

YILDIZ TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

Gerçek Zamanlı Bilgisayar Uygulaması

Ali Çulakçı

Yüksek Lisans Tezi

YILDIZ ÜNİVERSİTESİ
GENEL KİTAPLIĞI

368

Kot : 40
Alındığı Yer : Fen Bilimleri Enstitüsü
Tarih : 09/05/1991
Fatura :
Fiatı : 5000 TL
Ayniyat No : 1/3
Kayıt No : 47632
UDC : 001.64 378.242
Ek :

x

YILDIZ ÜNİVERSİTESİ

D.B. No 45359

**YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

Tez çalışmamı yöneten Bilgisayar Bilimleri Bölüm
Başkanı Sayın Prof. H. YILMAZ KARŞIĞIL'e ve çalışmamın
her aşamasında yardımlarını esirgemeyen Sayın Arş. Gör.
TEMFİK AKGÖR'e teşekkür ederim.

GERÇEK ZAMANLI BİLGİSAYAR UYGULAMASI

**OTOMASYAN FONKSİYONLARININ İŞ BİLGİSAYARLARINDA
ÇALIŞMASININ GERÇEKLENMESİ**

YÜKSEK LİSANS TEZİ

ELK. MÜH. ALİ ÇIKIKÇI

İSTANBUL 1988

İÇİNDEKİLER

1 - GİRİŞ	1
2 - GERÇEK ZAMANDA ÇALIŞMA	3
3 - IBM KİŞİSEL BİLGİSAYAR SİSTEMİ	4
3 - 1 - Sistem Kartı	4
3 - 2 - Donanım Kesmeleri	5
3 - 3 - Yazılım Kesmeleri	6
3 - 4 - Giriş / Çıkış Adresleri	7
3 - 5 - Giriş / Çıkış Konnektörü	8
3 - 6 - IBM PC nin G/Ç Konnektör Yapısı	9
4 - PC DE GERÇEK ZAMANDA ÇALIŞMA	13
4 - 1 - PC de Gerçek zamanda Çalışma Prensipleri	13
4 - 2 - PC de INT 1CH Yazılım Kesmesi	15
5 - KONTROL SİSTEMİ	17
5 - 1 - Kontrol Sisteminin Tanıtılması	17
5 - 2 - Yazılımın Bilgisayara Yerleştirilmesi	19
5 - 3 - Akış Diyagramı	21
5 - 4 - Kontrol Sistemi Blok Diyagramı	24
5 - 5 - Kontrol Birimi Konnektör Diyagramı	25
5 - 6 - Kontrol Sistemi Test Programı	26
6 - ARABAĞ SİSTEMİ	31
6 - 1 - Arabağ Sisteminin G/Ç Konnektör Yapısı	31
6 - 2 - Arabağ G/Ç Konnektör Diyagramı	34
6 - 3 - Arabağ Yazılımının Yerleştirilmesi	35
6 - 4 - Hizmet Veren Altprogramlar	35
6 - 5 - Simgesel Dil ile Çağırma	36
6 - 6 - Yüksek Seviyeli Dillerden Çağırma	41
6 - 7 - BASIC Dili ile Çağırma	41
6 - 8 - PASCAL Dili ile Çağırma	51
6 - 9 - C Dili ile Çağırma	55
7 - SONUÇ	58
8 - YARARLANILAN KAYNAKLAR	59
9 - ÖZGEÇMİŞ	60

ÖZET

Günümüzde, kişisel bilgisayarların gelişmesi ve bunların üreticileri arasındaki rekabet büyük bir hızla sürmektedir. Bu alandaki en büyük payı, IBM PC / XT - AT ve uyumlu bilgisayarları oluşturmaktadır.

Dünyada olduğu gibi ülkemizde de oldukça yaygınlaşan bu bilgisayarların genel kullanım alanı, klasik paket programları ile özel amaçlı uygulama programlarıdır.

Bu tez çalışmasının amacı, kişisel bilgisayarların donanımından ve çeşitli yazılımlardan da yararlanarak gerçek zamanlı çalışmalara uygun duruma getirilmesi ve kullanım alanlarının arttırılmasını sağlamaktır.

Bu çalışmada tasarlanan karta, bellek eklenerek bilgisayarın belleği optimum ölçülerde kullanılmıştır.

Sadece PC nin çalışma zamanının çok küçük bir bölümü gerçek zamanlı çalışma için ayrılmıştır. Bu eklenen yazılım ve donanımın diğer özel veya genel amaçlı programların çalışmasına karşı önemli bir dezavantajı yoktur. Böylece eklenen donanım ve yazılım, paket ve özel programlara etkisi olmadan çalışabilmektedir.

Sonuçta bilgisayarın birim zamanda yaptığı iş sayısı da böylece arttırılmaktadır.

Ayrıca PC tipi bilgisayarlarda, değişik amaçlar için tasarlanacak olan kartların, tasarımının basitleştirilmesi ve kullanıcının bu ek kartlara kolayca erişmesi sağlanmıştır.

SUMMARY

The development of personal computers and the competition between their manufacturers have a great acceleration, in these times. The largest share of this field belongs to IBM PC/XT - AT and those are compatible with it.

These type of computers are widely used in our country as well as around the world. This widespread use caused to be made a great number of general purpose software and special purpose application programs.

The objective of this study of thesis is to make the personal computers suitable for real time operations by making use of their hardware and several softwares and so is to widen the fields of use.

