YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR KONTROLLÜ, PROGRAMLANABİLİR, BASKILI DEVRE MATKAP TEZGAHI

Elektronik Müh. Altuğ ORHAN

F.B.E. Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Elektronik Programında hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Şefik SARIKAYALAR

İSTANBUL, 1994

İÇİNDEKİLER

TEŞEKKUR

Bu çalışma, günümüzde her alana girmiş olan otomasyon kavramı hakkında bilgi verme, bu konuya ilgi duyanlara yapacakları çalışmalarda yol gösterme, karşılaşılabilecek zorluklar ve bunların giderilmesi hakkında fikir verme amacı güdülerek yapılmıştır.

Kendi adıma, çok ilgi duyduğum bu konuda ve özellikle step motorlarla kontrol konusunda çok önemli bilgiler edinmemi sağlayan bu projenin gerçekleşmesinde, her türlü yardımı benden esirgemeyen, başta tez yönetici hocam sayın Prof. Şefik SARIKAYALAR olmak üzere Ekip Ltd. Gen. Müd. Elektronik Yük. Müh. sayın Şevket KOÇAK' a, sayın Elektronik Müh. Müşerref KAHRIMAN' a, sayın Elektronik Müh. Ahmet Oğuz BARAN' a ve sayın Nazlı SARIKAYA' ya teşekkürlerimi bir borç bilirim.

Altuğ ORHAN

07-02-1994

ÖZET

Bilgisayar kontrollü, programlanabilir baskılı devre matkap tezgahı üç ayrı sistemin biraraya getirilmesiyle oluşmuştur:

1. Donanım,
2. Yazılım,
3. Mekanik.

Sistem tasarlanırken, donanım ve yazılım bölümleri ön planda tutulmuştur. Mekanik ise, yazılım ve donanımın ihtiyaçlarına cevap verecek minimum şekilde tasarlanmıştır. Sistem yazılım ağırlıklı olup, motorlara ilişkin tüm kontrol işaretleri yazılım tarafından oluşturulmaktadır.

Donanımın en önemli parçası, iki adet 8255 PIA (Programmable Interface Adapter) içeren giriş/çıkış kartıdır. Toplam 48 adet giriş ve çıkışa olanak veren bu kart, yazılımla, motor sürücüleri ve referans noktalarına ait giriş bilgileri gibi diğer elektronik devreler arasındaki bilgi alışverişini sağlayan bir arabirimdir. Bu kartı PC ile step motor sürücü kartlarını elektriksel olarak birbirinden ayıran opto-kuplör kartı izlemektedir ve PC' nin toprak hattıyla, motor sürücü kartlarının ve diğer elektronik devrelerin toprak hatları birbirinden tamamen izoledir. Bu kartın ardından step motor sürücü kartları ve matkap motoru sürücü kartıyla giriş devreleri gelir. Sürücü kartların step motorları çalıştırmasıyla mekanik hareket sağlanır.

Sistemin yazılımı Turbo Pascal v7.0 kullanılarak yazılmıştır. Tüm kontroller yazılımın denetimi altındadır. Step motorlar, açık çevrim kontrolü ile çalıştırılmaktadır. Yük koşulları fazla ağır olmadığından bu tip kontrol uygun görülmüştür. Yeni bir kart dosyası oluşturma, hazır bir dosyayı çağırarak kart delme gibi özellikler oldukça anlaşılır bir şekilde kullanıcıya sunulmuştur. Giriş çıkış sayısının kullanılandan fazla olması nedeniyle step motorlara ilişkin 4 bitlik faz kontrol bilgileri herhangi bir ek devre kullanılmaksızın doğrudan PC tarafından üretilip gönderilmektedir.

Bu çalışmanın ilk iki bölümünde, step motorlar ve sürücü devreleri, üçüncü bölümde ise mikroişlemci temelli step motor sürücü devreleri ayrıntılarıyla incelenmiş ve ilerleyen bölümlerde bu bilgiler kullanılarak baskılı devre matkap tezgahının tasarımı yapılmıştır.

ABSTRACT

PC controlled, programmable PCB drill workbench consists of three different systems:

1. Hardware,
2. Software,
3. Mechanics.

When the system is being built, hardware and software is considered urgent and mechanics is constructed with minimum structure to cope with hardware and software's simplicities. The system is software intensive and all the control signals of motors is produced by software.

The most important part of hardware is that an I/O card which has two 8255 PIA (Programmable Interface Adapter). This I/O card makes possible overall 48 inputs and outputs and is an interface that provides data communucations between software and other electronics circuits like that stepping motor drivers and input datas belong to reference points. An opto-coupler card which isolates PC and stepping motor driver circuits follows I/O card and so the ground of PC and the ground of all the other circuits are electrically isolated from each other. After opto-coupler card there comes stepping motor driver cards, drill driver card and input circuits. Mechanical motion is produced with running stepping motors by driver circuits.

The software of system is written by using of Turbo Pascal v7.0. All the control signals is controlled by software. Stepping motors is being run by open loop control system. This control type is considered suitable as the load conditions are not too heavy. Properties of system like making a new PCB files, loading a PCB file from hard disk have been presented in comprehensible way. The four bits phase control signals of stepping motors is directly produced and transmitted by PC without using an additional circuits as the number of inputs and outputs is more than used in the system.

In the first two sections of this study, stepping motors and driver circuits and also in the third section microcontoller based stepping motor control systems are examined in details and in following sections building of PCB drilling system has been planned by using this knowledges.

1. GİRİŞ

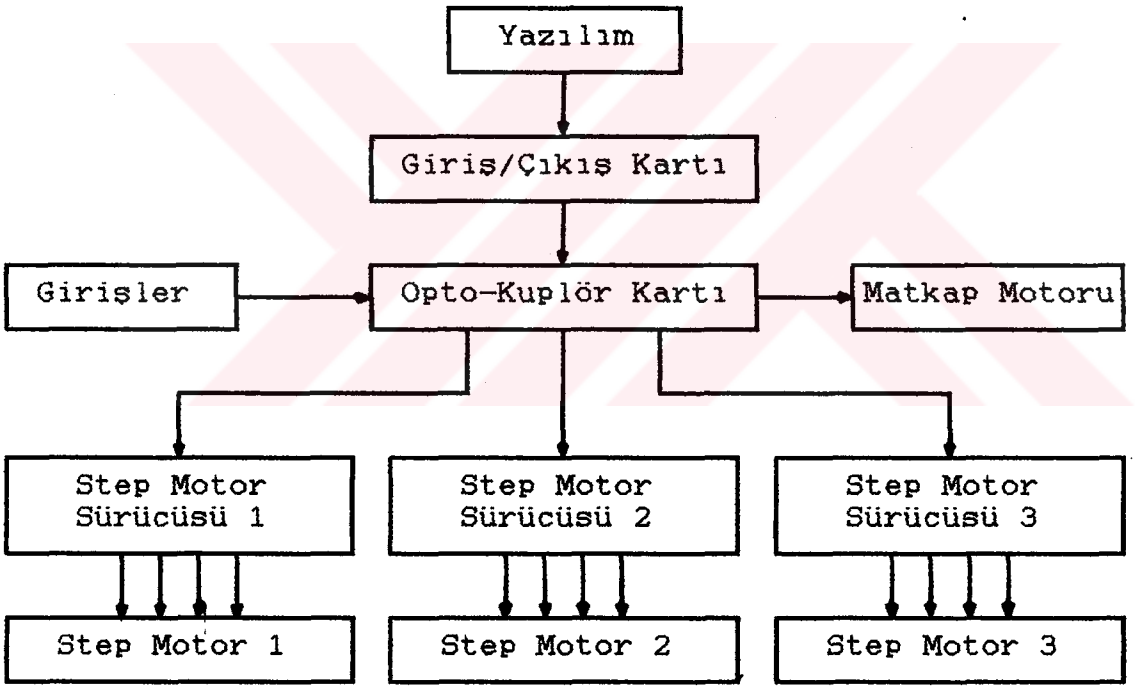
Günümüzde mekanik sistemlerin kontrolünde mikroişlemci ve bilgisayarın kullanımı oldukça önemli bir yer tutmuştur. Kontrol işaretlerinin doğru zamanlı olarak üretilmesi, dış ortam bilgilerinin toplanıp işlenmesi gibi uygulamalarda insan faktörü giderek azaltılarak hatasız sistem işletimlerine ulaşılmaya çalışılmaktadır. Bu amaca yönelik olarak bir bilgisayar, çalışan bir sistemin belli bir andaki tüm saha bilgilerini toplayıp ekranda gösterebilir ve aynı anda sistem parametrelerini yazılıma göre değiştirebilir.

Günümüz endüstrisinde "Kumanda ve Kontrol" kavramları, tek bir başlık altında toplanarak "PROSES KONTROL" adını almıştır. Proses Kontrol kısaca "Bir sistemin ürettiği çıkış bilgilerinden faydalanılarak, o sisteme girecek olan bilgiyi ayarlamak" olarak tanımlanabilir. Bu tanımdan yola çıkarak matkap tezgahını ele alalım:

Örnek olarak matkap motorunun X ekseninde 30 birimlik bir ilerleme kaydedebilmesi için bilgisayar, motora hareket verirken, aynı zamanda kaç adım attığını da kontrol etmektedir. Bu kontrolün sonucuna göre, hareket motorunu durdurur veya hareketine devam ettirir.

Kumanda edilen step motorlar özellikle hassas hareket istenilen ortamlarda yaygın olarak kullanılırlar ve sayısal bilgilerle yönetilmeye son derece uygun bir yapıya

sahiptirler. Step motorlar DC-AC motorlarda olduğu gibi sürekli bir dönme yapmak yerine ayrık açısal hareketler yapabilen bir yapiya sahiptirler. Analog gerilimler yada akımlar yerine sayısal bilgilerin oluşturduğu darbe dizileri ile yönetilirler. Sayısal bir darbe motorun belirli bir açı kadar dönmesini sağlar. Bu açı, tamamen motorun yapısıyla ilgilidir. Step motorlarla ilgili ayrıntılı bilgi, sonraki üç bölümde geniş şekilde verilmiştir.



Şekil 1.1 Baskılı devre matkap tezgahının blok diyagramı

Bu çalışmada, baski devre matkap tezgahının mekanik kısmı ve mekanikle ilgili çalışma prensiplerinden çok, mekanik aksama hareket kazandıran step motorlar, algılama

devreleri ve sürücü devreler üzerinde durulmuştur. Mekanik aksam geliştirilmeye açık olup, step motorlar yerine hidrolik veya pnömatik pistonlardan oluşan bir yapı kullanılarak daha sağlam bir yapıya ulaşılabilir.

Sistemin genel yapısı Şekil 1.1' deki gibi olup ilerleyen bölümler, bu şekil ele alınarak sondan başa doğru giden bir konu sırası halinde ele alınmıştır.

2. STEP MOTORLAR

Sayısal girişli bir eleman olan step motor, ilk olarak 1930'da denendi. 1960'lara doğru elektronik teknolojisindeki gelişmeler, yüksek güçlü anahtarlama transistörlerinin ve devrelerinin yapılabilme olanağını da beraberinde getirdi. Bu gelişmeler sonrası step motor daha etkin olarak kullanılmaya başlandı.

Step motorların ilk günlerinden bu zamana kadar çok çeşitli tipleri üretilmiş fakat görülmüştür ki belli sınırlar içinde büyük güçleri üretebilecek magnetik akının sağlanabilmesi için motor içindeki hareketli ve hareketsiz parçaların çok sayıda demir dişlere sahip olması gerekmektedir. Bu temel gereksinimi sağlayacak çeşitli metodlar ise ticari başarılara engel olmuş ve günümüzde ciddi olarak ele alınabilecek 2 tip step motor ortaya çıkmıştır:

1 - Değişken Relüktanslı Step Motor

2 - Hibrid Step Motor

Step motorda bulunması zorunlu olan özellik, anahtarlamalı uyarma bilgisinin, motorun rotor pozisyonunda konumu tam olarak belirli değişimler oluşturmasıdır. Doğru pozisyonlama stator ve rotorda bulunan demir dişlerin magnetik olarak karşı karşıya düşürülmesiyle sağlanır.

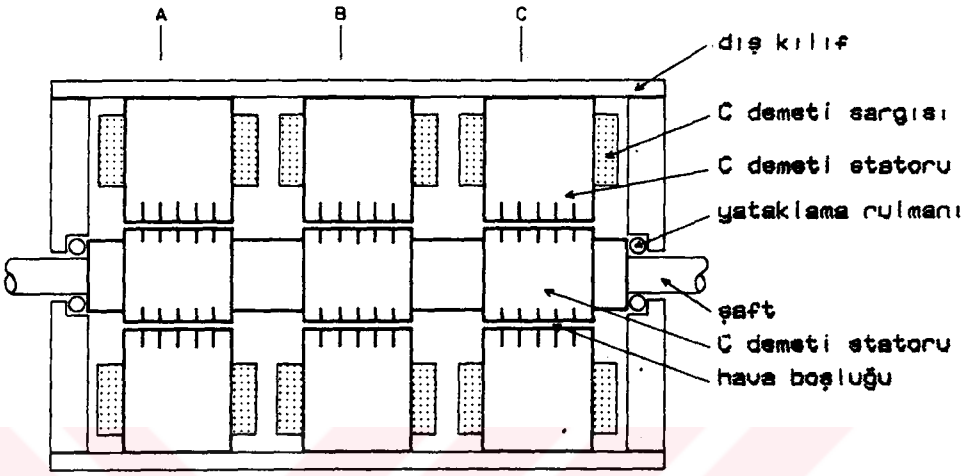
2.1. Değişken Relüktanslı Step Motor

Bu tip step motorlarda, magnetik akı yalnizca sargılardan geçen akım tarafından üretilir. Hibrid tip step motorlarda ise rotor üzerinde bulunan sürekli mıknatıs tarafından da akı üretilir. Değişken relüktanslı step motorların çok demetli ve tek demetli olmak üzere iki tipi vardır.

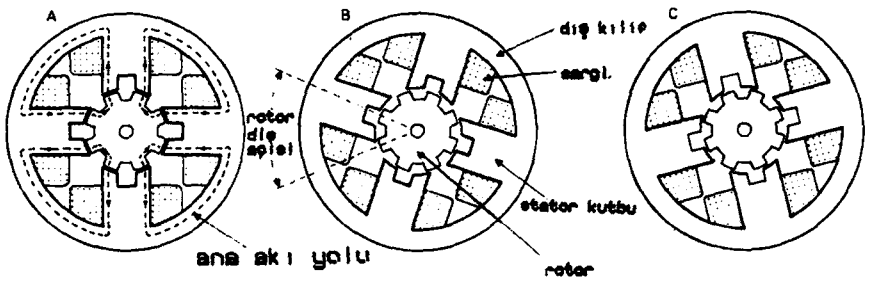
2.1.1. Çok Demetli Değişken Relüktanslı Step Motor

Çok demetli motor, eksen boyunca, birbirleri ile magnetik olarak yalıtılmıs demet olarak adlandırılan bölümlere ayrılmıştır. Her demet motorun dış kılıfına konumlandırılmış bir sabit eleman (stator) ve bir de hareketli eleman (rotor) içerir. Her demetin ayrı bir stator sargısı (faz) vardır. (Şekil 2.1.a) Stator ve rotor sıkıştırılmış demir levhalardan yapılmıştır. Bunun nedeni, motor içindeki magnetik alanın, aşırı eddy akımı kayıplarına neden olmaksızın hızla değişebilmesini sağlamaktır.

Her demetin statoru belli sayıda kutuba sahiptir. Şekil 2.1.b 'de 4 kutuplu bir örnek görülmektedir. Faz sargıları her kutuba, kutup üzerinde radyal magnetik alan oluşturacak şekilde sarılmıştır. Bitişik kutuplar üzerindeki sargılar ters yönde radyal magnetik alan oluşturabilmek için birbirlerine ters yönde sarılmışlardır. Her demet için kapalı



ekil 2.1.a 3 demetli değişken relüktanslı step motorun şafta paralel kesiti.



iekil 2.1.b 3 demetli değişken relüktanslı step motorun şafta dik kesiti.

magnetik çevrim bir stator kutbundan başlar, hava boşluğundan rotora geçer, rotor boyunca ilerler, hava boşluğundan geçerek bitişik kutba ulaşır, bu kutbu geçer ve kendi orjinal kutbunda son bulur. Her komşu kutup için bu magnetik çevrim tekrarlanır ve böylece Şekil 2.1.b' deki örnekte olduğu gibi 4 ana akı yolu oluşur.

Rotorun pozisyonu, belirli bir demete ait faz sargısı uyarıldığında tam olarak o demetin statoruna bağlıdır. Pozisyonun doğruluğu, demet üzerindeki relüktansı azaltmak için karşı karşıya dizlmeye meyilli olan, stator ve rotor üzerinde eşit sayıda bulunan dişler tarafından sağlanır. Stator ve rotor dişinin bulunduğu pozisyon, devrenin relüktansını minimumda tutacak şekilde olup, demet üzerindeki magnetik akı maksimum düzeydedir.

Şekil 2.1.b' deki motor 8 stator/rotor dişine sahıptir ve A demetinin uyarılmasıyla sağlanan pozisyondadır. Eksen boyunca bakılacak olursa rotor dişleri her demet için dizilidir. Halbuki stator dişlerinin dizilimleri her demet için farklıdır. Uyarmanın A demetinden B demetine geçirilmesiyle, B demetindeki rotor ve stator dişlerinin dizilmesi sağlanır. Bu yeni dizilim rotorun saat yönünde hareket etmesiyle mümkündür. Yani uyarmanın değişmesi sonucu motor bir adım atmıştır.

Saat yönünde bir sonrakı adım ise B demetindeki

uyarmanın C demetine geçirilmesiyle sağlanır. Sıralamanın son adımı uyarmanın yeniden A demetine uygulanmasıdır. Bu durumda A demetindeki stator ve rotor dişleri yeniden tam olarak dizilmiştir. Saat yönünde sürekli bir dönüş, uyarmanın A,B,C,A,B,C,A... şeklinde yapılmasıyla sağlanır. Saat yönünün tersine dönüş ise uyarma sırası ters çevrilerek sağlanır. Çok demetli motorda iki yönlü dönme işlemi isteniyorsa, 2 farklı uyarma sıralamasının yapılabilmesi için, motor en az 3 demetli olmalıdır.

Çok demetli değişken relüktanslı motor için stator/rotor dişlerinin sayısıyla adım açısı ve demet sayısı arasında basit bir bağıntı vardır. Motorun N demeti varsa (N faz), temel uyarma çevrimi her demetin sırasıyla uyarılmasıdır. Bu durumda motor toplam N adım atmıştır. Bir sargı, uyarma çevriminin başında ve sonunda uyarılır ve o demete ait stator ve rotor dişleri karşılıklı dizilirse, rotor 1 diş aralığı kadar hareket etmiştir. 1 diş aralığı 360/p derece olduğundan uyarmanın bir değişimiyle

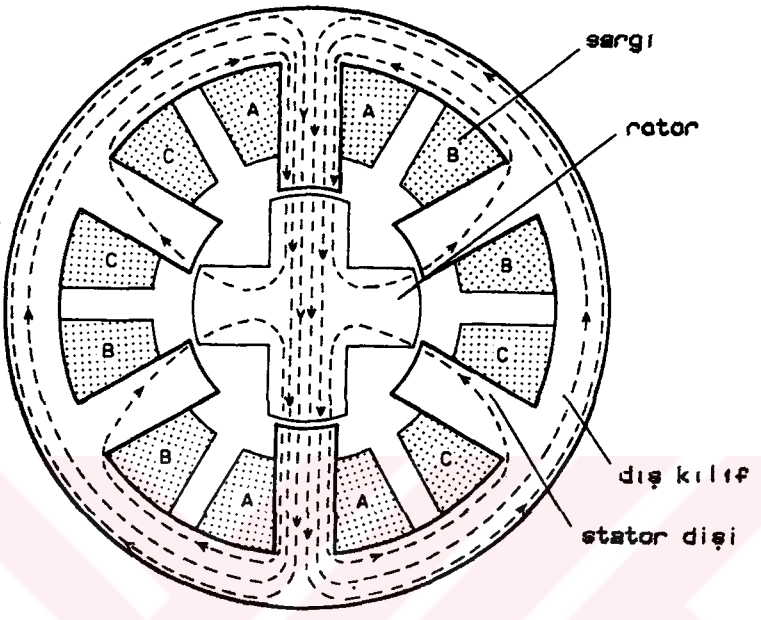
$$\text{Adım Açısı} = 360 / N.p \quad \text{derece} \qquad (2.1)$$

olarak bulunur. Burada p, rotor diş sayısıdır. Şekil 2.3'deki motorun 3 demeti ve 8 rotor dişi olduğundan adım açısı 15 derecedir. Çok demetli değişken relüktanslı step motorlar için adım açısı 2 - 15 derece arasında değişmektedir.

2.1.2 Tek Demetli Değişken Relüktanslı Step Motor

İsminden de anlaşılacağı gibi, bu tip motor tek bir birimden yapılmıştır ve bu nedenle şafta paralel olarak alınan kesiti, Şekil 2.1' deki motorun tek bir demetine benzer. Şafta dik kesiti gösteren Şekil 2.2, tek ve çok demetli motorlar arasındaki belirgin farklılıkları göstermektedir.

Stator yapısına bakıldığında, stator dişlerinin stator/rotor hava boşluğundan dış kılıfa doğru olduğu görülür. Her diş, d.c. akımla uyarıldığında radyal magnetik alan üreten, kendine ait sargıya sahiptir. Şekil 2.2' deki motorun 6 stator dişi vardır ve karşı dişteki sargılar birlikte bağlanarak bir sargı oluşturulmuştur. Bu nedenle şekildeki motorda 3 faz vardır ki bu da iki yönde dönüş sağlamak için gerekli minimum sayıdır. Karşı stator dişindeki sarımlar zıt yönde sarılmışlardır. Çünkü bir dişteki magnetik alan hava boşluğuna doğru iken, diğer dişteki alanın hava boşluğundan dışarı doğru olması gerekmektedir. Bir faz uyarımı geldiğinde ana akı yolu bir stator dişinden çikar, hava boşluğundan rotor dişine geçer, rotoru geçip dış kılıf üzerinden devresini tamamlar. Şekil 2.2' de görüldüğü gibi akının belli bir miktarı, uyarılmamış stator dişinden sızıntı şeklinde geçer. Bu ikinci akı yolu, tek demetli motorun faz sargıları arasında karşılıklı kuplaj üretir.



Şekil 2.2 Tek Demetli Değişken Relüktanslı Step Motorun Şafta Dik Kesiti (Akı yolu A fazının uyarıldığı durum için çizilmiştir.)

Rotorun en belirgin özelliği, diş sayısının stator diş sayısından farklı olmasıdır. Şekil 2.2.' deki motorun 4 dişi vardır. Bir fazın uyarılmasıyla, ana akı, iki rotor dişi üzerinden taşınır. Rotor dişlerinin kalan çifti uyarılmamış stator dişlerine komşu durumda bulunur. Faz uyarması değiştiğinde bu 2 rotor dişi, yeni uyarılan stator dişleriyle karşı karşıya gelecektir. Şekil 2.2, A fazının uyarıldığı durumdaki rotor pozisyonunu göstermektedir. Rotor ana akı

relüktansını minimize edecek şekilde yerini almıştır. Bu anda uyarma B fazına geçirilirse motor, saat yönünün tersi yönde bir adım atacak şekilde döner ve diğer rotor dişleri, B fazına ait stator dişleriyle karşı karşıya gelir. C fazının uyarılması ise diğer bir adıma neden olur. Böylece saat yönünün tersine sürekli bir dönüş için A,B,C,A,B,C,A... şeklinde bir uyarma sıralamasının yapılması gerekir. Saat yönünde sürekli bir dönüş için ise sıralamanın A,C,B,A,C,B,A,... şeklinde olması gerekir. ılginç olan bir nokta da, rotor hareketi stator magnetik alanının değişim yönüne ters yöndedir.

