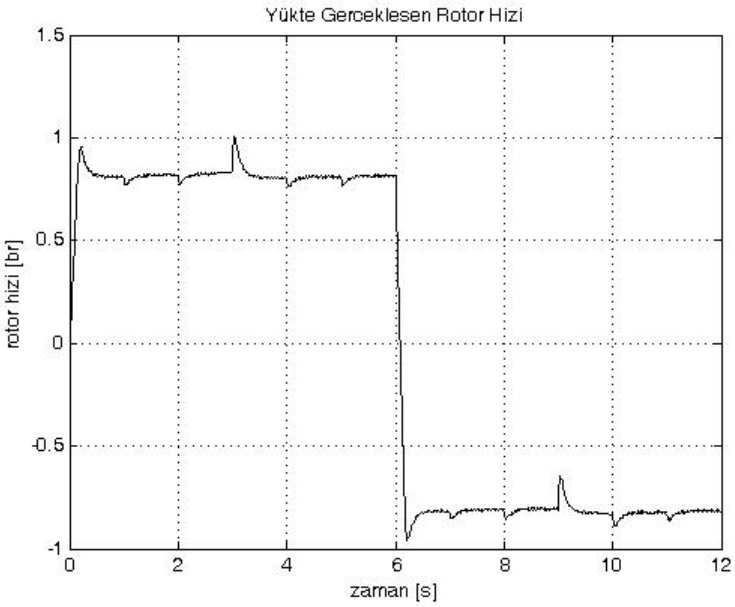
Tabii ki, sistemin uygulama aşamasındaki performansı gerçek sistemle yapılan deneyler ile anlaşılabilir. Buraya kadar yapılan simülasyon çalışmalarından, sistemin uygulanabilir olduğu anlaşılmaktadır. Uygulama aşamasında simülasyon çalışmasından farklı olarak parametre ve ölçüm gürültüleriyle karşılaşılacaktır. Ancak makine modelini oluşturan durum denklemlerindeki durum değişkenlerinin kestirimi için kullanılan EKF algoritması gürültüleri filtre etme yeteneğine sahiptir.



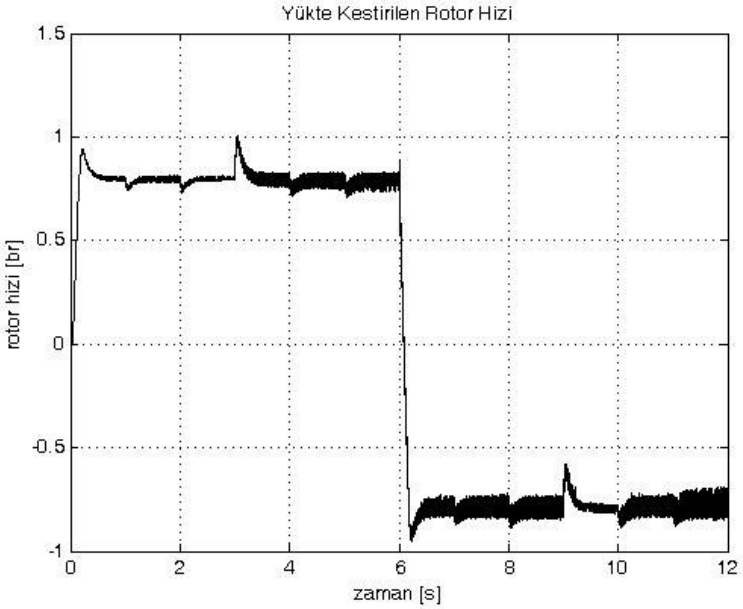
Şekil 6.4 Yüklü çalışma simülasyonunda makine yükleri



Şekil 6.5 Yüklü çalışma simülasyonunda referans rotor hızları



Şekil 6.6 Yüklü çalışma simülasyonunda gerçekleşen rotor hızları



Şekil 6.7 Yüklü çalışma simülasyonunda kestirilen rotor hızları

Tablo 6.2 Yüklü çalışma için elde edilen simülasyon sonuçları

Zaman [s]	Referans Rotor Hızı [br]	Makine Yükü [br]	Gerçeklş. Rotor Hızı [br]	Kestirilen Rotor Hızı [br]	Referans Hıza Uyma [br]	Kestirilen Hızın Doğr. [br]	Cevap Zamanı [s]
0-1	0.8	0.0	0.81	0.8	0.0125	0.0125	0.28
1-2	0.8	0.5	0.82	0.8	0.0250	0.0250	0.00
2-3	0.8	1.0	0.83	0.8	0.0375	0.0375	0.00
3-4	0.8	-1.0	0.80	0.8	0.0000	0.0000	0.15
4-5	0.8	-0.5	0.81	0.8	0.0125	0.0125	0.00
5-6	0.8	0.0	0.81	0.8	0.0125	0.0125	0.00
6-7	-0.8	0.0	-0.81	-0.8	0.0125	0.0125	0.29
7-8	-0.8	0.5	-0.81	-0.8	0.0125	0.0125	0.00
8-9	-0.8	1.0	-0.83	-0.8	0.0375	0.0375	0.00
9-10	-0.8	-1.0	-0.83	-0.8	0.0375	0.0375	0.11
10-11	-0.8	-0.5	-0.82	-0.8	0.0250	0.0250	0.00
11-12	-0.8	0.0	-0.81	-0.8	0.0250	0.0250	0.00

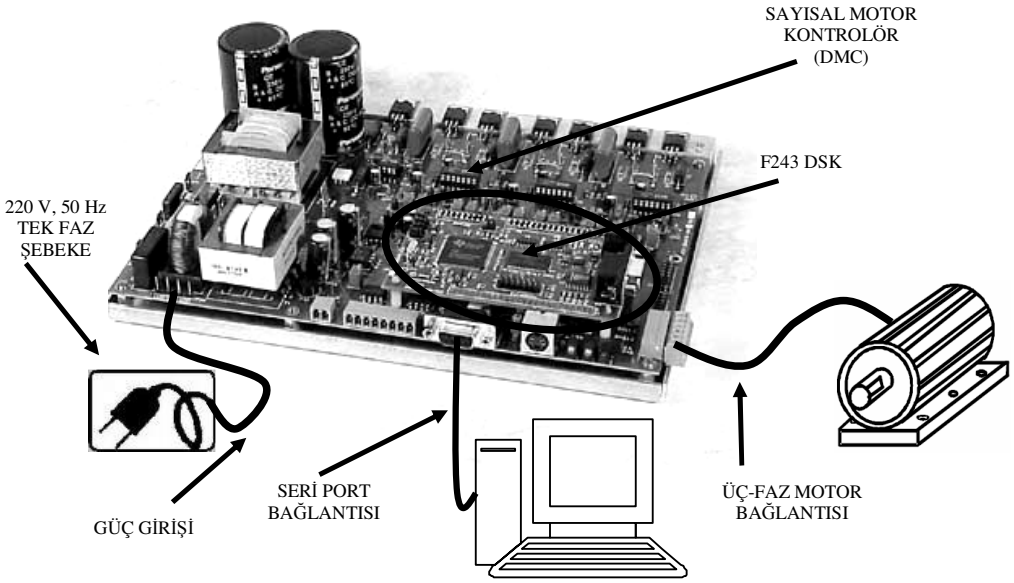
6.2. TMS320F243 Tabanlı Asenkron Motor Sürücüsü ile Yapılan Uygulama Sonuçları

Denemeler bölüm 5'de verilen donanım üzerinde ve yine aynı bölümde verilmiş olan simülasyon tasarımına uygun olarak düzenlenmiş önerilen yazılım ile yapılmıştır. TMS320F243 Tabanlı motor kontrolör kartı devre şeması EK-B'de ve bu motor kontrolör kartına uygun olarak düzenlenmiş asenkron motorun EKF tabanlı vektör kontrol yazılımı EK-C'de verilmiştir. F243 DSK kartı devre şeması için ise Bölüm 8'de verilen "F243 DSK Technical Reference" isimli kaynağa bakılmalıdır.

Şekil 6.8'de denemelerin yapıldığı düzenek görülmektedir. Görüldüğü gibi düzenek kişisel bilgisayar (PC), asenkron makine, sayısal motor kontrolör kartı (DMC) ve F243 DSK kartından oluşmaktadır. Şekilde kuvvetli akım geçen hatlar diğerlerinden daha kalın çizilmiştir. Sistemde PC haricinde kalan tüm kartların beslemesi tek-faz 220 V, 50 Hz şebekeden sağlanmaktadır. Gerekli diğer beslemeler motor kontrolör üzerinde 220 V AC girişten üretilmektedir. Yine F243 DSK kartı için gerekli 5 V DC voltaj da motor kontrolör kartından sağlanmaktadır.

Şekilde görülen PC ölçümlerin ve F243 DSKnın izlenmesi ile kontrol yazılımının F243 DSKya yüklenmesi amacıyla kullanılmaktadır. PC ile F243 DSK kartı arasındaki haberleşmeler seri port üzerinden sağlanmaktadır. Bu amaçla F243 DSK kartı ile birlikte gelen "*Code Explorer*" yazılımı kullanılmaktadır. Sisteme enerji verildiğinde F243 DSK içerisinde bulunan monitör yazılım çalışmaya başlamakta ve PCden bir bağlantı isteğini beklemektedir. PC tarafında "*Code Explorer*" yazılımı çalıştırıldığında PC ile F243 DSK arasında bir seri bağlantı oluşturmaktadır. Böylece kullanıcı F243 DSK üzerindeki verileri izleyebilmekte ve denemek istediği kontrol yazılımını F243 DSK kartına yükleyebilmektedir. "*Code Explorer*" yazılımının bunun dışında herhangi bir özelliği yoktur. F243 DSK içerisindeki monitör yazılım ise kalıcı belleğe imalat aşamasında yüklenmiştir. Kullanıcının bu yazılım üzerinde herhangi bir düzenleme yapma imkanı yoktur.

Sayısal motor kontrolör (DMC) kartı ve F243 DSK kartı üzerinde tamamen birbirlerinin aynı olan portlar mevcuttur. Bunlar genel giriş/çıkış uçları ve 5 V, DC beslemenin bulunduğu I/O genişletme portu ve ADC terminallerinin bulunduğu



Şekil 13 TMS320F243 Tabanlı asenkron motor sürücü düzeneği

analog genişletme portudur. Bu portlar ve uç bağlantıları için EK-B'de verilen DMC kartı devre şemasına bakılabilir. Bağlantı için DMCde ve F243 DSKda dişi konnektörler vardır. İki kart bu konnektörler aracılığıyla birbirlerine doğrudan bağlanmışlardır. I/O genişletme portu PWM sinyallerinin aktarılması ve analog genişletme portu voltaj ve akım ölçümleri için kullanılırlar.

Denemeler sırasında kullanılan asenkron motor kısım 3.2'de değerleri verilen 0.75 kW gücündeki sincap-kafes rotorlu asenkron makinedir. Şekil 6.8'de görüldüğü gibi motor ile kontrolör arasında güç ileten hattan başka herhangi bir bağlantı yoktur. Bu hatlar üzerindeki gerekli tüm ölçümler motora gelmeden önce kontrolörde yapılmaktadır. Bunun dışında motor üzerinden herhangi bir ölçüm alınmamaktadır. Yine motor miline bağlı herhangi bir yük de yoktur. Yapılan denemeler motor boşta çalışırken yapılmaktadır.

Şekil 6.9, Şekil 6.10 ve Şekil 6.11'de sırasıyla ADC ile doğrudan ölçülmüş olan gerçekleşen stator faz-A akımı, EKF ile kestirilen rotor alan akısı "*d*" bileşeni ve EKF ile kestirilen rotor hızı verilmiştir. Şekillerde verilen grafikler deney sistemi çalışırken çevrim-içi olarak izlenememektedir. Verilen grafikleri elde etmek için, öncelikle, her kontrol çevrimindeki ADCler tarafından ölçülüp birim sistemde 4.12

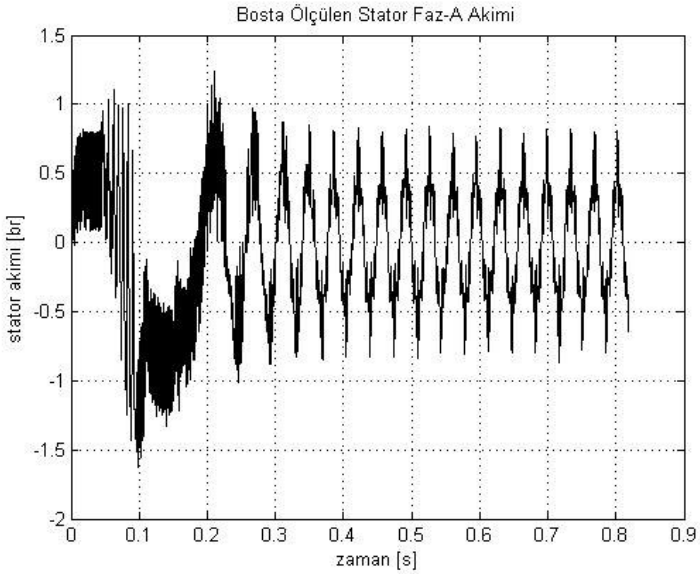
formatında saklanan değerler, sırayla F243 DSK RAM belleğinin boş bir alanına yazdırılmaktadır. F243 DSK RAM üzerinde boş 12 KB bellek bulunmaktadır. Bu bellek `0x9000-0xbfff` arasındaki bölgeden oluşmaktadır. Bu bölge her deneme sonunda üç adet veriyi saklayacak şekilde `0x9000-0x9fff`, `0xa000-0xafff` ve `0xb000-0xbfff` olarak üç alt bölgeye ayrılmıştır. Böylece, her bir bölgede, her biri bir örnekleme anından elde edilen 4096 adet veri sırayla saklanabilmektedir. Daha sonra, RAM bellekte saklanan veriler "*Code Explorer*" programı yardımıyla, PCde data-file olarak saklanmaktadır. Elde edilen bu data-file ise M/S ortamında matris veriler olarak okunmakta ve yine M/S figure olarak çizdirilmektedir. Şekil 6.9, Şekil 6.10 ve Şekil 6.11'de verilen grafikler bu yolla elde edilmişlerdir. Tabii ki, bu verilerin, EKF kestiriminden elde edildiği açıktır. Şekil 6.8'de verilen deney düzeneğinden de görüleceği gibi asenkron makine üzerinden ölçüm almak için herhangi bir ölçü cihazı yoktur.

Şekil 6.9, Şekil 6.10 ve Şekil 6.11'de verilen grafikleri elde etmek için birim referans rotor hızı komutu uygulanmıştır. Görüldüğü gibi makine bu referans rotor hızına uymaktadır. Birim değerler TMS320F243 DSP üzerinde 4096 sayısına karşılık gelir. Bunun nedeni 4.12 sabit noktalı gösterim formatıdır. Bu formatta 1.0 değeri `0x1000` olarak temsil edilmektedir. Bilindiği gibi, `0x1000` onaltı tabanında bir sayıdır ve onluk sistemde 4096 sayısına karşılık gelir. F243 DSKdan okunan değerler 4096'ya bölünerek grafikler çizdirilmiştir.

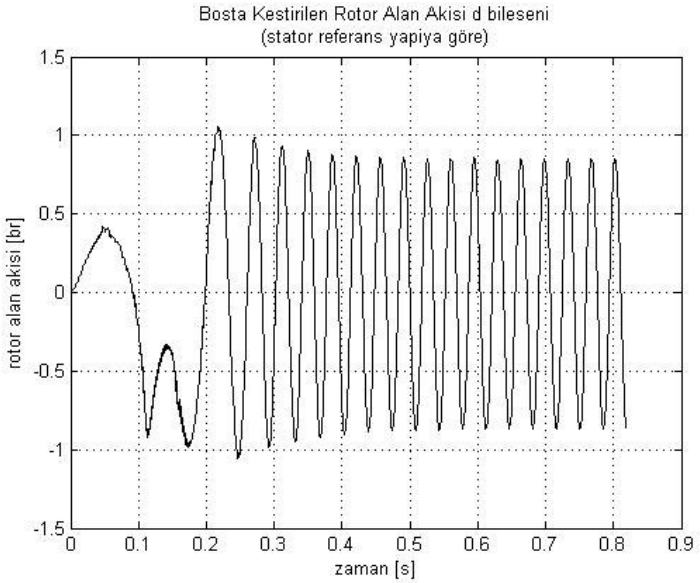
İki örnek arasındaki süre örnekleme süresi olup 200 μs dir. Grafiklerde yatay eksen de 4096 örnek alındığı görülmektedir. Böylece bu örneklerin alındığı toplam süre $4096*200*10^{-6}$ = 819.2 ms dir. Buna göre, sistemin adım cevabı 0.36 ms civarındadır. Ayrıca, referans rotor hızı ile kestirilen rotor hızı arasındaki birim hata, simülasyon çalışmasında olduğu gibi sıfır civarında gerçekleşmektedir. Cevap zamanı da simülasyon çalışması ile uyumludur.

Bunun dışında, alan yönlendirme kontrolü yerine makine boşta iken altı-adım inverter ile frekans kontrolü yapılmıştır. Bunun amacı hız kestiriminin doğruluğunu test etmektir. Bilindiği gibi makine boşta çalışırken yaklaşık senkron devirde dönmektedir. Bu çalışmada, makineye altı-adım inverter ile anma voltajında ve frekansında çalıştırılarak senkron devirde dönmesi sağlanmıştır. Bununla birlikte EKF algoritması da aynı anda işletilerek rotor hız kestirimi de yapılmıştır. Bu

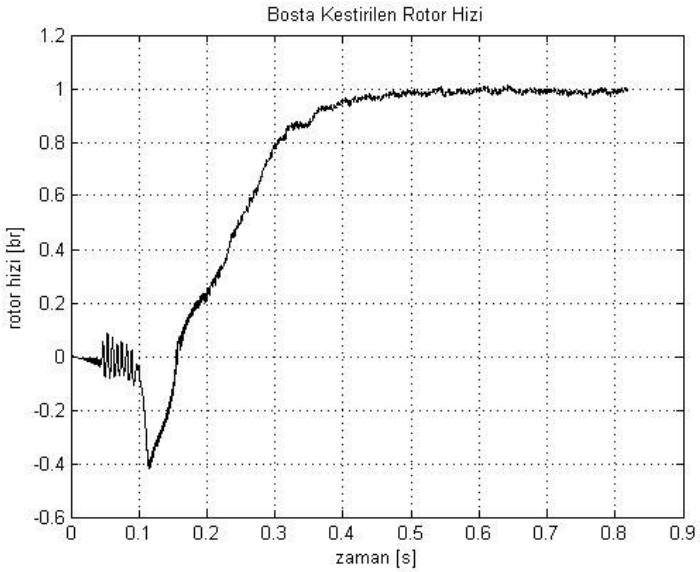
çalışma sonucunda da, senkron devirde dönen asenkron makinenin rotor hızının sıfır hata ile EKF tarafından kestirildiği gözlenmiştir. Bu şekilde hız kestiriminin doğruluğu test edildikten sonra makineye alan yönlendirme kontrolü uygulanarak yukarıda açıklanan sonuçlar elde edilmiştir.



Şekil 6.9 TMS320F243 Tabanlı sürücü ile yapılan boşta çalışma deneyinde gerçekleşen stator Faz-A akımı



Şekil 6.10 TMS320F243 Tabanlı sürücü ile yapılan boşta çalışma deneyinde kestirilen rotor alanı akısı "*d*" bileşeni



Şekil 6.11 TMS320F243 Tabanlı sürücü ile yapılan boşta çalışma deneyinde kestirilen rotor hızı

7. SONUÇLAR VE ÖNERİLER

7.1. Sonuçlar

Bu tez çalışması sırasında, alan yönlendirme kontrolü, hız ve parametre kestirim yöntemleri, inverter modülasyon türleri, simülasyon ortamları, assembly veya C ortamında program geliştirme gibi, bir çok konuda oldukça fazla yöntem üzerinde incelemeler yapılmış, mevcut olan imkanlar çerçevesinde en uygun olan yöntemlerin seçilmesine dikkat edilmiştir. Bu çalışmalar sonucunda, en uygun yöntemlerin, önceki bölümlerde açıklanmaya çalışılan ve bu tez çalışmasında gerçekleştirilen yöntemler olduğu sonucuna varılmıştır.

Burada verilen kaynaklardaki yöntemler öncelikle simülasyon ortamında test edilerek, uygulanabilirliği araştırılmıştır. Literatürde bahsi geçen bir çok yöntemin simülasyon ortamında iyi sonuçlar vermesine rağmen, gerçek bir sistem üzerinde test edildiğinde çalışmasının rotor hızı gibi bir çok sınırlar dahilinde olabileceği görülmüştür.

En fazla üzerinde durulan konu hız kestirimi olmuştur. Özellikle, MRAS yöntemi üzerinde durulmuştur. Yöntem çıkışları aynı durum değişkeni olan iki makine modelinin karşılaştırılmasına dayanır. Kestirilmeye çalışılan değişken adaptif modelinin girişidir. İki modelin çıkışı kestirilmeye çalışılan değişken ayarlanarak eşitlenir. Böylece ölçülemeyen değişken kestirilmiş olur. Bu yöntem simülasyon ortamında oldukça iyi sonuçlar vermektedir. Ancak, gerçek sistem üzerinde kullanıldığında hiçbir zaman anlamlı sonuçlar elde edilememiştir. Sistemin kararsızlığa neden olan tarafı açık çevrim integrasyon içermesidir. Açık çevrim integrasyon simülasyon ortamında oldukça rahat çalıştırılabilmektedir. Ancak, gerçek sistemdeki gürültüler, DSPde sayıların yeterince hassas ifade edilememesi, ölçüm devrelerinde meydana gelen DC bileşenler gibi nedenlerle, integrasyon off-set noktasında kayma meydana gelmekte ve sürüklenme olarak adlandırılan durum oluşmamaktadır. Sürüklenme sonucunda sayılar DSPnin gösterebileceği sınırları aşmakta ve taşmalar meydana gelmektedir. Bu durumun giderilebilmesi için de yöntemler mevcuttur, ancak bu yöntemlerle elde edilen sonuçlarda asıl veri

bozulmakta ve kontrolü kararsız hale getirecek yanlış veriler elde edilmektedir. Sonuç olarak, bu yöntemin işletilebilmesi için harcanan oldukça fazla emeğe rağmen, bu yöntemden vazgeçilmiş ve başka yöntemler üzerinde durulmuştur.