In this study, the memory of the computer is optimally used by adding some memory on the designed card.

A very small portion of the computer's operating time is assigned to the real time operation. Because the hardware and the software that we added does not affect the running of any other special or general purpose software, we can say any important disadvantage does not exist.

As a result, the work performed by the computer within a unit time is increased.

In addition, with PC type computers, it is simplified to design the cards to be for several purposes and supplied to the user easy access to these add-on-cards.

1 - GİRİŞ

Bu çalışmada, temelde büro amaçlı kullanım alanı bulan PC lerin gerçek zamanlı olarak kullanılma yöntemleri incelenmiştir.

Sonuçta PC nin iki türlü kullanım şekli ortaya çıkmıştır.

- 1 - PC gerçek zamanlı bir bilgisayar olarak bir göreve atanabilir.
- 2 - PC gerçek zamandaki bir görevi, başka bir görev ile eşzamanlı olarak yürütebilir.

Birinci seçenekteki çalışma şekli açıklamasından da anlaşılacağı gibi bilgisayarın gerçek zamanlı çalışmasını sağlayan arabirimler (kartlar) eklenerek tüm sistem bu görevi yürütmeye çalışmaktadır.

Bu çalışmanın amacı, 8 bitlik mikroişlemcilere, nispeten alışmış (kullanabilen) kişilerin, daha karmaşık yapıdaki işlemcinin (8086 - 8088) ve bir bilgisayar sisteminin (PC) gerçek zamanlı olarak kullanım zorluklarını ortadan kaldırmaktır.

Bu işlem iki aşamada gerçekleştirilmiştir.

Donanım aşamasında, 8 bitlik mikroişlemcilerin kullandıkları kartların sisteme bağlanması kolaylaşmıştır. Dolayısı ile kullanıcı bu kartları kendisi de tasarlamış olsa PC donanımını bilmeden sisteme yerleştirebilir.

Yazılım aşamasında ise, kullanıcının 8086-8088 makina diline ve PC sisteminde bu dili nasıl kullanması gerektiğini öğrenme zorunluluğu olmadan, eklenen kartlara erişebilmesini sağlayan yazılım araçları geliştirilmiştir. Kullanıcı herhangi bir yüksek seviye dili ile bu yazılım araçlarını kolaylıkla kullanabilir. Kullanıcıya Basic, Pascal ve C programlama dilleri ile çalışabilmesi için altprogramlar sunulmuştur.

Böylece kullanıcının, PC donanım sistemini öğrenmesine gerek kalmadan, daha kolay tasarlanabilir ek kartları yüksek seviyeli diller ile kontrol edebilmesi imkanı doğmuştur.

Uygulaması yapılan çalışma, bir gerçek zamanda çalışma olmamasına rağmen gerek donanım ve gerek ise yazılım olanaklarını daha iyi sunması açısından tercih edilmiştir.

Kullanıcının, bilgisayar dışındaki sistemleri kontrol edip uygulamalarını da deneyebileceği bilgisayar - dışdünya bağlantısını sağlayan bu sisteme, ARABAĞ sistemi adı verilmiştir.

ikinci çalışma türünde temel amaç, PC yi tamamı ile gerçek zamandaki göreve atamamaktır. Gerçek zamandaki görev ile eşzamanlı yürütülen görev, başka bir gerçek zamandaki görev olabilir.

Ancak bu çalışmadaki amaç, PC nin standart işlevlerinin (örneğin büro otomasyonunda kullanımı gibi) yanısıra bir gerçek zamanlı görevi de yürütmesidir.

Böylece maliyeti yüksek bir PC, standart işlevinde kullanılır iken aynı zamanda gerektiğinde de PC nin bütün olanaklarını (ekran, tuştakımı, disket birimi, yazıcı arabirimi vs. gibi) kullanarak gerçek zamandaki görevinide yerine getirebilir.

Bu çalışma şekli için de bir uygulama gerçekleştirilmiş ve bu sisteme ise, Kontrol Sistemi adı verilmiştir.

2 - GERÇEK ZAMANDA ÇALIŞMA

Bilgisayarların (mikroişlemcilerin) denetçi veya yorumlayıcı olarak kullanıldığı şekli temelde iki türdür.

- 1 - Gerçek zamanlı olmayan çalışma
- 2 - Gerçek zamanda çalışma

Gerçek zamanlı olmayan çalışmada önce veriler toplanır ve bilgisayar bu veriler üzerinde kendisine tanımlanan işlemleri yürüterek sonuçları bildirir. Veri toplama ile veri işlemenin aynı anda yürütülmesi gerekmemektedir.

Genellikle belirli bir sürede toplanan verilerin, bilgisayarın daha geniş bir zaman dilimi içerisinde, gerekiyor ise tekrar tekrar verileri uzun sürede işleyebildiği ve elde edilen sonuçların sadece bilgi değerini taşıması yüzünden bu çalışma önemlidir.

Yani sonuçlar veri kaynağını, veri toplama sırasında etkilememektedir.

Nüfus sayımından toplanan verilerden , istatistiksel sonuçlar elde etmek bu çalışmaya örnek olarak verilebilir.

Gerçek zamanda çalışmada, toplanan veriler anında işlenmeli ve sonuçlar veri kaynağına iletilmelidir. Veri kaynağı ise bu sonuçlara dayanarak yeni veriler üretecektir.