Adım açısı, basit bir ifadeyle, faz ve rotor diş sayılarıyla ifade edilebilir. p rotor dişli bir motor için, 1 diş aralığı 360/p derece olup N adım için:

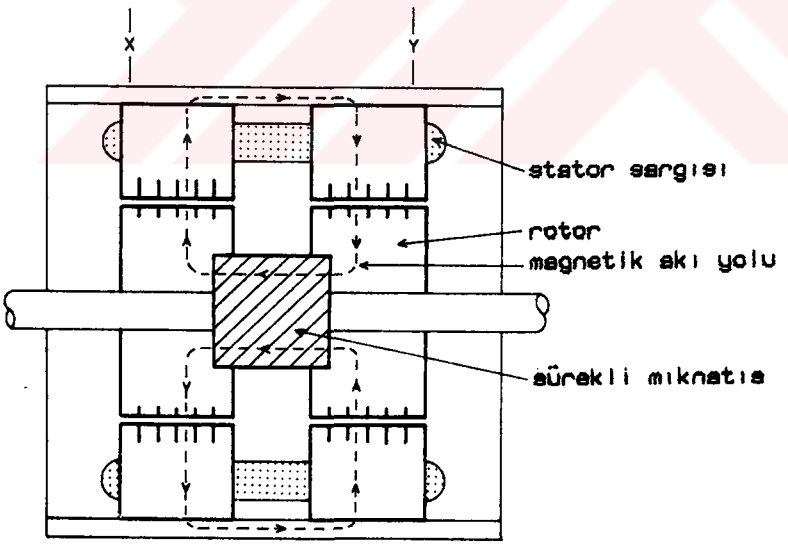
$$\text{Adım açısı} = 360 / N.p \text{ derecedir.} \qquad (2.2)$$

Şekil 2.2' deki motorun 3 fazı ve 4 rotor dişi olduğundan adım açısı 30 derecedir.

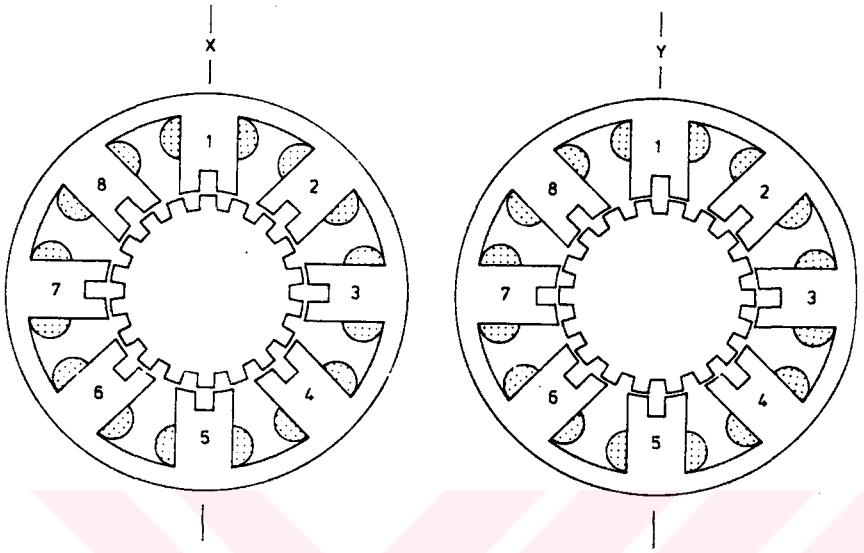
Stator dişlerinin sayısı, faz sayısi ve rotor dişlerinin sayısıyla sınırlanmıştır. Her faz birkaç stator dişi üzerine dağılmıştır ve akıyı, kendi üzerinden rotora doğru çevirecek sayıda stator dişi olmalıdır. Stator dişlerinin sayısı, faz sayısının tam katı şeklindedir. 3 fazlı bir motor için bu sayı, 6,12,18,24 olabilir.

2.2. Hibrid Step Motor

Hibrid step motorda, rotor üzerinde sabit, sürekli bir mıknatıs vardır. Şekil 2.3.a' da görüldüğü gibi, ana akı yolu mıknatısın N kutbundan başlar, radyal olarak rotoru geçer, hava boşluğunu aşar, X bölümünün stator kutuplarını geçer, stator metalını aşar ve Y bölümüne ait stator kutuplarını geçtikten sonra hava boşluğunu da aşarak sürekli mıknatısın S kutbuna ulaşır.



Şekil 2.3.a Hibrid Motorun Şafta Paralel Kesiti



Şekil 2.3.b Hibrid Motorun Şafta Dik Kesiti

Şekil 2.3.b' de 8 stator kutbu vardır ve her kutbun 2 ile 6 arasında dişi vardır. Ayrıca stator kutuplarında, istenen rotor pozisyonuna bağlı olarak sürekli mıknatıs akısını bazı kutularda artırmada ya da azaltmada kullanılan sargılar vardır. Motor üzerinde 2 temel sargı vardır ve her sargı (faz), 8 stator kutbunun 4 tanesi üzerine yerleşmiştir. A sargısı 1,3,5,7 ve B sargısı 2,4,6,8 nolu kutuplar üzerindedir. Her fazın ardışık kutupları zıt yönde sarılmışlardır. Örnek olarak, A sargısı pozitif yönde bir akımla uyarıldığında, sonuç magnetik alan, radyal olarak 3 ve 7 nolu kutuplardan dışarı, 1 ve 5 nolu kutuplardan içeri doğru yönlendirilir. Benzer yol B fazı içinde kullanılır. Tüm

olası durumlar Tablo 2.1' den görülebilir.

Tablo 2.1. Sargılardan geçen akım yönüne bağlı olarak stator kutuplarındaki magnetik alanın yönü

Sargı	Akım Yönü	Magnetik Alanın Yönü	
		Dışa Doğru	İçe Doğru
A	Pozitif	3,7	1,5
A	Negatif	1,5	3,7
B	Pozitif	4,8	2,6
B	Negatif	2,6	4,8

Sargı uyarımlarının, magnetik akı yoluna etkisini A sargısının pozitif yönlü akımla uyarıldığını göz önüne alarak inceleyelim. X bölümündeki sürekli mıknatıs akısı radyal olarak dışa doğrudur ve A sargısının uyarılmasıyla magnetik akının büyük bir bölümü 3 ve 7 nolu kutuplardan akar. Bununla birlikte Y bölümünde bu olayın tersi olur. Magnetik akı, radyal olarak içeri doğru olacağından, 1 ve 5 nolu kutuplarda yoğunlaşır.

Şekil 2.3.b' deki 8 kutbun her birinde 2 olmak üzere, 16 kutbu vardır. Rotordaki diş sayısı ise 18'dir. X ve Y bölümlerinin stator dişlerinin konumları birbirleriyle aynıdır. Halbuki rotor dişleri iki bölümde de farklı dizilmişlerdir. Magnetik akı, sargıların uyarılmasıyla belli kutuplarda yoğunlaşırsa, rotor dişleri kendini hava boşluğundaki akı yolunu minimize etmek için stator dişleri karşısında dizilmeye zorlar. A sargısının pozitif yönde

akımla uyarılması örneğinde, stator ve rotor dişleri X bölümünde 3 ve 7, Y bölümünde ise 1 ve 5 nolu kutuplarda dizilmişlerdir.(Şekil 2.3.b)

Motorun sürekli hareketi, faz sargılarının sıralı bir şekilde uyarılmasıyla sağlanır. A uyarması kaldırılıp B sargısı pozitif akımla uyarılırsa stator ve rotor dişleri, X bölümünde 3 ve 7, Y bölümünde ise 1 ve 5 nolu kutuplar üzerinde dizilir. Rotor, doğru pozisyona ulaşmak için saat yönünde bir adım atmıştır. Saat yönünde dönüşe A fazının, ardından da B fazının negatif akımlarla uyarılmasıyla devam edilir. Bu sıra A+,B+,A−,B−,A+,B+,... şeklindedir. Ters yönde dönüş bilgisi ise A+,B−,A−,B+,A+,B−,... şeklindedir.

Her adımın açısı, rotor diş sayısı p ile basitçe ifade edilebilir. Hibrid step motorlar için, uyarma bilgisinin bir tam çevrimi 4 durum içerir ve 4 adımlık rotor hareketi oluşturur. Bu 4 adımdan önceki ve sonraki adım sıraları aynıdır. Bu nedenle her 4 adımda bir, stator ve rotor dişleri aynı stator kutupları üzerine denk gelir. Böylece bir diş aralığı 360/p derece olup hibrid motor için:

$$\text{Adım açısı} = 90/p \text{ derecedir.} \qquad (2.3)$$

Şekil 2.3' deki motorun 18 rotor dişi vardır ve adım açısı 5 derecedir. Hibrid motorlar genellikle bu değerden daha küçük adım açılarında üretilir.

2.3. Motor Tiplerinin Karşılaştırılması

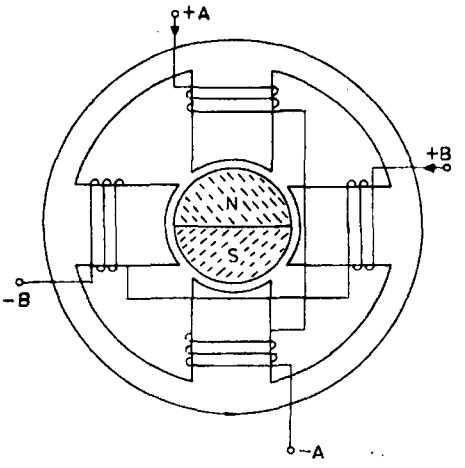
Hibrid motorlar, yüksek hassasiyetli pozisyon konumlandırmada büyük avantaj sağlayan düşük adım açılarına sahiptir. (Tipik değer 1.8 derece). Hibrid motorlarda tork üretme kapasitesi, değişken relüktanslı motorlara göre daha büyüktür. Bu nedenle hibrid motor, sınırlı çalışma sahalarında, küçük adım açısı ve yüksek tork istenen uygulamalarda doğal bir seçim olmaktadır. Hibrid motor sargıları üzerinden uyarma akımı kaldırıldığında sürekli mıknatıs, rotoru o anki adım pozisyonunda tutan, küçük bir kontrol torku sağlar. Kontrol torku, motorun bir veya iki sargısının birden uyarılmasıyla sağlanan motor torkundan daha küçüktür ve enerji kesilmesi durumunda pozisyonun korunması gereken uygulamalarda yararlı bir özelliktir.

Değişken relüktanslı motorlar, yükün önemli mesafelere taşınması sözkonusu olduğunda 2 önemli avantaja sahiptirler. İlk olarak adım açıları, hibrid motorlara göre daha küçüktür. (Tipik değer 15 derece). Bu nedenle daha az adımda istenilen mesafeye varırlar. Adım sayısının azalması, uyarmanın daha az sayıda olmasını, bu da istenen mesafeye ulaşmak için gereken zamanı belirleyen hızın artmasını sağlar. Bir diğer avantaj ise değişken relüktanslı motorun rotorunun ataleti, hibrid motora göre daha azdır. Çünkü rotoru üzerinde sürekli mıknatıs yoktur. Birçok durumda rotor ataleti, toplam yük ataletinin önemli bir oranını oluşturur ve bu ataletin

azalması daha çabuk hızlanmaya imkan verir.

Bu bölümde anlatılan 2 temel step motordan başka, adım hareketi yapan çeşitli elemanlar vardır. Bunlardan biri de sürekli mıknatıslı step motordur. Bu tip motor, tek demetli değişken relüktanslı step motora benzer yapıdadır, fakat rotorunda dişler yoktur. Sürekli mıknatıs özelliği taşıyan bir malzemeden yapılmıştır. Şekil 2.4' deki örnekte rotorun, sargı uyarmalarına bağlı olarak 2 stator dişinin karşısında dizilecek şekilde 2 magnetik kutbu vardır. 2 sargı arasında uyarmada yapılacak bir değişiklik 90 derecelik bir adım üretir. Akımın polaritesi bu tip motorda önemlidir. Şekildeki rotor pozisyonu A sargısının pozitif uyarımı içindir. Uyarmanın B sargısına pozitif yönde bir akım verecek şekilde değiştirilmesi saat yönünde bir adım sağlar. Halbuki negatif akım ters yönlü bir adıma neden olur. Çok sayıda kutbu olan küçük ebatlı sürekli mıknatıs yapmak zor olduğundan bu tip motorlar 30 - 90 derece arasındaki adım açılarıyla sınırlıdır.

Farklı bir tip ise, çok yüksek tork gerektiren uygulamalarda kullanılan elektrohidrolik step motorlardır. Temel olarak motor, giriş işaretini bilinen step motorlardan alan kapalı çevrim hidrolik kontrol sistemidir. Birkaç yüz mertebesinde tork kazancı vardır ve birçok makinede kullanılmıştır. (Acarnley, 1984)



Şekil 2.4 Sürekli Mıknatıslı Step Motor

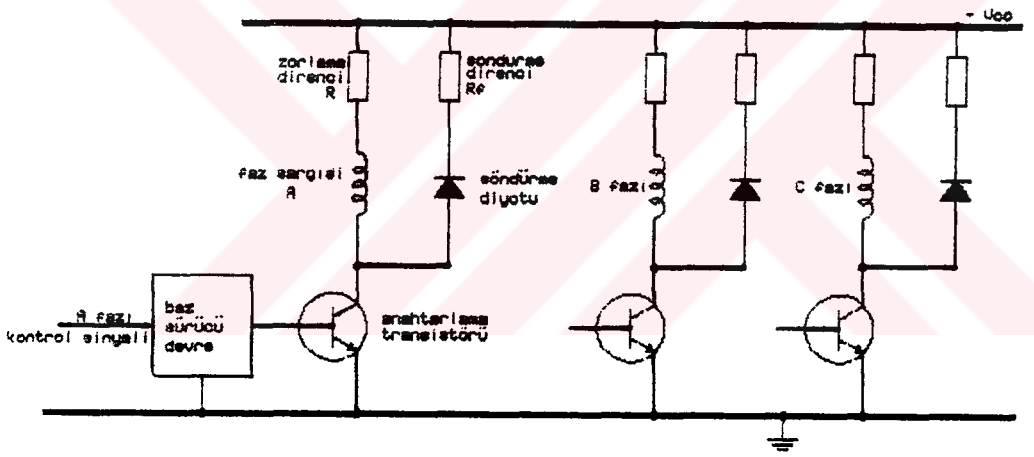
3. STEP MOTOR SÜRÜCÜ DEVRELERİ

Step motorlar için kullanılan kontrol işaretleri düşük güçlüdür ve TTL entegre devrelerle gerçekleştirilebilirler. Halbuki 1.2 Nm' lik tipik bir değişken relüktanslı step motor, 5V,3A gibi bir güce gereksinim duyar. Bu nedenle sürücü devre ile motor arasına bir anahtarlama kuvvetlendiricisi konulmalıdır. Birçok üretici, motorlarına uygun sürücü devreler de sunmaktadır. Fakat piyasada oldukça geniş bir sürücü yelpazesi vardır. Bu bölümde temel sürücü devreleri ele alınacaktır.

Değişken relüktanslı step motorun en azından 3 fazı vardır. Fakat faz akımlarının sadece on veya off olması gerekmektedir. Akım yönü tork üretiminde rol oynamaz. Bölüm 3.1' de ele alınacak olan tek kutuplu sürücü devre, bu tip motor için yeterlidir. Hibrid motorda veya sürekli mıknatıs içeren herhangi bir motorda sadece 2 faz vardır fakat akım yönü önemlidir. Bu nedenle çift kutuplu sürücü devre faz akımlarını iki yönde de akıtabilmek için gereklidir. Bölüm 3.2' de tanıtılacak olan transistörlü köprü sürücü devresi, tek kutuplu sürücüye göre daha fazla yarı iletken eleman içerir. Fakat hibrid motora eklenen ilave sargılar motorun sürülmesini basitleştirir. Bu teknik Bölüm 3.3' de anlatılacaktır.

3.1 Tek Kutuplu Sürücü Devre

3 fazlı değişken relüktanslı step motor için uygun, tek kutuplu basit bir sürücü devre Şekil 3.1' de görülmektedir. Her faz sargısı, düşük güçlü bir faz kontrol sinyali tarafından kontrol edilen, birbirinden bağımsız sürücü devreler tarafından uyarılır. Bu kontrol, faz transistörü için gerekli baz akımını sağlamak için birkaç anahtarlama kuvvetlendiricisi katı gerektirebilir.



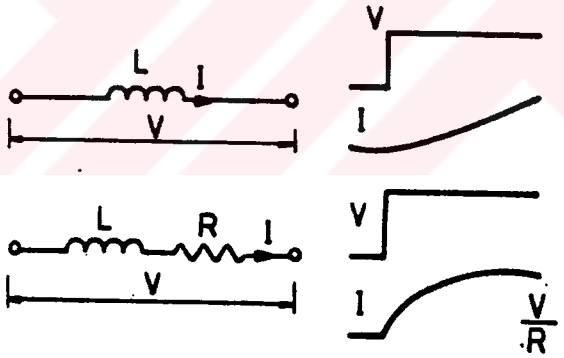
Şekil 3.1 3 fazlı tek kutuplu sürücü devre

Faz sargısı, ait olduğu anahtarlama transistörünün, yeterli baz akımıyla doyuma sokulmasıyla uyarılır. Bu durumda bütün kaynak gerilimi, faz sargısı ve zorlama direnci (forcing resistance) üzerine düşecektir. Kaynak gerilim (Vs), faz sargı direnci (r) ve zorlama direnci (R)' nin toplamından oluşan direnç üzerinde, istenen sargı akımını oluşturacak

şekilde seçilmelidir.(3.1)

$$Vs = I.(r + R) \qquad (3.1)$$

Genelde faz sargısının önemli büyüklükte bir endüktansı vardır. Bu nedenle zaman sabiti (endüktans/rezistans) uzundur. Faz akımının istenen değere ulaşması, yüksek hız bölgelerinde oldukça yavaş kalır. Zorlama direncinin ilavesiyle ve kaynak geriliminde buna göre artırılmasıyla, fazın zaman sabiti daha geniş hız bölgelerinde çalışmaya imkan verecek şekilde azaltılabilir. Zorlama direncinin faz akımına etkisi Şekil 3.2' de gösterilmiştir.



Şekil 3.2 Zorlama direncinin faz akımına etkisi

Önemli bir diğer nokta da; faz akımı, faz sargısının sonlu endüktansından dolayı ani olarak söndürülemez. Anahtarlama transistörünün baz akımı ani olarak kesilirse, transistörün kollektör-emiter uçları arasında sürücü devreye

önemli hasarlar verecek yükseklikte bir gerilim endüklenir. Bu olasılık, faz akımına alternatif bir yol sağlanarak engellenir. (Söndürme Devresi). Anahtarlama transistörü kesime gittiğinde faz akımı, söndürme diyotu (freewheeling diode) ve söndürme direnci (freewheeling resistance) tarafından oluşturulan yol üzerinden akmaya devam edebilir. Faz akımı istenen değerine oturduğunda, anahtarlama transistörü üzerindeki maksimum gerilim (Vce max), transistör kesime gittiğinde bir an ortaya çıkar. I akımı henüz düşmeye başlamamıştır ve söndürme direnci (Rf) üzerinden akar. Bu nedenle maksimum kollektör-emetör gerilimi, söndürme diyotu üzerine düşen gerilim ihmal edilerek:

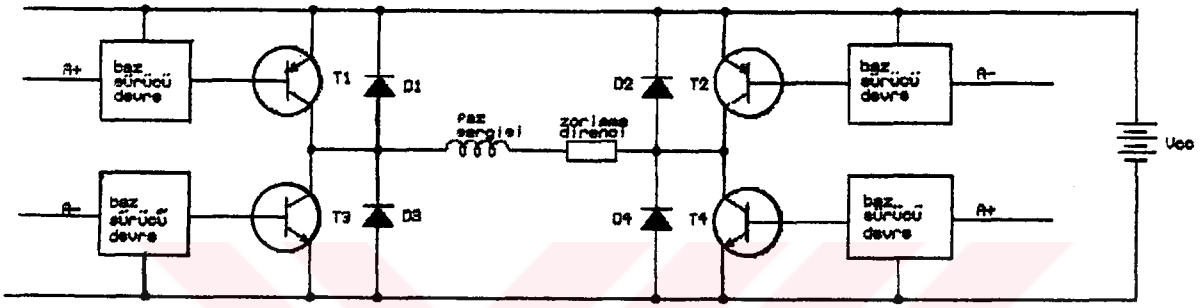
$$Vce\ max = Vs + Rf.I \text{ olur.} \qquad (3.2)$$

Böylece faz akımı söndürme devresi sayesinde hızla azalır ve kesim anında, faz endüktansı üzerinde depolanmış magnetik enerji, söndürme devresi direnci (sargı + zorlama + söndürme dirençleri) tarafndan dağıtılır. (Çölkesen, 1990)

3.2. Çift Kutuplu Sürücü Devre

Hibrid veya sürekli mıknatıslı step motorlar için uygun olan transistörlü köprü devresinin bir fazı için gerekli olan donanım, Şekil 3.3' de görülmektedir. İstenen akım yönüne bağlı olarak transistörler ikişer ikişer anahtarlanırlar. Faz sargısının pozitif uyarılmasında T1 ve T2 transistörleri doyumdadır. Böylece akım yolu, kaynaktan T1'e geçip faz

sargısını aştıktan sonra zorlama direncini de geçerek T4 üzerinden kaynağa ulaşır. Tersi durumda, T2 ve T3 transistörleri doyuma geçerek faz sargısındaki akım ters çevrilir.



Şekil 3.3 Çift kutuplu transistör köprü devresinin bir fazı

Köprü devresindeki 4 ana transistörün, 2 faz kontrol sinyalini (pozitif ve negatif) kuvvetlendirmek için ayrı baz sürücüleri ile sürülmeleri gerekir. Baz sürücüleri değişik besleme gerilimleri ile beslendiğinden, faz kontrol sinyalleri sürücülere optik izolasyon katıyla iletilirler.

Anahtarlama transistörlerine ters paralel bağlanmış 4 diyot köprüsü, söndürme akımı yolunu oluştururlar. Şekil 3.3' deki gösterimde söndürme akımı yolu T1 ve T4 kesime gittiği anda D2 ve D3 diyotları üzerinden oluşmuştur. Söndürme yolu kaynağı da içerdiğinden, kesim anında faz sargısı endüktansında depolanan enerjinin bir kısmı kaynağa geri

döner. Bu nedenle sistem verimi açısından, tek kutulu sürmeye göre çift kutuplu sürmenin avantajı vardır. Bu yüzden birçok büyük güçlü step motor (>1kW), değişken relüktanslı tipler de dahil olmak üzere, çiftkutuplu sürücülerle çalıştırılırlar.

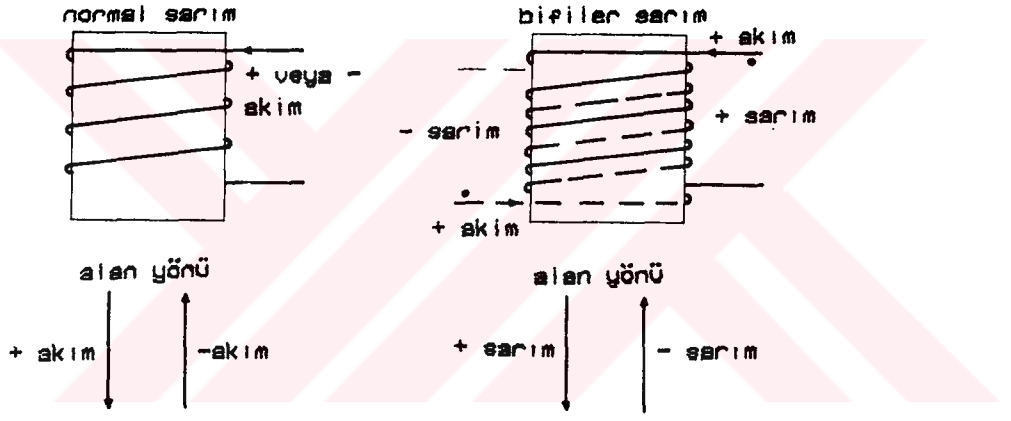
Söndürme akımı tekkutuplu sürücülere göre, kaynağa ters olduğundan daha hızlı bir şekilde eritilir. Bu nedenle çiftkutuplu sürücüde ilave söndürme direncine gerek yoktur.