Hız kestirimi için yapılan araştırmalar sonucunda EKF algoritmasının bu iş için uygun olduğuna karar verilmiştir. Bu algoritma sayesinde, önceki bölümde de açıklandığı gibi, hız algılayıcılı veya hız algılayıcısız olarak çalıştırılan diğer pek çok yöntem ile elde edilemeyen, sıfır hızda çalışmanın gerçekleştirilebileceği görülmüştür. EKF algoritmasının tek kötü tarafı oldukça fazla matris işlemidir ve bu işlemler tüm alan yönlendirme algoritmasının yaklaşık 15 katı süre almaktadır. Bu durum ise örnekleme zamanını uzatmaktadır. Örnekleme zamanının uzaması, kontrol için tehlikeli bir durum olup, burada da yapılan, örnekleme zamanı içerisinde bir değişkenin yaklaşık olarak sabit kaldığı gibi bazı kabulleri, geçersiz kılabilir. Ancak bu çalışmadaki örnekleme zamanı yapılan bu tür kabuller açısından sorun oluşturmamıştır.

Bunlardan başka, inverter modülasyon teknikleri üzerinde durulmuştur. Başlangıçta, uzay vektör darbe genişlik modülasyonu (SVPWM) kullanılarak çalışmalar yapılmıştır. Bu yöntem stator akımının veya voltajının uzay vektörü olarak düşünülmesine ve bu vektörün altı-adım inverterde de kullanılan altı adet vektör kullanılarak elde edilmesine dayanır. Ancak, bu yöntem için oldukça fazla hesaplama gerektiğinde EKF ile birlikte kullanılmamasına karar verilmiştir. Aksi halde, örnekleme zamanında biraz daha uzama olacaktır. Ayrıca, bu yöntemde akım örneklemesi, histerisiz band akım regülatörü modülasyonuna göre daha zordur. Histerisiz band akım regülatörü için kullanılan algoritma kısım 3.4 ve kısım 5.3'te açıklandığı gibi oldukça basittir ve kısa zaman alır. Bu nedenle SVPWM modülasyon yönteminden vazgeçilip HBCR modülasyon yöntemi tercih edilmiştir.

Diğer bir inceleme konusu da simülasyon ortamıdır. Öncelikle, asenkron makineyi ve denetleyici sistemi simüle etmek için sürekli-simülasyon ortamı gereklidir. İncelenen simülasyon paketleri MATLAB/SIMULINK, acslXtreme, ve SCILAB/SICOS paketleridir. Başlangıçta, ilk paket kullanılmış ve Simulinkte hazır bulunan bloklar yardımıyla tasarım yapılmıştır, ancak bu yöntemle simülasyon oldukça uzun sürmektedir. İkinci olarak acslXtreme paketi üzerinde durulmuştur. Bu pakette grafik ortam olmayıp, metin tabanlıdır. Bu paketle yapılan denemelerde yine

oldukça uzun simülasyon zamanları meydana gelmiştir. Üçüncü paket olan SCILAB/SCICOS paketi grafik ortama sahiptir, ve serbest dağıtılmaktadır. Ancak bu paket sürekli geliştirilmekte olup, çalışması çok kararsızdır ve anlamlı sonuçlar elde edilememektedir. Son olarak, simülasyonun C ortamında yapılmasına karar verilmiştir, ancak, bu durumda her yapılacak iş için yazılımın tek tek oluşturulması gereklidir. Ayrıca, standart sonuçların elde edilemeyeceği sonucuna varılmıştır. Tüm bunlar dikkate alındığında, standart simülasyon ortamı, hazır fonksiyonlar, grafik ve metin ortamının aynı anda kullanılabilmesi, simülasyon için kullanılacak modelin sadeliği, simülasyon hızı gibi bir çok faydayı bir arada sağlayan MATLAB/SIMULINK paketinde C S-Function'ın kullanılmasına karar verilmiştir. Bu şekilde elde edilen simülasyon hızı gerçek hızı dahi aşabilmektedir.

Son olarak, DSP yazılım ortamları üzerinde durulmuştur. Bu konudan daha önce bahsedilmiştir. Başlangıçta, yazılımın kolay ve anlaşılır olacağı düşüncesiyle C ortamı kullanılmıştır. Ancak bu şekilde edilen DSP programı, her işlem için bir çok alt-program ve dallanma içerdiğinden, bir çevrimin çalışma süresi örnekleme süresini aşmaktadır. Bu nedenle, DSP yazılım ortamı olarak metin editörü assebmly dili derleyicisi kullanılmasına karar verilmiştir. Bu şekilde elde edilen DSP programının bir çevriminin çalışması örnekleme zamanından daha kısa sürmektedir.

Bunun dışında önerilen alan yönlendirme kontrol sistemin beklentileri fazlasıyla karşıladığı simülasyon ve deney sonuçlarından açıkça görülebilir.

7.2. Öneriler

Simülasyon açısından bu çalışmanın yeterli olacağı düşünülmektedir. Ancak, uygulama açısından sistemi daha da geliştirmek mümkün olabilir. Bu çalışmada sistem maliyeti minimum düzeyde tutulmaya çalışılmıştır. Maliyetin azaltılması ve verimin artırılması için çeşitli yöntemler kullanılmıştır. Örneğin, akım ölçümleri basit bir direnç üzerinde yapılmaktadır. Sistemin özellikle hız sensörsüz çalışması sağlanmaya çalışılmıştır. Hız algılayıcısının sürücüye eklenmesinin maliyeti şimdiki maliyeti iki katına çıkarabilecek düzeydedir.

Bütün bunlara rağmen, tez çalışmasında makine parametre değişimlerin etkileri ihmal edilmiştir. Buda, ayrıca incelenmesi gereken bir konudur. Ancak EKF algoritmasının parametre değişimlerinin sistem kararlılığına ve doğruluğuna etkisini azaltacağı beklenmektedir. Literatürde bu konuyla ilgili bir çok yöntem bulunmaktadır. Ayrıca, bu çalışmada, makine parametrelerinin kullanıcı tarafından elde edilerek sisteme verilmesi gereklidir. İleride bu konuda yapılacak olan çalışmalarda, makine parametre değişimlerinin kontrol işlemine etkileri incelenebilir. Ayrıca, bu parametrelerin, makine akım ve gerilim ölçümlerinden otomatik olarak elde edildiği yöntemler, önerilen kontrol yöntemine eklenebilir.

Bunların dışında donanım kullanıcı ile daha uyumlu olacak biçimde, kontrol girişleri ve göstergeler eklenerek geliştirilebilir. Bu çalışmada herhangi bir kullanıcı ara yüzü olmayıp, giriş ve çıkış basit bir biçimde bilgisayar yardımıyla sağlanmaktadır. Bilgisayar sistemi olmadan sistem kendi kendine işleyecektir. Ayrıca yine, uygulama aşamasında, makinenin değişik yük şartlarında çalışması incelenmemiştir. Simülasyon çalışmasından elde edilen sonuçlar yeterli olduğu düşünülmüştür. Bu da ayrıca incelenebilir. Makine yükü doğrudan miline bağlanacak bir DC motor yardımıyla ayarlanabilir.

Sonuç olarak, uygulama aşamasında makinenin değişik yük şartları altında incelenmesi ve kullanıcı ara yüzü ek çalışmalar yapılabilir.

8. KAYNAKLAR

BAYBURAN, Bahattin; Microsoft Standart C; Beta, İstanbul, 1999

Beguenane, Rachid; Ouhrouche, Mohand A.; Trzynadlowski, Andrzej M.; "A new scheme for sensorless induction motor control drives operating in low speed region"; Mathematics and Computers in Simulation 71 (2006); Elsevier

Bharadwaj, R, A.; G. Parlos, and H. A. Toliyat, "Adaptive Neural Networks-Based State Filter for Induction Motor Speed Estimation," to be presented in the 1999 IEEE Industrial Electronics Society Annual Meeting, San Jose, CA, December 1999

Bharadwaj, Raj M.; Parlos, Alexander G.; Toliyat, Hamid A.; "Neural speed filtering for sensorless induction motor drives"; Control Engineering Practice 12 (2004); Elsevier

BOSE, Bimal K.; "High Performance Control of Induction Motor Drives"; On-line at http://sant.bradley.edu/ienews/98_3/Bose2.pdf

BOSE, Bimal K.; PATEL, Nitin R.; "Quasi-Fuzzy Estimation of Stator Resistance of Induction Motor"; IEEE Transaction on Power Electronics, Vol. 13, No. 3, May 1998.

BROWN, Robert Grover; HWANG, Patrick Y. C.; Introduction to Random Signals and Applied Kalman Filtering; Wiley, USA, 1997

CHAO, K. H.; LIAW, C. M.; "Speed Sensorless Control Performance Improvement of Induction Motor Drive Using Uncertainty Cancellation; IEEE Proc.-Electr. Power Apll., Vol. 147, No. 4, July 2000

CHAPRA, Steven C.; CANALE, Raymond P.; Mühendisler için Sayısal Yöntemler; Literatür, İstanbul, 2003

F243 DSK Technical Reference; Spectrum Digital, Inc.; 1999

FAIZ, J.; HOSSIENI, S. H.; GHANEEI, M.; KEYHANI, A.; PROCA, A.; "Direct Torque Control of Induction Motors for Electric Propulsion Systems"; On-line at http://eewww.eng.ohio-state.edu/~procaa/pdf/direct_torque.pdf

FINN, David; "General Three Phase Motor Controllers"; The University of Queensland; October 2000

Fitzgerald, A.E.; Kingsley, Charles, Jr.; Umans , Stephen D.; Electric Machinery; McGraw-Hill, USA, 1990

Ge, Qiongxuan; Feng, Zhiyue; “Speed Estimated for Vector Control of Induction Motor Using Reduced-order Extended Kalman Filter”; PIMC2000 Conference; IEEE, 2000

Guzinski, Jaroslaw; Abu-Rub, Haithem; Toliyat, Hamid A.; “An Advanced Low-Cost Sensorless Induction Motor Drive”; IEEE Transactions on Industry Applications, Vol. 39, No. 6, November/December 2003

H. de la Vallée Poussin, D. Grenier, F. Labrique, J.-D. Legat, “Looking into the implementation of AC motor control on a fixed point processor”, IEMDC'99, Seattle (USA)

Hilairet, M.; Auger, F.; Darengosse, C.; “Two efficient Kalman filters for flux and velocity estimation of induction motor”; IEEE; 2000

HOLTZ, Joachim; “Sensorless Speed and Position Control of Induction Motors”; 27th Annual Conference of the IEEE Industrial Electronics Society, IECON, Denver/CO, Nov. 29 - Dec. 2, 2001

Kojabadi, H. Madadi; Chang, L.; “Model Reference Adaptive System Pseudoreduced-Order Flux Observer for Very Low Speed and Zero Speed Estimation in Sensorless Induction Motor Drives”; IEEE; 2002

Koubaa, Yassine; “Recursive identification of induction motor parameters”; Simulation Modelling Practice and Theory 12 (2004); Elsevier

KREYSZIG, Erwin; Advanced Engineering Mathematics; Wiley, Singapore, 2006

KUO, Benjamin C.; Automatic Control Systems Fift Edition; Prentice-Hall Inc., New Jersey, 1987

Li, Mengwei; Chiasson, John; Bodson, Marc; Tolbert, Leon M.; “A Differential-Algebraic Approach to Speed Estimation in an Induction Motor”; IEEE Transactions on Automatic Control, Vol. 51, No. 7, July 2006

MOHAN. M; “TMS320F243 Based Sensorless Rotor Field Oriented Control of Induction Motor Using MRAS”; On-line at www.cranessoftware.com/ events/conference/ab11.html

MOHAN, Ned; UNDELAND, Tore M.; ROBBINS, William P.; Power Electronics; Wiley; USA, 2003

Motor Drive and Inverter Designer’s Manual; International Rectifier; 2002

MURPHY, J. M. D.; TURNBULL, F. G.; Power Electronic Control of AC Motors; Pergamon Press, GB, 1989

NOVOTNY, D. W.; LIPO, T. A.; Vector Control and Dynamics of AC Drives; Clarendon Pres, GB, 1998

NOVOTNY, D. W.; LIPO, T. A.; ECE 411 Electromechanical Systems Course Notes; University of Wisconsin WEMPEC, USA, 1986

OUHROUCE, Mohand A.; VOLAT C.; “Simulation of a Direct Field-Oriented Controller for an Induction Motor Using MATLAB/SIMULINK Software Package”; Proceeding of the IASTED International Conference Modelling and Simulation (MS'2000); May 15-17, 2000 - Pittsburgh, Pennsylvania, USA.

Power System Blockset User Guide for Use with Simulink; Mathwork Inc; 2000

RASHID, Muhammad H.; Power Electronics; Prentice Hall, USA, 1993

ROBERTS, G. Stephen; ”Design and implementation of a Three-Phase Induction Motor Control Scheme”; The University of Queensland; October 2001

SAVITCH, Walter; ProblemSolving with C++; Addison Wesley, USA, 2001

Shi, K. L.; Chan, T. F.; Wong, Y. K.; Ho, S. L.; “Speed Estimation of an Induction Motor Drive Using an Optimized Extended Kalman Filter”; IEEE Transactions on Industrial Electronics, Vol. 49, No. 1, February 2002

Simulink Model Based and System Based Design User Guide; Mathwork Inc; 2004

TMS320F243/F241/F242 DSP Controller Reference Guide System and Peripherals; On-line at http://ti.com; January 2000

TMS320C2x/2xx/5x Optimizing C Compiler User’s Guide; On-line at http://ti.com; August 1999

Toqeer, Raja S.; Bayindir, N. Suha; “Speed estimation of an induction motor using Elman neural network”; Neurocomputing 55 (2003); Esevier

Toliyat, Hamid A.; Levi, Emil; Raina, Mona; “A Review of RFO Induction Motor Parameter Estimation Techniques”; IEEE Transactions on Energy Conversion, Vol. 18, No. 2, June 2003

TRZYNADLOWSKI, Andrzej M.; The Field Orientation Priciple in Control of Induction Motors; Kluwer Academic Publishers, USA, 1994

Vaclavec, Pavel; Blaha, Petr; “Lyapunov-Function-Based Flux and Speed Observer for AC Induction Motor Sensorless Control and Parameters Estimation; IEEE Transactions on Industrial Electronics, Vol. 53, No. 1, February 2006

WILDIE, Theodore; Electrical Machines, Driver and Power Systems; Prentice Hall, New Jersey, 2000

Zidan, H.; Fujii, S.; Hanamoto, T.; Tsuji, T.; “Simple Sensor-less Vector Control System for Variable Speed Induction Motor”; T.IEE Japan, Vol. 120-D, No. 10, 2000

EKLER

EK-A SİMÜLASYON DOSYALARI

EK-B SAYISAL MOTOR KONTROLÖR KARTI ORCAD ŞEMALARI

EK-C DSP YAZILIMI

EK-D SİMÜLASYON ÇALIŞMASINDAN ELDE EDİLEN GRAFİKLER

EK-A SİMÜLASYON DOSYALARI

Bu ekte simülasyon aşamasında kullanılan C S-Function ve birlikte kullanılan dosyalar verilmiştir. Aşağıda sırasıyla bu dosyalar bulunmaktadır.

im_par.h dosyası (asenkron makine parametrelerini içerir)

```
#define Rs  9.9
#define Rr  12.345
#define Ls  0.661
#define Lr  0.661
#define Lm  0.590
#define Pn  4.
#define J   0.008
#define In  1.9
#define Vn  127.
#define fn  28.8636
#define dt  200.e-6
#define pi  3.14
#define iDn 0.55982
#define id_lim 1.8
#define hb 0.05
#define G0 (80*dt)
#define G1 (10.)
#define G2 (2./Pn*Lr/Lm)
```

matopt.c dosyası (matris operasyonlarını için kullanılan fonksiyonları içerir)

```
void matT(real_T *in, real_T *out, int_T r, int_T c)
    {
        int_T i,j;
        real_T temp[25], *t, *ptr;
        ptr = in;
        for(i=0;i<r;i++)
        {
            for(j=0;j<c;j++)
            {
                t = temp + i + r*j;
                *t = *ptr++;
            }
        }
        t = temp; ptr = out;
        for(i=0;i<(r*c);i++){*ptr++ = *t++;}
    }

void matMul(real_T *in1, real_T *in2, real_T *out, int_T r, int_T
rc, int_T c)
    {
        int_T i,j,k;
        real_T temp[25], *ptr1, *ptr2, *ptr3, tmp;
        ptr1 = temp;
        for(i=0;i<r;i++)
        {
            for(j=0;j<c;j++)
            {
                tmp = 0.; ptr2 = in1 + i*rc;
```

```
                for(k=0;k<rc;k++)
                {
                    ptr3 = in2 + j + c*k;
                    tmp += (*ptr2++)*(*ptr3);
                }
                *ptr1++ = tmp;
            }
        }
        ptr1 = temp;
        for(i=0;i<(r*c);i++){*out++ = *ptr1++;}
    }

void matInv(real_T *in, real_T *out)
    {
        int_T i;
        real_T temp[4], tmp;
        tmp = in[0]*in[3] - in[1]*in[2];
        temp[0] = in[3]/tmp; temp[1] = -in[1]/tmp;
        temp[2] = -in[2]/tmp; temp[3] = in[0]/tmp;
        for(i=0;i<4;i++){out[i] = temp[i];}
    }
```

impu.c dosyası (asenkron makine için kullanılan C S-Function, makine sürekli-zaman birim modelini içerir)

```
#define S_FUNCTION_NAME  impu
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include "im_par.h"
#include <math.h>
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }
    ssSetNumContStates(S, 5);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 3);
    ssSetInputPortRequiredContiguous(S, 0, true);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 5);
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 26);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
    ssSetOptions(S, 0);
}
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
```

```
    ssSetOffsetTime(S, 0, 0.0);
}
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
  static void mdlInitializeConditions(SimStruct *S)
  {
    real_T *x0 = ssGetContStates(S);
    int_T ncs = ssGetNumContStates(S);
    int_T  lp;
    for (lp=0;lp<ncs;lp++) {
        *x0++=0.0;
    }
  }
#endif /* MDL_INITIALIZE_CONDITIONS */
#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
  static void mdlStart(SimStruct *S)
  {
      int_T i, j;
      real_T *A = ssGetRWork(S);
      real_T *a = ssGetRWork(S);
      real_T Ib = In*sqrt(2.);
      real_T Vb = Vn*sqrt(2.);
      real_T Wb = 2.*pi*fn;
      real_T Phib = Vb/Wb;
      real_T Zb = Vb/Ib;
      real_T Tb = 3./2.*Vb*Ib/Wb;
      real_T rs = Rs/Zb;
      real_T rr = Rr/Zb;
      real_T xs = Wb*Ls/Zb;
      real_T xr = Wb*Lr/Zb;
      real_T xsig = (Ls - Lm*Lm/Lr)*Wb/Zb;
      real_T xm = Wb*Lm/Zb;
      real_T Tr = xr/rr/Wb;
      real_T Tm = J*Wb/Tb/(Pn/2.);
      real_T Ts2 = 1./(rs*Wb/xsig + xm*xm/xsig/xr/Tr);
      a[0] = -1./Ts2; a[1] = 0; a[2] = xm/xsig/xr/Tr;
      a[3] = xm/xsig/xr*Wb;
      a[4] = 0; a[5] = -1./Ts2; a[6] = -xm/xsig/xr*Wb;
      a[7] = xm/xsig/xr/Tr;
      a[8] = xm/Tr; a[9] = 0; a[10] = -1./Tr; a[11] = -Wb;
      a[12] = 0; a[13] = xm/Tr; a[14] = Wb; a[15] = -1./Tr;
      a = a + 16;
      a[0] = Wb/xsig; a[1] = 0;
      a[2] = 0; a[3] = Wb/xsig;
      a[4] = 0; a[5] = 0;
      a[6] = 0; a[7] = 0;
      a = a + 8;
      a[0] = xm/Tm/xr*(Pn/2); a[1] = 1./Tm;
  }
#endif /*  MDL_START */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T  i;
    real_T *y = ssGetOutputPortRealSignal(S,0);
    int_T  ny = ssGetOutputPortWidth(S,0);
    real_T *x = ssGetContStates(S);
    for (i = 0; i < ny; i++) {*y++ = *x++;}
}
```

```
#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  static void mdlDerivatives(SimStruct *S)
  {
      const real_T *u  = (const real_T*) ssGetInputPortSignal(S,0);
      real_T *dx  = ssGetdX(S);
      real_T *x = ssGetContStates(S);
      real_T *A = ssGetRWork(S);
      real_T *B, *G;
      B = A + 16; G = B + 8;
      dx[0] = A[0]*x[0] + A[2]*x[2] + A[3]*x[3]*x[4] + B[0]*u[0];
      dx[1] = A[5]*x[1] + A[6]*x[2]*x[4] + A[7]*x[3] + B[3]*u[1];
      dx[2] = A[8]*x[0] + A[10]*x[2] + A[11]*x[3]*x[4];
      dx[3] = A[13]*x[1] + A[14]*x[2]*x[4] + A[15]*x[3];
      dx[4] = G[0]*(x[2]*x[1] - x[3]*x[0]) - G[1]*u[2];
  }
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

ekfpu.c dosyası (alan yönlendirme ve EKF için kullanılan C S-Funtion, denetleyici ayrık-zaman birim modelini içerir)

```
#define S_FUNCTION_NAME  ekfpu
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include "im_par.h"
#include <math.h>
#include "matopt.c"
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 8);
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 3);
    ssSetInputPortRequiredContiguous(S, 0, true);
    ssSetInputPortDirectFeedThrough(S, 0, 0);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 7);
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 89);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
    ssSetOptions(S, 0);
}
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, dt);
```