Bilgisayar denetimli kontrol sistemleri gerçek zamanda çalışmaya örnektir. Bu çalışmada göz önüne alınması gereken noktalar, bilgisayarın veri kaynağı ile ilişkisinin hızlı kurulabilmesi ve veri kaynağının, veri üretme anları arasında bilgisayarın bu verileri işleyebilecek hızda çalışmasıdır.

Örnek olarak bilgisayar denetimli sıcaklık kontrolünü göz önüne alalım. Isı bir limit değerini aştığı taktirde bilgisayarın, ısıtıcı elemanı anında kontrol etmesi gerekmektedir.

3 - IBM Kişisel Bilgisayar Sistemi

3 - 1 - Sistem Kartı

Sistem kartının kalbi Intel 8088 mikroişlemcidir. Bu işlemci, Intel'in 16 bitlik 8086 işlemcisinin 8 bitlik veri yoluna sahip tipidir ve 8086 ile yazılım uyumludur. Şöyleki, 8088 mikroişlemci de 16 bit işlem yapar ve 20 bitlik (1 megabyte) adres alanına sahiptir.

Bu kart , yardımcı işlemci de eklenebilmesi için maksimum modda tasarlanmıştır. işlemci 4.77 MHz frekansta çalıştırılır. Bu frekans, 14.31818 MHz kristal osilatörden 3 e bölünerek elde edilir ve ayrıca kristal osilatör frekansı yine 4 e bölünerek 3.58 MHz de üretilir. 3.58 MHz lik işaret renkli televizyonların ihtiyaç duyduğu renk bilgisi işareti (color burst signal) dir.

4.77 MHz saat hızında, 8088 yol çevrimleri, 210 ns lik 4 saat darbesi olan $210 \times 4 = 840$ ns lik bir süreyi, giriş / çıkış ise 210 ns lik 5 saat darbesi olan 1.05 mikrosaniye süreyi kapsar.

işlemci, 4 kanallı 20 bit direkt bellek erişimi (DMA), 3 adet 16 bit zamanlayıcı ve 8 adet girişli kesme öncelik denetleyicisi gibi yüksek fonksiyonlu birimler ile desteklenir.

4 adet DMA kanalının 3 ü, bellek ve G/C birimleri arasındaki yüksek-hızlı veri transferini gerçekleştirir. 4. DMA kanalı ise sistemin dinamik belleklerini tazelemek için programlanıp kullanılır.

DMA transferinin periyodik olarak yapılması için zamanlayıcısının bir kanalı programlanmıştır. Bellek tazeleme kanalı hariç olmak üzere, bütün DMA veri transfer kanallarının etkinleşmesi, 5 adet 210 ns lik işlemci saat darbe süresini (1.05 mikrosaniye süreyi) alır. DMA üzerinden tazeleme çevrimi ise 4 saat darbe süresi olan 840 ns lik süreyi kapsar.

Sistemdeki 3 adet programlanabilir zamanlayıcı aşağıdaki işlevlerde kullanılır.

Bunlardan ;

Kanal 0, time-of-day programına sabit bir zaman sağlayan, genel amaçlı zamanlayıcı olarak kullanılır.

Kanal 1, DMA kanalından dinamik bellekleri tazelemeyi sağlayan zamanlayıcı olarak kullanılır.

Kanal 2, ses hoparlörünün ton üretimini sağlamak için kullanılır.

Her kanalın, en az 1.05 mikrosaniyelik zaman hassasiyeti vardır.

Ayrıca sistemde 1 adet G/Ç arabirimi kullanılmıştır. G/Ç arabirimi yardımıyla çeşitli sistem parametreleri mikroişlemciye aktarılmakta, tuştakımı ile haberleşme sağlanmaktadır.

Bu sistemde maksimum 640Kb RAM bellek, 16Kb ekran belleği, 256Kb ROM bellek kullanılabılır.

Ek denetim kartları ile sistemde çeşitli tipte disket sürücüler ve yazıcılar (paralel-seri) kullanılabılır.

3 - 2 - Donanım Kesmeleri

Aşağıda 8088 donanım kesmesi listesi verilmiştir.

8088 Hardware Interrupt Listing

Number	Usage
NMI	Parity
0	Timer
1	Keyboard
2	Reserved
3	Asynchronous Communications (Secondary) SDLC Communications
4	Asynchronous Communications (Secondary) SDLC Communications
5	Fixed Disk
6	Diskette
7	Parallel Printer

3 - 3 - Yazılım Kesmeleri

Aşağıda 8088 Yazılım kesmesi listesi verilmiştir.

8088 Software Interrupt Listing

Interrupt Number	Name
0	Divide by Zero
1	Single Step
2	Nonmaskable
3	Breakpoint
4	Overflow
5	Print Screen
6	Reserved
7	Reserved
8	Time of Day
9	Keyboard
A	Reserved
B	Communications
C	Communications
D	Disk
E	Diskette
F	Printer
10	Video
11	Equipment Check
12	Memory
13	Diskette/Disk
14	Communications
15	Cassette
16	Keyboard
17	Printer
18	Resident BASIC
19	Bootstrap
1A	Time of Day
1B	Keyboard Break
1C	Timer Tick
1D	Video Initialization
1E	Diskette Parameters
1F	Video Graphics Chars

3 - 4 - Giriş/Çıkış Adresleri

512 G/Ç birimi, G/Ç adres haritasına yerleştirilebilir.

Aşağıda G/Ç adres haritası verilmiştir.