3.3 Bifiler Sarım

Transistörlü çift kutuplu köprü devresi, her faz için 4 transistör/diyot çifti gerektirmektedir. Halbuki basit, tek kutuplu devrede her faz için 1 çift eleman kullanılır. Bu durum hibrid step motor sürücülerinin, değişken relüktanslı motorlara göre oldukça yüksek maliyet getirmelerine neden olmaktadır. Köprü düzeneğinin ek bir zorluğu baz sürücü devrelerinin kontrol işaretlerinin izole olması gereksinimidir. Ticari hibrid motorların getirdiği bu dezavantajlar nedeniyle üreticiler tekkutuplu sürücü devereleriyle çalıştırılabilen bifiler sarımlı hibrid motorları ürettiler.

Hibrid motor sargılarındaki iki yönlü akım akışı, stator kutuplarında iki yönlü alan üretir. Bifiler sarımla aynı sonuç, Şekil 3.4' de bir kutup için görüldüğü gibi, 2 kutup sargısının zıt yönde sarılmasıyla elde edilir. Alan yönüne

bağlı olarak sarımlardan biri, tek yönlü bir akımla uyarılır. Şekil 3.4' de normal sarım biçiminin pozitif akımla uyarılması sonucu elde edilen etki, bifiler sarım şeklinde pozitif sarımın pozitif akımla uyarılmasıyla elde edilir. Normal sarımın negatif akımla uyarılması sonucu elde edilen etki ise bifiler sarımda negatif sarımın pozitif akımla uyarılmasıyla elde edilir.



Şekil 3.4 Normal ve bifiler sarım biçiminin karşılaştırılması

Bifiler kutup sarımlarının her biri normal sarım değeri kadar sarılmalıdır ve aynı değerde akımla uyarılmalıdırlar. Bu fazlalık tabii ki maliyet artışına neden olur. Fakat sürücü devredeki basitleşmenin getirdiği kazanç bunu fazlasıyla telafi eder. 2 bifiler sarımlı tek bir faz, Bölüm 3.1' de tartışılan ayrık, tekkutuplu sürücülerle

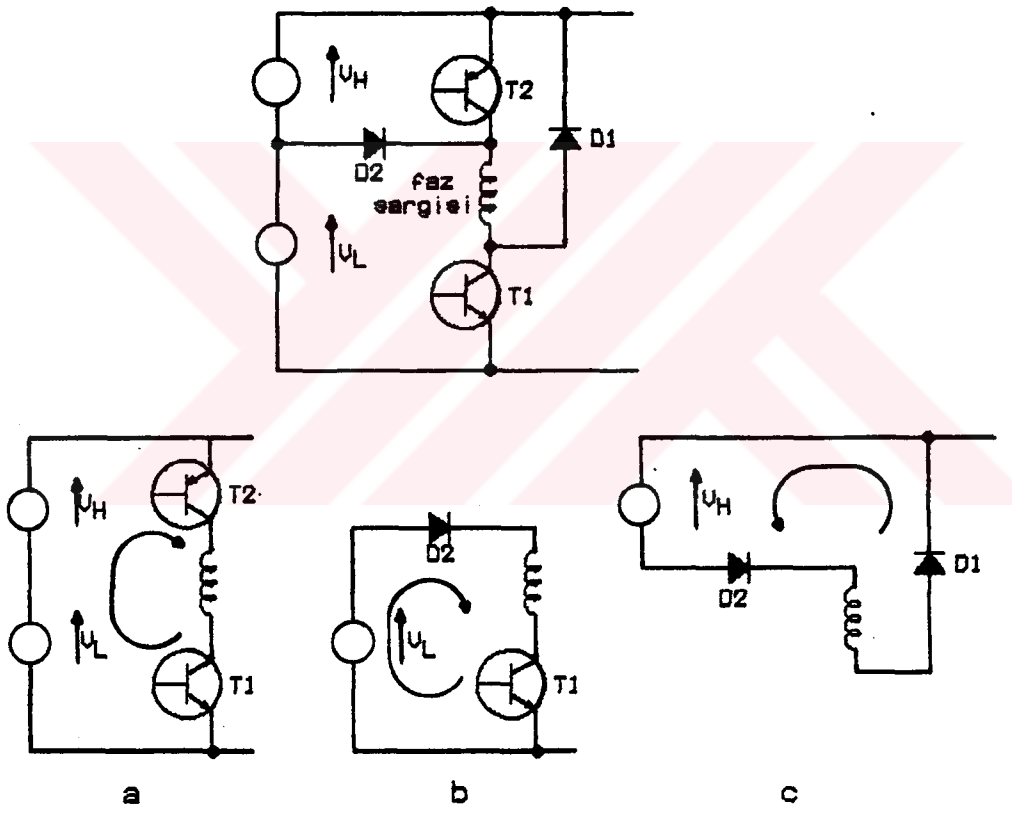
uyarılabilir. Bir diğer alternatif ise Şekil 3.5' de görüldüğü gibi zorlama direncinin paylaşılmasıdır. Bu durumda her fazda sadece 2 transistör/diyot çifti bulunur. Böylece bifiler sarımlı, 2 fazlı hibrid step motor sürücü maliyeti, 3 fazlı, değişken relüktanslı step motor sürücüsününkine yakın seviyeye iner. Bifiler sürücünün söndürme yolu, kesim anında endüktansta biriken enerjiyi kaynağa geri göndermediğinden bu tip sürücünün verimi bilinen çift kutuplu köprü tipi sürücüye göre daha düşüktür. Verimdeki bu düşüş, extra sargı maliyetiyle birlikte büyük ebattaki step motorlar için çok önemlidir ve bu tip motorlar çok nadiren bifiler sarım tekniğiyle yapılır. (Acarnley, 1984)

Önceki bölümlerde anlatılan sürücü devreler bazı uygulama alanları için çözüm olabilir. Bunların olumsuz yanlarını içermeyen birçok sürücü devre tekniği vardır. Doğal olarak, bu devreler daha karmaşık yapıdadır. Yüksek hızlarda, erişim ve sönüm zamanı küçük olan sürücü devrelerin en çok bilinen iki tanesi, ikiseviyeli sürücü (bilevel drive) ve kıyımlı sürücü (chopper drive) dır.

3.4. İki Seviyeli Sürme

Bu yöntemde iki besleme kaynağı vardır. örnek bir devre Şekil 3.5' deki gibidir. Motor ilk hareket edeceği zaman T1 ve T2 transistörleri doyumdadır. Diğer bir deyişle faz sargısına düşen gerilim, iki kaynağın gerilimi toplamına

eşittir. Sargıdaki akım yeterli seviyeye ulaştığında T2 transistörü kesime gider. Böylece motor sürekli durumda V1 kaynağını kullanmış olur. Hareket bitiminde, T1 ve T2 transistörleri kesime sokulur. Böylece, sargıda biriken enerji, sargı, D1 ve D2 diyotu ve V2 kaynağı üzerinden boşalır. V2 kaynağının kutuplanması sönüm zamanını küçültecek biçimdedir.



Şekil 3.5 İki seviyeli sürücü devre
a) iletim anı
b) sürekli uyarma
c) kesim anı

Akım dalga biçimindeki erişim ve sönüm zamanları (t1 ve t2) aşağıdaki biçimde bulunur.

$$t1=[V1/(V2+V1)].(L/R) \quad s \qquad (3.3)$$

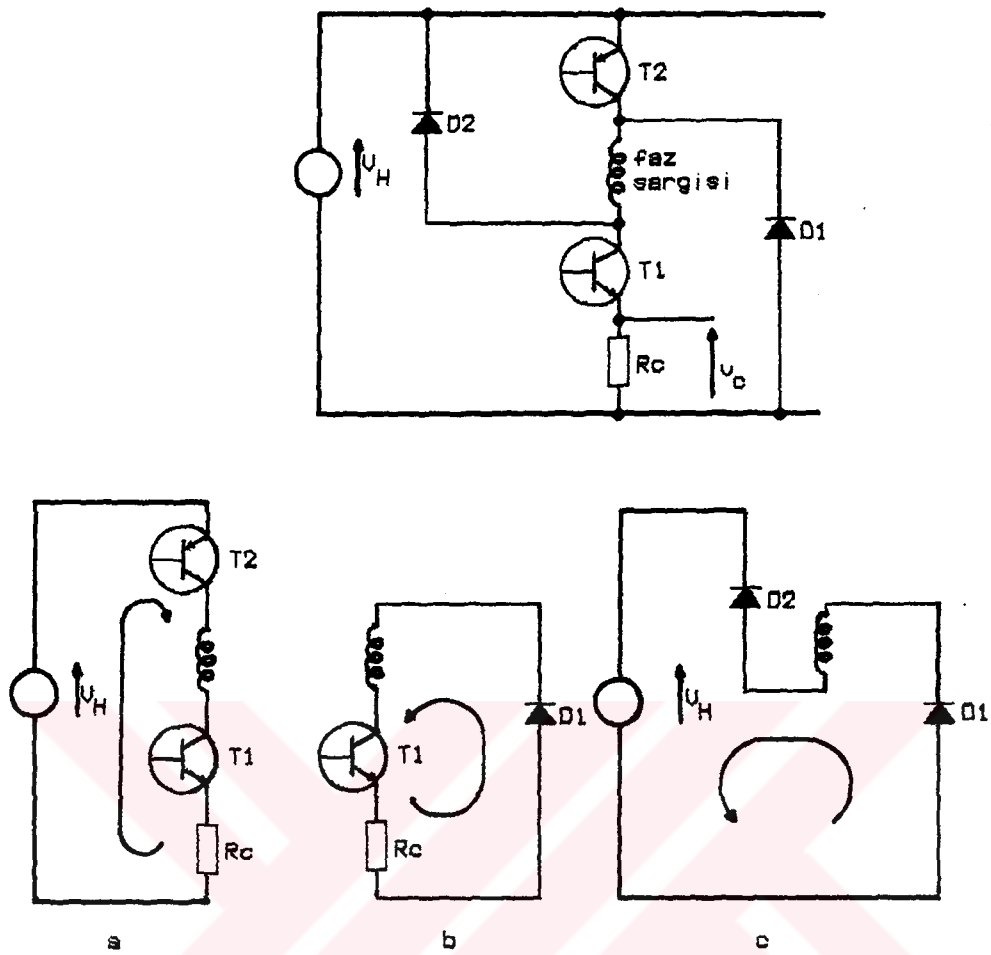
$$t2=(L/R)/[(V2/V1)+1] \quad s \qquad (3.4)$$

V2, V1' den büyük seçilirse, t1 erişim zamanı ve t2 sönüm zamanının küçüleceği görülür.

3.5 Kıyımlı Sürücü

Tek kaynaklı, sıradan step motor sürücü devrelerinde, faz sargısına, güç kaybına neden olan zorlama direnci üzerinden gerilim uygulanır. Kontrol devresi istenen hareketi yapacak biçimde, fazdan faza gerilimi anahtarlar. Faz sargısına zorlama direnci üzerinden gerilim verilirken $I^2.R$' lik bir güç kaybı olmaktadır. Bu güç kaybının sıradan yöntemler ile giderilmesi olanaksızdır. Bunu giderecek yöntem kıyımlı çalışmadır. Kıyımlı çalışma, güç kaybını önlediği gibi yüksek hızlarda çalışmaya da olanak vermektedir. İki seviyeli sürmedeki iki kaynak gereksinimi de yoktur. Tek kaynak kullanılır. Kaynağın gerilimi yüksek akım oluşacak şekilde büyük olmalıdır.

Şekil 3.6' daki kıyımlı sürücü devrede tek kaynak, iki transistör ve iki diyot temel ögedir. Buradaki amaç, sargıya sabit bir akım yerine belirli aralıklarla değişen bir akım vermektir. Şekil 3.7' de bu akımın dalga şekli görülmektedir. Şekil 3.6' daki devrenin çalışması şu şekildedir. Akım

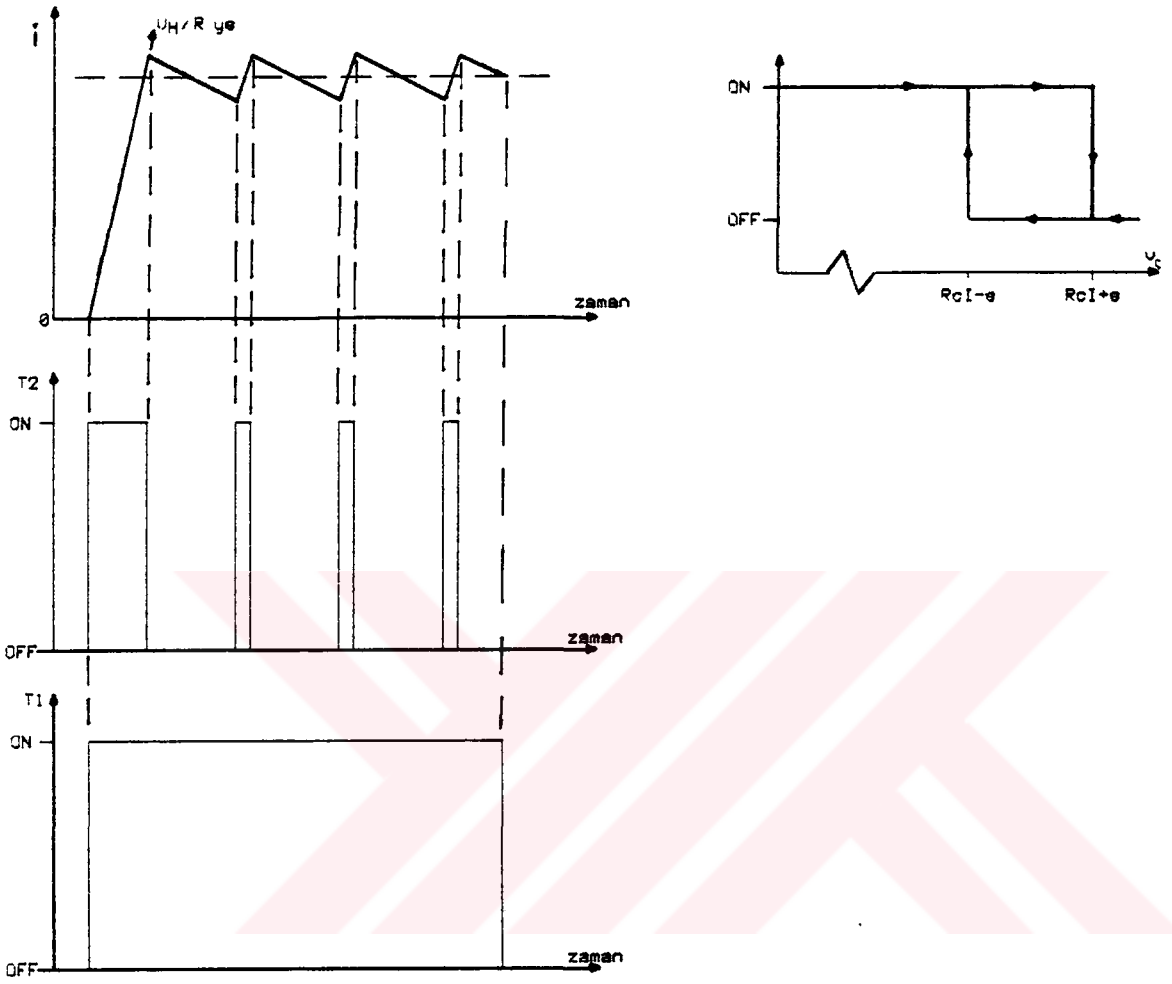


Şekil 3.6 Kıyımlı sürücü devre
a) akım sınır değerden az
b) akım sınır değerden çok
c) kesim anı

istenen aralıktan küçük olduğunda T1 ve T2 transistörleri doyumda ve böylece sargı akımı artmaktadır. Büyük olduğunda T2 transistörü kesimdedir ve T1 transistörü doyumdadır. Hareketin bitirilmesi T1 ve T2 transistörlerinin birlikte kesime sokulmasıyla gerçekleşir.

Bu sürücünün olumlu yanı, kaynağı en etkin biçimde kullanmasıdır. En yüksek hıza çıkma olanağı verir ve iyi

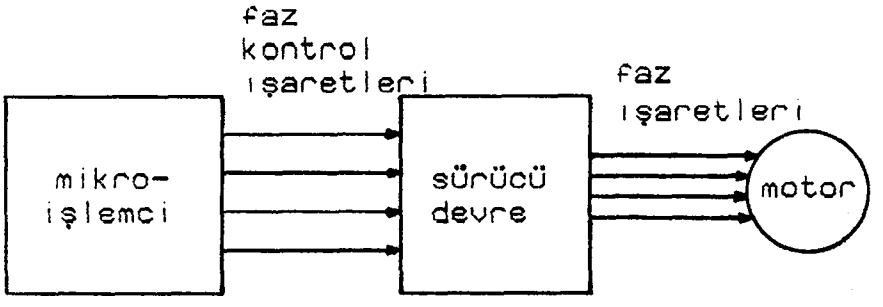
verim sağlar.



Şekil 3.7 Kıyımlı akımın dalga biçimi

4. MİKROİŞLEMCİ TEMELLİ STEP MOTOR KONTROL SİSTEMLERİ

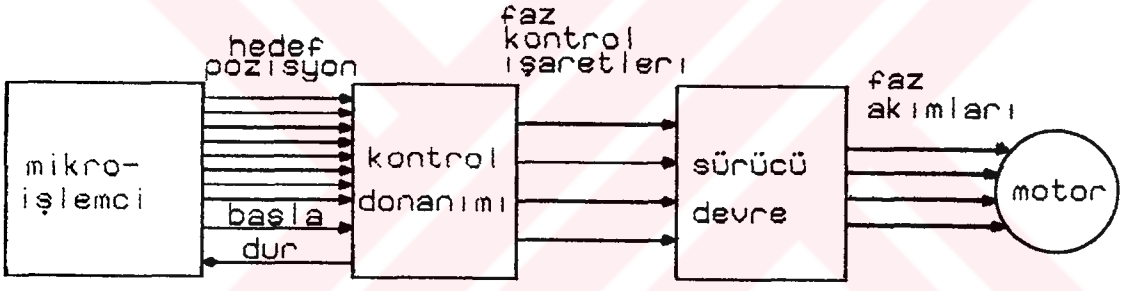
Step motorlar mikroişlemci temelli kontrol sistemlerinde çıkış elemanı olarak sıkça kullanılırlar. Örneğin, bir grafik çizicisinin kalemine ait x ve y bileşenleri, mikroişlemci tarafından kontrol edilen step motorlarla hareket ettirilebilir. Bu bölümde, step motorların kontrolünde mikroişlemcilerle yapılabilecekleri ele alacağız. Örneğin Şekil 4.1 yazılım ağırlıklı bir açık çevrim sistemidir. Çünkü faz kontrol işaretlerini mikroişlemci üretir. Program, motoru istenen pozisyona getirmek için zamanlama ve işaretlerin sıralanmasından sorumludur. Şekil 4.2' deki örnek ise donanım ağırlıklı bir sistemdir. Burada mikroişlemci sadece hedef pozisyon bilgisini ve donanım kontrol devresine başlama işaretini üretir.



Şekil 4.1 Yazılım ağırlıklı açık çevrim kontrolü

Yazılım veya donanım ağırlıklı arabirim arasından seçim yapılırken, sistem tasarımcısının kullanacağı mikroişlemci ve

motorları birçok yönden ele alması gerekmektedir. Bu noktada ilk göze alınacak husus, işlemci kapasitesinin yazılımın ihtiyacını karşılayabilmesidir. Bu kapasite mevcutsa, yazılım ağırlıklı tasarım, hazır step motor sürücü programlarının da olduğu göze alınırsa, düşük bir maliyetle tasarlanabilir. Bir takım gerçek zamanlı çalışan cihazların kontrolü uygulamalarında donanım yoğunluklu tasarım, yazılımın getirebileceği zorluklar nedeniyle daha gerçekçi bir yaklaşım olacaktır.



Şekil 4.2 Donanım ağırlıklı açık çevrim kontrolü

Yazılım ağırlıklı yaklaşım oldukça karmaşık kontrol algoritmalarına olanak vererek motor performansını maksimuma çıkarırlar. Donanım ağırlıklı yaklaşım ise kontrolün karmaşıklığına olanak vermez ve kontrol devresi optimum çalışma şartlarına yaklaşmayı amaçlar. Örneğin ileride inceleyeceğimiz hızlanma yavaşlama olayı, lineer rampa fonksiyonuyla yapılmaya çalışılır.

Sonuç olarak, mikroişlemci ve motor hızları uyum içinde olmalıdır. Kapalı çevrim kontrollü motorlar saniyede 20.000 adımlık hıza erişebilir ve bu hızda bir adım 50μs içinde gerçekleşir. Gerçek zamanlı kontrol uygulamalarında temel mikroişlemciler (Örneğin Motorola 6800, Intel 8080), 1 - 2μs' lik komut işleme süreleri olduğundan, yazılım temelli kapalı çevrim kontrolünde, yüksek hızlarda her motor adımı için sadece 25-50 komut kullanılabilir ki bu da adım zamanlaması, adım sayma ve faz sıralama gibi fonksiyonlara sınır getirir. Bu nedenle yazılım ağırlıklı yaklaşım kullanılıyorsa, karmaşık kontrolleri sağlamak işlemcinin hızına bağlıdır.

Bu bölümün devamında çeşitli kontrol düzenlerinin yazılım ve donanım ağırlıklı gerçekleştirilmesi, özellikle işlemci gereksinimleri, motor performansının optimizasyonu ve mikroişlemci temelli kontrolün getirdiği sınırlamalar açısından ele alınacaktır.

4.1 Açık Çevrim Kontrolü İçin Yazılım ve Donanım

Birçok step motor kontrol sistemi, çalışma hızı zorlayıcı yük koşullarının altında olan, sabit hızlı açık çevrim kontrol sistemi ile çalıştırılır. Bu düzenek motorun yüke konum verirken, zamanın çok kritik önemi olmadığı uygulamalarda oldukça yeterlidir. Bu sistemin avantajı basitliğidir. Oldukça sınırlı sayıda kontrol fonksiyonu istenir ve yazılım veya donanım olanağı her fonksiyon için

mevcuttur. Bu fonksiyonlar ve karşılaştırmaları bir alt bölümde ele alınacaktır.

İkinci alt bölümde ise daha karmaşık açık çevrim kontrol sistemlerini inceleyecek, yazılım ve donanım ağırlıklı kontroller performans yönünden tartışılacaktır.

4.1.1 Sabit Hızda Çalışma

Açık çevrim kontrolünün bu basit şekli 3 temel kontrol fonksiyonu içerir.

i) Adım hızı: Motor hızını start/stop hızından daha az olan bir değere set eder. Start hızı, motorun adım kaybetmeden yük ataletini karşılayabileceği maksimum adım frekansıdır. Stop rate ise motorun dur emri verildiğinde adım kaçırmadan durabileceği maksimum adım frekansıdır. Bu değerler birbirine çok yakındır ve içlerinde en küçük olanı kullanılarak ortak bir start/stop hızından bahsedilebilir.

ii) Faz sıralama: Motor fazlarının istenen yönde dönmeyi sağlayacak sırada uyarılmasını sağlar.

iii) Adım sayma: Motor tarafından atılan adım sayısını kaydeder ve hedef pozisyona gelindiğinde adım komutlarının motora ulaşmasını engeller.