```
    ssSetOffsetTime(S, 0, 0.0);
}
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
  static void mdlInitializeConditions(SimStruct *S)
  {
    real_T *x0 = ssGetDiscStates(S);
    int_T nds = ssGetNumDiscStates(S);
    int_T  lp;
    for (lp=0;lp<nds;lp++) {
        *x0++=0.0;
    }
  }
#endif /* MDL_INITIALIZE_CONDITIONS */
#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
  static void mdlStart(SimStruct *S)
  {
      int_T i,j;
      real_T *a = ssGetRWork(S);
      real_T Ib = In*sqrt(2.);
      real_T Vb = Vn*sqrt(2.);
      real_T Wb = 2.*pi*fn;
      real_T Phib = Vb/Wb;
      real_T Zb = Vb/Ib;
      real_T Tb = 3./2.*Vb*Ib/Wb;
      real_T rs = Rs/Zb;
      real_T rr = Rr/Zb;
      real_T xs = Wb*Ls/Zb;
      real_T xr = Wb*Lr/Zb;
      real_T xsig = (Ls - Lm*Lm/Lr)*Wb/Zb;
      real_T xm = Wb*Lm/Zb;
      real_T Tr = xr/rr/Wb;
      real_T Tm = J*Wb/(Pn/2.)/Tb;
      real_T Ts2 = 1./(rs*Wb/xsig + xm*xm/xsig/xr/Tr);
      a[0] = 1. -dt/Ts2; a[1] = 0.; a[2] = xm*dt/xsig/xr/Tr;
      a[3] = 0.; a[4] = 0.; a[5] = 0.; a[6] = 1. -dt/Ts2; a[7] = 0.;
      a[8] = xm*dt/xsig/xr/Tr; a[9] = 0.; a[10] = xm*dt/Tr;
      a[11] = 0.; a[12] = 1. - dt/Tr; a[13] = 0.; a[14] = 0.;
      a[15] = 0.; a[16] = xm*dt/Tr; a[17] = 0.;
      a[18] = 1. - dt/Tr; a[19] = 0.;
      a[20] = 0.; a[21] = 0.; a[22] = 0.; a[23] = 0.; a[24] = 1.;
      a[25] = xm*Wb*dt/xsig/xr; a[26] = Wb*dt;
      a = a + 27;
      a[0] = Wb*dt/xsig; a[1] = 0;
      a[2] = 0; a[3] = Wb*dt/xsig;
      a[4] = 0; a[5] = 0;
      a[6] = 0; a[7] = 0;
      a[8] = 0; a[9] = 0;
      a = a + 10;
      a[0] = 1.; a[1] = 0.; a[2] = 0.; a[3] = 0.; a[4] = 0.;
      a[5] = 0.; a[6] = 1.; a[7] = 0.; a[8] = 0.; a[9] = 0.;
      a = a + 10;
      a[0] = 0.003; a[1] = 0.003; a[2] = 0.00091;
      a[3] = 0.00091; a[4] = 0.00003;
      a = a + 5;
      a[0] = 0.61; a[1] = 0.61;
      a = a + 2;
      for(i=0;i<5;i++){
```

```
            for(j=0;j<5;j++)
            {
                *a = 0.;
                if (i == j)
                {
                    *a = 0.001;
                }
                *a++;
            }
        }
        for(i=0;i<10;i++){*a++ = 0.;}
}
#endif /*  MDL_START */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *y = ssGetOutputPortRealSignal(S,0);
    int_T  ny = ssGetOutputPortWidth(S,0);
    real_T *x = ssGetDiscStates(S);

    y[0] = x[0];
    y[1] = x[1];
    y[2] = x[2];
    y[3] = x[3];
    y[4] = x[4];
    y[5] = x[6];
    y[6] = x[7];
}
#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
      const real_T *u  = (const real_T*) ssGetInputPortSignal(S,0);
      real_T *x = ssGetDiscStates(S);
      real_T *F = ssGetRWork(S);
      real_T A[16], *B, *C, *Q, *R, *P, *K, *f, *a, *tmp;
      int_T i,j;
      real_T x_1[5], rs0[25], rs1[25];
      static int_T l;

      real_T werrorc = 0;
      real_T torq_com = 0.;
      real_T phir = 0;
      real_T sin_th = 0;
      real_T cos_th = 0.;
      real_T iD_com = 0;
      real_T iQ_com = 0;
      real_T id_com = 0;
      real_T iq_com = 0;
      real_T ia = 0.;
      real_T ib = 0.;
      real_T ic = 0.;
      real_T ia_com = 0.;
      real_T ib_com = 0.;
      real_T ic_com = 0.;
      real_T sa = 1.;
      real_T sb = 0.;
      real_T sc = 0.;
```

```
  B = F + 27; C = B + 10; Q = C + 10; R = Q + 5; P = R + 2;
  K = P + 25;
  //******** EKF routine starting ********
  // update diff_f and A matrix
  F[3] = F[25]*x[4]; F[4] = F[25]*x[3];
  F[7] = -F[25]*x[4]; F[9] = -F[25]*x[2];
  F[13] = -F[26]*x[4]; F[14] = -F[26]*x[3];
  F[17] = F[26]*x[4]; F[19] = F[26]*x[2];

  a = A, f = F;
  for(i=0;i<19;i++)
      {
          *a++ = *f++;
          if(i==4||i==9||i==14){*a--;}
      }
// state prediction
  x_1[4] = x[4];
  matMul(A, x, rs0, 4,4,1); tmp = x + 6;
  matMul(B, tmp, rs1, 4,2,1);
  for(i=0;i<4;i++){x_1[i] = rs0[i] + rs1[i];}
  //for(i=0;i<5;i++){x[i]=x_1[i];}
  //state prediction is OK!

  // update prediction error covariance matrix
  matMul(F,P,rs0, 5,5,5);
  matT(F,rs1, 5,5);
  matMul(rs0,rs1,P, 5,5,5);

  for(i=0;i<5;i++){for(j=0;j<5;j++){if(i==j){P[6*j]=P[6*j]+Q[j];
       }}}
  // update prediction error covariance matrix is OK!

  // update Kalman gain
  matT(C,rs0, 2,5);
  matMul(P,rs0,rs0, 5,5,2);
  matMul(C,rs0,rs1, 2,5,2);
  rs1[0] = rs1[0] + R[0]; rs1[3] = rs1[3] + R[1];
  matInv(rs1,rs1);
  matMul(rs0,rs1,K, 5,2,2);
  // update Kalman gain is OK!

  //state estimation
  matMul(C,x_1,rs0, 2,5,1);
  rs0[0] = u[0] - rs0[0]; rs0[1] = u[1] - rs0[1];
  matMul(K,rs0,rs0, 5,2,1);
  for(i=0;i<5;i++){x[i] = x_1[i] + rs0[i];}
  //state estimation is OK!

  // state estimation covariance matrix
  matMul(K,C,rs0, 5,2,5);
  matMul(rs0,P,rs0, 5,5,5);
  for(i=0;i<24;i++){P[i]= P[i] - rs0[i];}
  //******  end of EKF routine *******

  //****start of controller routine ****
  werrorc = u[2] - x[4];
  x[5] = x[5] + G0*werrorc;
  if (x[5] > id_lim) {x[5] = id_lim;}
```

```
      if (x[5] < -id_lim) {x[5] = -id_lim;}
      torq_com = x[5] + G1*werrorc;
      if (torq_com > id_lim) {torq_com = id_lim;}
      if (torq_com < -id_lim) {torq_com = -id_lim;}

      phir = sqrt(x[2]*x[2] + x[3]*x[3]) + 1e-6;
      cos_th = x[2]/phir; sin_th = x[3]/phir;

      iQ_com = G2*torq_com/phir;
      if (iQ_com > id_lim) {iQ_com = id_lim;}
      if (iQ_com < -id_lim) {iQ_com = -id_lim;}

      if(u[2] > 0.93){iD_com = iDn/(0.93/u[2])/(0.93/u[2]);}
      else{iD_com = iDn;}

      id_com = iD_com*cos_th - iQ_com*sin_th;
      iq_com = iD_com*sin_th + iQ_com*cos_th;

      ia_com = id_com; ib_com = -1./2.*id_com + sqrt(3.)/2.*iq_com;
      ic_com = -(ia_com + ib_com);

      ia = x[0]; ib = -1./2.*x[0] + sqrt(3.)/2.*x[1];
      ic = -(ia + ib);

      if ((ia_com - ia) > hb) {sa = 1;}
      if ((ia_com - ia) < -hb) {sa = 0;}

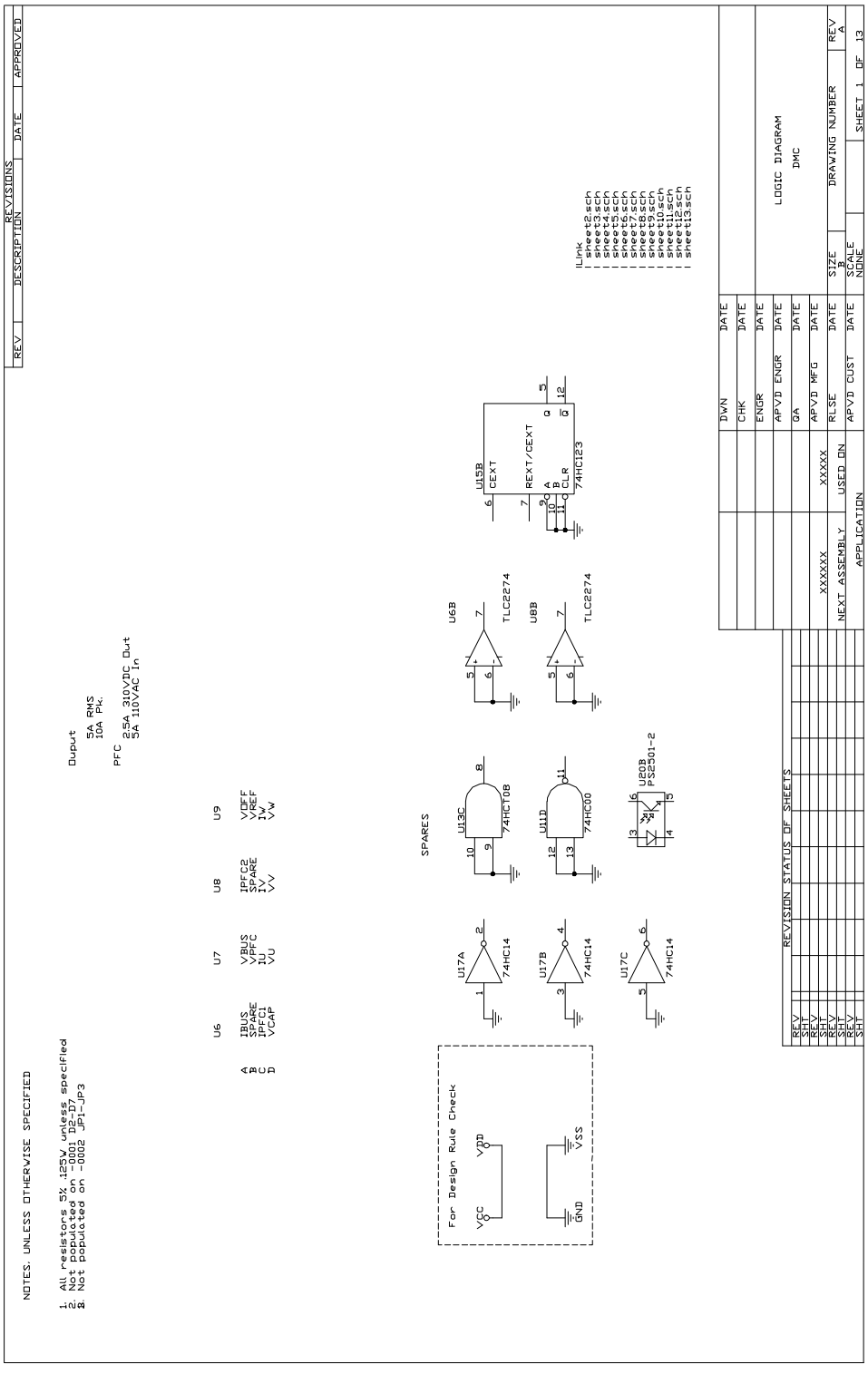
      if ((ib_com - ib) > hb) {sb = 1;}
      if ((ib_com - ib) < -hb) {sb = 0;}

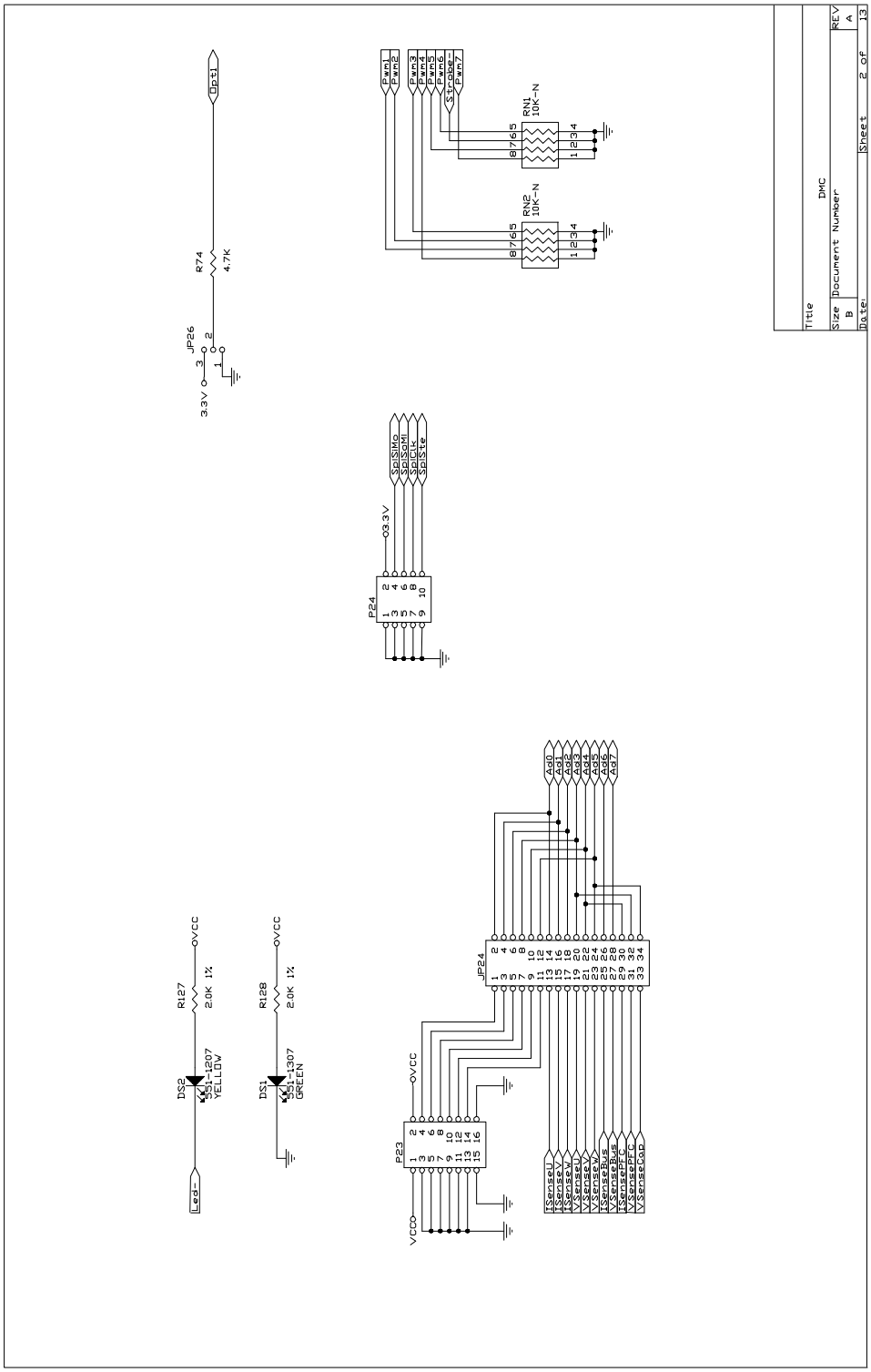
      if ((ic_com - ic) > hb) {sc = 1;}
      if ((ic_com - ic) < -hb) {sc = 0;}

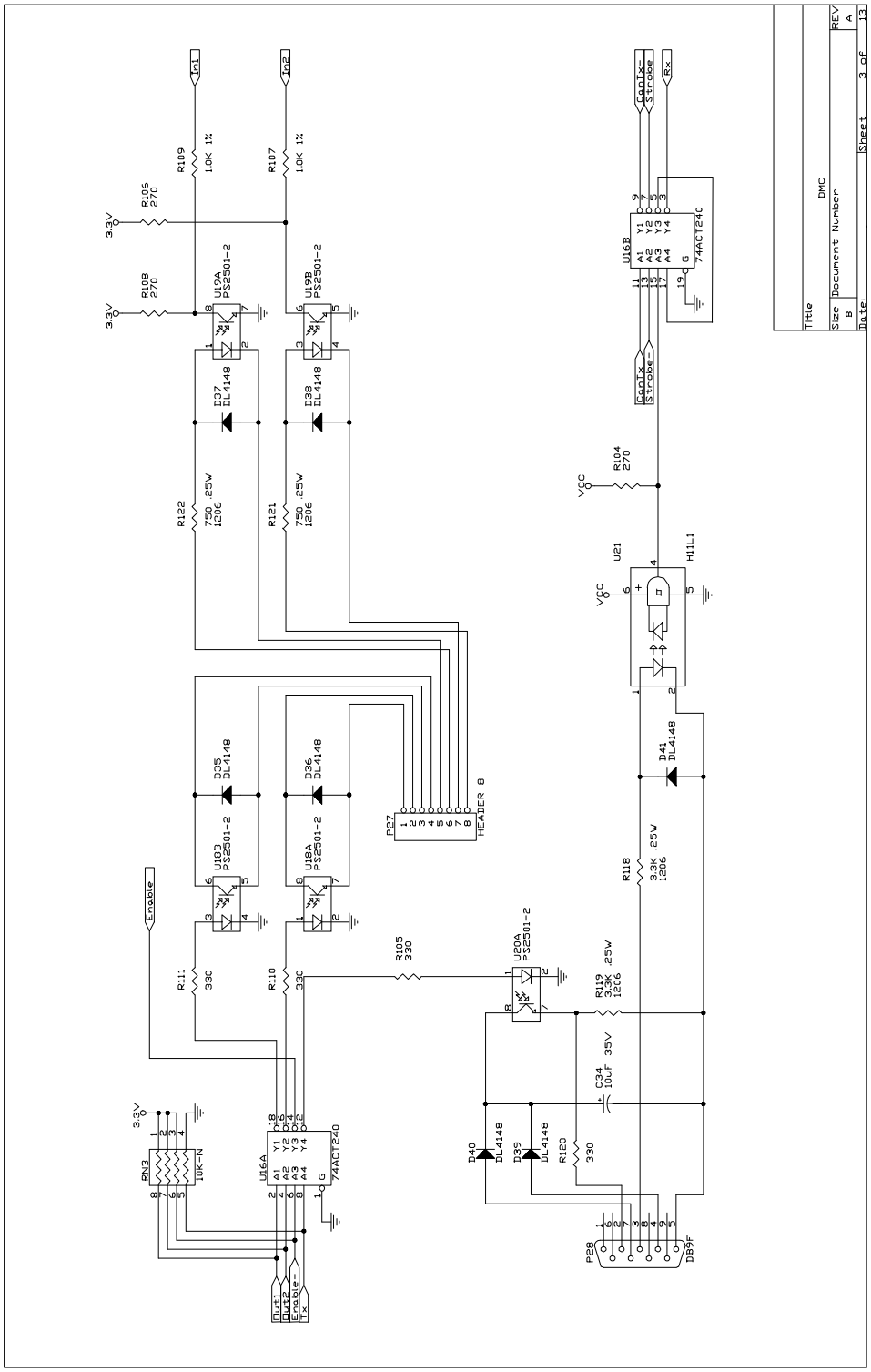
      x[6] = 1./sqrt(3.)*(2.*sa - sb - sc);
      x[7] = (sb - sc);

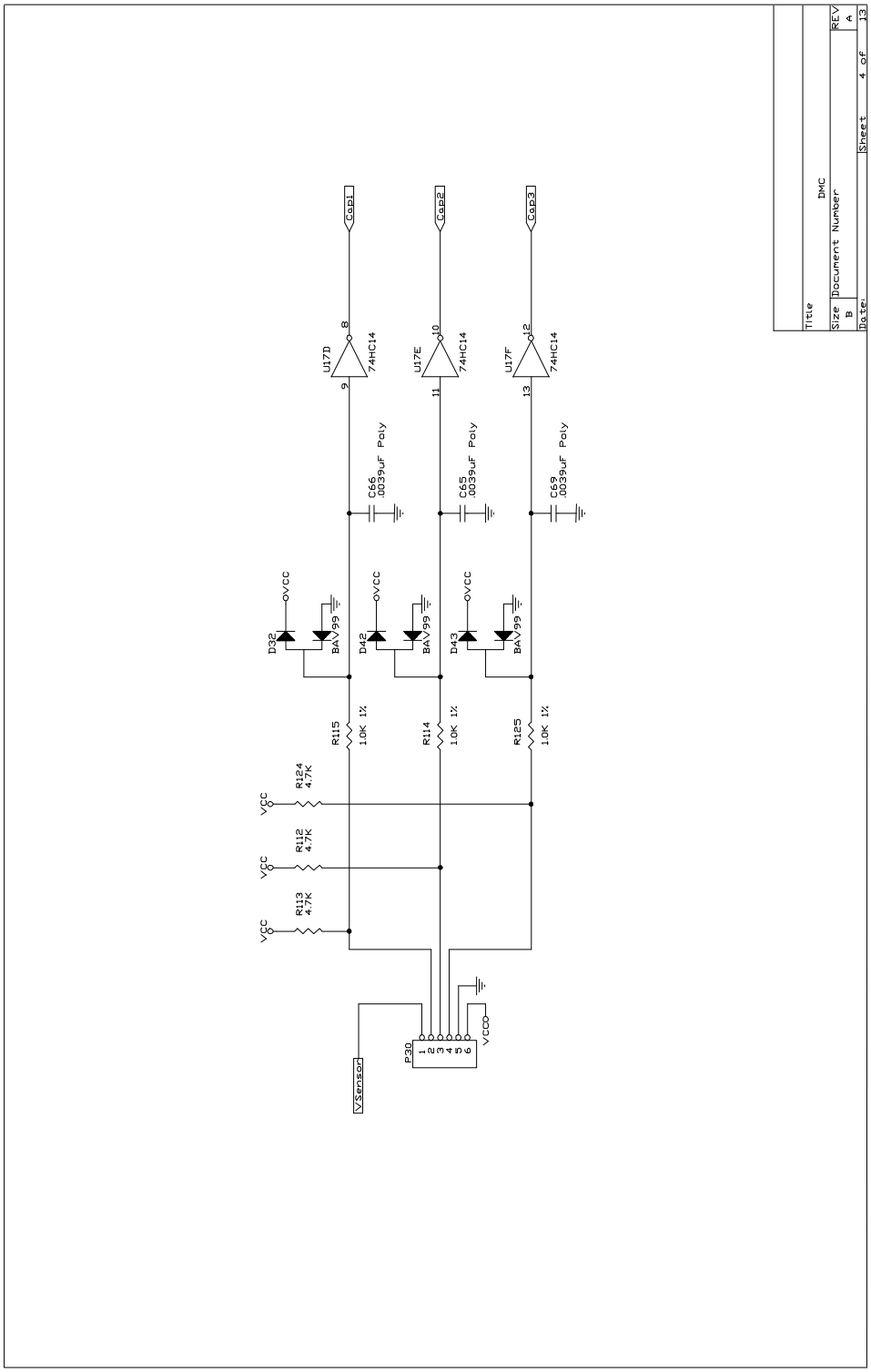
      // for sinusoidal suply
      //l++; x[6] = cos(2.*pi*50.*l*dt);
      //x[7] = sin(2.*pi*50.*l*dt);
 }
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