I/O Address Map

Hex Range	Usage
000-00F	DMA Chip 8237A-5
020-021	Interrupt 8259A
040-043	Timer 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Registers
0Ax	NMI Mask Register
0Cx	Reserved
0Ex	Reserved
100-1FF	Not Usable
200-20F	Game Control
210-217	Expansion Unit
220-24F	Reserved
278-27F	Reserved
2F0-2F7	Reserved
2F8-2FF	Asynchronous Communications (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Parallel Printer
380-38F	SDLC Communications
3A0-3AF	Reserved
3B0-3BF	IBM Monochrome Display/Printer
3C0-3CF	Reserved
3D0-3DF	Color/Graphics
3E0-3E7	Reserved
3F0-3F7	Diskette
3F8-3FF	Asynchronous Communications (Primary)

3 - 5 - Giriş / Çıkış Konnektörü

8088 mikroişlemcinin yollarını bilgisayardan dışarıya çıkarır.

Yollarla birlikte besleme, kesme hatları, doğrudan bellek erişimi ve kontrol hatları gibi fonksiyonlarına sahip genelde 5 veya 8 adet kanaldır.

Bir giriş/çıkış kanalı, 8 bitlik çift yönlü veri yolunu, 20 adres hattını, kesmenin 6 seviyesini, bellek ve G/Ç (giriş/çıkış) ile okuma ve yazma için gerekli kontrol hatlarını, saat ve zamanlama hatlarını, doğrudan bellek erişiminin 3 kanalının kontrol hatlarını, bellek tazelemesi için zaman kontrol hatlarını, gerilim kaynağının besleme ve toprak hatlarını içerir.

Giriş / Çıkış kartları için 4 gerilim seviyesi sağlanmıştır.

Bunlar +5 Vdc, -5 Vdc, +12 Vdc ve -12 Vdc dir.

Bu fonksiyonlar, bir $2 \times 31 = 62$ uçlu konnektörde bulunur.

Yavaş giriş/çıkış veya bellek birimleri ile çalışmak için G/Ç kanalında bir 'ready' hattı vardır. Kanalin ready hattı adreslenen birim tarafından aktif durumda değil ise, bellek oku ve yaz çevrimleri 4 tane 210 ns lik süre yani, 840 ns/byte tutar. G/Ç oku ve yaz çevrimi, 1.05 mikrosaniye/byte lık bir çevrim süresi için 5 saat darbesine ihtiyaç duyar.

Tazeleme çevrimi, her 72 saat darbesinde (yaklaşık olarak 15 mikrosaniyede) bir oluşur ve 4 saat darbesi ile başlatılır.

Gerilim kaynağının özellikleri ;

+12 Vdc	2 A
+ 5 Vdc	7 A
- 5 Vdc	0.3 A
-12 Vdc	0.25 A

3 - 6 - IBM PC nin G/Ç Konnektör Yapısı

IBM Kişisel bilgisayarının giriş/çıkış kanallarının ayrıntılı açıklaması aşağıda verilmiştir.
Bütün hatlar TTL uyumludur.

İŞARET	G/Ç	AÇIKLAMA
OSC	Ç	Osilator : 70 ns periyodlu (14.31818MHz) % 50 darbe/boşluk süreliidir.
CLK	Ç	Sistem saati : Osilatörün 3 ile bölünmesi sonucu elde edilir ve 210 ns periyodlu (4.77 MHz) dur. Saat işareti %33 darbe/boşluk süreliidir.
RESET DRV	Ç	Bu işaret sistem saatinin düşen kenarı ile senkronize edilmiştir ve lojik .1. konumunda aktiftir.
A0 - A19	Ç	0 dan 19 a kadar olan adres yolu bitleri: Sistemdeki giriş/çıkış arabirimlerinin ve belleğin adreslenmesi için kullanılır. Bu 20 adres hattı ile 1 megabyte lık bellek adreslenir. A0 düşük anlamlı bit (LSB "Least Significant Bit") ve A19 yüksek anlamlı bit (MSB "Most Significant Bit ") dir.
D0 - D7	G/Ç	0 dan 7 ye kadar olan veri yolu bitleri: işlemci, bellek ve G/Ç arabirimlerinin arasındaki veri alışverişisindeki 8 bitlik veri yolunu oluşturur. D0 düşük anlamlı bit ve D7 yüksek anlamlı bit tir.
ALE	Ç	Adres tutma izni : Bu hat, işlemciden geçerli adresin çıktığı bilgisini taşır ve 8288 yol kontrol tümdevresi tarafından üretilir. AEN ile kullanıldığında geçerli işlemci adresini tanımlar. işlemci adresleri ALE nin düşen kenarı ile saklanır.