Bu kontrol fonksiyonlarının her biri yazılım veya

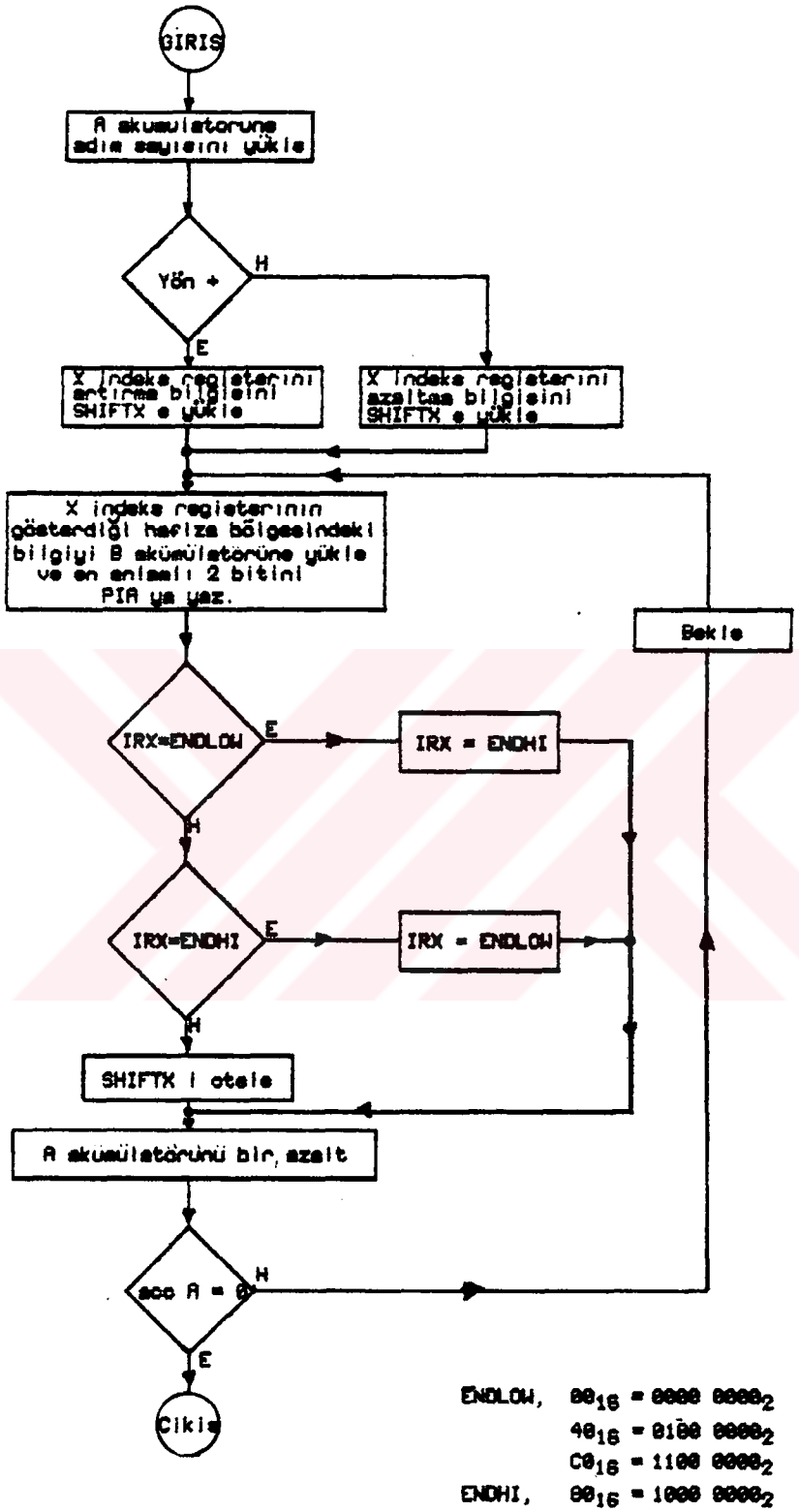
donanımla gerçekleştirilir ve tüm bir step motor denetçisi bu iki alternatifin bir kombinasyonundan oluşabilir.

4.1.1.1. Yazılım Kontrollü Hız, Faz Sıralama ve Adım Sayma

Bu sistemde mikroişlemci, motor sürücü devresi için faz kontrol işaretlerini üretir (Şekil 4.1) ve 3 kontrol fonksiyonu da yazılım tarafından gerçekleştirilir. örnek olarak Şekil 4.3' de, 8 bitlik bir mikroişlemci ile 2 fazlı hibrid step motorun kontrolü için bir akış diyagramı görülmektedir. Motor iki fazının sırayla uyarılmasıyla çalıştırılır ve istenen dönüş yönüne göre program, PIA (Parallel Interface Adaptor)' nın 2 bitini kontrol eder.

Pozitif Dönme		Negatif Dönme	
Bit 0	Bit 1	Bit 0	Bit 1
0	0	0	0
0	1	1	0
1	1	1	1
1	0	0	1
0	0	0	0

PIA' nın 0 nolu biti A fazını, 1 nolu biti B fazının uyarılmasını kontrol eder. Yani bit0=0 ise A fazı pozitif, 1 ise negatif akımla uyarılır. Motor uyarma sırası pozitif dönme için, A+B+,A+B-,A-B-,A-B+,A+B+,..., negatif dönme için, A+B+,A-B+,A-B-,A+B-,A+B+,... şeklindedir.



Şekil 4.3 Yazılım kontrollü hız, faz sıralama ve adım sayma programına ait örnek akış diyagramı.

Şekil 4.3' de görülen motor kontrol akışında adım sayısı akümülatör A'ya yüklenir. Adım sayısı 8 bitle ifade edilemiyorsa rutin tekrarlanmalıdır. X indeks registeri ENDLOW ve ENDHI arasındaki listenin hangi konumunda olunduğunu gösterir. İstenen dönüş yönüne bağlı olarak SHIFTX'e, X'i artırma (pozitif dönüş) veya azaltma (negatif dönüş) bilgisi yüklenir. Daha sonra B akümülatörüne X registerının gösterdiği adresteki bilgi (ındexed adresleme) yüklenir ve en anlamlı 2 biti faz uyarmasını değiştirecek olan PIA' ya yazılır. Örneğin, indeks registerın ENDLOW+1 olmasıyla, 40(Hex) sayısı B akümülatörüne yüklenir ve bit0=0, bıt1=1 olduğunda faz uyarması A+B- halini alır.

Motor hareketinin yönüne bağlı olarak indeks register ENDLOW ve ENDHI arasında hareket eder. Motor negatif yönde hareket ediyorsa faz uyarımının bir turu ENDLOW noktasında biter ve bir sonraki tur, registerın ENDHI noktasına ötelenmesiyle başlar. Benzer şekilde, pozitif yönde dönüşte register ENDHI noktasından ENDLOW noktasına ötelenır. Register, uyarma sırasını gösteren listenin sonunda değilse SHIFTX komutunun icra edilmesiyle bir artırılır veya bir azaltılır.

A akümülatörü hedefe göre bağıl olarak motorun pozisyonunu depolar ve faz uyarmasının değiştirilmesiyle motor bir adım attığında, bu değer bir azaltılır. Daha sonra A akümülatörü kontrol edilir ve motorun hedefe geldiğini

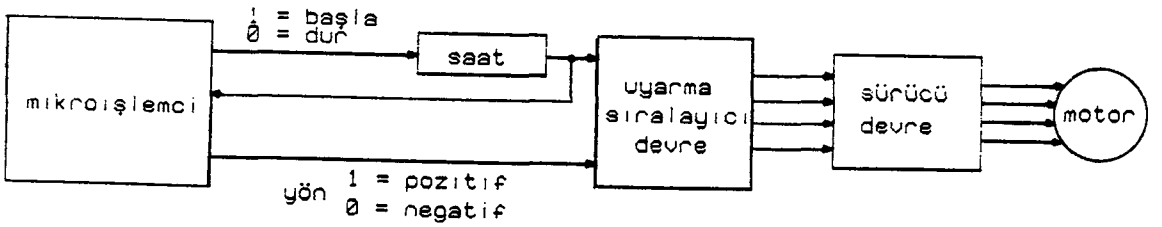
gösteren 0 değerine geldiğinde motor kontrol rutini terkedilir. Sonraki adımlar gerçekleştirilecekse bir sonraki uyarma değişiminden önce bir zaman gecikmesi gerekir. Bu bekleme sırasında mikroişlemci başka motorların çalıştırılması gibi işleri yapabilir. Bu işler için harcanacak zamanın tek sınırlaması, iki uyarma arasındaki zamana eşit olması gerektiğidir. Bekleme rutininden çıkıldığında bir sonraki uyarma değişimi, uyarma listesindeki yeni değerin B akümülatörüne yüklenmesiyle başlatılır.

Yazılım temelli bu kontrol sisteminin en önemli avantajı, faz kontrol sinyallerinin doğrudan PIA kullanılarak gerçekleştirilmesiyle yapılan mikroişlemci/motor arabiriminin basitliğidir. Motor kontrol programının diğer alternatiflere göre mikroişlemci hafızasının çoğunu kullanmasına rağmen, hafıza gereksinimi yine de oldukça azdır. Esas programlama sorunu faz uyarma değişimlerinin doğru zamanlamasında oluşabilmektedir. Adımlar arasındaki zaman gecikmesinden sorumlu olan program parçasının, doğru zamanlama için diğer bölümler gibi oldukça dikkatli bir şekilde yazılması gerekmektedir.

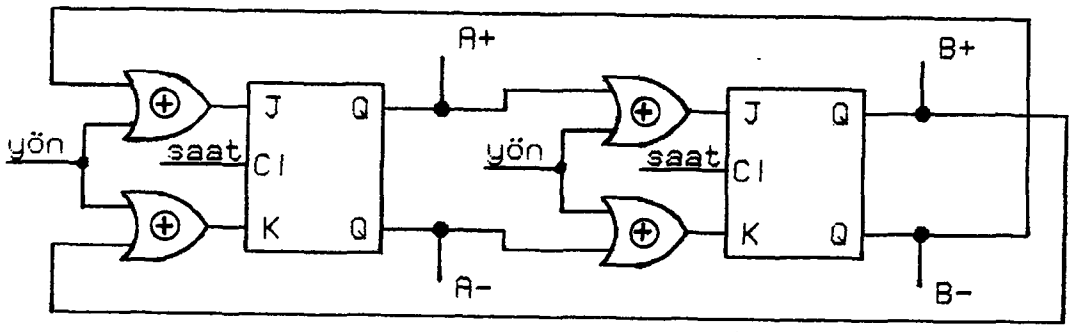
4.1.1.2 Yazılımla Adım Sayma, Donanımla Zamanlama ve Faz Sıralama

Bu metodla, açık çevrim kontrolünün gerçekleştirilmesinin blok diyagramı Şekil 4.4.a' da

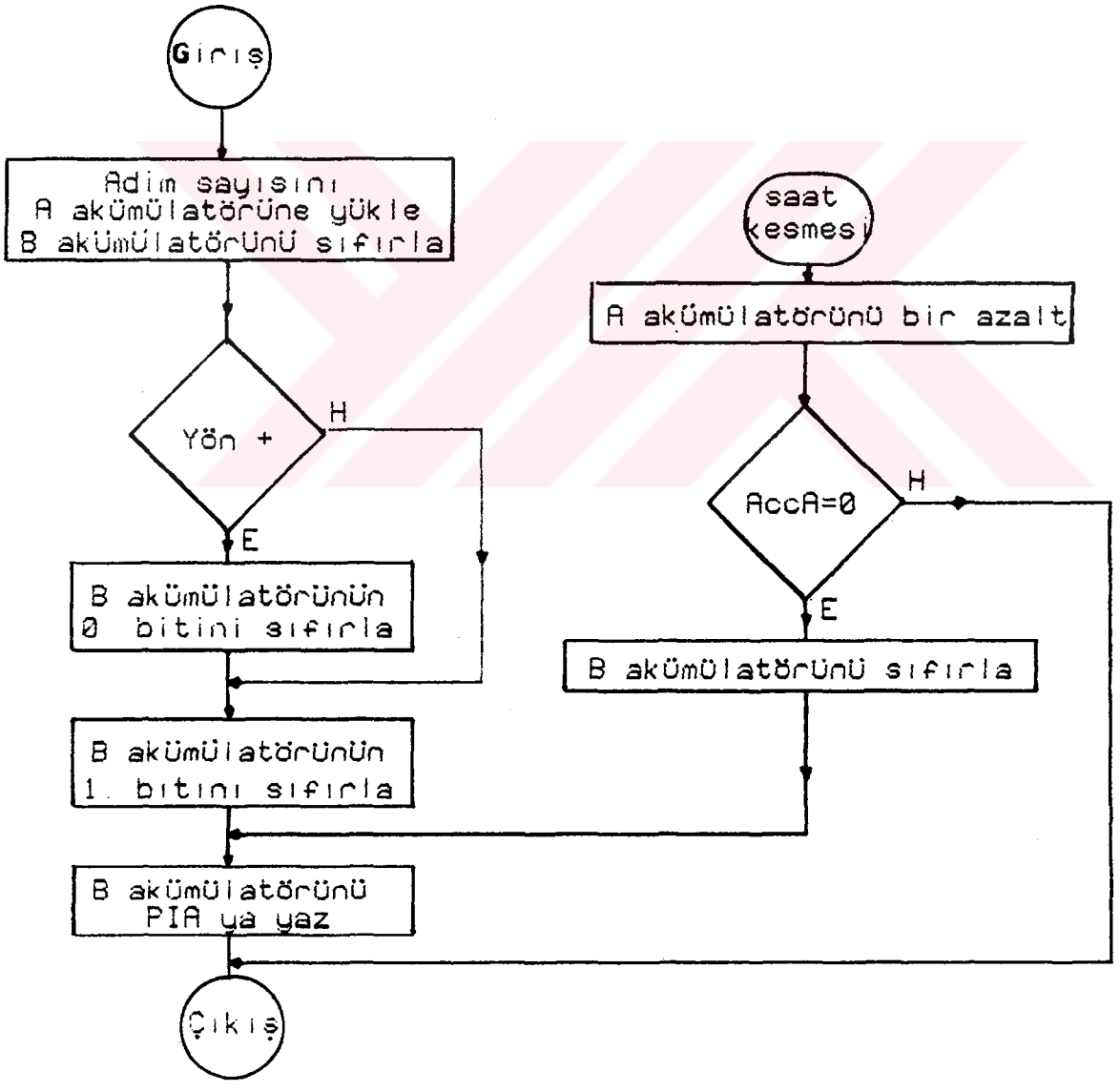
görülmektedir. Adım atma hızı mikroişlemciden çıkan tek bitlik bir sinyalin kontrol ettiği, tek frekanslı saat işaretiyle sabitlenmiştir. Saat darbeleri motor pozisyonunun hedefe göre bağıl durumunu kaydetmesi için mikroişlemciye ve mikroişlemcinin gönderdiği yön işareti tarafından, faz kontrol işaretini üreten uyarmaları sıralayan donanıma gönderilir. Şekil 4.4.b, 2 J-K flip flop ve 4 EX-OR kapısından oluşan ıkı fazlı hibrid step motorun uyarılması için tipik bir uyarma sıralayıcı devre örneğidir. Bu devrenin girişleri saat darbesi ve yön işareti. çıkışları ise 4 adet faz kontrol sinyalidir. (A+,A-,B+,B-)



Şekıl 4.4.a Yazılımla adım sayma, donanımla zamanlama ve sıralama sisteminin blok diyagramı.



Şekil 4.4.b Sıralama devresi



Şekil 4.4.c Mikroişlemci yazılımının akış diyagramı

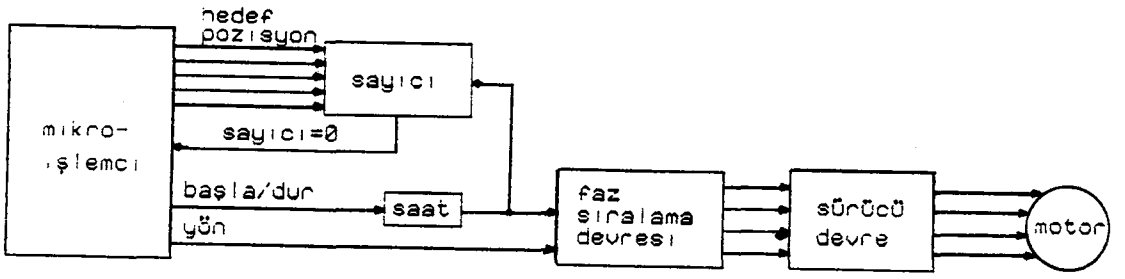
Mikroişlemci yazılımı saat işaretlerini sayar ve saat start/stop işaretiyle yön işaretini üretir. Motor kontrol rutininin başında atılacak adım sayısı sayıcıya yüklenir. Bu örnekte sayıcı, akümülatördür. B akümülatörü ilk olarak sıfırlandıktan sonra, pozitif yönde dönme isteniyorsa en anlamlı biti olan bit0=1 olur. Bir sonraki anlamlı bit olan bit1' de bu anda 1 yapılır ve akümülatörün bu ıkı bıtı daha sonra yön ve saat start/stop sinyali olmak üzere PIA' ya yazılır. Böylece B akümülatörünün 0. bıtı yön sinyali, 1. biti ise start/stop sinyali olarak çıkar. Bundan sonra mikroişlemcinin diğer işleri yapabilmesi için motor kontrol rutininden çıkılabilir.

Saat start sinyalini aldığında darbe üretmeye başlar. Bu darbeler program akışını motor kontrol programının INTERRUPT girişine yönlendirir. Uyarmaları sıralayan devreye gelen bir saat darbesi step motora bir adım attıracak uyarma değişimine neden olur. Bundan dolayı motor pozisyonunun hedefe göre bağıl durumunun bilinmesi için A akümülatörünün içeriği yeni durum bilgisini içerecek şekilde bir azaltılmalıdır. Akümülatördeki sayı 0 değilse kontrol rutininden çıkılır, 0 ise hedefe varılmış olduğundan saat işareti durdurulmalıdır. Bu iş B akümülatörünün sıfırlanması ve en anlamlı iki bit olan yön ve start/stop sinyallerinin her ikisinin de sıfırlanması için PIA' ya yazılmasıyla yapılır. Bu yazılım/donanım kombinasyonunun avantajı, tüm bileşenlerinin

oldukça basit olmasıdır. Start/stop sinyali ile kontrol edilebilen sabit frekanslı saat devresi tek bir entegre devre ile. Şekil 4.4.b' deki uyarma sıralama devresi ise, sadece 2 entegre devre ile kurulabilir. Mikroişlemci yazılımı program geliştirme, zamanlama kaygılarından uzak şekilde yapılabilir. Çünkü mikroişlemci her motor adımı arasında diğer işleri yapabilmek için serbesttir.

4.1.1.3 Donanımla hız, faz sıralama ve adım sayma

Şekil 4.5' deki 3 temel kontrol fonksiyonu da donanımla yapılır. Motor kontrol rutininin başında mikroişlemci hedefin pozisyonunu bir geri sayıcıya yükler, sabit frekanslı saat darbelerini başlatır, uyarma sıralama devresi için yön sinyalini oluşturur ve hedefe varılana kadar herhangi bir görevi yoktur.



Şekil 4.5 Donanımla hız, faz sıralama ve adım sayma

Sabit frekanslı saat darbesi motor/yük kombinasyonunun start/stop hızında adım komutlarını üretir ve bu komutlar hem uyarma sıralama devresine hem de motorun o anda, hedefe göre bağıl durumunu gösteren geri sayıcıya gider. Hedefe varıldığında sayıcının içeriği 0' dır ve bu durum daha sonra saati durdurma sinyalini üretecek olan mikroişlemciye iletilir. Alternatif olarak, sayıcının 0' a ulaşması da doğrudan saati durdurabilir.

Bu sistemin en önemli avantajı önceki iki sisteme göre mikroişlemciyi en az meşgul etmesidir. Mikroişlemci her motor adımında meşgul edilmez. Donanım ağırlıklı olduğundan açıklanan sistemler içinde en kompleksi olmakla birlikte devreler, az sayıda entegre devrelerle ve özel step motor kontrol entegreleriyle birlikte kolayca oluşturulabilir. Donanım temelli bu sistemin olası dezavantajı, mikroişlemci ile kontrol devreleri arasındaki arabirim için çok sayıda işaret hattı gerekmesidir. Örneğin maksimum 128 adımlık bir sayıcının paralel olarak yüklenmesi için 7 hat gerekmektedir.

4.1.2 Rampalı Hızlanma/Yavaşlama

Açık çevrim step motor kontrolü start/stop hızının üstünde yapılıyorsa, bu andaki adım hızına belli sayıda adımdan sonra ulaşılmalıdır. Hedef pozisyona yaklaşıldığında ise, hedefe varma anında motorun start/stop hızının altında

çalışması için adım hızı yavaşlatılmalıdır. Optimum hız davranışı, analitik teknikler kullanılarak pull-out tork/hız karakteristiğinden bulunabilir. Pull-out tork/hız karakteristiği, step motorun çeşitli hız değerlerinde verebileceği maksimum torku gösterir ve her motor için grafik şeklinde kullanıcılara sunulur. Bu bölümde adım hızının karmaşık kontrolü için yazılım ve donanım ağirliklı sistemlerin birbirlerine göre üstünlükleri tartışılacaktır.

4.1.2.1 Yazılım Ağırlıklı Yaklaşım

Açık çevrim kontrolünün bu tipinde mikroişlemci faz kontrol işaretlerini (Şekil 3.1) üretir ve yazılım, uyarma sıralama, adım sayma ve her uyarma arasındaki zamanlamadan sorumludur. Rampalı hızlanma/yavaşlama kontrolünde uyarma zamanlaması özellikle dikkat isteyen bir çalışmadır ve hızlanma ve yavaşlama sırasında her adımın uzunluğu ayrı bir hafıza bölgesi işgal ettiğinden büyük miktarda hafıza gerektirir.

Tipik bir yazılım temelli kontrolün akış diyagramı Şekil 4.6' da görülmektedir. Her uyarma arasındaki zaman, hızlanma veya yavaşlama tablosundan alınan gecikme değeri ile belirlenir. Her uyarma değişiminin arasında işlemci, gecikme değerinin bir sayıcıyla 0' a doğru sayıldığı bir BEKLE prosedürü icra eder. Yüksek gecikme değerleri uzun gecikmelere neden olduğundan adım hızı düşüktür. Şekil 4.6'

daki örnekte hızlanma için tipik gecikme değerleri ACCST bölgesinden başlayıp ACCEND' de biten ve ACCVAL değerlerini içeren tabloda depolanmıştır. Benzer şekilde DECST bölgesinden DECEND' e kadar bölgeye yerleşmiş DECVAL değerleri ise yavaşlama tablosudur.

Program, adım sayısı cinsinden hedef pozisyonunun COUNT değerine yüklenmesiyle başlar ve bir POINTER ilk gecikme değeri olan ACCST' i gösterir. Uyarmanın istenen dönme yönüne bağlı olarak değişmesi motora hareket verir ve kalan adımları gösteren COUNT değeri bir azaltılır. Program, POINTER' ı ilerleterek hızlanma tablosundan bir sonraki değeri alır ve yeni uyarma değişiminden önce adım arasındaki sürenin tamamlanmasını bekler. Bunun yanında POINTER' ın gerçek değer gösterdiğinin anlaşılması için birkaç basit test uygulanmalıdır:

i) Hızlanma tablosunun sonuna gelinmişse motor, ACCEND değeriyle hareketine sabit hızda devam eder. Bu koşul POINTER ve ACCEND değerlerinin eşitliği ile algılanır.