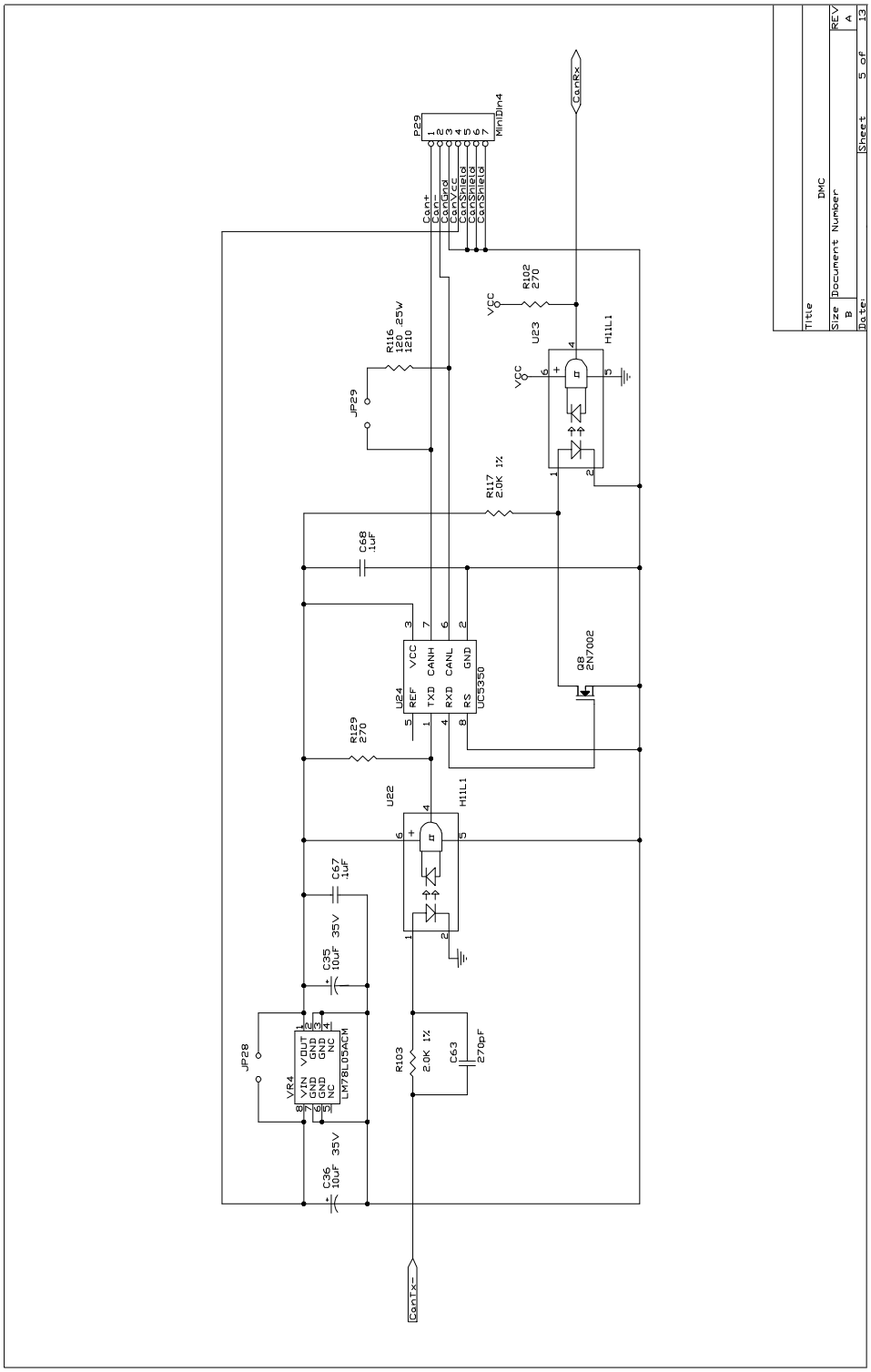
EK-B SAYISAL MOTOR KONTROLÖR KARTI ORCAD ŞEMALARI

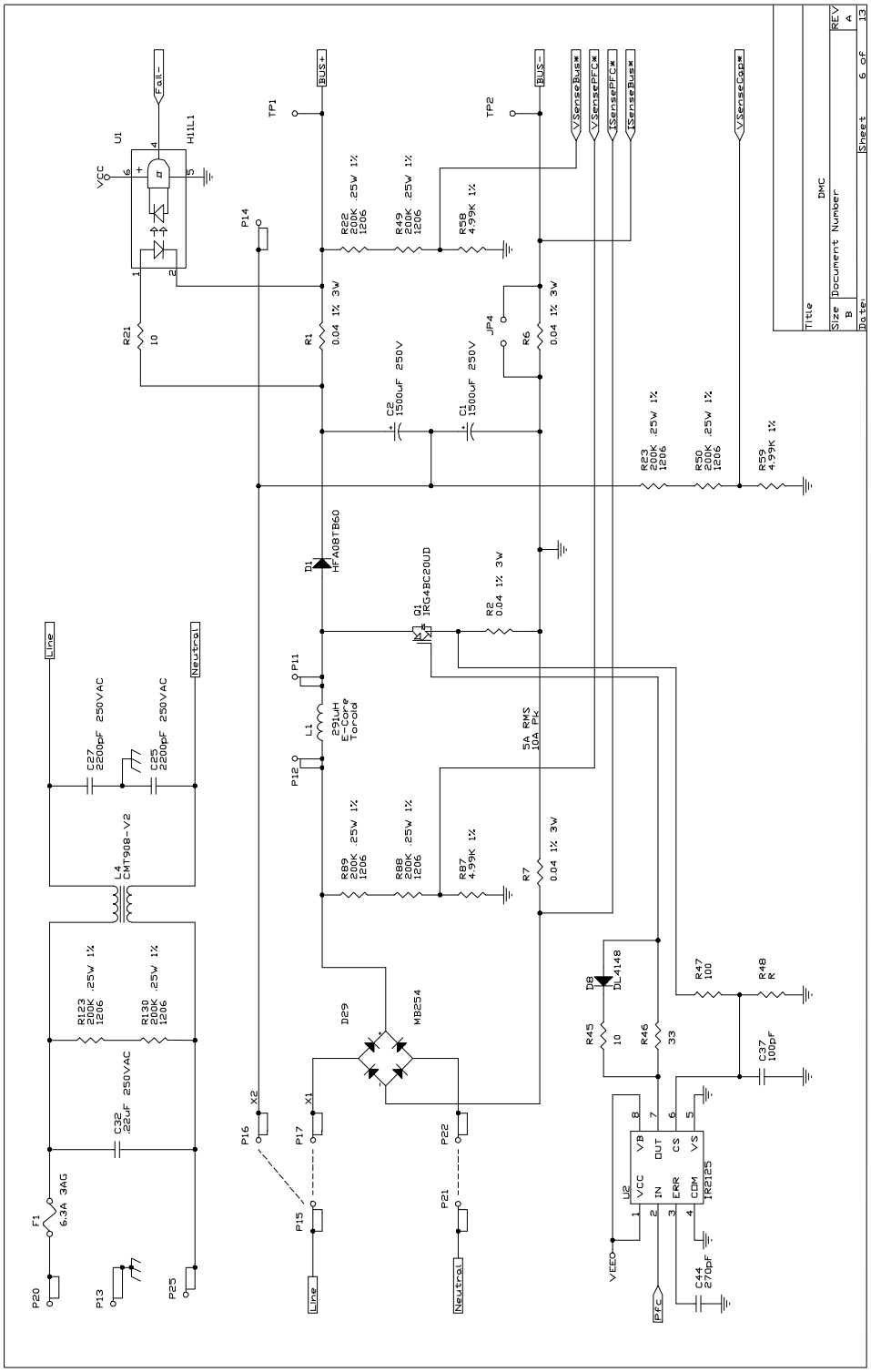


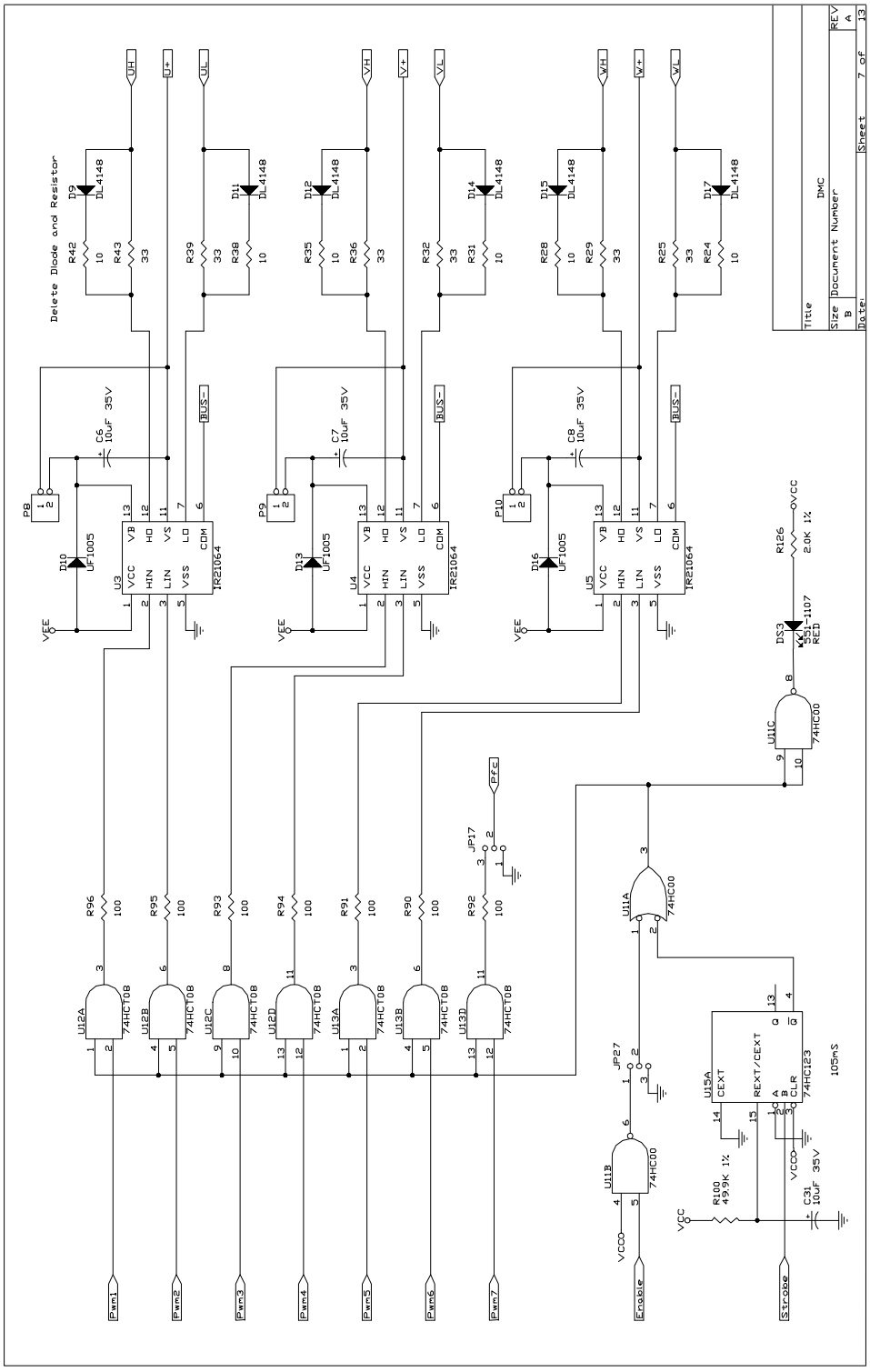

Led-
DS2
SSI-1207
YELLOW
R127
2.0K 1%
OVCC
DS1
SSI-1307
GREEN
R128
2.0K 1%
OVCC
P23
VCC
OVCC
JP24
A40
A41
A42
A43
A44
A45
A46
A47
ISenseU
ISenseV
ISenseW
VSenseU
VSenseV
VSenseW
ISenseBus
VSensePFC
ISensePFC
VSenseCap
P24
3.3V
SpiSIMo
SpiSOMI
SpiClk
SpiSte
JP26
3.3V
R74
4.7K
Dpti
Pwm1
Pwm2
Pwm3
Pwm4
Pwm5
Pwm6
Strobe
Pwm7
RN1
10K-N
RN2
10K-N
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet
2 of
13

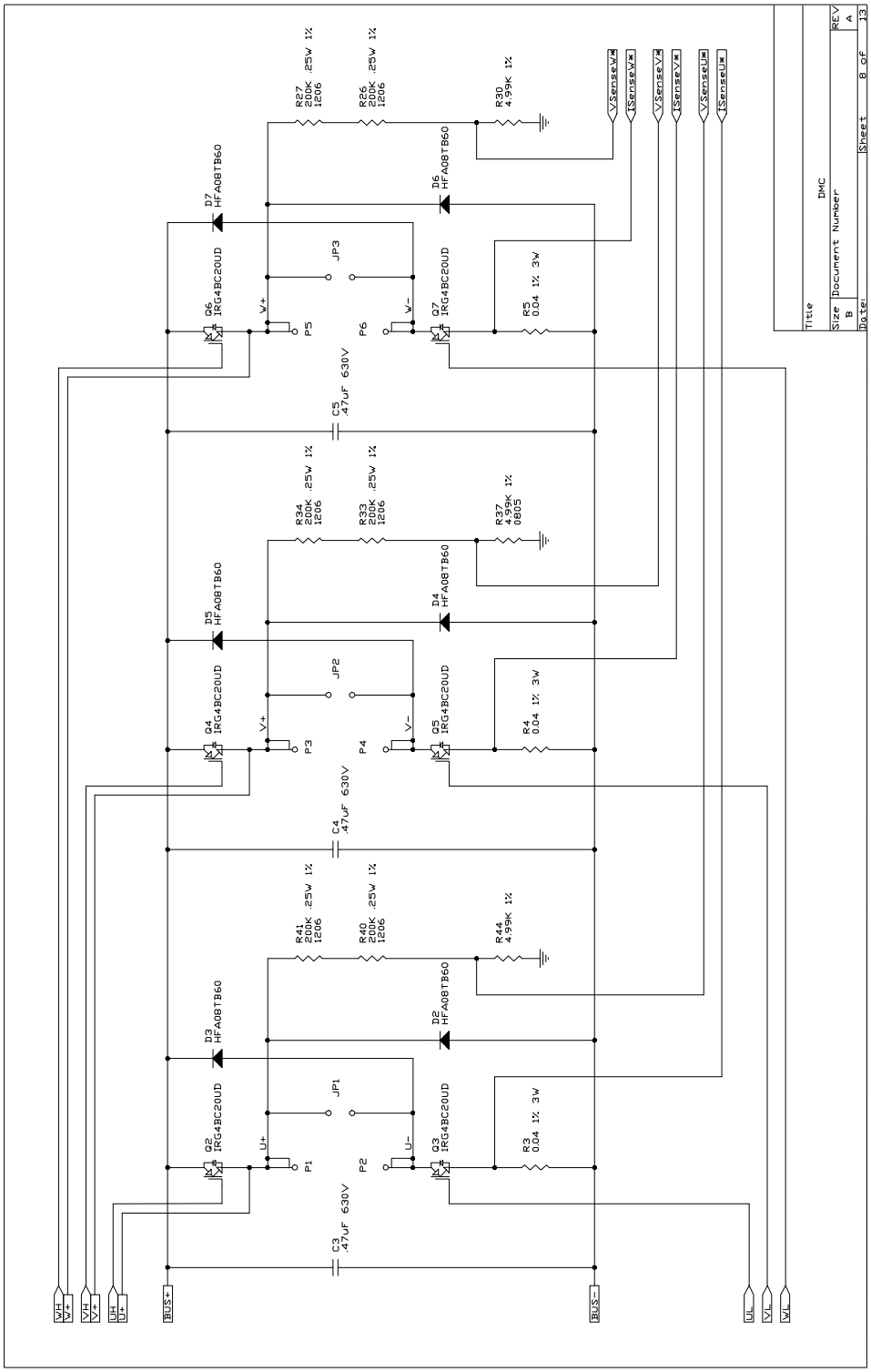

In1
In2
R109
10K 1%
R107
10K 1%
R106
270
R108
270
3.3V
U19A
PS2501-2
U19B
PS2501-2
D37
DL4148
D38
DL4148
R122
750 25W
1206
R121
750 25W
1206
D35
DL4148
D36
DL4148
U18B
PS2501-2
U18A
PS2501-2
P27
HEADER 8
Enable
R111
330
R110
330
R105
330
U20A
PS2501-2
RN3
10K-N
U16A
74ACT240
Out1
Out2
Enable-
Tx
D40
DL4148
D39
DL4148
R120
330
C34
10uF 35V
R119
3.3K 1206 .25W
R118
3.3K 1206 .25W
P28
DB9F
CanTx-
CanTx
Stroke-
Stroke
Rx
U16B
74ACT240
R104
270
VCC
U21
H11L1
D41
DL4148
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet 3 of 13

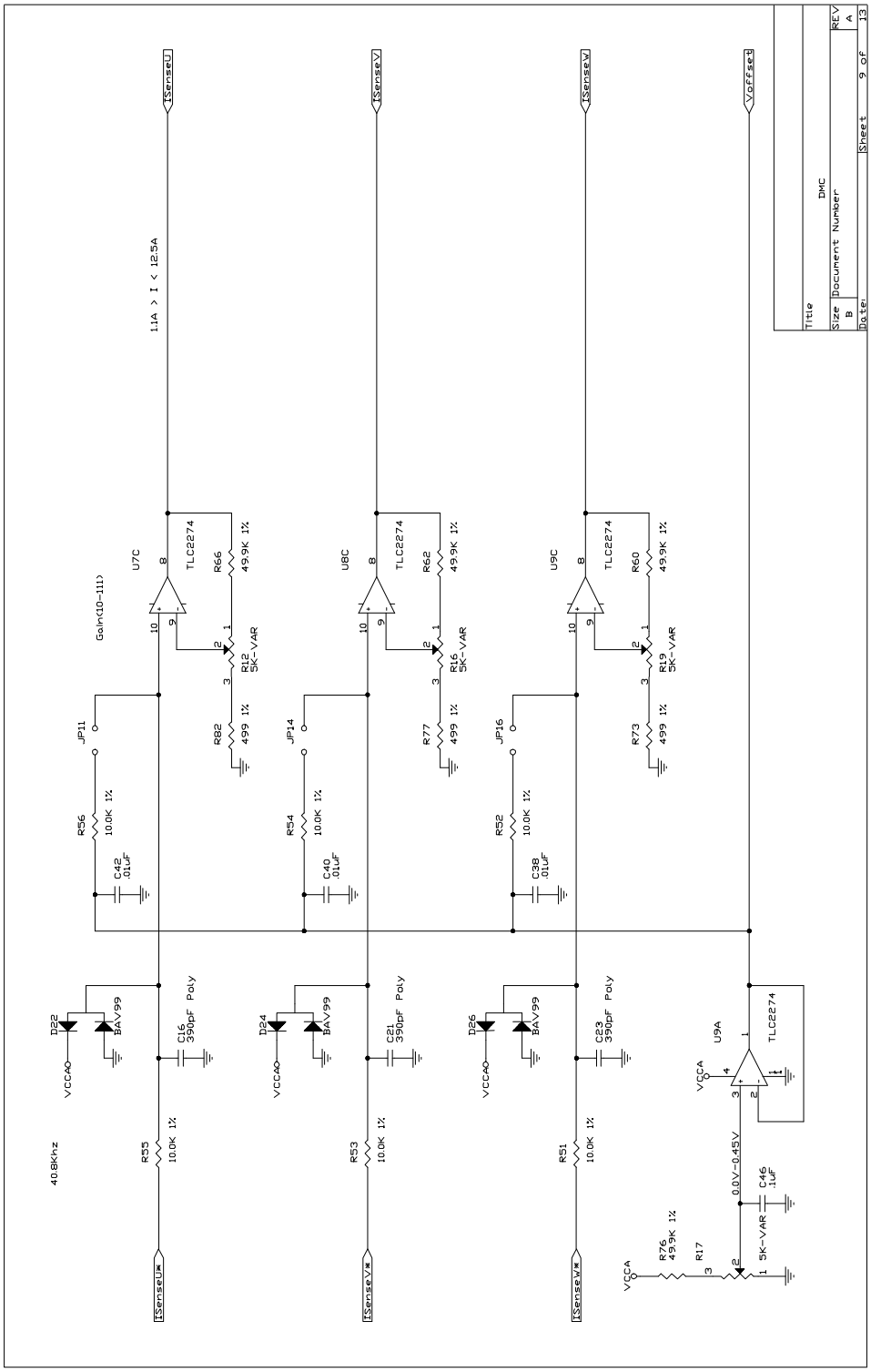

Cap1
Cap2
Cap3
U17D
U17E
U17F
74HC14
C66
.0039uF Poly
C65
.0039uF Poly
C69
.0039uF Poly
D32
D42
D43
BAV99
OVCC
R115
R114
R125
10K 1%
R124
R112
R113
4.7K
VCC
P30
VCCO
VSensor
Title
DMC
Size
B
Document Number
REV
A
Date
Sheet
4 of
13