I/O CH CK	G	G/Ç kanal kontrolü : Bu hat bellekde veya G/Ç kanalındaki cihazlarda eşlik (parity) hatası bilgisini işlemciye bildirir. Bu işaret, bir eşlik hatası olduğunda, lojik .0. konumunda etkindir.
I/O CH RDY	G	G/Ç kanalı hazır : Bu hat normalde lojik .1. seviyesindedir. G/Ç veya bellek erişim çevrimi sürelerini uzatmak için bir bellek veya G/Ç birimi tarafından lojik .0. a çekilir (hazır değil). Bu hat 10 saat çevrim süresinden daha uzun süre lojik .0. da tutulmamalıdır.
IRQ2 - IRQ7	G	2 den 7 ye kadar kesme istek girişleri: G/Ç arabirimlerinin kesme isteklerini işlemciye bildirmek için kullanılır. IRQ2 en yüksek öncelikli ve IRQ7 en düşük öncelikli seviyeli. Bir kesme isteği IRQ hattından yükselen kenar üretilmesi ile işlemciye bildirilir ve işlemci (kesme hizmet programı) tarafından kabul edilinceye kadar lojik .1. seviyesinde tutulur.
IOR	Ç	G/Ç okuma bilgisi : Bu hat veri yolundan G/Ç arabiriminden veri okuma işlemi yapacağını bildirir. Bu işaret işlemci veya DMA denetleyicisi tarafından üretilir ve lojik .0. konumunda aktiftir.
IOW	Ç	G/Ç yazma bilgisi : Bu hat veri yolundan G/Ç arabirimine veri yazma işlemi yapılacağını bildirir. Bu işaret işlemci veya DMA kontrol birimi tarafından üretilir ve lojik .0. konumunda aktiftir.
MEMR	Ç	Bellek okuma bilgisi : Bu hat veri yolu ile bellekten veri okuma işlemi yapılacağını bildirir. Bu işaret işlemci veya DMA kontrol birimi tarafından üretilir ve lojik .0. konumunda aktiftir.

MEMW	Ç	Bellek yazma bilgisi : Bu hat veri yolu ile belleğe veri yazma işlemi yapılacağını bildirir. Bu işaret işlemci veya DMA kontrol birimi tarafından üretilir ve lojik .0. konumunda aktiftir.
DRQ1 - DRQ3	G	1 den 3 e kadar DMA isteği : DMA servisi için çevre birimleri tarafından üretilen işaretlerdir. DRQ3 endüyük ve DRQ1 en yüksek seviye önceliklidir. Bir DRQ hattı, DACK hattı aktif oluncaya kadar lojik .1. konumunda tutulmalıdır.
DACK0 - DACK3	Ç	0 dan 3e kadar DMA kabul bilgisi : Bu hatlar DMA isteğinin kabulü (DRQ1-DRQ3) için ve sistemin dinamik belleklerinin tazelenmesinde (DACK0) kullanılır. Bu işaretler lojik .0. konumunda aktiftir.
AEN	Ç	Adres izni : Bu hat G/Ç kanalından DMA transferinin gerçekleşmesinde işlemci ve diğer birimlerin kapı kontrolunu sağlamak için kullanılır. Bu hat aktif (lojik .1.) olduğunda adres yolunun, veri yolunun, okuma bilgisi hattı (bellek ve G/Ç) ve yazma bilgisi hatlarının (bellek ve G/Ç) kontrolü, DMA denetleyicisinin kontrolundadır.
T/C	Ç	Terminal sayıcı : Bu hatla, herhangi DMA kanalı için terminal sayısına erişildiğinde bir darbe işareti sağlanır. Bu işaret lojik .1. konumda aktiftir.

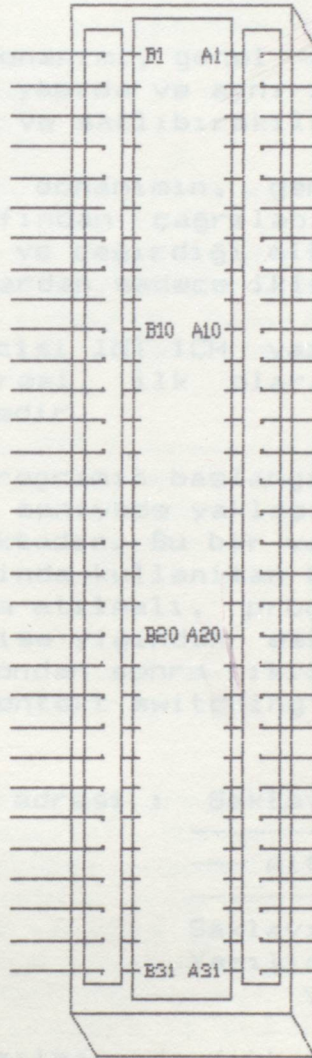
Aşağıdaki voltajlar sistem kartındaki G/Ç konnektöründe bulunmaktadır.

+5 Vdc	, 2 adet uçda yer alır.
-5 Vdc	, 1 adet uçda yer alır.
+12 Vdc	, 1 adet uçda yer alır.
-12 Vdc	, 1 adet uçda yer alır.
GND (Toprak)	, 3 adet uçda yer alır.

Rear Panel

Signal Name

GND
+RESET DRV
+5V
+IRQ2
-5VDC
+DRQ2
-12V
Reserved
+12V
GND
-MEMW
-MEMR
-IDW
-IDR
-DACK3
+DRQ3
-DACK1
+DRQ1
-DACK0
CLOCK
+IRQ7
+IRQ6
+IRQ5
+IRQ4
+IRQ3
-DACK2
+T/C
+ALE
+5V
OSC
GND



Signal Name

-I/O CH CK
D7
D6
D5
D4
D3
D2
D1
D0
+I/O CH RDY
+AEN
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0

Component Side

IBM PC I/O Channel Diagram

(IBM PC G/C Konnektoru Diyagramı)

4 - PC DE GERÇEK ZAMANDA ÇALIŞMA

4 - 1- PC de Gerçek Zamanda Çalışma Prensipleri

IBM PC/XT nin donanımı, genel kullanım kartları ve BIOS u incelenip bu yapıda ve aynı zamanda IBM AT de de henüz kullanılmayan ve saklı bırakılmış G/Ç arabirim adresleri seçilmiştir.