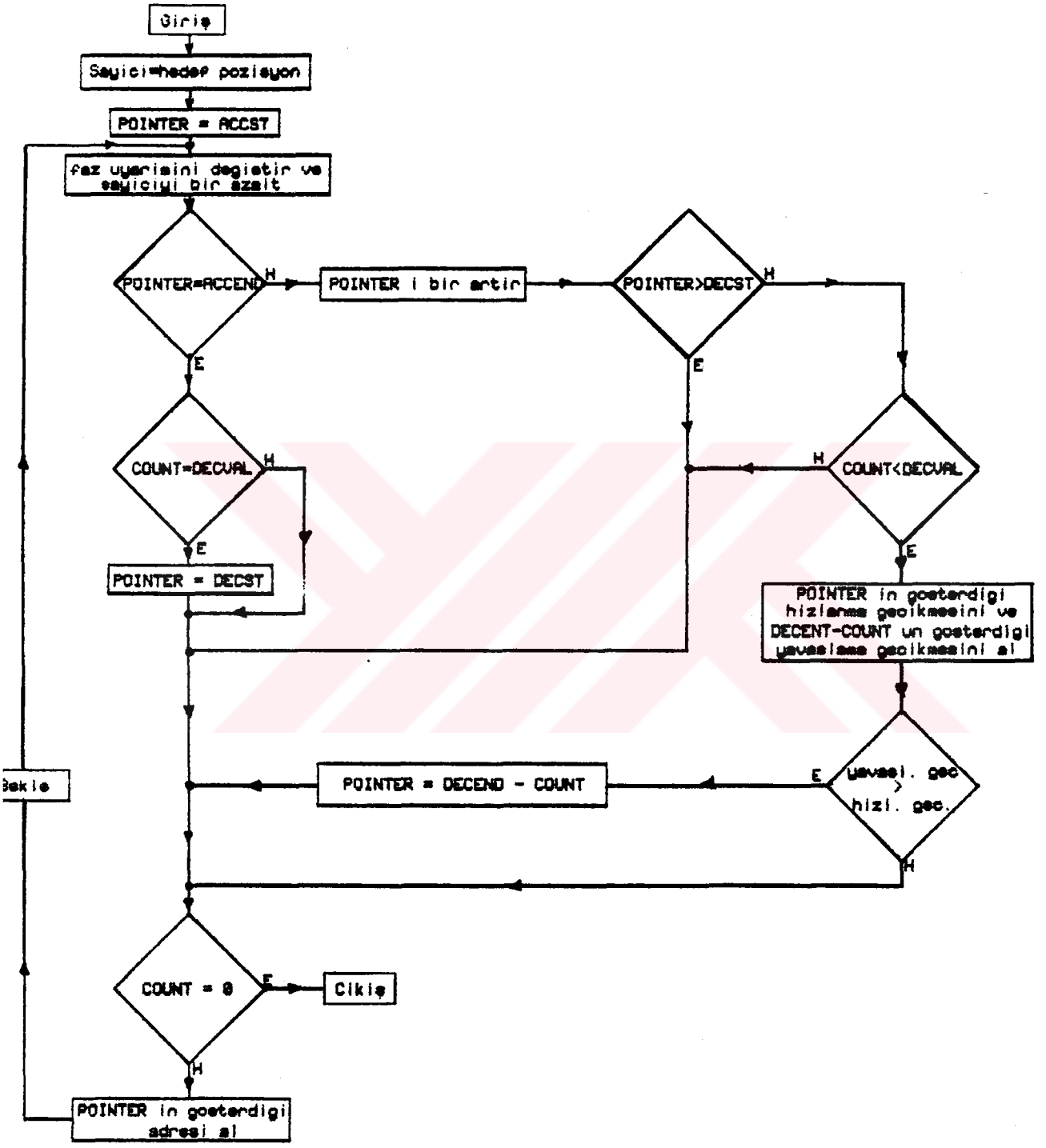
ii) Maksimum hıza ulaşıldığında, yavaşlamanın başlayacağı adım sayısıyla, hedefe varmak için kalan adım sayısı eşit olur ve adım hızının yavaşlatılmasına başlanmalıdır. Yavaşlatma, sonraki gecikme değerlerinin tablodan okunması için POINTER=DECST yapılarak başlatılır.

iii) Motor, atılacak adım sayısı hızlanma ve yavaşlama için gerekli adım sayısından küçükse, motor maksimum hızına ulaşamaz. Yani COUNT' un ilk değeri ACCVAL+DECVAL' den küçüktür.

Hızlanma sırasında hedefe ulaşmak için kalan adım sayısı (COUNT), yavaşlama değerlerinin sayısı ile karşılaştırılır. COUNT değeri DECVAL' den büyük olduğu sürece motor hızlanmaya devam eder. Fakat başka testler de gereklidir. O anki hızlanma gecikmesi yavaşlama gecikmesi değeriyle, yavaşlamanın bu pozisyonda başlayıp başlamayacağının anlaşılması için karşılaştırılır. (Akış diyagramında DECENT-COUNT bölümünde) Yavaşlama gecikmesi hızlanma gecikmesinde büyükse yavaşlatma, POINTER' a yavaşlatma tablosundaki ilgili değer yüklenerek başlatılır.

Tablo 4.1 Örnek hızlanma/yavaşlama tablosu

Hızlanma Değerleri	Yavaşlama Değerleri
ACCST, 300	DECST, 75
240	77
190	80
150	84
115	.
90	.
.	.
.	.
.	230
ACCEND, 70	DECEND,300
ACCVAL=ACCEND-ACCST.	DECVAL=DECEND-DECST



Şekil 4.6 Rampalı hızlanma/yavaşlama yazılımının akış diyagramı

Yazılım ağırlıklı kontrol düzeni, adım aralıklarının hızlanma ve yavaşlama sırasında doğru ve hassas zamanlamasını sağlar. Bir gecikme rutini çevriminin icrası için geçen zaman T1 ve uyarma değişimi, pozisyon sayımı, yeni pozisyonun hafızaya alınması ve gecikme pointerının hareketi için geçen zaman T2, gecikme değeri d ise uyarma değişimı için geçen süre:

$$\text{Adım Aralığı} = 1 / (dT1+T2) \text{ sn dir.} \qquad (4.1)$$

T1 ve T2, yazılımın benzer bölümlerinde işlemcinin komut çevriminin değeri kadar süre sabittir. Motorola 6800 ve Intel 8080 gibi işlemciler için tipik değerler T1=10μs ve T2=50μs' dir. Değişik gecikme değerleri için adım hızı tabloları yapılabilir.

Tablo 4.2 Örnek adım hızı tablosu

Adım Hızı (adım s)	Gecikme Değeri
99.9	996
100.0	995
100.1	994
.	.
.	.
.	.
990	96
1000	95
1010	94
.	.
.	.
.	.
4760	16
5000	15
5270	14

Bu örnek tablo yazılım temelli sistemin önemli

dezavantajını göstermektedir. Artan hızlarda, mümkün olan hız sınırları içinde hassasiyet azalmaktadır. Örneğin saniyede 100 adım için gecikme değerindeki birim değişim, adım hızında %0.1' lik bir değişme oluştururken, 5000 adım için aynı gecikme, adım hızında %5' lik bir fark oluşturmaktadır.

İşlemci gereksinimleri ile ilgilendiğimizden Şekil 4.6' daki akış diyagramı mikroişlemcinin sadece tek bir step motoru kontrol ettiği varsayılarak yapılmasına rağmen yazılım, birkaç motorun kontrolü için adapte edilebilir. En büyük sorun büyük miktarda hafıza gereksinimidir. Özellikle step motor sistemi, ataleti büyük bir yüke sahipse hızlanma ve yavaşlama için çok miktarda adım gerekmektedir. Optimum hız profilinden bazı sapmalar gözardı edilirse oldukça yüksek maliyetli ilave hafıza miktarı iki yolla düşürülebilir.

i) Hızlanma gecikme değerleri, ters olarak yavaşlama için kullanılabilir.

ii) Yaklaşık olarak tekrarlanan bir formül, her gecikme değerinin önceki değerlerden hesaplanması için çıkarılabilir.

Bu tartışmalardan çıkan sonuç, yazılım temelli bir sistem orta hız değerleri için hız karakterinin hassas bir şekilde kontrolüne olanak vermesine rağmen yüksek hızda çalışmayı engelleyebilir. Bundan dolayı hızlanma/yavaşlama işlemlerinin ağırlıklı olduğu uygulamalarda en uygun

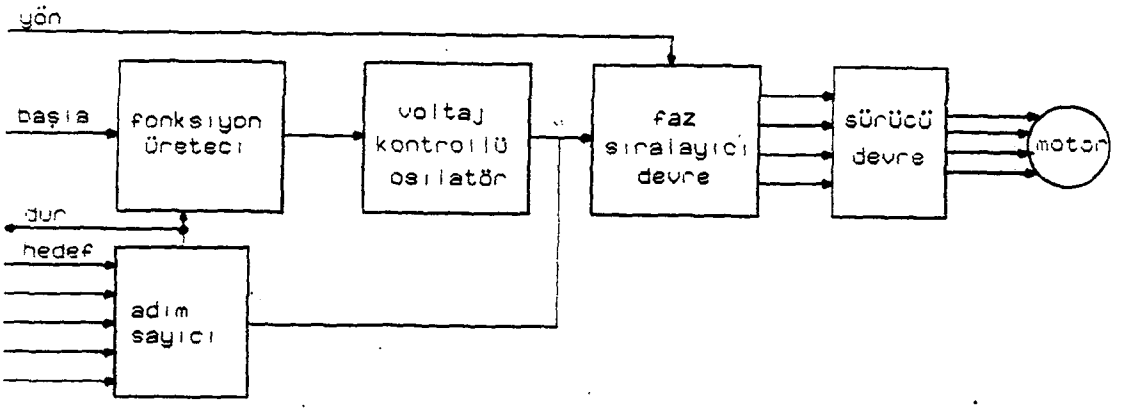
yöntemdir.

4.1.2.2 Donanım Ağırlıklı Yaklaşım

Donanım temelli kontrolde, mikroişlemcinin kullanımı minimumdur. Hedef pozisyon ve yön bilgisi işlemciden kontrol devresine iletilir (Şekil 4.2) ve işlemci donanımdan, hedefe varıldığına dair bir işaret alana kadar diğer işlerle meşgul olur. Kontrol devreleri bu nedenle, motor pozisyonunun hedefe göre durumunu kaydettiği kadar faz uyarmalarının sıralanması ve zamanlamasıyla da meşguldür.

Birçok kontrol devresi, Şekil 4.7' de şematik olarak gösterilen yapıdadır. Bu sistemin kalbi, adım komutlarını faz uyarma devresine gönderen gerilim kontrollü osilatördür. Osilatör girişi mikroişlemci tarafından başlatılan ve adım sayıcısı tarafından durdurulan bir fonksiyon üreteci ile sağlanır. Bu sistemin yazılım ağırlıklı yaklaşıma göre avantajı hemen görülebilir: Gerilim kontrollü osilatörün çıkışı sürekli değişkendir. Bu nedenle ayrık adım hızlarına sıçrama nedeniyle motor çalışma hızı fazla sınırlanmamış olur.

Optimum açık çevrim çalışma için hızlanma eğrisi yaklaşık eksponansiyel, yavaşlama için ise ters eksponansiyel olması gerekir. (Şekil 4.8.a). Ne yazık ki bu son fonksiyonun basit analog devrelerle oluşturulması çok zor olduğundan hız



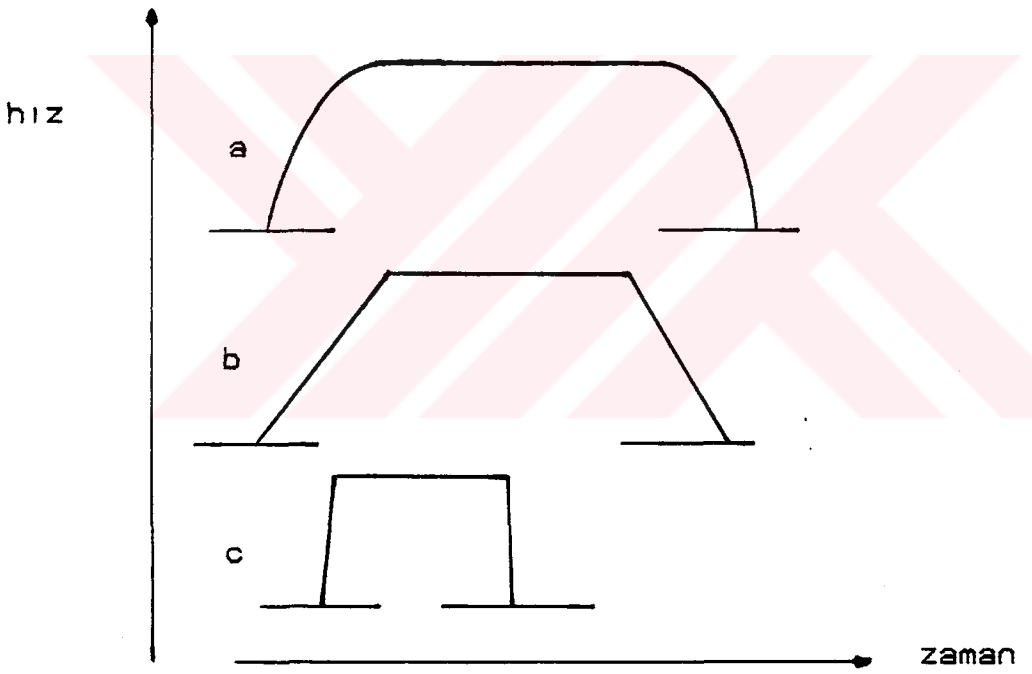
Şekil 4.7 Rampalı hızlanma/yavaşlama için donanım temelli kontrol

eğrisi, pratik yaklaşıklıkla lineer rampa fonksiyonu halini alır. (Şekil 4.8.b-c). Bununla birlikte bu yaklaşım rampa fonksiyonunun bir sonucu olarak, sistemin maksimum hızına bir sınırlama getirir. Herhangi bir hızda mümkün olan hızlanma şu eşitlikle verilir:

$$\text{Hızlanma} = (\text{motor torku} - \text{yük torku})/\text{atalet} \qquad (3.2)$$

Örnek olarak pull-out hızında motor ve yük torkları eşit olduğundan pull-out hızı, sıfır hızlanmaya yaklaşır. Lineer rampa hız fonksiyonunda hızlanma sabittir ve izin verilen maksimum hızlanma değeriyle sınırlıdır. Maksimum adım hızı, gerçekte motor torku yük torkundan büyük olduğundan ve sistem çabuk hızlanabildiğinden sınırlı tutulmalıdır. Artan hızlarda

motor torku azaldığından düşük adım hızı sınırı yüksek motor torku ve böylece çabuk hızlanma anlamına gelir. Bu nedenle hız sınırına kısa bir sürede varılır. (Şekil 4.8.c). Aksine yüksek adım hızı sınırı yavaş hızlanmayı doğurur. (Şekil 4.8.b). Optimum rampa eğrisi ve adım hızı sınırı, atılacak adım sayısına bağlıdır. Ayrıca fonksiyon üretecinin karakteristiği hedef pozisyon bilgisine göre oluşturulabilir.



Şekil 4.8 Açık çevrim kontrolü hız profili
a) optimum hız profili
b) uzun mesafeler için lineer rampa fonksiyonu
c) kısa mesafeler için lineer rampa fonksiyonu

Çeşitli genel amaçlı step motor kontrolörü bulunmaktadır (Genellikle MSI entegre devre şeklindedir) ve

çeşitli karmaşıklık seviyelerinde kontrol mümkündür. ilginç bir gelişme, yüksek maliyetli, sınırlı hız bölgesi fakat iyi hızlanma/yavaşlama karakteristiğine sahip hassas mikroişlemci kontrolü ile, düşük maliyetli, geniş hız bölgesi fakat zayıf hızlanma/yavaşlama karakteristiğine sahıp donanım yoğunluklu kontrol arasında bir orta yol bulan programlanabilir step motor kontrol devreleridir.

4.2 Mikroişlemci Temelli Kapalı Çevrim Kontrolü

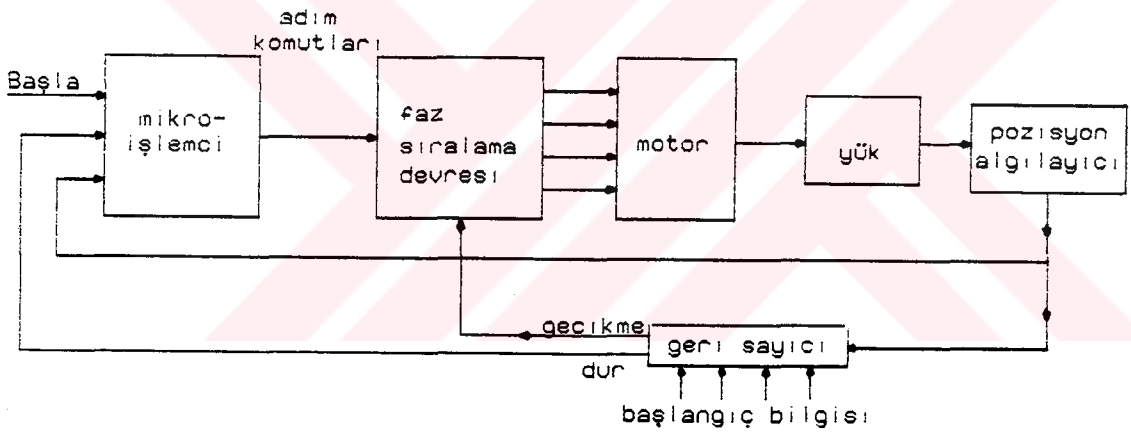
Tez konusu olan baskılı devre matkap tezgahı, tamamen açık çevrim kontrolü kullanılarak gerçekleştirildiği için, kapalı çevrim kontrolü yüzeysel olarak ele alınacaktır.

Kapalı çevrim step motor kontrol sisteminde, herhangi bir andaki motor pozisyonu algılanır ve kontrol birimine geri besleme yoluyla iletilir. Her adım komutu, sadece önceki komut motor tarafından başarılı şekilde yerine getirilmişse gönderilerek, motorun yükle senkron bir şekilde çalışması sağlanır.

Şematik bir kapalı çevrim kontrolü Şrkil 4.9' da görülmektedir. Başlangıçta sistem, bir veya daha fazla fazı uyarılmış şekilde sabittir. Hedef pozisyonu bir geri sayıcıya yüklenir ve darbe şeklinde bir start sinyali kontrol birimine gönderildiği anda, faz sıralama üretecine adım komutları gönderilir. Sonuç olarak uyarmada bir değişiklik olur ve

motor verilen yük parametrelerine göre hızlanmaya başlar.

İlk adım bitmek üzereyken pozisyon üreteci geri sayıcıya ve kontrol birimine bir darbe gönderir. Geri sayıcı 1 eksiltilerek hedefe göre yükün pozisyonunu içermesi sağlanır. Burada açık ve kapalı çevrim arasındaki fark görülmektedir: Açık çevrim kontrolünde geri sayıcı, motora gönderilecek adım sayısını tutar ve bu adımların atıldığının garantisini



Şekil 4.9 Kapalı çevrim step motor kontrolünün blok diyagramı

vermez. Kapalı çevrim kontrolünde ise geri sayıcı yükün gerçek pozisyonunu tutar.

Kontrol birimine gönderilen pozisyon algılama darbesi bir sonraki adım komutunun üretilmesinde kullanılır. Büyük yükler için ilk adım pozisyonuna ulaşmak için geçen süre daha

uzundur ve bundan dolayı ard arda gelen adım komutları arasındaki süre, hızlanmanın yavaş olmasına izin vermek için kendiliğinden ayarlanır. Neticede motor, açık çevrim kazancında olduğu gibi motor ve tork/hız karakteristiği tarafından belirlenen maksimum çalışma hızına ulaşır ve hedef pozisyona yaklaşana kadar bu hızda çalışmaya devam eder. Daha sonra geri sayıcı motoru, hedefe doğru yavaşlatmak için istenen faz sıralamasındaki değişikliği başlatan bir yavaşla sinyali üretir. Geri sayıcı sıfır olduğunda istenen sayıda adım atılmıştır ve bir STOP sinyali, daha fazla adım komutunu engellemek için kontrol birimine gönderilir.

Kapalı çevrim kontrolü böylece, yük koşullarını uyarma zamanlamasına uydurur ve hızlı yük pozisyonlama sonucunu üreten, yaklaşık optimal hız eğrisini verme yeteneğine sahiptir. Birçok kullanıcı için en etkileyici özellik, kötü yük koşullarında bile adım komutları ile rotor pozisyonu arasındaki senkronizasyonun bozulma olasılığı olmamasını sağlayan, yük pozisyonunun doğrudan izlenmesidir. (Acarnley, 1984)

5. MATKAP TEZGAHI KONTROL BİRİMİ

Sistemde olması gereken özellikler, matkap motorunun delinecek kart üzerinde X ve Y koordinatlarında, delme işlemi için de Z ekseni üzerinde hareket etmesidir. Bu hareketlerin sağlanması için temin edilen 3 adet, adım açısı 1.8 derece/adım olan çift kutuplu ve 1 adet, adım açısı 15 derece/adım olan bifiler sarımlı hibrid step motor kullanılmıştır.

Sistemin kontrolü bir PC tarafından yapılmaktadır. Bunun için, piyasada hazır olarak bulunan, 2 adet8255 paralel giriş/çıkış tümdevresi içeren bir giriş/çıkış kartı kullanılmıştır. Kartın çalışma şeklini ve dolayısıyla Ek A' da verilen yazılımı anlayabilmek için, 8255 tümdevresinin çalışma şeklinin bilinmesi gerekmektedir.

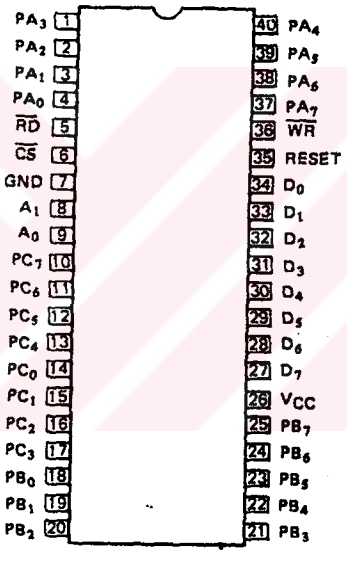
5.1 8255 Paralel Giriş/Çıkış Tümdevresi

8255 paralel giriş/çıkış tümdevresi, her biri 8 uca sahip 3 adet giriş/çıkış portu içermektedir. Bu üç port iki gruba ayrılmıştır. 1. grup A portunu (PA0...PA7) ve C portunun PC4...PC7 uçlarını kapsar. B portu (PB0...PB7) ile C portunun PC0...PC3 hatları 2. grubu oluşturur. 8255 tümdevresi şu, iç fonksiyon bloklarından oluşmaktadır:

Veri arabelleği, okuma/yazma kumanda lojiği, bir

kumanda/durum kütüğü, 2 port grubunun kumandası ve her kapının sürücüsü.

8255 tümdevresinde port hatlarından başka, veri yolu bağlantısı (D0...D7), seçme ucu ($\overline{CS}$), sıfırlama (Reset) ucu, A0 ve A1 adres hatları ile $\overline{RD}$ ve $\overline{WR}$ kumanda uçları bulunmaktadır. (Şekil 5.1). "Reset" işaretinin alınmasından sonra, 8255 tümdevresinin bütün kapı uçları giriş olarak tanımlanır.



Şekil 5.1 8255 paralel giriş/çıkış tümdevresi

Elemanın içinde 2 kütük bulunmaktadır. Bunlar kumanda kütüğü (Control Word Register) ile durum kütüğüdür (Status Register).

1. ve 2. grubun çalışabileceği 3 farklı çalışma türü bulunmaktadır.

A ve B portlarının çalışma türleri birbirlerinden farklı olabilirken, C portunun yarıları, ait oldukları port grubunun çalışma türüne uyarlar. Durum kütüğü (Status Register) 1. ve 2. çalışma türlerinde veri iletişiminin durumunu verir.