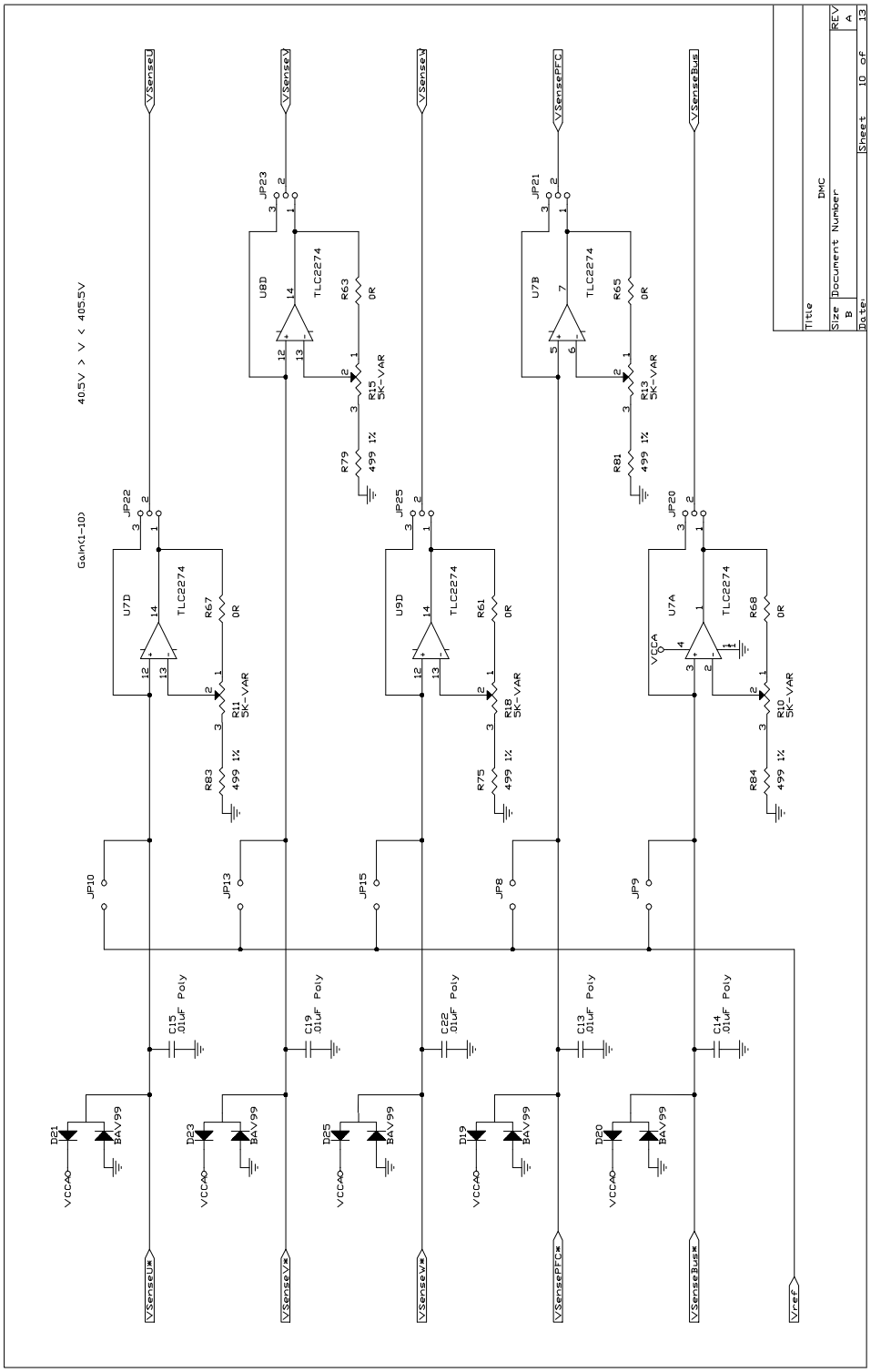

P29
MiniDin4
Can+
Can-
CanGnd
CanVcc
CanShield
CanShield
CanShield
CanShield
CanRx
CanTx
R102
270
VCC
U23
H11L1
R116
120
1210
25W
JP29
R117
2.0K
1%
C68
.1uF
U24
UC5350
REF
TXD
RXD
RS
VCC
CANH
CANL
GND
Q8
2N7002
R129
270
U22
H11L1
C67
.1uF
C35
10uF
35V
JP28
VR4
LM78L05ACM
VIN
VOUT
GND
GND
NC
R103
2.0K
1%
C63
270pF
C36
10uF
35V
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet
5 of
13

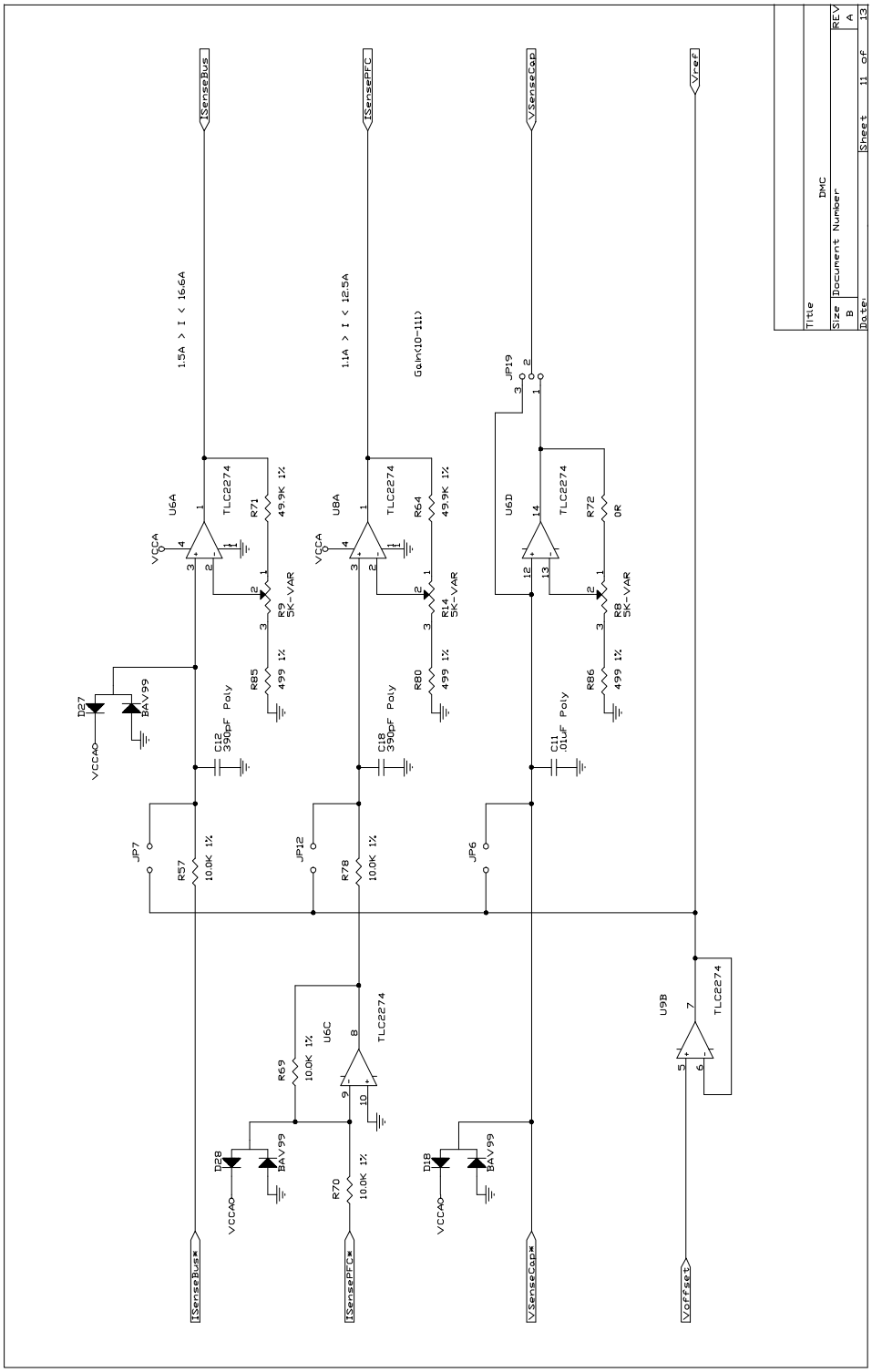

Line
Neutral
C27 2200pF 250VAC
C25 2200pF 250VAC
L4 CMT908-V2
R123 200K .25W 1%
1206
R130 200K .25W 1%
1206
C32 .22uF 250VAC
F1 6.3A 3AG
P20
P13
P25
P16
X2
P17
X1
P15
Line
P22
P21
Neutral
D29 MB254
P12
P11
L1 291uH Eccore Toroid
5A RMS 10A PK
R89 200K .25W 1%
1206
R88 200K .25W 1%
1206
R87 4.99K 1%
R7 0.04 1% 3W
Q1 IRG4BC20UD
R2 0.04 1% 3W
D1 HFA08TB60
C2 1500uF 250V
C1 1500uF 250V
JP4
R6 0.04 1% 3W
R1 0.04 1% 3W
R21 10
U1 H11L1
VCC
Fail-
P14
TP1
BUS+
R22 200K .25W 1%
1206
R49 200K .25W 1%
1206
R38 4.99K 1%
TP2
BUS-
VSenseBus*
VSensePFC*
ISensePFC*
ISenseBus*
VSenseCap*
R23 200K .25W 1%
1206
R50 200K .25W 1%
1206
R39 4.99K 1%
D8 DL-4148
R45 10
R46 33
R47 100
R48 R
C37 100pF
U2 IR2125
VCC
IN
ERR
VB
DUT
CS
VS
COM
VEE
PFC
C44 270pF
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet 6 of 13

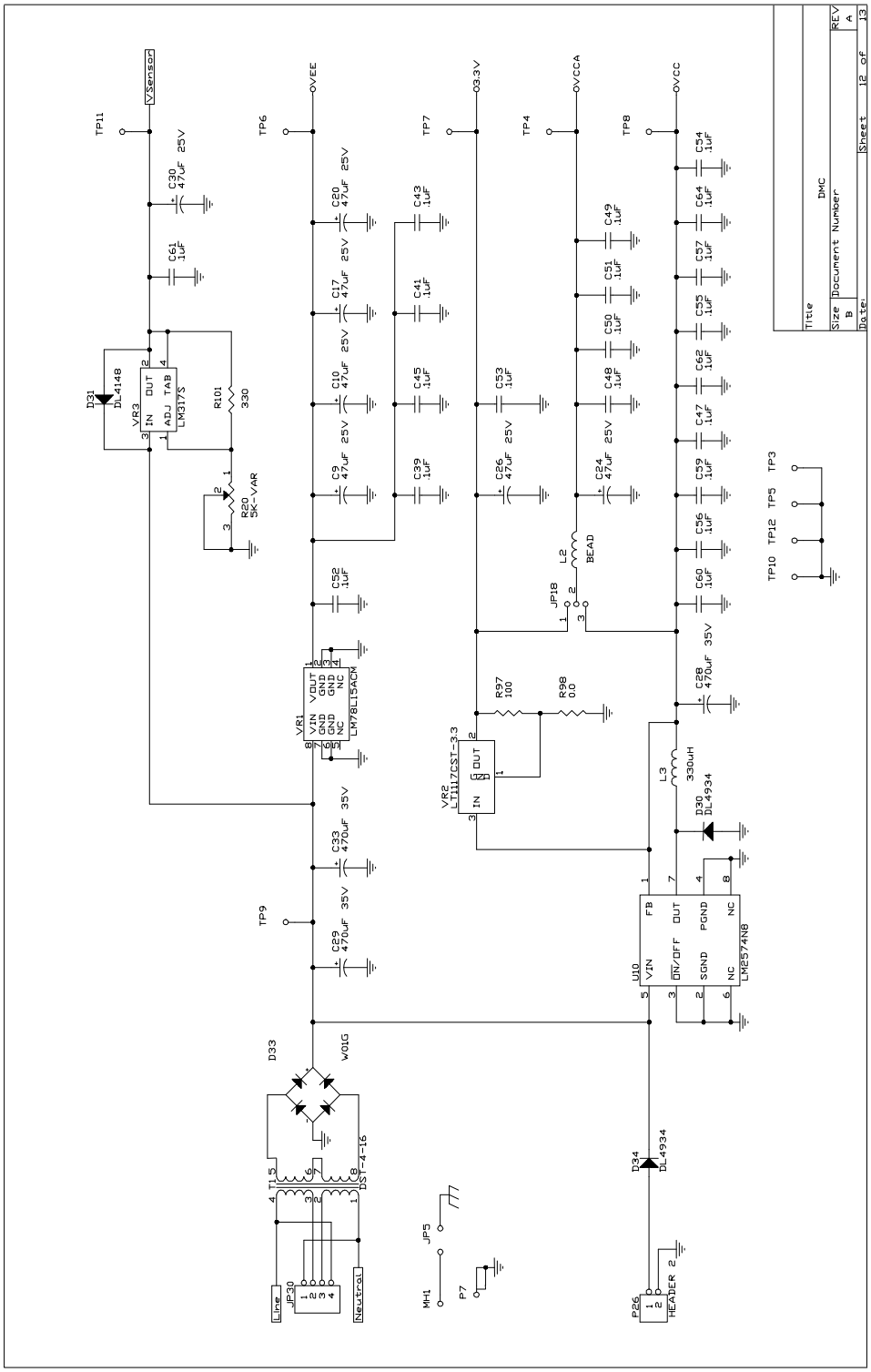

Delete Diode and Resistor
UH
U+
UL
VH
V+
VL
WH
W+
WL
BUS-
VCC
VEE
IR21064
UF1005
74HCT08
74HC00
74HC123
LM4148
Pwm1
Pwm2
Pwm3
Pwm4
Pwm5
Pwm6
Pwm7
Enable
Strobe
105mS
Title
DMC
Size
B
Document Number
REV
A
Sheet
7 of 13
Date:

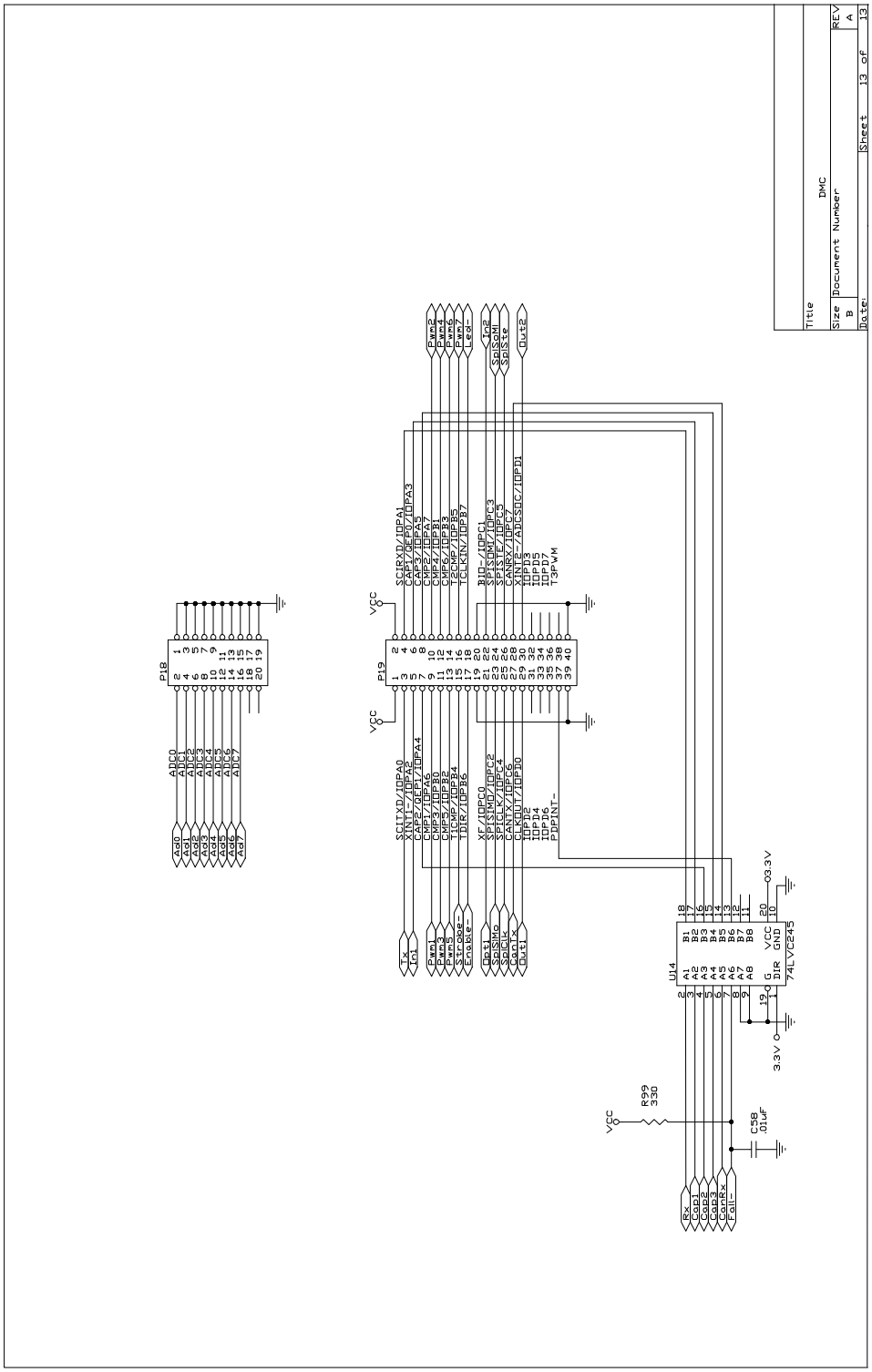

R27 200K .25W 1% 1206
R26 200K .25W 1% 1206
R30 4.99K 1%
D7 HFA08TB60
D6 HFA08TB60
JP3
Q6 IRG4BC20UD
Q7 IRG4BC20UD
R5 0.04 1% 3W
P5
P6
C5 .47uF 630V
R34 200K .25W 1% 1206
R33 200K .25W 1% 1206
R37 4.99K 1% 0805
D5 HFA08TB60
D4 HFA08TB60
JP2
Q4 IRG4BC20UD
Q5 IRG4BC20UD
R4 0.04 1% 3W
P3
P4
C4 .47uF 630V
R41 200K .25W 1% 1206
R40 200K .25W 1% 1206
R44 4.99K 1%
D3 HFA08TB60
D2 HFA08TB60
JP1
Q2 IRG4BC20UD
Q3 IRG4BC20UD
R3 0.04 1% 3W
P1
P2
C3 .47uF 630V
VSenseW
ISenseW
VSenseV
ISenseV
VSenseU
ISenseU
VH
V+
VH
V+
UH
U+
BUS+
BUS-
UL
VL
VL
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet 8 of 13


1.1A > I < 12.5A
Gain<10-111>
TLC2274
U7C
U8C
U9C
U9A
R66
R62
R60
49.9K 1%
R12
R16
R19
5K-VAR
R82
R77
R73
499 1%
JP11
JP14
JP16
R56
R54
R52
10.0K 1%
C42
C40
C38
.01uF
D22
D24
D26
BAV99
VCCA0
C16
C21
C23
390pF Poly
R55
R53
R51
40.8Khz
ISenseU
ISenseV
ISenseW
VOffset
VCCA
R76
R17
0.0V-0.45V
C46
1uF
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet
9 of
13


405V > V < 405V
Gain(1-10)
VSenseU
VSenseV
VSenseW
VSensePFC
VSenseBus
Vref
JP10
JP13
JP15
JP8
JP9
JP22
JP23
JP25
JP21
JP20
U7D
U8D
U9D
U7B
U7A
TLC2274
R67
R63
R61
R65
R68
0R
R11
R12
R18
R13
R10
5K-VAR
R83
R79
R75
R81
R84
499 1%
C15
C19
C22
C13
C14
.01uF Poly
D21
D23
D25
D19
D20
BAV99
VCCA
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet 10 of 13


1.5A > I < 16.6A
1.1A > I < 12.5A
Gain(10-111)
ISenseBus
ISensePFC
VSenseCap
Vref
U6A
U8A
U6D
U9B
U6C
TLC2274
R71
R64
R72
49.9K 1%
0R
R9
R14
R8
5K-VAR
R85
R80
R86
499 1%
C12
C18
C11
390pF Poly
.01uF Poly
D27
D28
D18
BAV99
VCCAO
JP7
JP12
JP6
JP19
R57
R78
R69
R70
10.0K 1%
ISenseBusx
ISensePFCx
VSenseCapx
Voffset
Title
DMC
Size
B
Document Number
REV
A
Date:
Sheet 11 of 13


VSensor
TP11
TP6
OVEE
TP7
O3.3V
TP4
OVCCA
TP8
OVCC
C30 47uF 25V
C61 1uF
C20 47uF 25V
C43 1uF
C17 47uF 25V
C41 1uF
C10 47uF 25V
C45 1uF
C9 47uF 25V
C39 1uF
C53 1uF
C26 47uF 25V
C49 1uF
C51 1uF
C50 1uF
C48 1uF
C24 47uF 25V
C54 1uF
C64 1uF
C57 1uF
C55 1uF
C62 1uF
C47 1uF
C59 1uF
C36 1uF
C60 1uF
C52 1uF
D31 DL4148
VR3 LM317S
R101 330
R20 5K VAR
L2 BEAD
JP18
VR1 LM78L15ACM
R97 100
R98 0.0
VR2 LT1117CST-3.3
C28 470uF 35V
L3 330uH
D30 DL4934
C33 470uF 35V
C29 470uF 35V
TP9
U10 LM2574N8
VIN FB DUT PGND SGND NC ON/OFF
D33 W01G
T1 DST-4-16
D34 DL4934
JP30
Line
Neutral
JP5
MH1
P7
P26 HEADER 2
TP10 TP12 TP5 TP3
Title
DMC
Size B
Document Number
REV A
Date:
Sheet 12 of 13


P18
P19
U14
74LVC245
R99
330
C58
01uF
VCC
3.3V
DMC
Title
Size
Document Number
REV
Date:
Sheet 13 of 13

EK-C DSP YAZILIMI

Bu kısımda DSP için TMS320F243 asembly dilende yazılan “dfo_ekf.asm” dosyasının listesi verilmiştir.