Bu hazırlanacak donanımın, gerçek zamanlı çalışma sırasında BIOS tarafından çağrılabilmesi için sistemin her zaman kullandığı ve çağırdığı altprogramlar araştırıldı. Bulunan altprogramlardan sadece ikisi bu çalışmaya olanak vermektedir.

- Bunlardan birincisi INT 1CH yazılım kesmesidir. Bu kesmenin vektör adresi, ilk olarak BIOS da bir IRET komutunu göstermektedir.

Yazılacak bir programın başlangıç adresi vektör olarak yüklendiği takdirde saniyede yaklaşık 18.2 kez bu programın çağırılması sağlanmaktadır. Bu bir yazılım kesmesi sayıldığı için program içerisinde kullanılan bütün saklayıcılar kesmeye girer iken yığına atılmalı, program icra edilmeli ve program bittiğinde ise yığından eski saklayıcı değerleri çıkarılıp kesmeden ondan sonra çıkılması gerekmektedir. (Bağlam değişimi "context switching").

Yani ;

Programın başlangıç adresi : Saklayıcılar yığına atılır
--- Alt program ---
Saklayıcılar yığından çekilir
Yazılım kesmesinden çıkılır
ve ana programa dönülür.

Altprogramın yazılmasında dikkat edilecek en önemli nokta icra süresinin oldukça kısa olmasıdır. Çünkü altprogram saniyede 18.2 kez çağrılmakta, program icra edilmekte ve buradan çıktıktan sonra ise esas iş olan kullanıcının hizmetine dönülmektedir.

Kullanıcıya hizmet zamanı içerisinde, ayrıca disk sürücülerini kullanıldığında DMA veri yollarını eline geçirerek MIB i pasif duruma sokup, çevre birimleri ile bellekler arası veri alışverişini sağlamaktadır.

Bu yazılım kesmesinin, sistemden nereden ve nasıl çağrıldığı araştırıldığında ise INT 8 kesmesi bulundu. Buna ait vektörün gösterdiği altprogramın sonunda, kullanıcıya ait bırakılan INT 1CH kesmesi çağrılmaktadır.

BIOS da ve işletim sisteminde INT 1CH hiçbir şekilde çağrılmamaktadır, ancak IBM PC/XT ve AT ye ait hazır paket programlar incelendiğinde INT 1CH yazılım kesmesinin kullanıldığı görülmektedir.

Program içerisinde bu kesmeye ait vektör değiştirilerek sürekli çalışan saat programları ve programlara ait kontrol işlemlerinin yerine getirildiği görüldü. Bu tip programlar, çalışacak programımızı etkileyebileceğinden bu yazılım kesmesinin hazırlanan devrede, birinci derecede kullanılmamasına karar verildi.

- ikinci kesme ise INT 8 kesmesidir. Bu kesmenin çağrılma süreside $1 / 18.2$ saniye aralıktadır. Kesmeyi yaratan yapı donanımdan kaynaklanmaktadır. Bu ise 8253 zamanlayıcı tümdevresinin 0. kanalından her saniyede 18.2 kez kesme oluşturmaya ile sağlanmıştır.

Bu süreyi 8253 un 0.kanalına tekrardan programlayarak değiştirmek mümkündür. Ancak bu sürenin değişmesi demek, INT 8 kesmesinin vektörünün gösterdiği altprogramın da değişmesi demektir.

Bunun sebebi ise bu altprogramın takvim ve saat kontrolunu gerçekleştirilmesidir. Sürenin değişmesi ile birlikte saatin ayarının bozulması problemi ortaya çıkmaktadır.

Bütün bunları düşünüp programı dikkatlice yazarak 8253 ün 0.kanalının kesme yaratma süresi değiştirilebilir.

Aynı zamanda INT 8 kesmesi programında disklerin aktif kalma süreleri de ayarlanmaktadır. Bu kesmenin sonunda ise INT 1CH kesmesi çağrılmakta ve kullanıcının bu amaca ilişkin kesme programı icra edilmektedir.

BIOS da bulunan INT 8 kesme hizmet programının yapısı aşağıdaki gibidir:

DISKET İÇİN AKTİF KALMA TESTİ

DEC MOTOR_COUNT ; disket kontrolü
JNZ 76
AND MOTOR_STATUS,CFOR ; motoru çalıştıran
; biti kapat