5.1.1 Çalışma Türü 0 (Standart Giriş/Çıkış)

Kullanıcıya 3 adet 8 Bit' lik giriş/çıkış portu verilir. Her port 8 Bit' lik çıkış veya 8 Bit' lik giriş olarak programlanabilir. C portu ise 4 Bit' lik 2 parçaya ayrılabilir ve bunlar, birbirinden bağımsız olarak giriş veya çıkış olarak kullanılabilir.

5.1.2 Çalışma Türü 1 (El Sıkışmalı Giriş/Çıkış)

El sıkışmalı (Hand Shake) arabirimli cihazlarla bağlantı için 2 kapı grubu bulunmaktadır. 1. kapı gruuna ait olan PC4...PC7 hatları A portunun el sıkışma hatları olarak çalışırlar. Benzer şekilde PC0...PC3 de B portunun el sıkışma hatlarıdır. Bir port grubu üzerinden ya sadece giriş, ya da sadece çıkış yapılabilir.

5.1.3 Çalışma Türü 2 (İki Yönlü Yol)

1. port grubu 2 yönlü ve el sıkışmalı arabirim olarak çalıştırılabilir. Aynı sırada 2. port grubu da, 0. veya 1. çalışma türünde işletilebilir.

Kumanda kütüğüne sadece yazılabilir. Bu kütüğün içeriğinin okunması mümkün değildir.

5.2. 8255' in Programlanması

Tümdevrenin kumanda kütüğü programlandıktan sonra, artık veri giriş ve çıkışi yapılabilir. İlk kumanda sözcüğü bütün çıkış portlarını `0' yapar. Bu sözcüğün yapısı Tablo 5.1' deki gibidir.

Tablo 5.1 Kumanda sözcüğünün yapısı

Bit	Anlamı
	1. Grup
D0:	C portu (Alt yarı) 1 = Giriş 0 = Çıkış
D1:	B portu 1 = Giriş 0 = Çıkış
D2:	Çalışma türü seçme 0 = 0. çalışma türü 1 = 1. çalışma türü
	2. Grup
D3:	C portu (Üst yarı) 1 = Giriş 0 = Çıkış
D4:	A portu 1 = Giriş 0 = Çıkış
D6,D5:	00 = 0. Çalışma türü 01 = 1. Çalışma türü 1X = 2. Çalışma türü
D7:	İlk kumanda formatı 1 = aktif

Elemana ilk koşullar verildikten sonra, her zaman, port çıkışlarının durumları değiştirilebilir ve giriş kapılarının durumları okunabilir. Tablo 5.2, hangi işaret kombinezonlarının söz konusu olabildiğini ve bunların neler yaptığını göstermektedir.

Tablo 5.2 8255 tümdevresinin adres yolu ve kontrol yolu işaretleriyle çalışması

A1	A2	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
					Giriş (Okuma komutu)
0	0	0	1	0	A portu -> Veri yolu
0	1	0	1	0	B portu -> Veri yolu
1	0	0	1	0	C portu -> Veri yolu
					Giriş (Yazma komutu)
0	0	1	0	0	Veri yolu -> A portu
0	1	1	0	0	Veri yolu -> B portu
1	0	1	0	0	Veri yolu -> C portu
1	1	1	0	0	Veri yolu -> Kontrol
					Eleman Kapalı
X	X	X	X	1	Çıkışlar yüksek empedanslı
1	1	0	1	0	Geçersiz işaret kombinezonu
X	X	1	1	0	Veri yolu yüksek empedanslı

8255 tümdevresinin programlanması için, elde bulunan olanaklar Tablo 5.3' de birarada görülmektedir. Kumanda sözcüğü "C" (Control Word) ile, çıkış "O" (Output) ve giriş de "I" (Input) şeklinde kısaltılmıştır. (OKI)

Örneğin, A ve B kapılarını çıkış, C kapısını da giriş olarak programlamak isteyelim. Tablo 5.3' den kumanda

kütüğüne bu amaçla onluk düzende 137 yazmak gerektiği görülmektedir.

Tablo 5.3 PIA 8255'ın programlama olanakları (C=Kumanda kütüğü değerleri, O=Çıkış, I=Giriş)

C=128	0-7 A-->O 4-7 C-->O 0-3 C-->O 0-7 B-->O	C=129	0-7 A-->O 4-7 C-->O 0-3 C-->I 0-7 B-->O	C=130	0-7 A-->O 4-7 C-->O 0-3 C-->O 0-7 B-->I
C=131	0-7 A-->O 4-7 C-->O 0-3 C-->I 0-7 B-->I	C=136	0-7 A-->O 4-7 C-->I 0-3 C-->O 0-7 B-->O	C=137	0-7 A-->O 4-7 C-->I 0-3 C-->I 0-7 B-->O
C=138	0-7 A-->O 4-7 C-->I 0-3 C-->O 0-7 B-->I	C=139	0-7 A-->O 4-7 C-->I 0-3 C-->I 0-7 B-->I	C=144	0-7 A-->I 4-7 C-->O 0-3 C-->O 0-7 B-->O
C=145	0-7 A-->I 4-7 C-->O 0-3 C-->I 0-7 B-->O	C=146	0-7 A-->I 4-7 C-->O 0-3 C-->O 0-7 B-->I	C=147	0-7 A-->I 4-7 C-->O 0-3 C-->I 0-7 B-->I
C=152	0-7 A-->I 4-7 C-->I 0-3 C-->O 0-7 B-->O	C=153	0-7 A-->I 4-7 C-->I 0-3 C-->I 0-7 B-->O	C=154	0-7 A-->I 4-7 C-->I 0-3 C-->O 0-7 B-->I
C=155	0-7 A-->I 4-7 C-->I 0-3 C-->I 0-7 B-->I				

5.3. Giriş/Çıkış Kartının Kullanımı

Kullanılan giriş/çıkış kartı, PC bellek haritası üzerinde 01B0(Hex) adresinden itibaren yerleştirilmiştir.

Kart üzerindeki 2 adet 8255 tümdevresinin erişim adresleri Tablo 5.4' de görülmektedir.

Örneğin 1 nolu 8255'i tamamen çıkış, 2 nolu 8255'i ise tamamen giriş için programlamak gerektiğinde, herhangi bir giriş çıkış işlemi yapılmadan önce şu şekilde iki Pascal komutunun icra edilmesi gerekmektedir:

```
port($01B3):=128;
port($01B7):=155;
```

Matkap tezgahına ilişkin programda 8255 tümdevreleri sadece 0. çalışma türünde kullanılmışlardır.

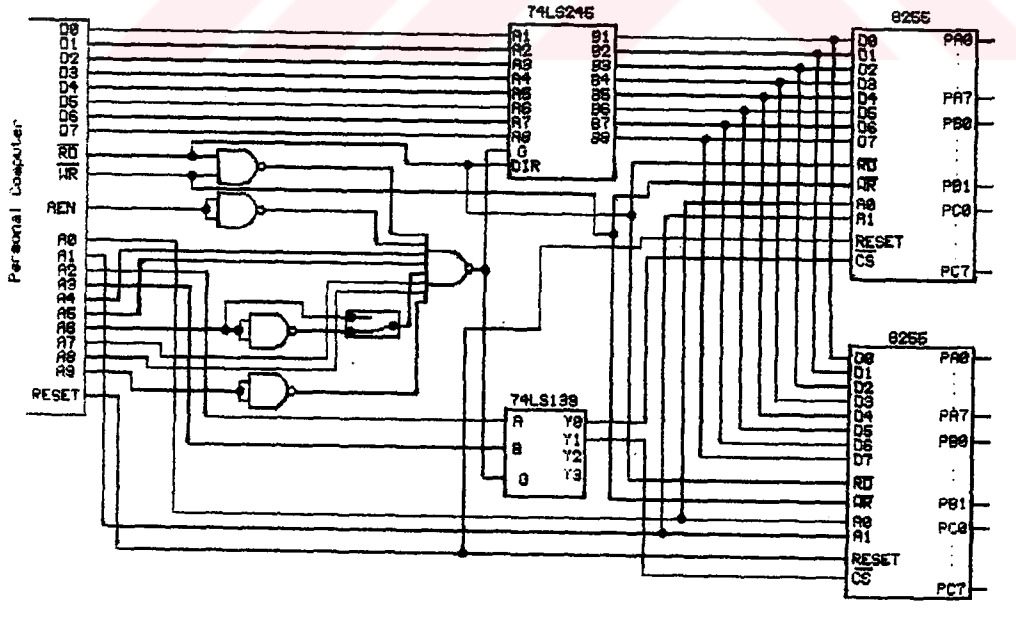
Tablo 5.4. I/O kartında kullanılan 8255' lerin PC bellek haritasındaki yerleşimleri

Adres	Anlamı
01B0(Hex)	1 nolu 8255 A portu
01B1(Hex)	1 nolu 8255 B portu
01B2(Hex)	1 nolu 8255 C portu
01B3(Hex)	1 nolu 8255 kumanda sözcüğü
01B4(Hex)	2 nolu 8255 A portu
01B5(Hex)	2 nolu 8255 B portu
01B6(Hex)	2 nolu 8255 C portu
01B7(Hex)	2 nolu 8255 kumanda sözcüğü

5.4. Giriş/Çıkış Kartının Çalışması

Giriş/Çıkış kartının devre şeması Şekil 5.2' de verilmiştir. PC' ye ait data bus' ın düşük anlamlı 8 biti

(D0-D7), çift yönlü bir bus entegresi olan 74LS245' den geçirilerek 8255' lerin data hatlarına bağlanmıştır. Çift yönlü bus devresinin yönü PC' nin RD sinyali ile belirlenir. Bu sinyal, `1' olduğunda data bus PC' den 8255' e, `0' olduğunda ise 8255' den PC' ye doğru bilgi aktarır. Çift yönlü bus devresinin bilgi aktarımı yapabilmesi için, adres decoder çıkışı olan $\overline{CE}$ bitinin `0' olması gerekmektedir. Bu durum aynı zamanda, 8255' lerin $\overline{CS}$ işaretini üreten bloğun da aktif olmasını sağlar. $\overline{CE}$ ve $\overline{CS}$ işaretleri, adres busta (1B0)Hex-(1B7)Hex arasında bir adres olduğu zaman üretilir. Adres decode işleminde $\overline{RD}$, $\overline{WR}$, RESET, AEN işaretleriyle, adres busın A4-A9 bitleri kullanılmıştır. AEN, PC' nin adres busindaki adresin geçerli olduğunu göstermektedir.



Şekil 5.2. Giriş/Çıkış kartı devre şeması

A2-A3 bitleri $\overline{CS1}$ ve $\overline{CS2}$ işaretlerini üreten 74LS139 decoder entegresine gider. A0-A1 bitleri ise 8255' lerin A0 ve A1 bacaklarına bağlıdır. Bu en az anlamlı 2 bit, (1B0)Hex-(1B3)Hex arasında birinci 8255' in, PortA, PortB, PortC ve kumanda sözcükleri arasında hareket etmeyi sağlar. (1B4)Hex-(1B7)Hex adresleri arasında ise aynı durum ikinci 8255 için geçerlidir.

6. SİSTEMİN DİĞER ELEMANLARI

Bu bölümde, giriş/çıkış kartının dışında sistemin mekanik yapısı ve sistemde kullanılan diğer devre blokları incelenecektir.

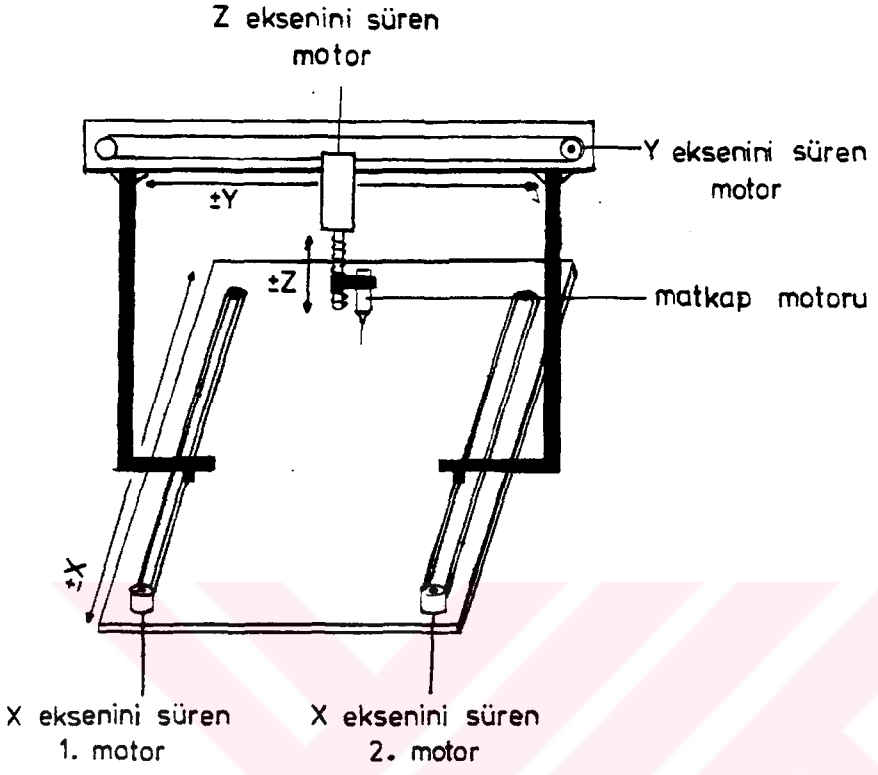
6.1. Mekanik Yapı

Matkap tezgahı, mekanik olarak, 3 boyutta hareket edecek olan bir matkap motorunun tüm hareketlerini yapacak şekilde dizayn edilmiştir. Şekil 6.1' de mekanik yapının basit bir çizimi gösterilmiştir. Şekilden de görüleceği gibi, matkap motoru köprü tipi bir yapı üzerinde bulunmaktadır. Bu köprünün iki ayağı, X eksenine ait 2 adet step motorun hareket ettirdiği hareketli parçalara bağlıdır. Bu iki motor birbirlerinin ayni olup aynı anda aynı faz bilgisiyle paralel olarak uyarılmaktadır.

Z eksenini süren step motor ise, köprü üzerinde, Y eksenini süren step motorun hareket ettirdiği parçaya bağlıdır. Matkap motoru ise, Z eksenini hareket ettiren motorun eksenine bağlı sonsuz vida üzerinde hareket eden parçaya bağlıdır.

Mekanik yapı bu haliyle, daha önceki bölümlerde de belirtildiği gibi sistemin ihtiyacını karşılayacak minimum

şekilde tasarlanmıştır ve geliştirilmeye açıktır.



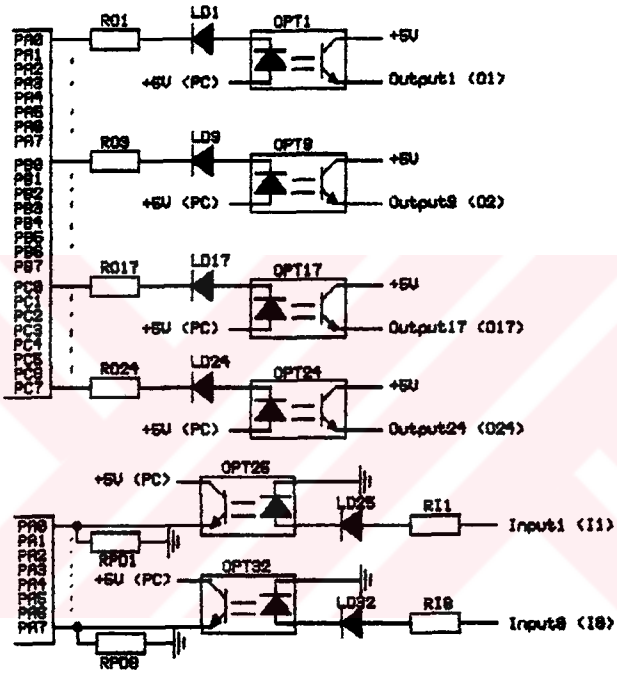
Şekil 6.1. Matkap tezgahının mekanik yapısı

6.2. Opto-Kuplör Kartı

Giriş/çıkış kartını ve dolayısıyla PC' yi, yüksek akım çeken step motor sürücü devrelerinden yalıtmak için opto-kuplörler kullanılmıştır. İlgili devre şeması Şekil 6.2' de görülmektedir. Giriş işlemlerinin sayısı 8' den az olduğu için opto-kuplör kartının giriş bölümü, maksimum 8 giriş yapılabilecek şekilde tasarlanmıştır.

Giriş/çıkış kartı üzerindeki 1 nolu 8255 çıkış, 2 nolu 8255 ise giriş işlemlerine ayrılmıştır. Giriş hatlarındaki

opto-kuplörlerin kollektör bacakları doğrudan PC' nin +5V hattına bağlıdır. Emitör bacakları ise giriş/çıkış kartına giriş olarak gider. Giriş/çıkış kartının giriş hatları `1' aktifdir. Giriş/çıkış kartına giden hatlar ayrıca birer 330 Ohm' luk pull-down dirençleriyle PC' nin toprak hattına bağlıdır.



Şekil 6.2. Opto-kuplör kartının devre şeması

Çıkış hattında kullanılan opto-kuplörlerin bağlantıları biraz daha değişiktir. Giriş/çıkış kartının çıkış hatları, bir direnç ve çıkış durumlarını gösteren bir led üzerinden opto-kuplör ledlerinin katot bacaklarına giderler. Ledlerin anot bacakları ise PC' nin +5V hattına bağlıdır. Giriş/çıkış kartının çıkış gerilimi lojik 0 olduğunda her iki led de yanar ve opto-kuplör aktif hale geçerek ilgili çıkışa izin

verir. Çıkış lojik 1 olduğunda hattın her iki ucunda da +5V olacağından ledler sönük kalır.

6.3. Step Motor Sürücü Arabirimi

Sistemin hareketini sağlamak için kullanılan 3 adet çiftkutuplu ve 1 adet hibrid step motor için Şekil 6.3' deki devre kullanılmıştır.

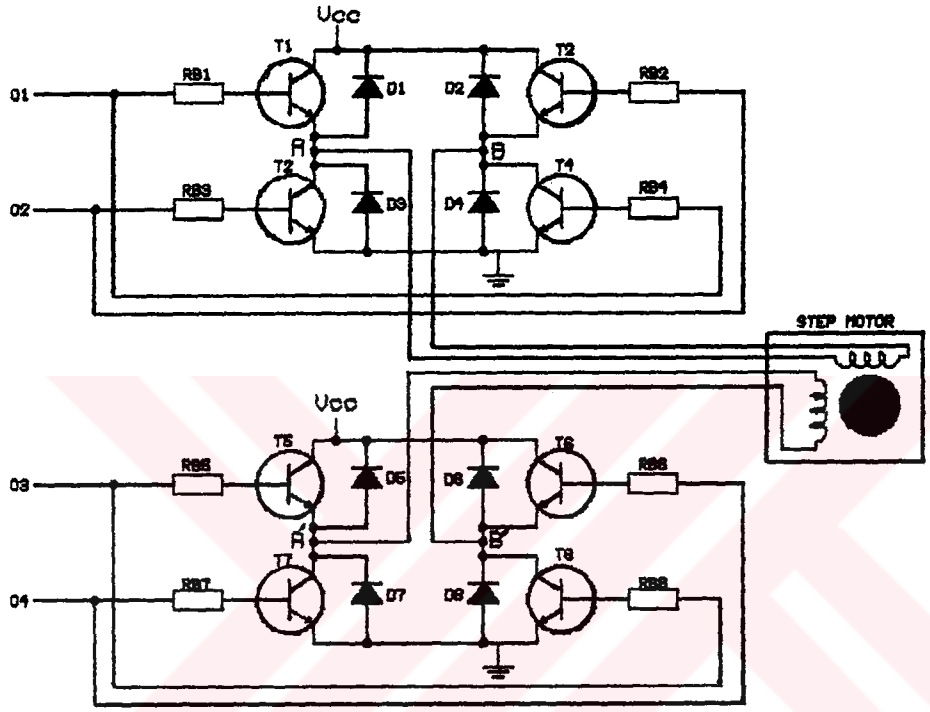
Step motor arabirimine başlamadan önce, her motor için kullanılan 4 adet sıralı motor adım bilgilerinin, nasıl olması gerektiğini görelim. Tablo 6.1 bu sırayı göstermektedir. (ETI)

Tablo 6.1 Step motor faz uyarma sıralaması

Zaman -->	1	2	3	4
L1a	+	+	-	-
L1b	-	-	+	+
L2a	-	+	+	-
L2b	+	-	-	+

Yukarıdaki tablodan da görüleceği gibi bir uyarma değişiminin bir anında sadece bir sarıma ait akımın yönü değişmektedir. Gelen faz bilgisi motor rotorunda 1 adımlık dönmeye yol açar. Motoru ters yönde döndürmek için bu sıra

ters yönde motora gönderilmelidir. Yani uyarma sırası 4-3-2-1-4... şeklinde olmalıdır.



Şekil 6.3. Step motor sürücü devresinin devre şeması

6.3.1 Sürücü Devrenin Çalışma Prensibi

Sürücü devreden istenen özellik, motor sargı uçlarındaki gerilimin polaritesinin, zaman içinde değiştirilerek sargılardan geçen akımın yönünü değiştirmektir. Bunu sağlamanın tek yolu her motor sargısı için 4 adet anahtarlama transistörü kullanmaktır. Böylece bir motor için 8 adet transistör gerekmektedir.

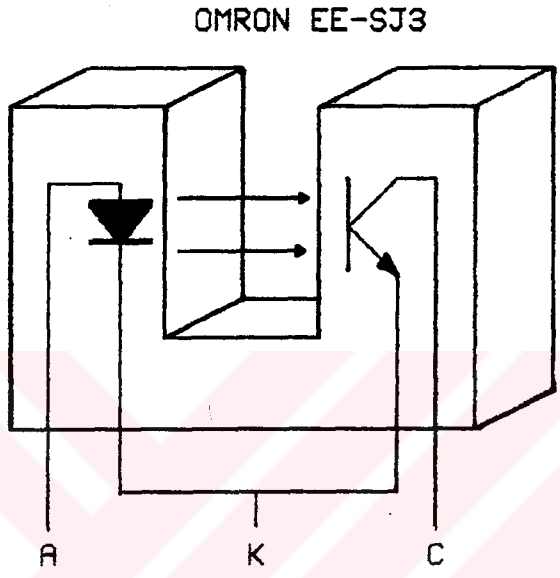
Şekil 6.3' de bir motor için gerekli sürücü devrenin yapısı görülmektedir. Tablo 6.1' deki sıralamayı sağlamak için O1 ile O2, aynı şekilde O3 ile O4 çıkışları birbirlerinin evriği konumda olmalıdır. O1 ile O4 lojik 1 ve O2 ile O3 lojik 0 iken tablodaki 1 numaralı uyarma bilgisi gelmiş olur. Bu nedenle T1,T4,T6 ve T7 transistörleri doyumda, T2,T3,T5 ve T8 transistörleri kesimdedir. Bunun sonucunda, devredeki A ve B' noktalarında yaklaşık besleme gerilimi, A' ve B noktalarında ise 0 Volt görülür.

Uyarma sıralayıcı devre bundan sonra, Tablo 6.1' deki 2 nolu adımı gerçekleştirmek üzere O1 ve O3 çıkışlarını lojik 1, O2 ve O4 çıkışlarını lojik 0 yapar. Burada uyarma sıralayıcı devre PC' nin kendisidir. Bu durumda doyumda olan transistörler T1,T4,T5 ve T8, kesimde olan transistörler ise T2,T3,T6 ve T7 dir. Bu anda faz uyarımları sırasıyla +,−,+,−' dir. Bu şekilde 4 faz uyarma bilgisi PC aracılığıyla sürekli olarak sıralanarak sürekli dönme elde edilir.