dfo_ekf.asm dosyası (assebmler kaynak dosyası)

```
****************************************
* motor equivalent circuit parameters    *
* rs = 9.9 ohm, rr = 12.345 ohm          *
* lm = .590 H, ls = .661 H, lr = .661 H  *
* dt = 200e-6                            *
****************************************

STACK .set  0x03f0           ; software stack adress
sqrt3_2     .set  0x0ddb     ; sqrt(3)/2 @4.12f
inv_sqrt3   .set  0x093c     ; 1/sqrt(3) @4.12f
inv3        .set  0x0555     ; 1/3 @4.12f

period      .set  4000  ; pwm period
adc_delay   .set  20    ; for current measurement by EVSOC (T1CMP)
sw_on       .set  100   ; time of switch on
            .eval (period+1), sw_off     ; switch off
            .setsect    ".data", 0x0300, 1
            .setsect    ".vect", 0x8000, 0
            .setsect    ".text", 0x8044, 0
            .include    "app_243.h"
            .sect       ".vect"
RESET b     SETUP
INT0  b     PDP_ISR
INT1  b     PER_ISR
            .data
*** data page EKF algorithm ***
u     .int    0x0000, 0x0000             ; measured input vector
y     .int    0x0000, 0x0000             ; measured output vector
dF    .int    0x0f88, 0x0000, 0x0025, 0x0000, 0x0000 ; dF/dt matrix
      .int    0x0000, 0x0f88, 0x0000, 0x0025, 0x0000
      .int    0x0018, 0x0000, 0x0ff1, 0x0000, 0x0000
      .int    0x0000, 0x0018, 0x0000, 0x0ff1, 0x0000
      .int    0x0000, 0x0000, 0x0000, 0x0000, 0x1000

;correction and B matrix coefficient
dF1   .int    0x016c, 0x0095, 0x0197
;estimated state vector
x     .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0000
;predicted state vector
x_1   .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0000
;output matrix
C     .int    0x1000, 0x0000, 0x0000, 0x0000, 0x0000
      .int    0x0000, 0x1000, 0x0000, 0x0000, 0x0000
;prediction error covariance matrix
P     .int    0x0028, 0x0000, 0x0000, 0x0000, 0x0000
      .int    0x0000, 0x0028, 0x0000, 0x0000, 0x0000
      .int    0x0000, 0x0000, 0x0028, 0x0000, 0x0000
      .int    0x0000, 0x0000, 0x0000, 0x0028, 0x0000
      .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0028
;filter error covariance matrix
```

```
P_1   .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0000
      .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0000
      .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0000
      .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0000
      .int    0x0000, 0x0000, 0x0000, 0x0000, 0x0000
;state noise covariance matrix
Q     .int    0x000c, 0x000c, 0x0004, 0x0004, 0x0001
; measure noise covariance matrix
R     .int    0x0800, 0x0800
K     .int    0x0000, 0x0000             ; kalman gain matrix
      .int    0x0000, 0x0000
      .int    0x0000, 0x0000
      .int    0x0000, 0x0000
      .int    0x0000, 0x0000

RSLT  .space  25*16

iDn   .set  0x08f5      ; if iD < iDn then iQ_cm = 0
ksqrt .set  0x00aa      ; for square root table
ST_UP .int  0x0000      ; in start-up period
phir  .int  0x0000      ; estimated rotor flux in sync. ref. frame
sin   .int  0x0000      ; sin(theta) @4.12
cos   .int  0x1000      ; cos(theta) @4.12

i_off .set  0x0200      ; current off-set
adc_rd1     .int  0x0000      ; for read current
adc_k1      .int  0x5952      ; 0.75*10/2.687/512 @0.22f
adc_rd2     .int  0x0000      ; for read current
adc_rd3     .int  0x0000      ; for read voltage
adc_k2      .int  0x8ac0      ; 400/1023/179.60512*(1-5.5/200)
include spare time @0.20f
vdc   .int  0x0000      ; dc bus voltage

ia    .int  0x0000      ; stator phase A current
ib    .int  0x0000      ; stator phase B current
ic    .int  0x0000      ; stator phase C current
ia_cm .int  0x0000      ; stator phase A current command
ib_cm .int  0x0000      ; stator phase B current command
ic_cm .int  0x0000      ; stator phase C current command
id_cm .int  0x0000      ; stator d-axis current command
iq_cm .int  0x0000      ; stator q-axis current command
iD_cm .int  0x0000      ; stator D-axis current command
iQ_cm .int  0x0000      ; stator Q-axis current command

hb    .set  0x00cc      ; hist. band upper limit 0.05pu
wr_cm .int  0x1000      ; rotor electrical speed command
wr_ind      .int  0x0000      ; for rotor speed command change
trq_cm      .int  0x0000      ; torque command
trq_int     .int  0x0000      ; torque controller integration part
trq_Ki      .set  0x0041      ; torq. int. const.
Ki*wbase*Ts/trqbase 80*dt @4.12f
trq_Kp      .int  0x7fff      ; torque command prop. const. Kp = 8
@4.12f
trq_Kc      .set  0x0008      ; torque command integ. corr. const.
@4.12f
trq_lim     .set  0x1000      ; torque command limit
G_iQ  .set  0x08f6      ; iQ command gain

SS    .set  1
```

```
DIR   .set  1
inv_step .set     60
inv_pw      .set  2050
sx_step     .int  (inv_step) ; for step count
sector      .int  (DIR+1)     ; for six-step sector
sa    .int  0x0001      ; inverter A switch status
sb    .int  0x0000      ; inverter B switch status
sc    .int  0x0000      ; inverter C switch status

l     .int  0x0000      ; for flashing XF led
t1cnt   .int      0x0000      ; for period sufficient control
gr_ind      .int  0x9000      ; for visualization
gr_ind1     .int  0x0000      ; for visualization

*** program is begin here!! ***
      .text
      b     SETUP       ; hardware and software initialization
*** T1 period inrrupt ***
PER_ISR
*** context save ***
      mar   *, AR7
      mar   *-
      sst   #01h, *-
      sst   #00h, *-
      sach  *-
      sacl  *-
******* inverter STROBE *******
***
      mar   *, AR0
      lar   AR0, #PBDATDIR
      lacc  *
      and   #0xffef
      sacl  *

*** IOPB4 is low
*******************************

***********************************
***   EKF algoritm            ***
***********************************
*** update dF/dt matrix ***
UPDF
      setc  sxm
      spm   0x02
      setc  ovm

      lar   AR2, #(x+4)
      lar   AR3, #(dF+3)
      lar   AR6, #(dF1)
      mar   *, AR2
      lt    *-, AR6
      mpy   *+, AR3
      pac
      sach  *
      neg
      adrk  #0x04
      sach  *, 0, AR6
      mpy   *-, AR3
      pac
```

```
      adrk  #0x0a
      sach  *
      neg
      sbrk  #0x04
      sach  *, 0, AR6

      lt    *, AR2
      mpy   *-, AR3
      pac
      lar   AR3, #(dF+4)
      sach  *, 0, AR2
      mpy   *+, AR3
      adrk  #0x05
      pac
      neg
      sach  *, 0, AR6
      adrk  #0x01
      lt    *, AR2
      mpy   *-, AR3
      adrk  #0x05
      pac
      neg
      sach  *, 0, AR2
      mpy   *, AR3
      adrk  #0x05
      pac
      sach  *

*** now dF/dt matrix was updated ***

*** update state vector prediction ***
SVP
            spm   0x02
            setc  ovm
            setc  sxm
            lar   AR3, #dF
            lar   AR4, #x_1
            lar   AR5, #u
            lar   AR6, #(dF1+2)
      .eval 0, i
      .loop
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #x
            .eval 0, j
            .loop
                  lta   *+, AR3
                  mpy   *+, AR2
            .eval j+1,j
            .break j==4
            .endloop
            .if ((i==0)|(i==1))
                  mar   *, AR5
                  lta   *+, AR6
                  mpy   *, AR2
            .endif
                  apac
                  mar   *, AR4
```

```
                  sach  *+, 0, AR3
                  adrk  #0x01
      .eval i+1, i
      .break i==4
      .endloop
            mar   *, AR2
            lacc  *, 0, AR4
            sacl  *

*** now state vector prediction was updated ***
*** update prediction error covariance matrix ***
ECM
            spm   0x02
            setc  ovm
            setc  sxm
            lar   AR4, #RSLT
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(dF+5*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(P+j)
                  .eval 0, k
                  .loop
                        lta   *, AR3
                        mpy   *+, AR2
                        adrk  #0x05
                  .eval k+1, k
                  .break k==5
                  .endloop
                  apac
                  mar   *, AR4
                  sach  *+
            .eval j+1, j
            .break j==5
            .endloop
      .eval i+1, i
      .break i==5
      .endloop

ECM1
            lar   AR4, #P_1
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(RSLT+5*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(dF+5*j)
                  .eval 0, k
                  .loop
                        lta   *+, AR3
                        mpy   *+, AR2
                  .eval k+1, k
```

```
                  .break k==5
                  .endloop
                  apac
                  mar   *, AR4
                  sach  *+
            .eval j+1, j
            .break j==5
            .endloop
      .eval i+1, i
      .break i==5
      .endloop

      lar   AR2, #P_1
      lar   AR3, #Q
      mar   *, AR2
      .eval 0, i
      .loop
            lacc  *, 0, AR3
            add   *+, 0, AR2
            sacl  *
            adrk  #0x06
      .eval i+1, i
      .break i==5
      .endloop
ECM_END

*** now prediction error covariance matrix was updated ***
*** update Kalman gain matrix ***
KGM
            spm   0x02
            setc  ovm
            setc  sxm
            lar   AR4, #RSLT
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(P_1 + 5*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(C + 5*j)
                  .eval 0, k
                  .loop
                        lta   *+, AR3
                        mpy   *+, AR2
                  .eval k+1, k
                  .break k==5
                  .endloop
                  apac
                  mar   *, AR4
                  sach  *+
            .eval j+1, j
            .break j==2
            .endloop
      .eval i+1, i
      .break i==5
      .endloop
KGM1
```

```
            spm   0x02
            setc  ovm
            setc  sxm
            lar   AR4, #K
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(C+5*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(RSLT+j)
                  .eval 0, k
                  .loop
                        lta   *, AR3
                        mpy   *+, AR2
                        adrk  #0x02
                  .eval k+1, k
                  .break k==5
                  .endloop
                  apac
                  mar   *, AR4
                  sach  *+
            .eval j+1, j
            .break j==2
            .endloop
      .eval i+1, i
      .break i==2
      .endloop
KGM2
      spm   0x02
      setc  ovm
      setc  sxm
      lar   AR2, #K
      lar   AR3, #R
      lar   AR4, #(P+3)
      mar   *, AR2
      lacc  *, 0, AR3
      add   *+, 0, AR4
      sacl  *, 0, AR2
      lar   AR2, #(K+3)
      lar   AR4, #P
      lacc  *, 0, AR3
      add   *+, 0, AR4
      sacl  *+, 0, AR2
      sbrk  #0x02
      lacc  *+, 0, AR4
      neg
      sacl  *+, 0, AR2
      lacc  *+, 0, AR4
      neg
      sacl  *+, 0, AR2

      lar   AR2, #P
      lar   AR3, #(P+3)
      lar   AR4, #(P+4)
      lt    *+, AR3
      mpy   *-, AR2
```

```
      ltp   *, AR3
      mpy   *, AR4
      spac
      sach  *

      spm   0x02
      setc  ovm
      setc  sxm
      lacc  *+
      bcnd  KGM_INVDET1,EQ
      abs
      sacl  *
      lacc  #0x1000, 12
      rpt   #0x0f
      subc  *
      sacl  *-
      lacc  *+
      bcnd  KGM_INVDET2,GEQ
      neg
      sacl  *
      b     KGM_INVDET2
KGM_INVDET1
      lacc  #0x00
      sacl  *
KGM_INVDET2

      spm   0x02
      setc  ovm
      setc  sxm
      lar   AR2, #P
      lt    *, AR2
      .eval 0, i
      .loop
            mpy   *
            pac
            sach  *+
      .eval i+1,i
      .break i==4
      .endloop
KGM3
      spm   0x02
      setc  ovm
      setc  sxm
      lar   AR4, #K
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(RSLT + 2*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(P + j)
                  .eval 0, k
                  .loop
                        lta   *, AR3
                        mpy   *+, AR2
                        adrk  #0x02
                  .eval k+1, k
```

```
                  .break k==2
                  .endloop
                  apac
                  mar   *, AR4
                  sach  *+
            .eval j+1, j
            .break j==2
            .endloop
      .eval i+1, i
      .break i==5
      .endloop
KGM_END
*** now Kalman gain matrix was updated ***
*** update state estimation ***
SEST
      spm   0x02
      setc  ovm
      setc  sxm

      lar   AR2, #y
      lar   AR3, #x_1
      lar   AR4, #P
      mar   *, AR2
      lacc  *+, 0, AR3
      sub   *+, 0, AR4
      sacl  *+, 0, AR2
      lacc  *+, 0, AR3
      sub   *+, 0, AR4
      sacl  *+, 0, AR2

      lar   AR4, #RSLT
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(K+2*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(P+j)
                  .eval 0, k
                  .loop
                        lta   *+, AR3
                        mpy   *+, AR2
                  .eval k+1, k
                  .break k==2
                  .endloop
                  apac
                  mar   *, AR4
                  sach  *+
            .eval j+1, j
            .break j==1
            .endloop
      .eval i+1, i
      .break i==5
      .endloop
      lar   AR2, #x_1
      lar   AR3, #RSLT
      lar   AR4, #x
```

```
      mar   *, AR2
      .eval 0, i
      .loop
            lacc  *+, 0, AR3
            add   *+, 0, AR4
            sacl  *+, 0, AR2
      .eval i+1, i
      .break i==5
      .endloop
SEST_END
*** now state estimation was updated ***
*** update state estimation covariance matrix ***
SECM
            spm   0x02
            setc  ovm
            setc  sxm
            lar   AR4, #RSLT
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(K+2*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(C+j)
                  .eval 0, k
                  .loop
                        lta   *, AR3
                        mpy   *+, AR2
                        adrk  #0x05
                  .eval k+1, k
                  .break k==2
                  .endloop
                  apac
                  mar   *, AR4
                  sach  *+, 0, AR3
            .eval j+1, j
            .break j==5
            .endloop
      .eval i+1, i
      .break i==5
      .endloop

SECM1
            spm   0x02
            setc  ovm
            setc  sxm
            lar   AR4, #P
      .eval 0, i
      .loop
            .eval 0, j
            .loop
                  lar   AR3, #(RSLT + 5*i)
                  mpy   #0x00
                  pac
                  mar   *, AR2
                  lar   AR2, #(P_1 + j)
                  .eval 0, k
```

```
                .loop
                      lta   *, AR3
                      mpy   *+, AR2
                      adrk  #0x05
                .eval k+1, k
                .break k==5
                .endloop
                apac
                mar   *, AR4
                sach  *+
          .eval j+1, j
          .break j==5
          .endloop
    .eval i+1, i
    .break i==5
    .endloop

    lar   AR2, #P_1
    lar   AR3, #P
    mar   *, AR2
    .eval 0, i
    .loop
          lacc  *+, 0, AR3
          sub   *
          sacl  *+, 0, AR2
    .eval i+1, i
    .break i==25
    .endloop
SECM_END

*** now state estimation covariance matrix was updated ***

**********************************
***   end of EKF algoritm     ***
**********************************

*** calculation theta, sin, cos, and rotor flux value ***

SIN_COS
    setc  sxm
    spm   0x02
    setc  ovm
    mar   *, AR2
    lar   AR2, #(x+2)
    lar   AR3, #RSLT
    lt    *
    mpy   *+
    pac
    lt    *
    mpy   *, AR3
    apac
    sach  *
    lacc  *+
    add   #0x1000
    sfr
    sacl  *

    .eval 0, i
    .loop
```

```
      mar   *, AR2
      lar   AR2, #RSLT
      lacc  *+, 12
      rpt   #0x0f
      subc  *
      mar   *+
      sacl  *
      lacc  *-
      add   *
      sfr
      sacl  *
      .eval i+1, i
      .break i==4
      .endloop
      lar   AR2, #(RSLT+1)
      lar   AR4, #phir
      mar   *, AR2
      lacc  *, 0, AR4
      sacl  *

COS
      setc  sxm
      spm   0x02
      setc  ovm
      mar   *, AR2
      lar   AR2, #(x+2)
      lar   AR3, #phir
      lar   AR4, #cos

      lacc  *, 0, AR4
      bcnd  SC_C1, EQ
      abs
      sacl  *, 0, AR3
      lacc  *, 0, AR4
      bcnd  SC_C1,EQ
      lacc  *, 12, AR3
      rpt   #0x0f
      subc  *
      mar   *, AR4
      sacl  *, 0, AR2
      lacc  *, 0, AR4
      bcnd  SC_C2,GEQ
      lacc  *
      neg
      sacl  *
      b     SC_C2
SC_C1
      lacc  #0x1000
      sacl  *
SC_C2
******
SIN
      setc  sxm
      spm   0x02
      setc  ovm
      mar   *, AR2
      lar   AR2, #(x+3)
      lar   AR3, #phir
      lar   AR4, #sin
```

```
      lacc  *, 0, AR4
      bcnd  SC_S1, EQ
      abs
      sacl  *, 0, AR3
      lacc  *, 0, AR4
      bcnd  SC_S1,EQ
      lacc  *, 12, AR3
      rpt   #0x0f
      subc  *
      mar   *, AR4
      sacl  *, 0, AR2
      lacc  *, 0, AR4
      bcnd  SC_S2,GEQ
      lacc  *
      neg
      sacl  *
      b     SC_S2
SC_S1
      lacc  #0x0000
      sacl  *
SC_S2

*******
*** calculation theta, sin, cos, and rotor flux value is OK!!***

*********************************************************
*           end of state estimation             *
*********************************************************

******* inverter STROBE *******
***
      mar   *, AR0
      lar   AR0, #PBDATDIR
      lacc  *
      or    #0x0010
      sacl  *

*** IOPB4 is high
*******************************

*********************************************************
*                 FOC Controller                *
*********************************************************

*** calculation iD command ***
IDCC
      setc  sxm
      spm   0x02
      lacc  #iDn
      mar   *, AR0
      lar   AR0, #iD_cm
      sacl  *
***

*** calculation torque commnad ***
TRQCC
      setc  sxm
      spm   0x02
```

```
      setc  ovm

      lar   AR0, #ST_UP
      lacc  *
      bcnd  ROT_SPEED, NEQ
      lar   AR0, #phir
      lacc  *
      sub   #0x0600
      bcnd  IQCC_L1, LT
      lacc  #0x01
      lar   AR0, #ST_UP
      sacl  *

ROT_SPEED
      ;b    TRQ_L0
      lar   AR0, #wr_cm
      lar   AR1, #wr_ind
      mar   *, AR1
      lacc  *
      add   #0x01
      sacl  *
      sub   #0x8fff
      bcnd  TRQ_L0, NEQ
      sacl  *, 0, AR0
      lacc  *
      sub   #0x1000
      bcnd  ROT_SPEED1, EQ
      lacc  #0x1000
      sacl  *
      b     ROT_SPEED2
ROT_SPEED1
      lacc  #0xf000
      sacl  *

ROT_SPEED2
TRQ_L0
      spm   0x02
      setc  sxm
      setc  ovm
      mar   *, AR3
      lar   AR2, #(x+4)
      lar   AR3, #wr_cm
      lar   AR4, #(trq_cm + 1)
      lar   AR5, #trq_Kp
      lar   AR6, #RSLT

        lacc    *, 0, AR2
      sub   *, 0, AR6
      sacl  *

      lt    *+, AR5
      mpy   *, AR4
      pac
      add   *-, 16
      sach  *
      lacc  *, 0, AR6
      sacl  *, 0, AR4

      .eval 0xffff - trq_lim + 1, trq_lim_neg
```

```
      lacc  *
      sub   #trq_lim
      bcnd  TRQ_L1, GT
      lacc  *
      sub   #trq_lim_neg
      bcnd  TRQ_L2, LT
      b     TRQ_L3
TRQ_L1
      lacc  #trq_lim
      sacl  *
      b     TRQ_L3
TRQ_L2
      lacc  #trq_lim_neg
      sacl  *
TRQ_L3
      lacc  *+, 0, AR6
      sub   *
      sacl  *
      lt    *-
      mpy   #trq_Kc
      ltp   *, AR4
      mpy   #trq_Ki
      apac
      add   *, 16
      sach  *
***

*** calculation iQ command ***

IQCC
      setc  sxm
      spm   0x02
      setc  ovm
      mar   *, AR2
      lar   AR2, #trq_cm
      lar   AR3, #phir
      lar   AR4, #iQ_cm

      lacc  *, 0, AR4
      bcnd  IQCC_L1, EQ
      abs
      sacl  *, 0, AR3
      lacc  *, 0, AR4
      bcnd  IQCC_L1,EQ
      lacc  *, 12, AR3
      rpt   #0x0f
      subc  *
      mar   *, AR4
      sacl  *, 0, AR2
      lacc  *, 0, AR4
      bcnd  IQCC_L2,GEQ
      lacc  *
      neg
      sacl  *
      b     IQCC_L2
IQCC_L1
      lacc  #0x0000
      sacl  *
IQCC_L2
```

```
      lt    *
      mpy   #G_iQ
      pac
      sach  *

***
***
*** inverse park projection (DQ=>dq) ***
IPPC
      setc  sxm
      spm   0x02
      setc  ovm

      lar   AR2, #cos
      lar   AR3, #iD_cm
      lar   AR4, #id_cm
      mar   *, AR3

      lt    *+, AR2
      mpy   *-, AR3
      ltp   *-, AR2
      mpy   *, AR4
      spac
      sach  *+, 0, AR3

      lt    *+, AR2
      mpy   *+, AR3
      ltp   *, AR2
      mpy   *, AR4
      apac
      sach  *
***

*** inverse clark projection (dq=>abc conversion) for command***
ICPC
      setc  sxm
      spm   0x02
      setc  ovm

      mar   *, AR2
      lar   AR2, #id_cm
      lar   AR4, #ia_cm

      lacc  *+, 0, AR4
      sacl  *+, 0, AR2
      lt    *-
      mpy   #sqrt3_2
      pac
      sub   *, 15, AR4
      sach  *-
      lacc  *+
      add   *+
      neg
        sacl    *
***


*** histerezis band current controller ***
HBCR
```

```
      .if   (SS==1)
      b     SX_STEP
      .endif

      setc  sxm
      spm   0x02
      setc  ovm
      .eval 0xffff - hb +1, hb_neg
HBA
      lar   AR2, #ia_cm
      lar   AR3, #ia
      lar   AR4, #sa
      lar   AR5, #CMPR1
      lar   AR6, #RSLT
      mar   *, AR2
      lacc  *, 0, AR3
      sub   *, 0, AR6
      sacl  *
      lacc  *
      sub   #hb
      bcnd  HBA_L1, GT
      lacc  *
      sub   #hb_neg
      bcnd  HBA_L2, LT
      b     HBA_L3
HBA_L1
      mar   *, AR4
      lacc  #0x01
      sacl  *, 0, AR5
      lacc  #sw_on
      sacl  *
      b     HBA_L3
HBA_L2
      mar   *, AR4
      lacc  #0x00
      sacl  *, 0, AR5
      lacc  #sw_off
      sacl  *
HBA_L3
***
HBB
      mar   *, AR2
      mar   *+, AR3
      mar   *+, AR4
      mar   *+, AR5
      mar   *+, AR2

      lacc  *, 0, AR3
      sub   *, 0, AR6
      sacl  *
      lacc  *
      sub   #hb
      bcnd  HBB_L1, GT
      lacc  *
      sub   #hb_neg
      bcnd  HBB_L2, LT
      b     HBB_L3
HBB_L1
      mar   *, AR4
```

```
      lacc  #0x01
      sacl  *, 0, AR5
      lacc  #sw_on
      sacl  *
      b     HBB_L3
HBB_L2
      mar   *, AR4
      lacc  #0x00
      sacl  *, 0, AR5
      lacc  #sw_off
      sacl  *
HBB_L3
***
HBC
      mar   *, AR2
      mar   *+, AR3
      mar   *+, AR4
      mar   *+, AR5
      mar   *+, AR2

      lacc  *, 0, AR3
      sub   *, 0, AR6
      sacl  *
      lacc  *
      sub   #hb
      bcnd  HBC_L1, GT
      lacc  *
      sub   #hb_neg
      bcnd  HBC_L2, LT
      b     HBC_L3
HBC_L1
      mar   *, AR4
      lacc  #0x01
      sacl  *, 0, AR5
      lacc  #sw_on
      sacl  *
      b     HBC_L3
HBC_L2
      mar   *, AR4
      lacc  #0x00
      sacl  *, 0, AR5
      lacc  #sw_off
      sacl  *
HBC_L3

*********************************************************
*           end of      FOC Controller                  *
*********************************************************


*****************************************************
*     six-step control  for v/f control             *
*     this routine used for speed estimation test   *
*     if SS = 1 then six-step control               *
*     else vector control                           *
*****************************************************

      .if   (SS!=1)
      b     SX_STEP_END
```

```
      .endif

SX_STEP
      setc  sxm
      spm   0x00
      mar   *, AR2
      lar   AR2, #sx_step
      lacc  *
      sub   #0x01
      sacl  *
      bcnd  SX_STEP_END, GEQ
      lacc  #inv_step
      sacl  *

      lar   AR2, #sector
      lar   AR3, #RSLT
      mar   *, AR2

      .if   (DIR!=1)
SX_STEP1
      lacc  *
      add   #0x01
      sacl  *
      sub   #0x02
      bcnd  SX_STEP2, LEQ
      lacc  #0x00
      sacl  *
      .endif

      .if   (DIR==1)
SX_STEP1
      lacc  *
      sub   #0x01
      sacl  *
      bcnd  SX_STEP2, GEQ
      lacc  #0x02
      sacl  *
      .endif

SX_STEP2
      lacc  *
      mar   *, AR3
      add   #(sector+1)
      sacl  *
      lar   AR4, *, AR2
      lacc  *, 0, AR3
      add   #CMPR1
      sacl  *
      lar   AR5, *, AR4
      lacc  *
      bcnd  SX_STEP_SW_ON, EQ

SX_STEP_SW_OFF
      lacc  #0x00
      sacl  *, 0, AR5
      lacc  #sw_off
      sacl  *, 0, AR2
      b     SX_STEP_END
```

```
SX_STEP_SW_ON
      lacc  #0x01
      sacl  *, 0, AR5
      lacc  #inv_pw
      sacl  *, 0, AR2

SX_STEP_END

*************************************
*     end of six-step control       *
*************************************

*********************************************************
*           ADC read routine             *
*********************************************************

*** get ADC ***
GET_ADC
      clrc  sxm
      mar   *, AR2
      lar   AR2, #ADCTRL1
      lacc  *
      or    #0x000e
      sacl  *
      nop
      nop
      nop
      lacc  *
      or    #0x0001
      sacl  *
      nop
      nop
      nop
GET_ADC0
      bit   *, (15-7)
      bcnd  GET_ADC0, TC

      mar   *, AR2
      lar   AR2, #ADCFIFO1
      lar   AR4, #adc_rd1
      lacc  *, 10, AR4
      sach  *, 0, AR2
      lacc  *, 10, AR4
      adrk  #0x03
      sach  *-, 0, AR2

      adrk  #0x02
      lacc  *, 10, AR4
      sach  *, 0, AR2
      lacc  *, 10

GET_ADC_END
*****************************************
*** next period   current select for read ***
*****************************************

      mar   *, AR2
      lar   AR2, #ADCTRL1
      lacc  *
```

```
      and   #0xff81
      or    #0x0010     ; for read ia, ib
      sacl  *

*********************************
*** rd1,2,3 => ia,b,c and vdc ***
*********************************

      setc  sxm
      spm   0x02

      lar   AR2, #adc_rd3
      mar   *, AR2
      lt    *+
      mpyu  *+
      pac
      sach  *

      .if   (SS == 1)
      lacc  *
      sfr
      sacl  *
      .endif

      spm   0x00
      lar   AR2, #adc_rd1
      lar   AR4, #ia
      mar   *, AR2
      lacc  #i_off
      sub   *+, 0, AR4
      sacl  *
      lt    *, AR2
      mpy   *+, AR4
      pac
      sach  *+, 6, AR2

      lacc  #i_off
      sub   *-, 0, AR4
      sacl  *
      lt    *, AR2
      mpy   *, AR4
      pac
      sach  *-, 6

      lacc  *+
      add   *+
      neg
      sacl  *

*****************************************
*** clark projection (abc=>dq conversion) ***
*****************************************

      setc  sxm
      spm   0x02

      mar   *, AR2
      lar   AR2, #ia
      lar   AR4, #y
```

```
      lacc  *+, 0, AR4
      sacl  *+, 0, AR2
      add   *, 1, AR4
      sacl  *
      lt    *
      mpy   #inv_sqrt3
      pac
      sach  *

      lar   AR2, #vdc
      lar   AR4, #u
      mar   *, AR2
      lt    *, AR4
      mpy   #inv3
      pac
      sach  *+
      mpy   #inv_sqrt3
      pac
      sach  *


      spm   0x00
      lar   AR2, #sa
      lar   AR3, #RSLT
      lar   AR4, #u
      mar   *, AR2
      lacc  *+, 1
      sub   *+
      sub   *-, 0, AR3
      sacl  *
      lt    *, AR4
      mpy   *
      pac
      sacl  *+, 0, AR2

      lacc  *+
      sub   *, 0, AR3
      sacl  *
      lt    *, AR4
      mpy   *
      pac
      sacl  *

*********************************************************
*           end of ADC read routine             *
*********************************************************

*************************
**** graphic routine ******
*************************

      b     gr1
      clrc  sxm
      lar   AR2, #gr_ind1
      mar   *, AR2
      lacc  *
      add   #0x01
      sacl  *
```

```
      sub   #0x10
      bcnd  graph_end, NEQ
      sacl  *
gr1
      lar   AR0, #0x1000
      lar   AR1, #gr_ind
      mar   *, AR1
      lar   AR2, *, AR3
      lar   AR3, #y
      lar   AR4, #x+2
      lar   AR5, #x+4
graph_l1
      lacc  *, 0, AR2
      sacl  *0+, 0, AR4
      lacc  *, 0, AR2
      sacl  *0+, 0, AR5
      lacc  *, 0, AR2
      sacl  *, 0, AR1
graph_l2
      lacc  *
      add   #0x01
      sacl  *
      sub   #0xa000
      bcnd  graph_end, LT
      lacc  #0xa000
      sacl  *
*****************************
graph_end
*****************************
*****************************
* period suffient control ****
      lar   AR5, #T1CNT
      mar   *, AR5
      bldd  *, #t1cnt
*******************************

*** flashing XF led counter ****
      lar   AR5, #1
      mar   *, AR5
      lacc  *
      add   #0x40
      sacl  *
*******************************

***
      lar   AR5, #EVIFRA
      mar   *, AR5
      lacc  #0x0200
      sacl  *
***
*** context restore ***
      mar   *,AR7
      mar   *+
      lacl  *+
      add   *+,16
      lst   #00h,*+
      lst   #01h,*+
      clrc  intm
***
```

```
      ret


*** setup hardware and software

SETUP
      setc  intm

      lacc  #0x0000
      mar   *, AR0
      sacl  *
      out   *, WSGR
      lar   AR7,#STACK

*** hardware init ***
      lacc  #0x00
      mar   *, AR0
      lar   AR0, #GPTCON
      sacl  *+    ; GPTCON
      sacl  *+    ; T1CNT
      sacl  *+    ; T1CMPR
      sacl  *+    ; T1PER
      sacl  *+    ; T1CON
      sacl  *+    ; T2CNT
      sacl  *+    ; T2CMPR
      sacl  *+    ; T2PER
      sacl  *+    ; T2CON
      lar   AR0, #COMCON
      sacl  *     ; COMCON
      adrk  #0x02
      sacl  *     ; ACTR
      adrk  #0x02
      sacl  *     ; DBTCON
      adrk  #0x02
      sacl  *+    ; CMPR1
      sacl  *+    ; CMPR2
      sacl  *+    ; CMPR3
      lar   AR0, #CAPCON
      sacl  *+    ; CAPCON
      mar   *+
      sacl  *+    ; CAPFIFO
      sacl  *+    ; CAP1FIFO
      sacl  *+    ; CAP2FIFO
      sacl  *+    ; CAP3FIFO

      lar   AR0, #T1PER
      lacc  #period
      sacl  *
      .eval (sw_on-adc_delay), adc_start_now
      lacc  #adc_start_now
      lar   AR0, #T1CMPR
      sacl  *-
      lacc  #0x0000
      sacl  *     ; T1CNT
      lacc  #0x8000
      lar   AR0, #COMCON
      sacl  *
      lacc  #0xd002
```

```
        lar   AR0, #T1CON
        sacl  *
        lacc  #0x0666
        lar   AR0, #ACTR
        sacl  *
        lacc  #0x05e0
        lar   AR0, #DBTCON
        sacl  *
        lacc  #sw_off
        lar   AR0, #CMPR1
        sacl  *+
        sacl  *+
        sacl  *+
        lacc  #0x8200
        lar   AR0, #COMCON
        sacl  *

        lar   AR0, #T1CON
        lacc  *
        or    #0x0040
        sacl  *

        lacc  #0x0180
        lar   AR0, #GPTCON
        sacl  *

        lacc  #0xd810
        lar   AR0, #ADCTRL1
        sacl  *+

        lacc  #0x0400
        mar   *+
        sacl  *     ; ADCTRL2

        lar   AR0, #OCRA
        lacc  *
        or    #0x0fc0
        sacl  *+

        lacc  #0x0000
        mar   *+
        sacl  *

        lacc  #0xffc0
        lar   AR0, #PBDATDIR
        sacl  *

*** interrupt init ***

        lacc  #0x0000
        lar   AR0, #EVIMRA
        sacl  *+    ; EVIMRA
        sacl  *+    ; EVIMRB
        sacl  *+    ; EVIMRC
        lacc  #0x0201
        lar   AR0, #EVIMRA
        sacl  *
        lacc  #0xffff
        lar   AR0, #EVIFRA
```

```
      sacl  *+
      sacl  *+
      sacl  *+

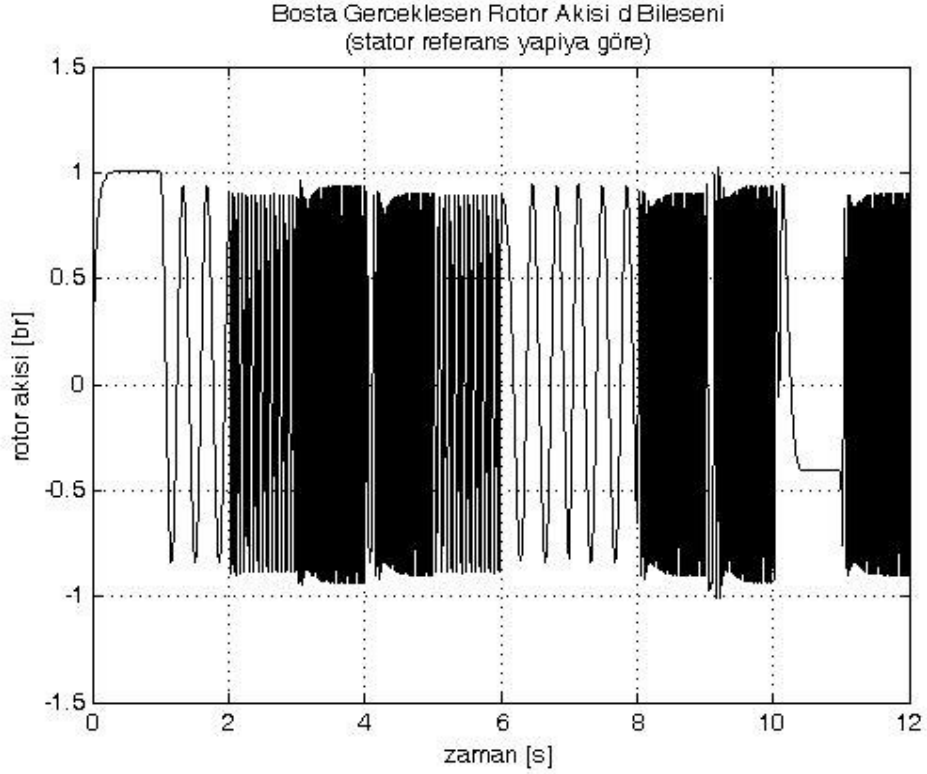
      lar   AR0, #IMR
      lacc  *
      or    #0x0003
      sacl  *

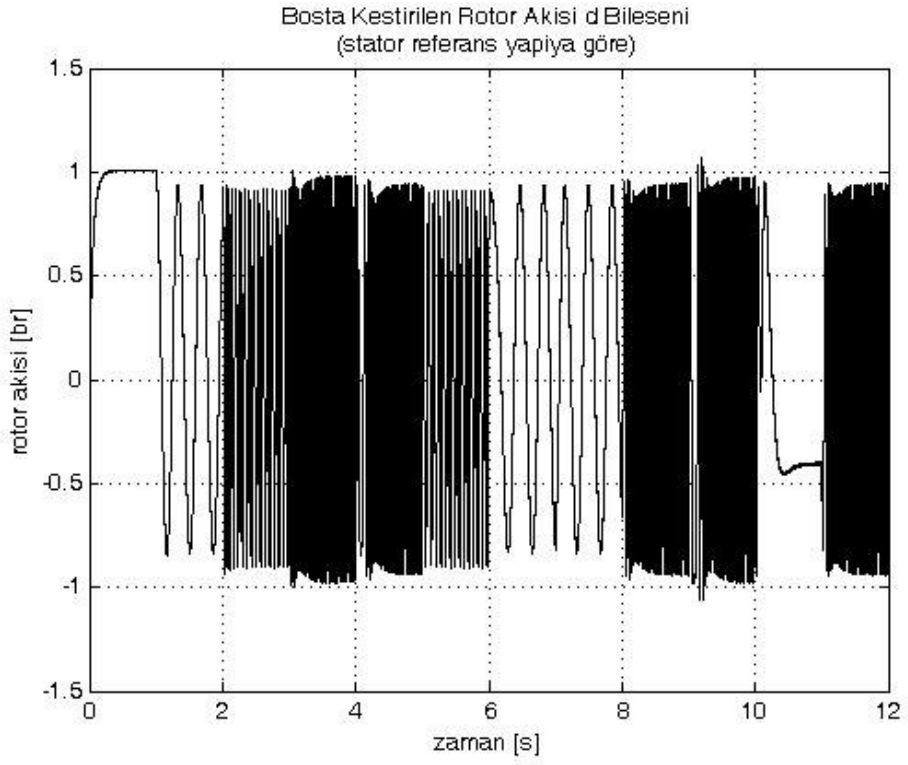
      clrc  intm
MAIN
      mar   *, AR0
      lar   AR0, #1
      lacc  *
      sub   #0x8000
      bcnd  M1, LT
      clrc  XF
      b     MAIN
M1
      setc  XF
      b     MAIN

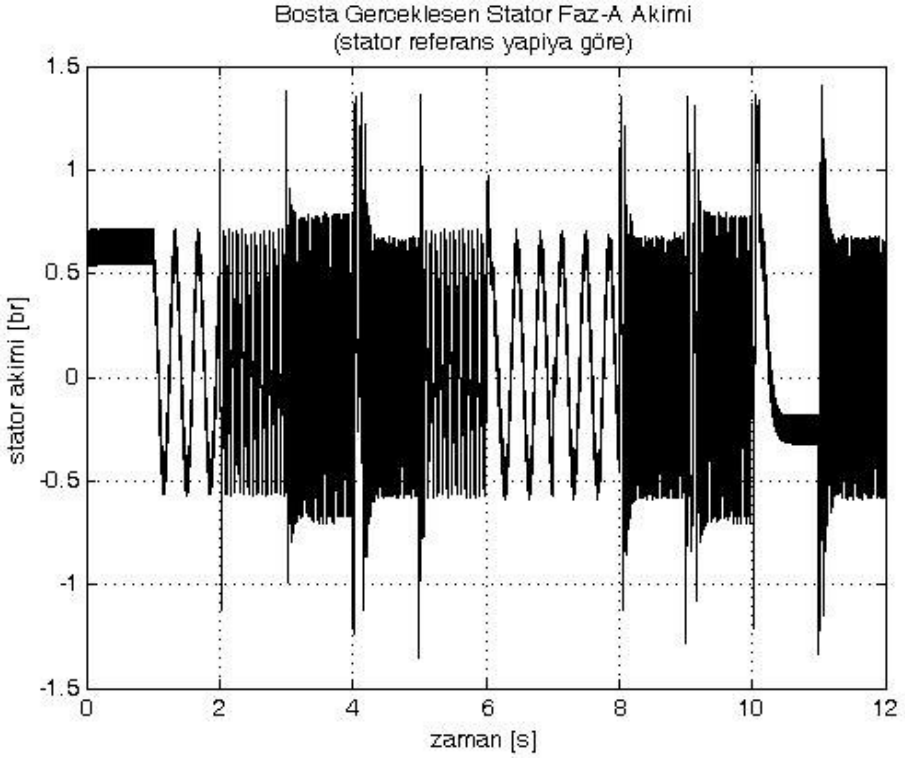
*** power drive protection interrupt ***
PDP_ISR
*** context save ***
      mar   *, AR7
      mar   *-
      sst   #01h,*-
      sst   #00h,*-
      sach  *-
      sacl  *-
***
      lar   AR0, #PBDATDIR
      lacc  *
      and   #0x0ff7f
      sacl  *
      lar   AR0, #IMR
      lacc  *
      and   #0xfffd
      sacl  *
***
      lar   AR0, #EVIFRA
      lacc  #0x0001
      sacl  *
*** context restore ***
      mar   *, AR7
      mar   *+
      lacl  *+
      add   *+, 16
      lst   #00h,*+
      lst   #01h,*+
***
      ret
```