4 - 2 - PC de INT 1CH Yazılım Kesmesi

```
-----
: Bu altprogram 8253 zamanlayıcının 0. kanalından :
: yaratılan kesme ile çalıştırılmaktadır. :
: Giriş frekansı 1.19318 MHZ ve bölücü 65536 dır. :
: Sonuç ise yaklaşık olarak her saniyede 18.2 kesme :
: demektir. Bu kesme hizmet programı bir kullanıcı kesme :
: programı olan INT 1CH kesmesine hizmet vermektedir. :
:-----
: Kullanıcı, kesme programını makine kodu olarak :
: düzenleyip program başlangıç adresini, vektör :
: tablosunda doğru adrese yerleştirmelidir. :
:-----

TIMER_INT PROC FAR
    STI
    PUSH DS
    PUSH AX
    PUSH DX
    MOV AX,40H
    MOV DS,AX
    INC TIMER_LOW ; alçak zaman sayacı
                  ; bir arttırılıyor

    JNZ T4
    INC TIMER_HIGH ; yüksek zaman sayacı
                  ; bir arttırılıyor
T4:                  ; gün kontrolü
    CMP TIMER_HIGH,018H ; saat 24 oldumu?
    JNZ T5 ; disket kontroluna
    CMP TIMER_LOW,0BOH
    JNZ T5 ; disket kontroluna

;----- SAYICIDA 24 SAAT SÜRESİ DOLDU -----

    SUB AX,AX
    MOV TIMER_HIGH,AX
    MOV TIMER_LOW,AX
    MOV TIMER_OFL,1

;----- DiSKET iÇin AKTİF KALMA TESTİ -----

T5:                  ; disket kontrolu
    DEC MOTOR_COUNT
    JNZ T6
    AND MOTOR_STATUS,0FOH ; motoru çalıştıran
                          ; biti kapat
```



```
MOV AL,0CH
MOV DX,03F2H
OUT DX,AL ; motor durduruldu

T6: INT 1CH ; kullanıcı altprogramını
      ; çalıştırmak için transfer

MOV AL,20H
OUT 020H,AL ; 8259 a kesme sonu
POP DX
POP AX
POP DS
IRET ; kesme programından dönüş

TIMER_INT ENDP
```

Bir çok sayıda devre yardımı ile bir makine yapı-
lan giriş çıkışların toplanarak tesbiti ve sayımı,
bir kontrol biriminde bulunan makine ve
belli bir adreste daha başka bir seviye yerine
geçirip daha sonra başka bir istatistik sonuc-
larını göstermektedir.

Bu gibi devreleri daha da çoğaltabiliriz.

Kontrol biriminde, bilgisayar ile giriş/çıkış
sinyalleri, 8255 programlanabilir giriş/çıkış birimi
kullanılarak kullanılmaktadır.

Bu devrenin, son 0 da olarak 3 ayrı 8 bitlik
giriş veya çıkış olarak programlanabilen portu vardır.

Bu yapılmış portlar
veya yollar, adres yolu ve kontrol hatları olarak ilgili
devre devreye girer kullanılır.

Bu sistemin yapısında kullanılan bütün sabitler,
değişkenler ve denetimler sistemin açık kapalı olduğu
durum, tarih ve saat olarak bir tablo halinde saklanarak
iletişim bir statik RAM kullanılmaktadır.

Statik RAM, bilgisayarın gelen bilgileri geriye
saklamakta olduğu bir ila beşer. Böylece bütün bu
bilgiler, değişkenler ve tarih-saat bilgileri saklanarak

5 - KONTROL SİSTEMİ

5 - 1 - Kontrol Sisteminin Tanıtılması

Önce kontrol sistemi ile neler yapılabileceğine bir bakalım.

Kontrol sistemi blok diyagramında görüldüğü gibi, denetlenen birime örnek olması açısından kontrol sistemi ile yapılabilecek uygulamaları birkaç örnek ile aşağıdaki gibi kısaca sıralıyabiliriz;

- Kapı, kasa.. gibi alarm kontrolü ; ikaz çıkışı ile de bir alarm sisteminin kontrolü,
- Bir optik sensörlü devre yardımı ile bir mekana yapılan giriş-çıkışların zaman olarak tesbiti ve sayımı,
- Bir hareketli band üzerindeki malların sayımı ve belli bir adetden sonra başka bir işlevi yerine getirip daha sonra zaman-adet gibi istatistik sonuçlarını değerlendirmek.

Bu gibi örnekleri daha da çoğaltabiliriz.

Kontrol biriminde, bilgisayar ile Giriş/Çıkış'ı çok sağlayan, 8255 programlanabilir giriş/çıkış arabirimi tümdevresi kullanılmıştır.

Bu tümdevrenin, mod 0 da olmak üzere 3 ayrı 8 bitlik Giriş veya Çıkış olarak programlanabilen portu vardır.

Bu uygulamamızda portlar ; birer tane besleme, veri yolu, adres yolu ve kontrol hatları olarak ilgili devre dizayn edilip kullanılmıştır.

Bu birimin yazılımında kullanılan bütün sabitler, değişkenler ve denetlenen birimin açık kapalı olduğu anları, tarih ve saat olarak bir tablo halinde saklamak için pilli bir statik RAM kullanılmıştır.

Statik RAM, bilgisayardan gelen besleme geriliminin kesilmesi halinde pil ile beslenir. Böylece bütün bu sabitler, değişkenler ve tarih-saat bilgileri korunmuş olur.

Kontrol konsolunda, bir adet tuştakımı, bir adet 7 kollu bir gösterge ve bir adet de LED bulunur.

Tuštakımı 0 dan 9 a kadar rakkam tuşlarını
C clear (silme) tuşunu ve
E enter (giriş) tuşunu içerir.

Görevi ise, bir şifre girilerek, kontrol biriminin kabul etmesini ve sonuçta denetlenen birimin, izin yolunu açmasını sağlamaktır.