6.4. Giriş Bilgileri ve Sistem Tarafından Algılanması

Matkap tezgahının, X ve Y eksenlerinin referans noktalarını belirtmek için, X=0 ve Y=0 değerlerine karşı düşen noktalara birer yarıklı tip opto-kuplör konulmuştur. Bu tip opto-kuplörler, pozisyon algilama islemlerinde kullanılmak üzere geliştirilmişlerdir. Özellikle printer,

fotokopi makinası gibi endüstriyel ortamlarda oldukça sık kullanılırlar. Sistemde kullanılan opto-kuplörler OMRON EE-SJ3 tipi olup Şekil 6.4' te kesiti görülmektedir.

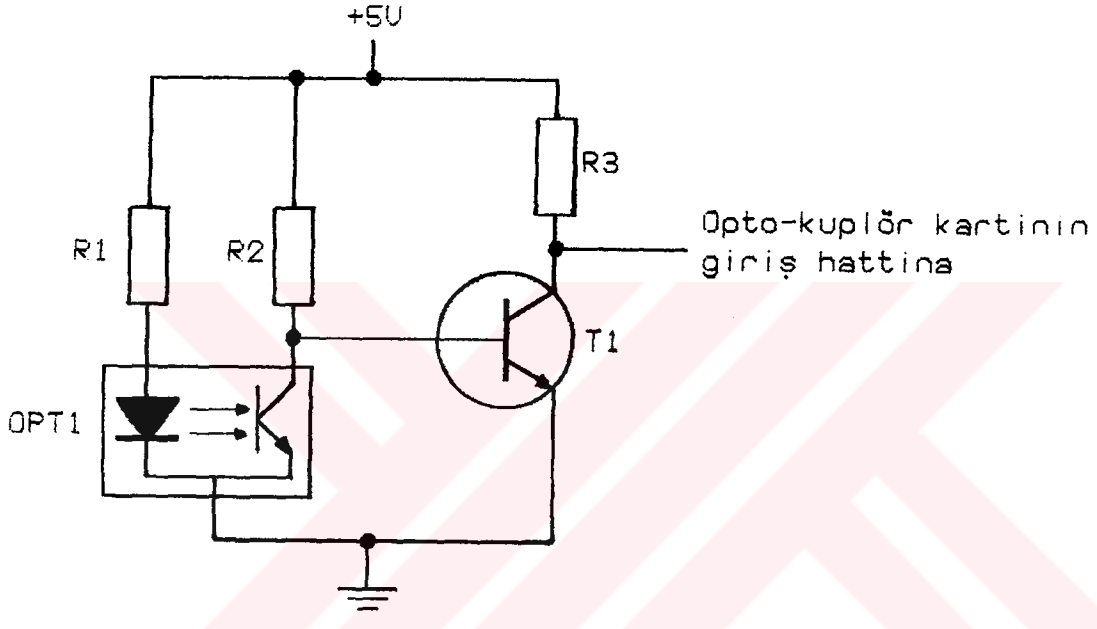


Şekil 6.4. Referans noktalarında kullanılan yarıklı tip opto-kuplörün yapısı

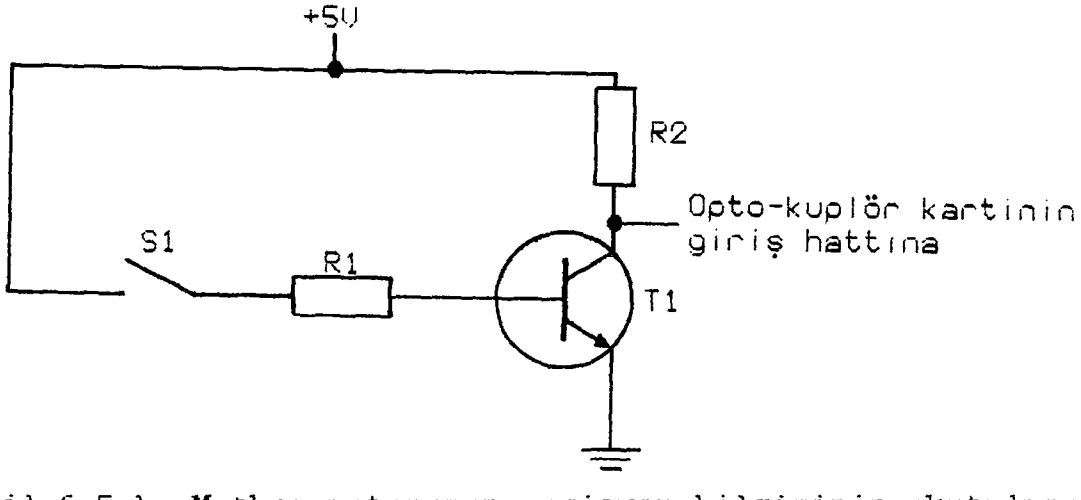
Şekil 6.4' teki opto kuplörden, bir adet X ve bir adet Y ekseninde olmak üzere 2 adet kullanılmıştır. Eksenlere ait hareketli parçaların altlarında bulunan pimlerin, led ile transistör arasına girmesiyle transistör kesime gider. Giriş çıkış kartı tarafından, ilgili girişin PC' ye aktarılmasıyla giriş algılanır. Bu opto-kuplörlerın kullanıldığı devre şeması Şekil 6.5.a' da görülmektedir.

Bir diğer giriş ise, matkap motorunun en alt noktaya

indiğini anlamada kullanılır. Matkap motoru en alt noktaya indiğinde kontak tipi bir anahtar kapanır. Bu durumda düşük güçlü bir transistörle yapılan basit bir anahtarlama katından, opto-kuplör kartına bir giriş bilgisi iletilerek PC' nin bu girişi okuması sağlanır. (Şekil 6.5.b)



Şekil 6.5.a Referans noktası bilgilerinin okutulması



Şekil 6.5.b Matkap motorunun pozisyon bilgisinin okutulması

Giriş sayısi bu aşamada üçtür. Fakat sistem geliştirilmeye açık olduğundan opto-kuplör kartı 8 girişe göre tasarlanmıştır.

Böylece sistemin tüm elemanları, ayrı ayri incelendi. Bundan sonraki bölümde sistem yazılımı ayrıntılı olarak incelenecektir.

7. SİSTEM YAZILIMI

Sistem yazılımı Turbo Pascal v7.0 kullanılarak yazılmıştır ve iki ana fonksiyonu gerçekleştirmektedir:

1) Delik noktalarının dosyaya aktarılması,
2) Dosyadan yüklenen herhangi bir kartın, PC kontrollü olarak delinmesi.

Genel olarak bu 3 fonksiyonun gerçekleştiren program, her çalıştırıldığında matkap motoru kendiliğinden X ve Y koordinatlarının sıfır olduğu referans noktasına gider. Bu nokta delinecek olan kartın da köşe noktası olarak kabul edilir. Burada dikkat edilmesi gereken bir husus, delinecek tüm kartların ilk delik noktalarının aynı X ve Y noktalarına sahip olmasıdır. Bu nedenle kartların teorik olarak birbirlerinin aynı olması gerektirmektedir. Bu sağlanmadığı takdirde delik noktalarındaki kaymalar kaçınılmazdır.

Program ilk çalıştırıldığında ekrana bir tanıtım sayfası çıkar. Bu sayfa herhangi bir tuşa basılarak geçildiğinde, ana çalışma ekranına geçilir. Bu sayfada, "Dosya Adı", "Delinecek Toplam Delik Sayısı", "Kalan Delik Sayısı", X ve Y koordinatlarını gösteren pencereler vardır. X ve Y koordinatları, X ve Y eksenlerini süren motorların adım sayılarını göstermektedir. Bu pencerelerin altında bir menü çubuğu bulunmaktadır. Gerektiğinde sistem uyarılarını

gösteren ve her zaman görülmeyen bir mesaj penceresi de bu ekrandadır.

Menü çubuğunda, program ilk çalıştırıldığında aşağıdaki gibi bir menü görülür.

F1:Yeni Dosya Oluşturma F2:Dosya Yükleme F3:Dosya Silme

7.1. Yeni Dosya Oluşturma

<F1> tuşuna basılarak bu menüye girildiğinde ilk olarak matkap motoru X ve Y koordinatlarının 0 olduğu referans noktasina gider ve mesaj penceresinde "Dosya Adını Giriniz" şeklinde bir mesaj çıkar. En fazla 8 karakterlik dosya adı girilip <ENTER> tuşuna basıldıktan sonra, program daha önceden oluşturulmuş kart dosyalarını tarayarak bu isimde bir dosya olup olmadığına bakar. Varsa "Bu İsimde Bir Dosya Mevcut! Herhangi Bir Tuşa Basınız." mesajı verir ve bir tuşa basıldığında bu bölümden çıkılır. Bu menüye tekrar girebilmek için yeniden <F1> tuşuna basılmalıdır.

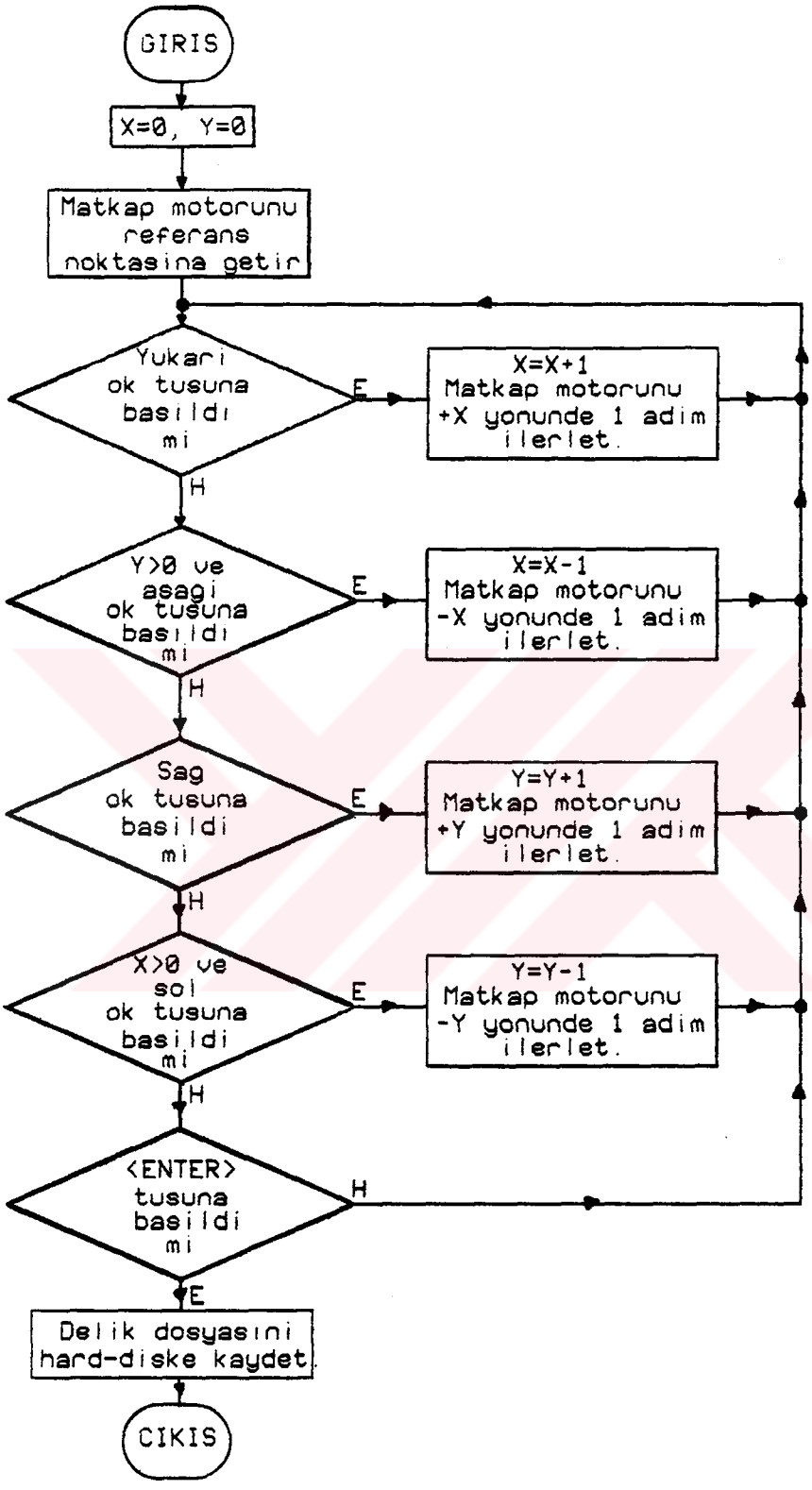
Girilen isimde bir dosya yoksa, mesaj pencerisinde, "Kartı Tezgah Üzerine Yerleştirerek <ENTER> Tuşuna Basınız" mesajı çıkar. Kullanıcı kartı yerleştirip <ENTER> tuşuna bastığında "Delinecek Toplam Delik Sayısı", X ve Y koordinatlarına ait pencereler aktif hale geçer ve bu

pencerelerde 0 sayısı görülür. Bu andan itibaren sağ, sol, yukarı ve aşağı ok tuşlarına basılarak matkap motoru tabla üzerinde, tuşlara her basışta bir step motor adımı kadar hareket eder. Yukarı ok tuşu +X, aşağı ok tuşu -X, sağ ok tuşu +Y, sol ok tuşu -Y yönlerinde birer adımlık hareket sağlarlar. İlgili delik noktasına gelindiğinde <ENTER> tuşuna basılarak bu koordinatların dosyaya atılması sağlanır. X veya Y sıfırken -X veya -Y işlemleri geçersiz kılınmıştır. Daha sonra diğer delik noktası için de aynı işlemler tekrarlanır.

Örnek olarak, (5,4),(9,4),(5,11) noktalarını dosyaya atalım. İlk olarak ekranda, koordinat pencerelerinde (0,0) görüldüğünden, ilk delik noktası için 5 defa <yukarı ok> ve 4 defa <sağ ok> tuşlarına basılıp ilgili delik noktasına gelinir ve <ENTER> tuşuna basılır. Ok tuşlarına her basışta, o koordinata ait değer ekranda değişmektedir. İlk delik noktasına gelindiğinde koordinat pencerelerinde (5,4) sayısı görülmektedir. Bundan sonraki delik noktasına gidebilmek için <yukarı ok> tuşuna 4 defa basıp tekrar <ENTER> tuşuna basılarak bu koordinatlar da dosyaya atılır. Üçüncü noktaya ulaşmak için 4 defa <aşağı ok> ve 7 kere de <sağ ok> tuşuna basılır.

"Yeni Dosya Oluşturma" menüsüne girildiği anda menü çubuğundaki menü değişir ve aşağıdaki şekli alır:

<Shift - F1> - Dosyaya Kaydetme	< ESC > - İptal



Şekil 7.1. Yeni dosya oluşturma bölümüne ilişkin blok diyagramı

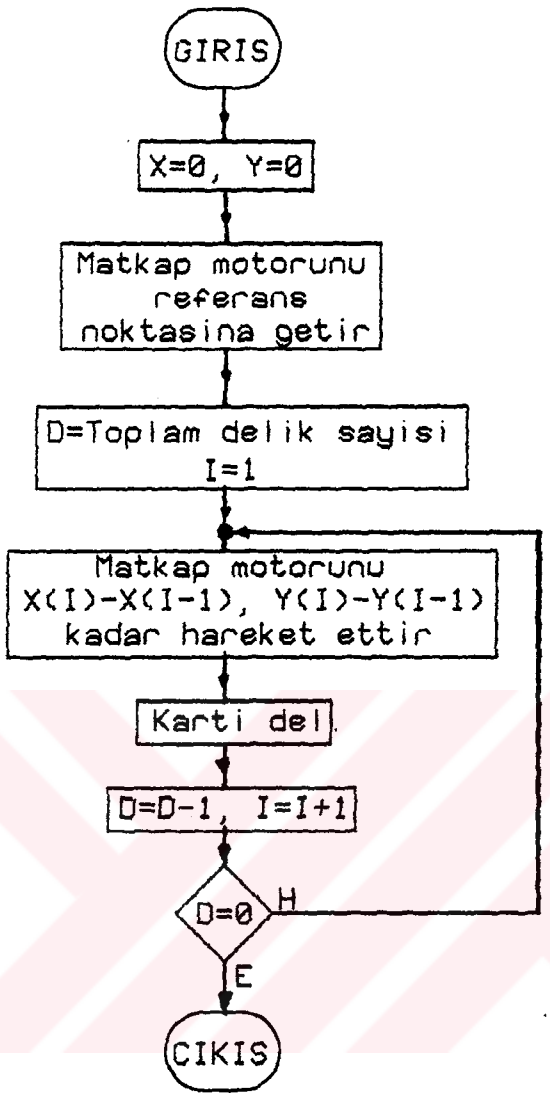
Bütün delik noktaları dosyaya atıldıktan sonra, <Shift-F1> tuşlarına basılarak, dosyaya verilen isim altında hard diske kaydedilebilir veya herhangi bir anda <ESC> tuşuna basılarak, yeni dosya oluşturma menüsünden çıkılır.

Yazılımın bu bölümüne ait akış diyagramı Şekil 7.1' de görülmektedir.

7.2. Dosya Yükleme ve Kart Delme

<F2> tuşuna basılarak bu menüye girildiğinde, yeni dosya oluşturma bölümünde olduğu gibi, matkap motoru referans noktasına gider. Daha sonra 8 karaktere kadar olan dosya adı girilir ve <ENTER> tuşuna basılır. Program hard diskte kayıtlı olan dosyaları tarar. Dosya bulunamazsa, "Bu isimde bir dosya yok!! Herhangi bir tuşa basınız." mesajı görülür. Kart yerleştirilip <ENTER> tuşuna basıldıktan sonra matkap motoru ilk delik noktası üzerine ok tuşları ile götürülür ve tekrar <ENTER> tuşuna basılır. Matkap motoru Z ekseninde inerek delme işlemini yapar. Bu anda "Kalan Delik Sayısı" penceresindeki değer bir azalır. Her delik için, o deliğe ait X ve Y koordinatları da gösterilir.

Bir delik delindiği zaman, bir sonraki deliğin koordinatlarından, o anda aktif olan deliğin koordinatları çıkarılır ve buna göre X ve Y eksenini süren motorlar ±X veya ±Y koordinatlarında bu fark kadar adım atar.



Şekil 7.2. Kart delme bölümünün blok diyagramı

Tüm delikler delindiğinde, mesaj penceresinde "Devam edecek misiniz? <E> - Evet <H> - Hayır" sorusu çıkar. Bu soruya <H> tuşu ile cevap verildiği zaman açılış ekranına geri dönülür. <E> tuşuna basılarak prosedür, aynı dosya için baştan itibaren tekrar edilir.

Şekil 7.2 kart delme prosedürünün blok diyagramını

göstermektedir .

7.3. Dosya Silme

Bu fonksiyonun matkap tezgahı üzerinde bir etkisi yoktur. <F3> tuşuyla bu menüye girildiğinde, mesaj ekranında "Dosya Adını Giriniz" cümlesi görülür. En fazla 8 karakterlik dosya adı girilerek, <ENTER> tuşuna basıldığında program hard diskteki dosyaları tarar. Verilen isimde bir dosya varsa mesaj penceresinde "Emin misiniz? <E> - Evet <H> - Hayır" sorusu görülür. <E> tuşuna basılarak dosya silinir veya <H> tuşuna basılarak işlem iptal edilir.

SONUÇLAR VE ÖNERİLER

Baskılı devre matkap tezgahının 3 temel bileşeni vardır: Mekanik, yazılım ve donanım. Tüm bu sistemin esas elemanları step motorlardır. Zaten sistem, step motorlara bir uygulama örneği oluşturmak amacıyla gelistirilmiştir. Kullanılan step motorlar, açık çevrim kontrolü ile çalıştırılmaktadır. Bu kontrol türü step motorların başlıca avantajlarından birini oluşturmaktadır. Böylece diğer motorlar için hemen hemen şart olan kapalı çevrim kontrol sistemleri nedeniyle artan maliyet azalmaktadır. Elektronik kontrol devresi ve yük koşulları motor için uygun olduğu takdirde, açık çevrim kontrolü sistem için oldukça uygun bir yöntemdir. Matkap tezgahının bu şartleri sağlaması nedeniyle bu tip bir kontrol uygun görülmüştür.

Sistem mekanik olarak hazırlanırken ortaya bazı problemler çıkmıştır. Bu problemlerin en önemlisi, hafif olması nedeniyle kullanılan el tipi küçük matkap motoruna takılan matkap ucunun z ekseni üzerinde sapmadan dönmesi gerekirken çok küçük salınımlarla konik bir hareket yaptığı gözlenmistir. Fakat bu salınım, denenen 3 matkap motorunda da önlenememıştir. Bu nedenle delme işlemlerinde delik noktalarından 1.5 mm' ye varan sapmalar gözlenmiştir. Bu sorun, bu tür makinalarda ve özellikle CNC tezgahlarında kullanılan yüksek frekans motorlarının kullanılmasıyla engellenebilir.

Bu sorunlar gözardı edildiğinde, sistemin yazılım ve donanım kısmıyla mekanik kısmı birbirleriyle uyum içinde çalışmaktadır. Bu tip makinalar için piyasada hazır olarak satılan mekanik yapılar kullanılarak değişik yapılarda matkap tezgahları hazırlanabilir. Bütün bu sistemler için Ek A' da verilen yazılım ve tasarlanan devreler, küçük değişiklerle bu yeni sistemlere adape edilebilecek esnekliktedir.

KAYNAKLAR

1- Acarnley,P.P., 1984. Stepping Motors: a guide to modern theory and practice. Peter Peregrinus Ltd., London, U.K.. 1-21,134-152.

2- Çölkesen,R., Şubat 1990. Bir çizici için sürücü, denetim, arabirim tasarımı ve gerekli yazılımların geliştirilmesi. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul. 17-30.

3- Electronics Today International, Volume 22, No:2, February 1993. Argus Specialist Publication, U.K.. 50-52.

4- Electronics Today International, Volume 22, No:3, March 1993. Argus Specialist Publication, U.K.. 34-37.

5- Turbo Pascal v5.0 - User's Guide, 1988. Borland International, U.S.A.

6- Turbo Pascal v5.0 - Reference Guide, 1988. Borland International, U.S.A.

7- Microprocessors and Peripherals Data Book, 1990. OKI, Japan.