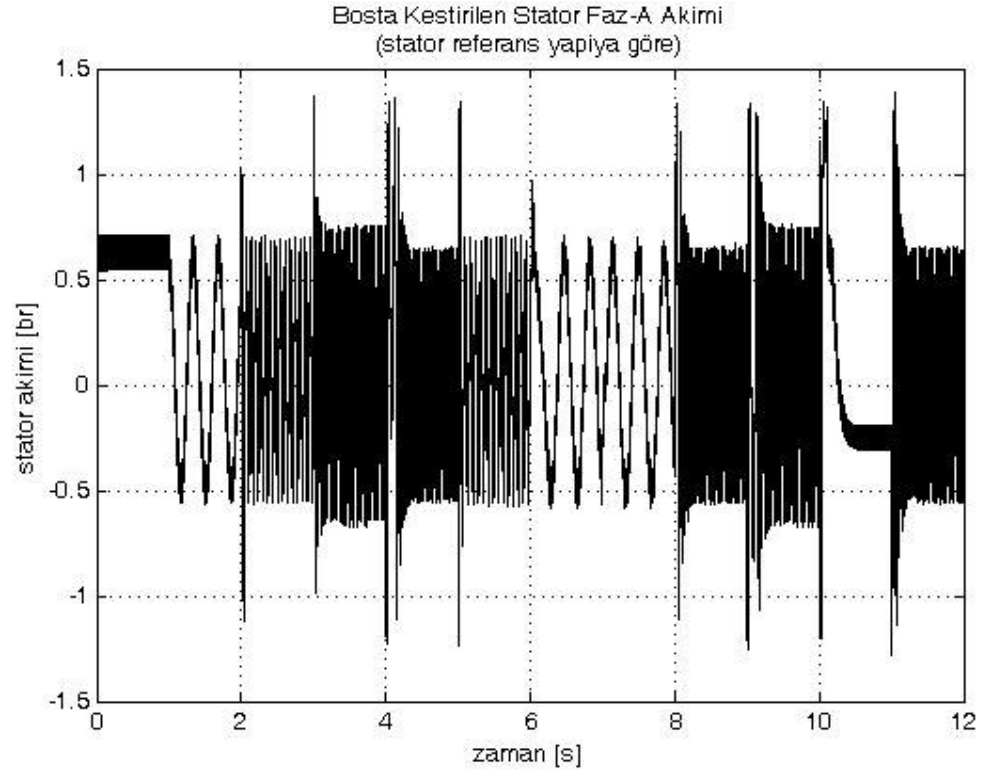
EK-D SİMÜLASYON ÇALIŞMASINDAN ELDE EDİLEN GRAFİKLER

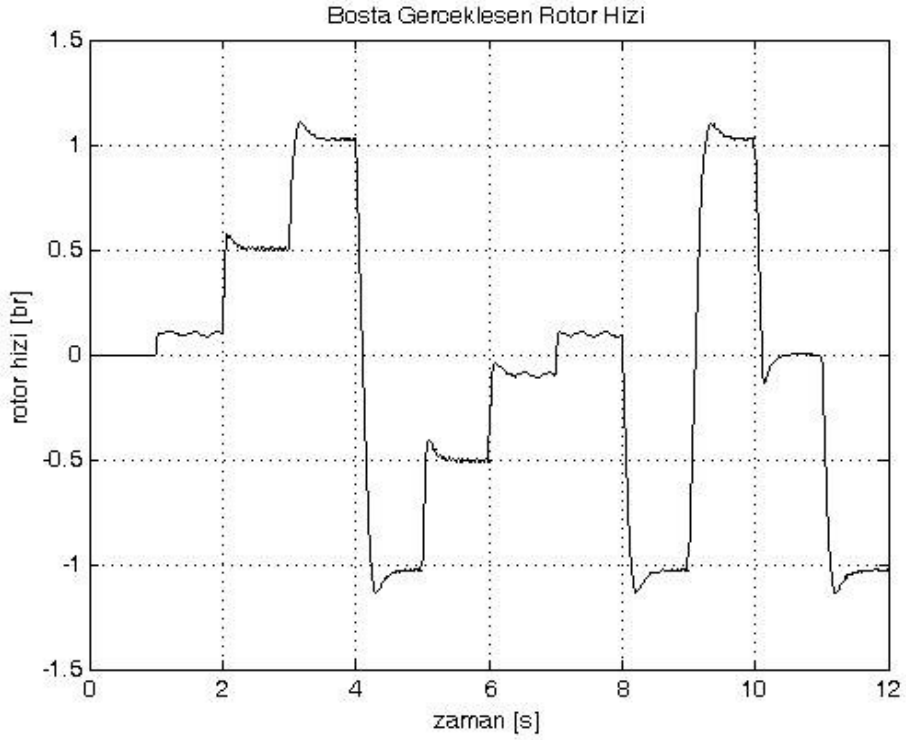
a. Boşta Çalışma için Yapılan Simülasyondan Elde Edilen Grafikler