EK A: MATKAP TEZGAHI YAZILIMI

```
*****************************************************************
**                                                             **
**                                                             **
**                  BİLGİSAYAR KONTROLLU,                      **
**                     PROGRAMLANABİLİR                        **
**               BASKILI DEVRE MATKAP TEZGAHI                  **
**                                                             **
**               Elektronik Müh. Altuğ ORHAN                   **
**                                                             **
**                       Şubat, 1994                           **
**                                                             **
*****************************************************************


uses graph,dos,crt;

var
g:pointer;
buyukluk:word;
xxx,yyy:string;
sira,deliksay,TK, gd,gm:integer;
FLNM:ARRAY[1..8] OF char;
tus:char;
out,loop,akis,hang,cik,cik2,bas,funckey:boolean;
dxk1,dxk2,i:integer;
koordinatx:array [1..200] of integer;
koordinaty:array [1..200] of integer;
koordinatx2:array [1..200] of integer;
koordinaty2:array [1..200] of integer;
label controller,baslangic;


PROCEDURE REFERANS;

var
inp:byte;
label as,bs;

BEGIN
inp:=port[$1b4];
if (inp and 4)<>0 then
    repeat
    port[$1b0]:=246;delay(9);
    port[$1b0]:=250;delay(9);
    port[$1b0]:=249;delay(9);
    port[$1b0]:=245;delay(9);
    port[$1b0]:=255;delay(9);
    inp:=port[$1b4];
```

```
    until (inp and 4)=0;
inp:=port[$1b4];
if (inp and 2)<>0 then
    repeat
    port[$1b0]:=111;delay(9);
    port[$1b0]:=175;delay(9);
    port[$1b0]:=159;delay(9);
    port[$1b0]:=95;delay(9);
    port[$1b0]:=255;delay(9);
    inp:=port[$1b4];
    until (inp and 2)=0;
END;


PROCEDURE DIZ;

type
pcb=record
pcbiz:char;
pcbname:string[8];
delsay:integer;
pcbkoordx:array[1..200] of integer;
pcbkoordy:array[1..200] of integer;
END;

var
koord:file of pcb;
krd:pcb;
thy,crp,qp,j,kn,sayim2:integer;
ret,karar:char;
qps,over:string;

BEGIN
buyukluk:=imagesize(0,130,639,255);
getmem(g,buyukluk);
getimage(0,130,639,255,G^);
setfillstyle(0,0);
bar(0,130,639,255);
settextstyle(0,0,1);
rectangle(0,135,638,250);
assign(koord,'koordinat.dat');
reset(koord);
qp:=1;
repeat
    if qp<11 then crp:=0
         else if qp<21 then crp:=1
         else if qp<31 then crp:=2
         else if qp<51 then crp:=4;
    seek(koord,qp-1);
    read(koord,krd);
    str(qp,qps);
    if length(qps)=1 then qps:=' '+qps;
    thy:=qp-crp*10;
    if krd.pcbiz='*' then
```

```
    outtextxy(10+130*crp,130+thy*10,qps+ ' '+krd.pcbname);
    qp:=qp+1;
    until eof(koord);
close(koord);
rectangle(0,260,639,300);
setfillstyle(1,0);
settextstyle(0,0,1);
outtextxy(40,280,'Devam Etmek Icin <ENTER> Tusuna
Basiniz!!');
repeat
    ret:=readkey;
until ret=#13;
setcolor(4);
rectangle(0,150,638,250);
setfillstyle(0,0);
bar(3,153,635,162) ;
putimage(0,130,g^,normalput);
freemem(g,buyukluk);
END;


PROCEDURE ARASTIR;

type
pcb=record
pcbiz:char;
pcbname:string[8];
delsay:integer;
pcbkoordx:array[1..200] of integer;
pcbkoordy:array[1..200] of integer;
END;

var
koord:file of pcb;
krd:pcb;
sayimm:integer;

BEGIN
assign(koord,'koordinat.dat');
reset(koord);
sayimm:=1;
repeat
    seek(koord,sayimm-1);
    sayimm:=sayimm+1;
    read(koord,krd);
    if (krd.pcbname=flnm) and (krd.pcbiz='*') then
          BEGIN
          if hang=false then
              BEGIN
              bar(33,263,548,298);
              settextstyle(0,0,1);
              outtextxy(40,270,'Bu isimde bir dosya mevcut!!
              Herhangi bir tusa basiniz');
              cik:=true;
```

```
                delay(1000);
                close(koord);
                repeat
                until keypressed;
                exit;
                END
            else
            if hang=true then
                BEGIN
                cik:=true;
                close(koord);
                exit;
                END;
            END;
until eof(koord);
if hang=true then
     BEGIN
     bar(33,263,548,298);
     settextstyle(0,0,1);
     outtextxy(40,270,'Bu isimde bir dosya yok!! Herhangi bir
     tusa basiniz');
     cik:=true;
     delay(1000);
     close(koord);
     out:=true;
     repeat
     until keypressed;
     END;
END;


PROCEDURE KAYIT;

type
pcb=record
pcbiz:char;
pcbname:string[8];
delsay:integer;
pcbkoordx:array[1..200] of integer;
pcbkoordy:array[1..200] of integer;
END;

var
koord:file of pcb;
krd:pcb;
delik,kn,j,sayim:integer;
tus3:char;

BEGIN
delik:=i-1;
setfillstyle(1,0);
settextstyle(0,0,1);
bar(32,447,547,460);
assign(koord,'koordinat.dat');
```

```
reset(koord);
sayim:=1;
repeat
     seek(koord,sayim-1);
     sayim:=sayim+1;
     read(koord,krd);
until krd.pcbiz<>'*';
krd.pcbiz:='*';
krd.pcbname:=flnm;
krd.delsay:=delik;
for j:=0 to 200 do
     BEGIN
     krd.pcbkoordx[j]:=koordinatx[j];
     krd.pcbkoordy[j]:=koordinaty[j];
     END;
seek(koord,sayim-2);
write(koord,krd);
close(koord);
bas:=true;
END;


PROCEDURE YUKCIK;

var
stepz2:integer;
label ddx;

BEGIN
stepz2:=170;
ddx:
port[$1b1]:=246;delay(90);
stepz2:=stepz2-1;
if stepz2=0 then
     BEGIN
     port[$1b1]:=255;
     exit;
     END;
port[$1b1]:=250;delay(90);
stepz2:=stepz2-1;
if stepz2=0 then
     BEGIN
     port[$1b1]:=255;
     exit;
     END;
port[$1b1]:=249;delay(90);
stepz2:=stepz2-1;
if stepz2=0 then
     BEGIN
     port[$1b1]:=255;
     exit;
     END;
port[$1b1]:=245;delay(90);
stepz2:=stepz2-1;
```

```
if stepz2=0 then
    BEGIN
    port[$1b1]:=255;
    exit;
    END;
goto ddx;
END;


PROCEDURE DEL;

var
break:boolean;
inp:word;
stepz:integer;
label ddd;

PROCEDURE YUKARI;

label ddx;

BEGIN
stepz:=45;
ddx:
port[$1b1]:=230;delay(130);
stepz:=stepz-1;
if stepz=0 then
    BEGIN
    break:=true;
    port[$1b1]:=255;
    exit;
    END;
port[$1b1]:=234;delay(130);
stepz:=stepz-1;
if stepz=0 then
    BEGIN
    break:=true;
    port[$1b1]:=255;
    exit;
    END;
port[$1b1]:=233;delay(130);
stepz:=stepz-1;
if stepz=0 then
    BEGIN
    break:=true;
    port[$1b1]:=255;
    exit;
    END;
port[$1b1]:=229;delay(130);
stepz:=stepz-1;
if stepz=0 then
    BEGIN
    break:=true;
    port[$1b1]:=255;
```

```
    exit;
    END;
goto ddx;
END;


PROCEDURE KONTROL;

BEGIN
inp:=port[$1b4];
stepz:=stepz+1;
if (inp AND 1)=1 then
    BEGIN
    port[$1b1]:=255;
    yukari;
    END
else
    exit;
END;

BEGIN
port[$1b0]:=255;
break:=false;
STEPZ:=0;
DDD:
port[$1b1]:=229;delay(130);
kontrol;
if break=true then exit;
port[$1b1]:=233;delay(130);
kontrol;
if break=true then exit;
port[$1b1]:=234;delay(130);
kontrol;
if break=true then exit;
port[$1b1]:=230;delay(130);
kontrol;
if break=true then exit;
goto ddd;
END;


PROCEDURE ADSOR;

var
qw:integer;
ad:char;

BEGIN
cik:=false;
bar(0,260,638,300);
setcolor(15);
rectangle(0,260,638,300);
setfillstyle(1,0);
settextstyle(0,0,1);
```

```
outtextxy(40,280,'Dosya Adini Giriniz');
settextstyle(0,0,2);
for qw:=1 to 8 do
    BEGIN
    repeat
        repeat
        until keypressed;
    ad:=readkey;
    until (UPCASE(ad) IN (['A'..'Z'])) or (upcase(ad) in
    (['0'..'9'])) or (upcase(ad)=' ') or (ad=#13) or (ad=#8)
    or (ad=#27);
    if ad=#27 then
        BEGIN
        cik:=true;
        cik2:=true;
        exit;
        END
    else
    if ad<>#13 then
        BEGIN
        flnm[qw]:=upcase(AD);
        if qw<8 then outtextxy(250+20*(QW+1),100,'_');
        bar(250+QW*20,107,270+QW*20,118);
        outtextxy(250+20*qw-1,100,flnm[qw]);
        END
    else
        BEGIN
        bar(250+qw*20,107,270+qw*20,118);
        qw:=8;
        if loop=false then arastir;
        END;
    END;
END;


PROCEDURE ILERI(stepcounti:integer;ycontrol:byte);

var
gecik:byte;
countspci:integer;
label il;

BEGIN
if ycontrol=1 then gecik:=4
else
gecik:=13;
port[$1b1]:=255;
countspci:=0;
il:
port[$1b0]:=(245-150*ycontrol);delay(gecik);
countspci:=countspci+1;
if countspci=stepcounti then exit;
port[$1b0]:=(249-90*ycontrol);delay(gecik);
countspci:=countspci+1;
```

```
if countspci=stepcounti then exit;
port[$1b0]:=(250-75*ycontrol);delay(gecik);
countspci:=countspci+1;
if countspci=stepcounti then exit;
port[$1b0]:=(246-135*ycontrol);delay(gecik);
countspci:=countspci+1;
if countspci=stepcounti then exit;
goto il;
END;


PROCEDURE GERI(stepcountg:integer;ycontrol:byte);

var
countspcg:integer;
gecik:byte;
label ger;

BEGIN
if ycontrol=1 then gecik:=4
else
   gecik:=14;
port[$1b1]:=255;
countspcg:=0;
ger:
port[$1b0]:=(246-135*ycontrol);delay(gecik);
countspcg:=countspcg+1;
if countspcg=-stepcountg then exit;
port[$1b0]:=(250-75*ycontrol);delay(gecik);
countspcg:=countspcg+1;
if countspcg=-stepcountg then exit;
port[$1b0]:=(249-90*ycontrol);delay(gecik);
countspcg:=countspcg+1;
if countspcg=-stepcountg then exit;
port[$1b0]:=(245-150*ycontrol);delay(gecik);
countspcg:=countspcg+1;
if countspcg=-stepcountg then exit;
goto ger;
END;


PROCEDURE HESAPLA;

var
counter,adimsayx,adimsayy,kartx,karty,referansx,referansy:int
eger;
sayyaz:string;
karar:char;

BEGIN
counter:=deliksay-1;
referansx:=koordinatx2[sira];
referansy:=koordinaty2[sira];
for sira:=2 to deliksay do
```

```
BEGIN
     settextstyle(0,0,9);
     bar(05,355,311,427);
     str(koordinatx2[sira],xxx);
     outtextxy(18,359,xxx);
     bar(319,355,632,427);
     str(koordinaty2[sira],yyy);
     outtextxy(319,359,yyy);
     kartx:=koordinatx2[sira];
     karty:=koordinaty2[sira];
     adimsayx:=kartx-referansx;
     adimsayy:=karty-referansy;
     referansx:=kartx;
     referansy:=karty;
     if adimsayx>0 then ileri(adimsayx,0)
     else
         if adimsayx<0 then geri(adimsayx,0);
     if adimsayy>0 then ileri(adimsayy,1)
     else
         if adimsayy<0 then geri(adimsayy,1);
     del;
     counter:=counter-1;
     if counter<0 then counter:=0;
     bar(319,173,632,243);
     str(counter,sayyaz);
     settextstyle(0,0,9);
     outtextxy(336,176,sayyaz);
     END;
END;


PROCEDURE ADIMATFF(xyxy:byte;akw:byte);

BEGIN
port[$1b1]:=255;
case akw of
     0:BEGIN port[$1b0]:=(245-150*xyxy);delay(9);END;
     1:BEGIN port[$1b0]:=(249-90*xyxy);delay(9);END;
     2:BEGIN port[$1b0]:=(250-75*xyxy);delay(9);END;
     3:BEGIN port[$1b0]:=(246-135*xyxy);delay(9);END;
     END;
END;


PROCEDURE ADIMATBW(xyxy:byte;akw:byte);

BEGIN
port[$1b1]:=255;
case akw of
     0:BEGIN port[$1b0]:=(245-150*xyxy);delay(9);END;
     1:BEGIN port[$1b0]:=(249-90*xyxy);delay(9);END;
     2:BEGIN port[$1b0]:=(250-75*xyxy);delay(9);END;
     3:BEGIN port[$1b0]:=(246-135*xyxy);delay(9);END;
```

```
    END;
END;


PROCEDURE INDZ;

var
inp:byte;

BEGIN
inp:=port[$1b4];
if (inp AND 1)<>1 then
   BEGIN
   port[$1b0]:=255;
   port[$1b1]:=245;delay(130);
   port[$1b1]:=249;delay(130);
   port[$1b1]:=250;delay(130);
   port[$1b1]:=246;delay(130);
   port[$1b1]:=255;
   END;
END;


PROCEDURE CIKZ;

var
inp:byte;

BEGIN
port[$1b0]:=255;
port[$1b1]:=246;delay(130);
port[$1b1]:=250;delay(130);
port[$1b1]:=249;delay(130);
port[$1b1]:=245;delay(130);
port[$1b1]:=255;
END;


PROCEDURE SIFIRLAMA;

var
tus2:char;
label a1;

BEGIN
a1:
tus2:=readkey;
if tus2=#13 then
    BEGIN
    bar(33,263,548,298);
    exit;
    END;
if tus2<>#0 then funckey:=false
else
```

```
    BEGIN
    funckey:=true;
    tus2:=readkey;
    if bas=true then exit;
    if tus2=#77 then
        BEGIN
        dxk1:=dxk1+1;
        if dxk1=4 then dxk1:=0;
        adimatff(1,dxk1);
        END;
    if tus2=#75 then
        BEGIN
        dxk1:=dxk1-1;
        if dxk1=(-1) then dxk1:=3;
        adimatbw(1,dxk1);
        END;
    if tus2=#72 then
        BEGIN
        dxk2:=dxk2+1;
        if dxk2=4 then dxk2:=0;
        adimatff(0,dxk2);
        END;
    if tus2=#80 then
        BEGIN
        dxk2:=dxk2-1;
        if dxk2=(-1) then dxk2:=3;
        adimatbw(0,dxk2);
        END;
    if tus2=#81 then indz;
    if tus2=#73 then cikz;
    goto a1;
    END;
END;


PROCEDURE YUKLE;

type
pcb=record
pcbiz:char;
pcbname:string[8];
delsay:integer;
pcbkoordx:array[1..200] of integer;
pcbkoordy:array[1..200] of integer;
END;

var
koord:file of pcb;
krd:pcb;
crp,qp,j,kn,sayim2:integer;
ret,karar:char;
qps,over:string;
BEGIN
hang:=true;
```

```
referans;
settextstyle(0,0,3);
outtextxy(160,60,'Dosya Yukleme');
diz;
adsor;
if (out=true) or (cik2=true) then exit;
bar(32,447,547,459);
bar(33,263,548,298);
assign(koord,'koordinat.dat');
reset(koord);
sayim2:=1;
repeat
    seek(koord,sayim2-1);
    sayim2:=sayim2+1;
    read(koord,krd);
until (krd.pcbname=flnm) and (krd.pcbiz='*') or (eof(koord));
deliksay:=krd.delsay-1;
if deliksay<0 then deliksay:=0;
str(deliksay,over);
settextstyle(0,0,9);
outtextxy(45,173,over);
outtextxy(336,173,over);
for j:=0 to 200 do
    BEGIN
    koordinatx2[j]:=krd.pcbkoordx[j];
    koordinaty2[j]:=krd.pcbkoordy[j];
    END;
close(koord);
settextstyle(0,0,1);
bar(33,263,548,298);
outtextxy(40,280,'Yeni Karti Tezgah Uzerine Sabitleyerek
<ENTER> Tusuna Basiniz');
repeat
    repeat
    until keypressed;
    karar:=readkey;
until KARAR=#13;
bar(33,263,548,298);
outtextxy(40,280,'Matkabi Ilk Delik Noktasina Getirip <ENTER>
Tusuna Basiniz');
sifirlama;
sira:=1;
settextstyle(0,0,9);
bar(05,355,311,427);
str(koordinatx2[sira],xxx);
outtextxy(18,359,xxx);
bar(319,355,632,427);
str(koordinaty2[sira],yyy);
outtextxy(319,359,yyy);
del;
hesapla;
END;
```

```
PROCEDURE SIL;

type
pcb=record
pcbiz:char;
pcbname:string[8];
delsay:integer;
pcbkoordx:array[1..200] of integer;
pcbkoordy:array[1..200] of integer;
END;

var
koord:file of pcb;
krd:pcb;
sayimm:integer;
tusf3:char;
BEGIN
diz;
loop:=true;
SETTEXTSTYLE(0,0,3);
outtextxy(70,60,'   DOSYA SILME  ');
adsor;
assign(koord,'koordinat.dat');
reset(koord);
sayimm:=1;
repeat
    seek(koord,sayimm-1);
    sayimm:=sayimm+1;
    read(koord,krd);
    if krd.pcbname=flnm then
        BEGIN
        bar(33,263,548,298);
        settextstyle(0,0,1);
        outtextxy(40,270,'  Emin misiniz ?      <E> - Evet
        <H> -Hayir');
        tusf3:=readkey;
        if upcase(tusf3)='E' then
            BEGIN
            krd.pcbiz:=' ';
            seek(koord,sayimm-2);
            write(koord,krd);
            close(koord);
            exit;
            END
        else if upcase(tusf3)='H' then
            BEGIN
            close(koord);
            exit;
            END;
        END;
until eof(koord);
END;
```

```
PROCEDURE DOSYA;

var
AD,tus2,cev:char;
QW,y,x:integer;
xyaz,yyaz,ii,yy,xx:string;
label a1;

BEGIN
x:=0;
y:=0;
i:=1;
hang:=false;
referans;
setfillstyle(0,0);
settextstyle(0,0,1);
bar(32,446,547,459);
OUTTEXTXY(50,450,'<Shift-F1> --  Dosyaya Kaydetme
<ESC> --Iptal');
SETTEXTSTYLE(0,0,3);
outtextxy(70,60,'YENI DOSYA OLUSTURMA');
adsor;
if cik=true then exit;
settextstyle(0,0,1);
bar(33,263,548,298);
outtextxy(40,280,'Karti Tezgah Uzerine Sabitleyerek <ENTER>
Tusuna Basiniz');
repeat
    cev:=readkey
until cev=#13;
bar(33,263,548,298);
settextstyle(0,0,9);
outtextxy(18,176,'0000');
outtextxy(18,359,'0000');
outtextxy(3 36,359, '0000');
a1:
tus2:=readkey;
if tus2=#13 then
    BEGIN
    koordinatx[i]:=x;
    koordinaty[i]:=y;
    str(i,ii);
    bar(05,173,311,24 3);
    outtextxy(18,176,ii);
    i:=i+1;
    goto a1;
    END
else if tus2=#8 then
    BEGIN
    koordinatx[i-1]:=0;
    koordinaty[i-1]:=0;
    str(koordinatx[i-2],xyaz);
    bar(05,355,311,427);
    outtextxy(18,359,xyaz);
```

```
     str(koordinaty[i-2],yyaz);
     bar(319,355,632,427);
     outtextxy(336,359,yyaz);
     i:=i-2;
     str(i,ii);
     bar(05,173,311,243);
     outtextxy(18,176,ii);
     i:=i+1;
     goto a1;
     END
else if tus=#27 then exit;
if tus2<>#0 then funckey:=false
else
     BEGIN
     funckey:=true;
     tus2:=readkey;
     if tus2=#84 then kayit;
     if bas=true then exit;
     if tus2=#77 then
          BEGIN
          y:=y+1;
          str(y,yy);
          bar(319,355,632,427);
          outtextxy(336,359,yy);
          dxk1:=dxk1+1;
          if dxk1=4 then dxk1:=0;
          adimatff(1,dxk1);
          END;
     if ((tus2=#75) and (y>0))  then
          BEGIN
          y:=y-1;
          str(y,yy);
          bar(319,355,632,427);
          outtextxy(336,359,yy);
          dxk1:=dxk1-1;
          if dxk1=(-1) then dxk1:=3;
          adimatbw(1,dxk1);
          END;
     if tus2=#72 then
          BEGIN
          x:=x+1;
          str(x,xx);
          bar(05,355,311,427);
          outtextxy(18,359,xx);
          dxk2:=dxk2+1;
          if dxk2=4 then dxk2:=0;
          adimatff(0,dxk2);
          END;
     if ((tus2=#80) and (x>0)) then
          BEGIN
          x:=x-1;
          str(x,xx);
          bar(05,355,311,427);
          outtextxy(18,359,xx);
```

```
        dxk2:=dxk2-1;
        if dxk2=(-1) then dxk2:=3;
        adimatbw(0,dxk2);
        END;
    if tus2=#81 then indz;
    if tus2=#73 then cikz;
    goto a1;
    END;
END;


PROCEDURE DEVAM;

var
karar:char;

BEGIN
settextstyle(0,0,1);
bar(33,263,548,298);
outtextxy(40,280,'Devam edecek misiniz?    <E> - Evet     <H>
-Hayir');
repeat
    repeat
    until keypressed;
    karar:=readkey;
until upcase(KARAR) in ['E','H'];
if upcase(karar)='E' then
    BEGIN
    referans;
    bar(33,263,548,298);
    bar(33,263,548,298);
    outtextxy(40,280,'Yeni Karti Tezgah Uzerine Sabitleyerek
    <ENTER> Tusuna Basiniz');
    repeat
        repeat
        until keypressed;
        karar:=readkey;
    until KARAR=#13;
    bar(33,263,548,298);
    hesapla
    END
else
if upcase(karar)='H' then
    BEGIN
    yukcik;
    akis:=false;
    END;
END;


BEGIN (* Ana Program*)
baslangic:
dxk1:=0;
dxk2:=0;
```

```
port[$1b3]:=128;
port[$1b7]:=146;
port[$1b0]:=255;
port[$1b1]:=255;
cik2:=false;
akis:=false;
loop:=false;
bas:=false;
out:=false;
FOR TK:=1 TO 8 DO FLNM[TK]:=' ';
gd:=detect;
initgraph(gd,gm,'');
if graphresult<>grok then halt(1);
setcolor(4);
bar3d(0,30,610,50,10,topon);
settextstyle(0,0,1);
outtextxy(120,38,'PCB  Drilling  System  By  ALTUG ORHAN');
settextstyle(0,0,2);
setcolor(15);
setcolor(9);
setlinestyle(0,0,3);
rectangle(0,90,638,120);
setfillstyle(7,4);
bar(3,93,205,117);
setcolor(10);
outtextxy(50,100,'DOSYA ADI:');
setcolor(11);
rectangle(0,150,638,250);
setfillstyle(4,4);
bar(3,153,635,162);
line(0,165,638,165);
line(313,150,313,250);
settextstyle(0,0,1);
setcolor(15);
outtextxy(40,155,'Delinecek Toplam Delik Sayisi          Kalan
Delik Sayisi ');
setfillstyle(2,4);
bar(3,313,635,349);
outtextxy(230,317,'KOORDINATLAR');
outtextxy(150,337,'X                              Y');
setcolor(2);
rectangle(0,310,638,430);
line(0,350,638,350);
line(313,330,313 ,430);
line(0,330,638,330);
setfillstyle(10,12);
bar(3,448,635,getmaxy-16);
setcolor(223);
rectangle(0,445,638,getmaxy-18);outtextxy(30,450,'F1:  Yeni
Dosya Olusturma    F2:  Dosya Yukleme    F3:   Dosya Silme');
if akis=false then
    BEGIN
    tus:=readkey;
    if tus<>#0 then funckey:=false
```

```
else
    BEGIN
    funckey:=true;
    tus:=readkey;
    if tus=#59 then dosya
    else if tus=#60 then
        BEGIN
        akis:=true;
        yukle;
        END
    else if tus=#61 then sil;
    END;
END;
if out=true then goto baslangic;
controller:
if akis=true then devam
else
goto baslangic;
goto controller;
END.
```

ÖZGEÇMİŞ

Doğum Tarihi	9 Şubat 1970,
Doğum Yeri	Merzifon,
İlkokul	S. Sabancı Yıldız İlkokulu, İstanbul, 1976-1981,
Ortaokul ve Lise	Özel Şişli Terakki Lisesi, Burslu, İstanbul, 1981-1987,
Üniversite	Yıldız Üniversitesi, Elektronik ve Haberleşme Müh, 1987-1991,
Yüksek Lisans	Yıldız Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik Programı, 1991-.