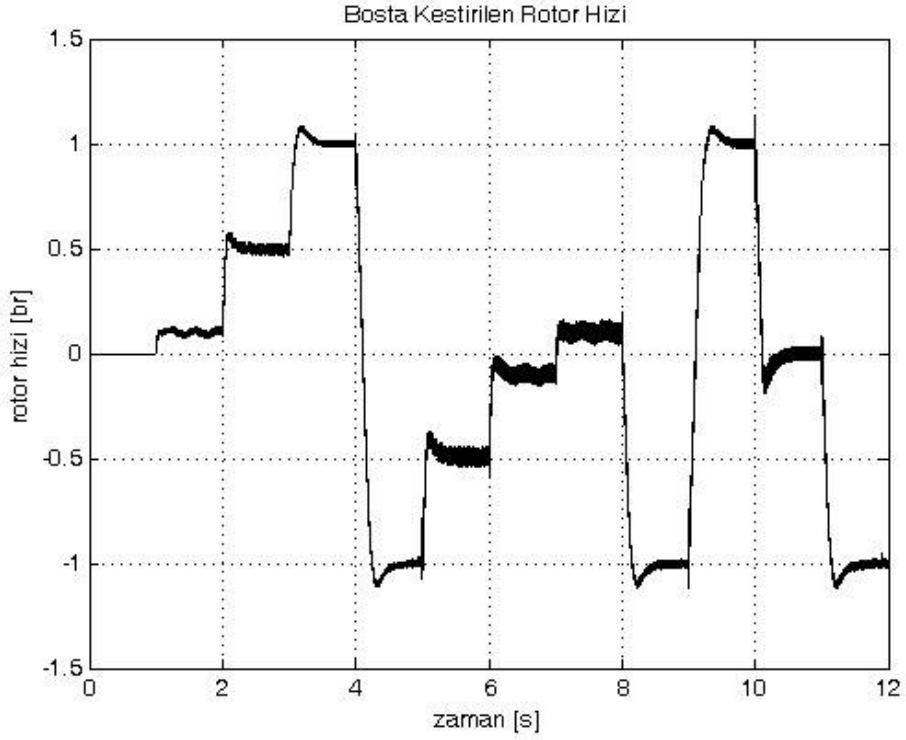


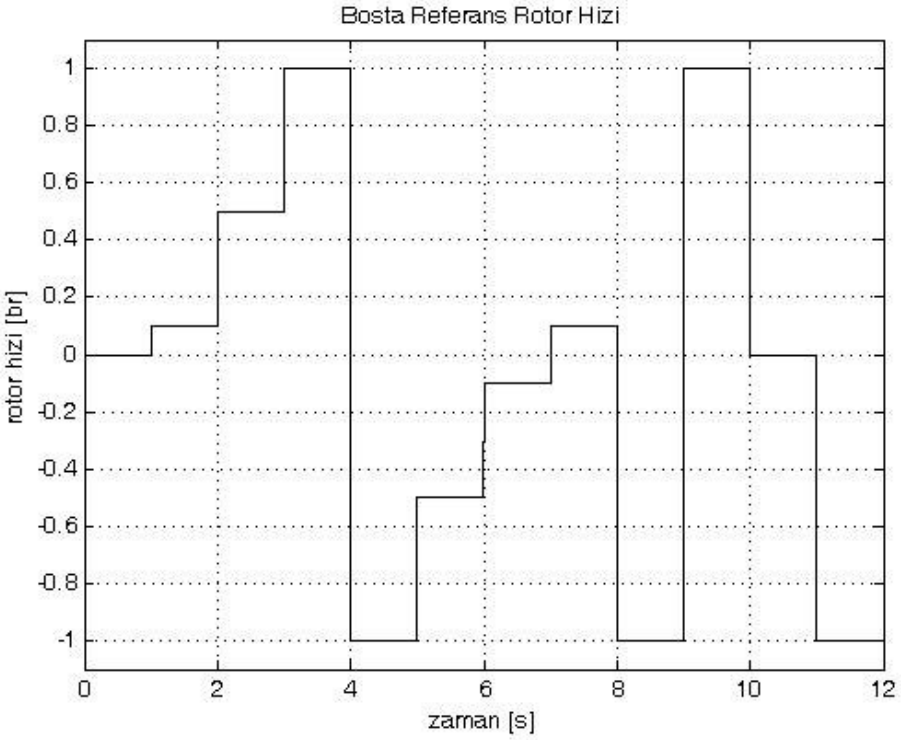



Bosta Gerceklesen Stator Faz-A Akimi
(stator referans yapiya göre)
stator akimi [br]
zaman [s]
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

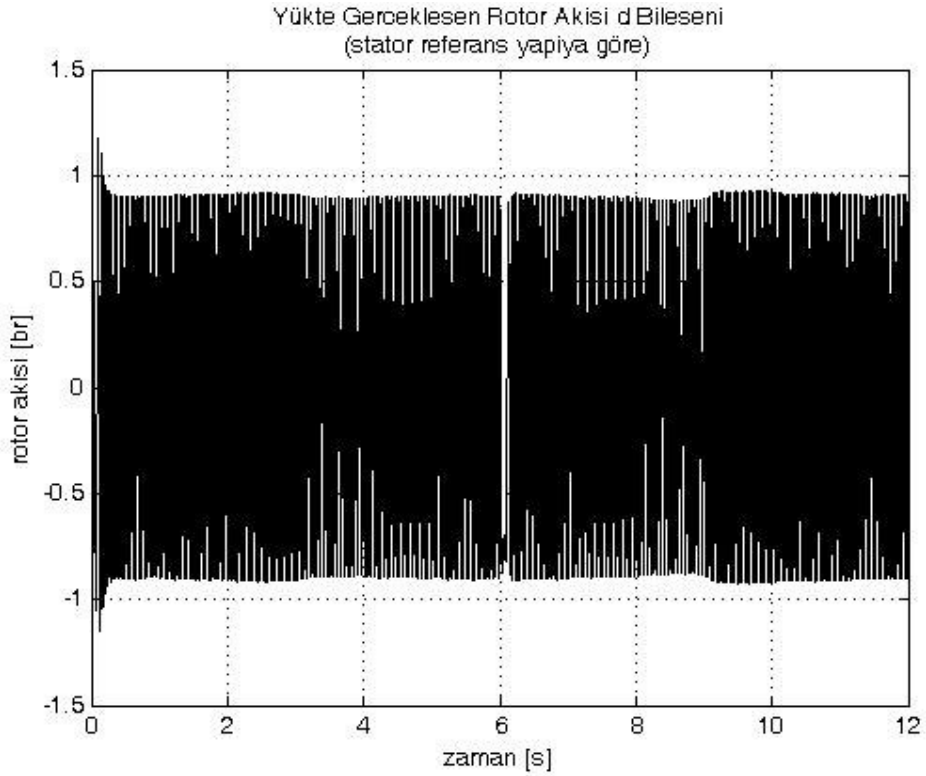

Bosta Kestirilen Stator Faz-A Akimi
(stator referans yapiya göre)
stator akimi [br]
zaman [s]
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

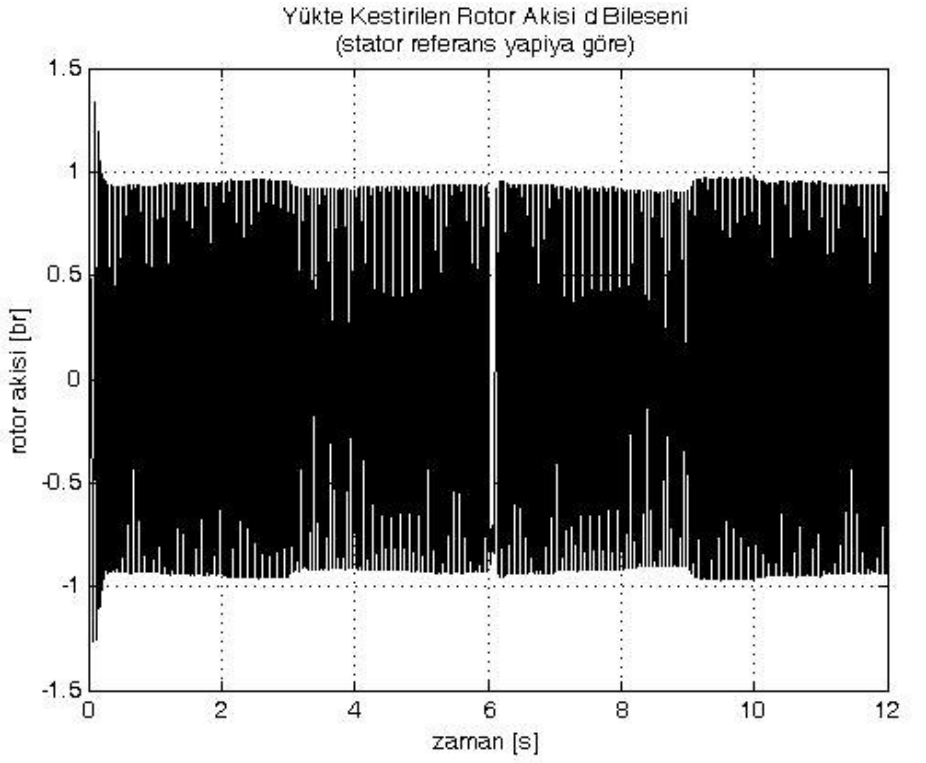

Bosta Gerceklesen Rotor Hizi
rotor hizi [br]
zaman [s]
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

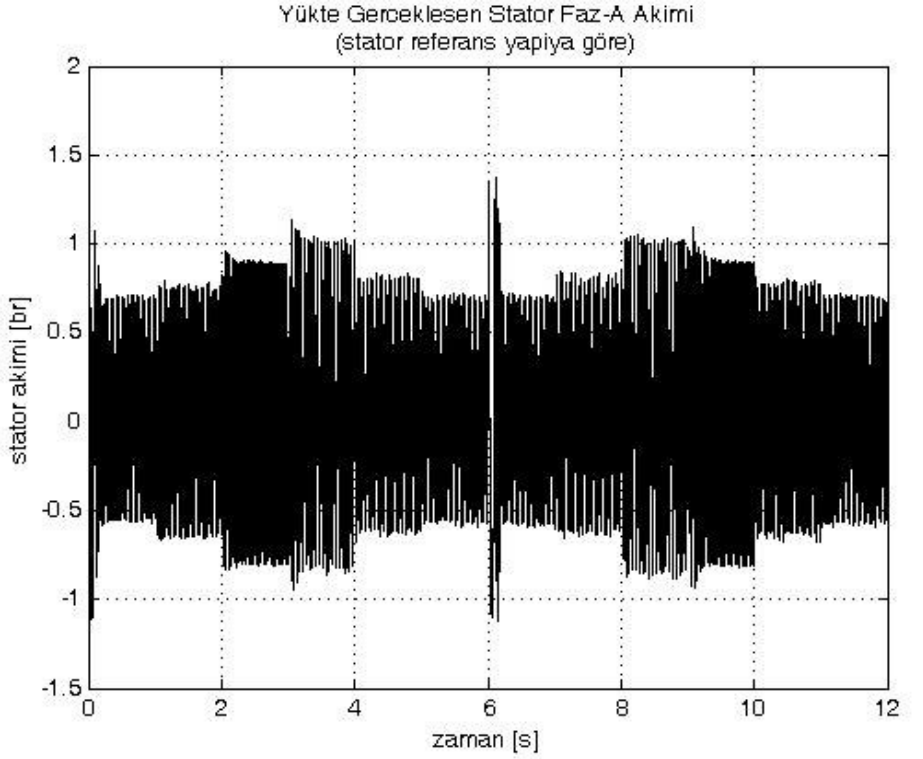

Bosta Kestirilen Rotor Hizi
rotor hizi [br]
zaman [s]
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12

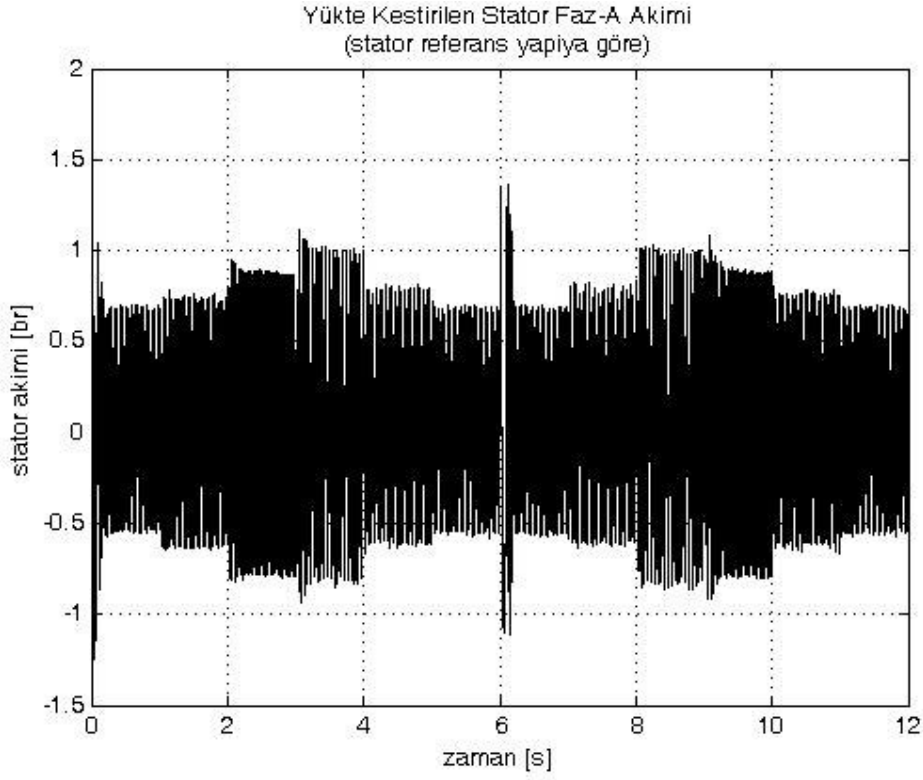

Bosta Referans Rotor Hizi
rotor hizi [br]
zaman [s]
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12

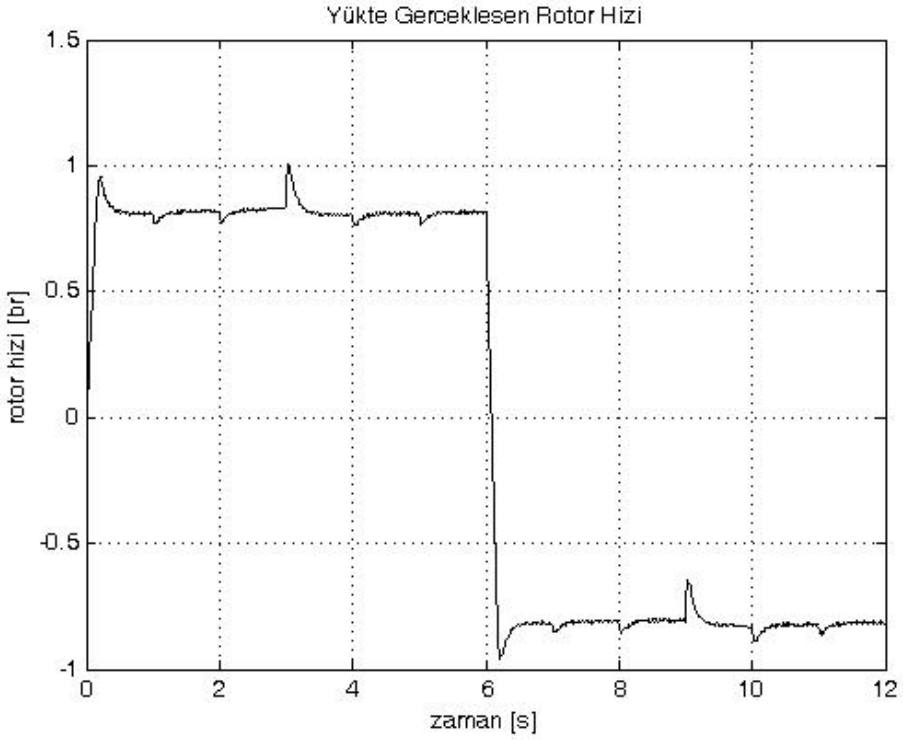
b. Yüklü Çalışma için Yapılan Simülasyondan Elde Edilen Grafikler

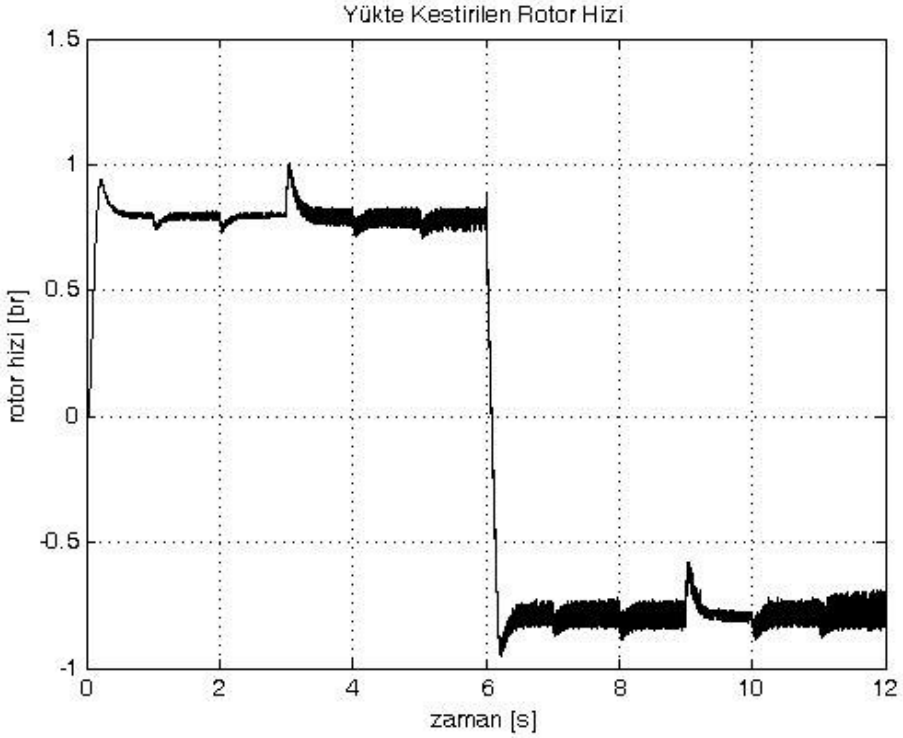


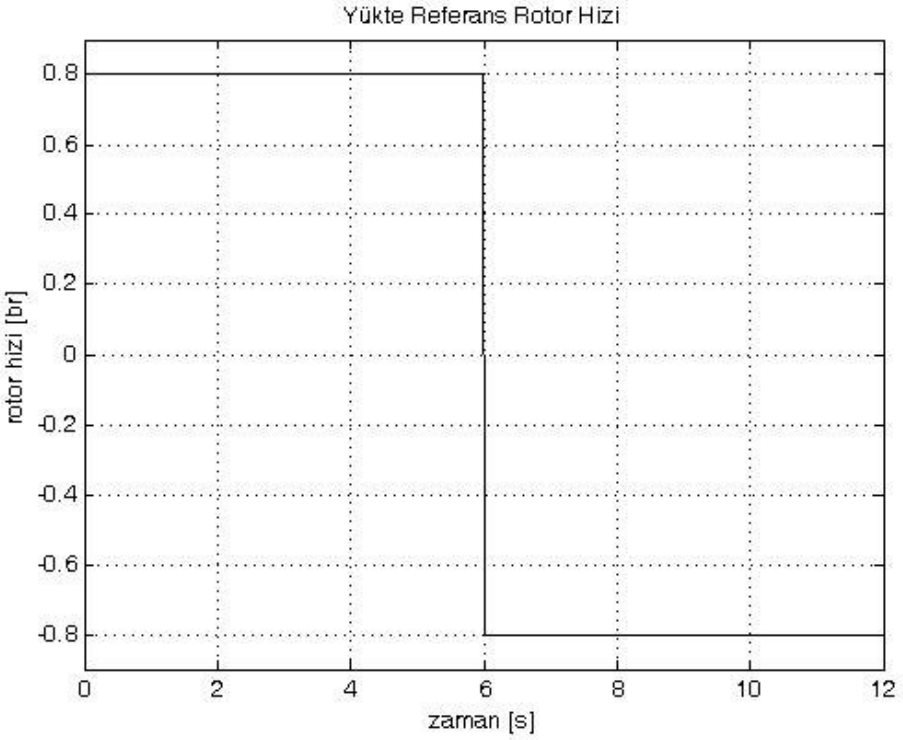


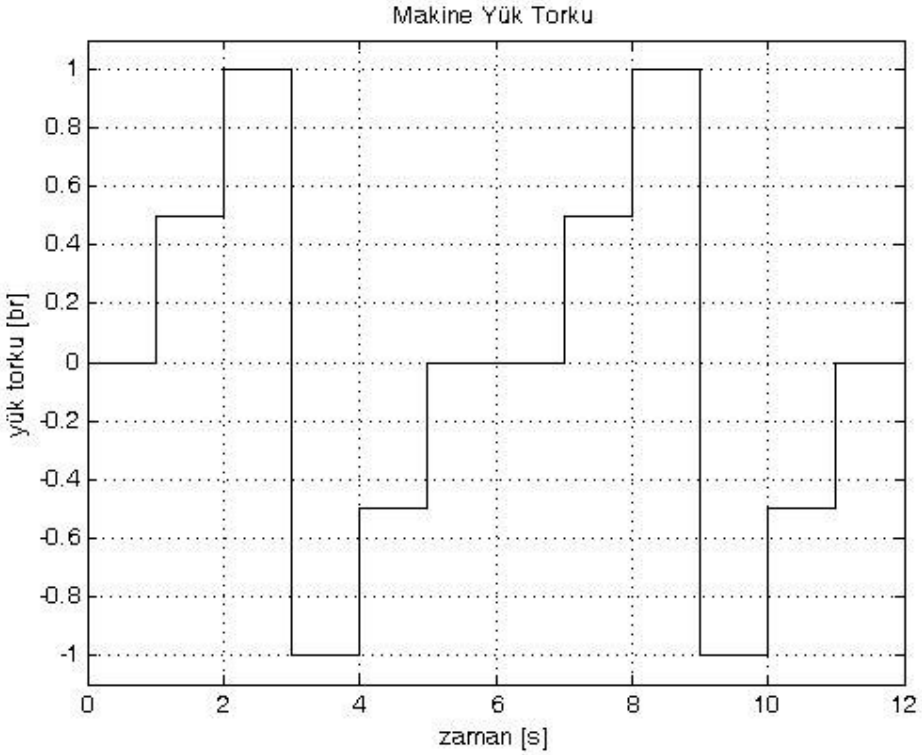

Yükte Gerceklesen Stator Faz-A Akimi
(stator referans yapiya göre)
stator akimi [br]
zaman [s]
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12


Yükte Kestirilen Stator Faz-A Akimi
(stator referans yapiya göre)
stator akimi [br]
zaman [s]
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12


Yükte Gerceklesen Rotor Hizi
rotor hizi [br]
zaman [s]
1.5
1
0.5
0
-0.5
-1
0
2
4
6
8
10
12


Yükte Kestirilen Rotor Hizi
rotor hizi [br]
zaman [s]
1.5
1
0.5
0
-0.5
-1
0
2
4
6
8
10
12


Yükte Referans Rotor Hizi
rotor hizi [br]
zaman [s]
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
2
4
6
8
10
12


Makine Yük Torku
yük torku [br]
zaman [s]
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12