Bu açık kalabilme süresi veya başka bir deyişle girilen şifrenin geçerli olduğu süre yazılımda değişken olarak tanımlandığından değiştirilebilmektedir. Bu değişken, akış diyagramında "ilksüre" olarak tanımlanmıştır.

Girilen şifrenin uzunluğu, en fazla 8 basamaklı bir sayı olabilir. Tuş takımından bir rakkam girildiği anda tuşsayıcı sı bir artar. Eğer 9 u geçmiş ise tuşsayıcı sı sıfırlanır. Bu değer 7 kollu göstergede görüntülenir.

Girilen rakkamların son 8 i şifre değerlendirilmesinde dikkate alınır.

Bir rakkam tuşuna sürekli basılıyor ise, o rakkam sadece bir kez değerlendirilir. Ayrıca bütün tuşlar çok kısa bir an serbest bırakıldığında (basılmadığında) yeni bir tuşa basılmasına izin verilir ve basılan tuş geçerli olur.

Yani kısaca, kontrol biriminin tuştakımından girilen 2 tuşun değerlendirmeye alınabilmesi için, 2 tuşa basılma süresi arasında en az birkez hiçbir tuşa basılmamış olunması gerekmektedir. Ancak böylece, uzun süreli tuşa basmalarında, tekrarlama olayından kurtulmuş olunur.

C (clear) tuşu hangi anda basılır ise basılsın, bütün şifre için girilen rakkamlar ve tuşsayıcısı silinir. Eğer denetlenen birime izin verilmiş ise girilen şifre silindiği için denetlenen birime verilen izinde ortadan kalkar.

E (enter) tuşuna basıldığı anda eğer şifre için girilen rakkamlar şifre koduna eşit ise, bu andan itibaren denetlenen birime izin verilir. Tersine eşit değil ise, bütün şifre için girilen rakkamlar ve tuşsayıcısı silinir.

Ayrıca akış diyagramında görüldüğü gibi "sınırsüresi" adında bir değişken bulunmaktadır. Bu değişkende ilksüre nin içeriğine göre çok daha küçük bir zamana sahiptir. Bu son anda (kalansüre, sınır süresine eşit ve daha küçük olduğu zamanlarda) kalansüre yi biraz daha uzatmak için konmuştur. Kalan sürenin uzama miktarı sınır süresi kadardır. Bu süre uzatım anı şifre doğru ise ve sürenin sonuna doğru gelinmiş ise E tuşuna basılması ile olur.

Yani özetle, her E tuşuna basışta ;
if (şifre) and (kalansüre <= sınırsüresi) then
kalansüre := sınırsüresi
olur.

Son olarakta şifre doğru iken, geçen zaman içinde 0 dan 9 a kadar olan rakamlar içerisinden, herhangi birine basıldığı anda, şifre için girilen rakamların son 8 i değerlendirildiğinden, şifre kodundan farklı bir değere sahip olmuş ise şifre iptal edilir. Denetlenen birime verilen izin de ortadan kalkar.

5 - 2 - Yazılımın Bilgisayara Yerleştirilmesi

Bu sistemi çalıştırabilmek için buna ait programında (KONTSIS.EXE) çalışması gerekmektedir.

Bu programın bir özelliği, daha öncede bahsedildiği gibi program çalışır iken, kullanıcının normal çalışmasını engellememesidir.

Çalışmasında, yaklaşık olarak saniyede 18 kez çağrılarak icra edilmesi ile gerçekleşir. NMI kesmesi sayılmaz ise en öncelikli kesme ile çağrılır. Yani, bütün çevre birimlerinden daha öncelikli olduğundan kesme zamanı gelmiş ise diğer çevre birimlerinin o an işi olsa bile iş bırakılır, kontrol biriminin programı icra edilir ve daha sonra yarıda bırakılan işe dönülür.

Kontrol biriminin yazılımı iki ayrı şekilde bilgisayara yerleştirilir. Bu programı belleğe yerleştirmede ;

ilk yol : Program, DOS işletim sistemi altında çağrılabilir ve program belleğe yerleştikten sonra otomatik olarak vektörü koyup, kendisini daha sonra bir başka programın ezmemesi için kullanıcıya açık olan çalışma bölgesini biraz daha yukarı kaydırarak sürekli çalışmaya başlar.

Bilgisayar RESET oluncaya kadar program yerleştii o bölgede kalır ve çalışmaya devam eder.

Çalışmasının durması ise, ancak vektörün kaldırılması ile mümkündür.

ikinci yol : Program bir EPROM a yüklenir, genellikle bütün IBM PC/XT veya IBM PC/XT uyumlu bilgisayarlarda olan boş ROM soketlerine (IBM PC/XT de bu soketlerde BASIC ROM lar bulunur. Bu ROM lar çıkarılabilir.) bu EPROM takılarak programın sürekli olarak bellekte bulunması sağlanabilir.

Bu soketlerin adresleri genellikle F4000H - FFFFFH adresleri arasındaki bölgelerdir. Yanlız dikkat edilmesi gereken şey, her bilgisayarın BIOS u 8K olmayıp, daha uzun olanlarda bu bölgelerin bazılarında kullanılması sözkonusu olmaktadır.

Daha sonra seçilecek bir yol ile programın başlangıç adresi kesme vektörü olarak yerleştirilir.

Vektör yerleştirme de iki şekilde yapılabilir.

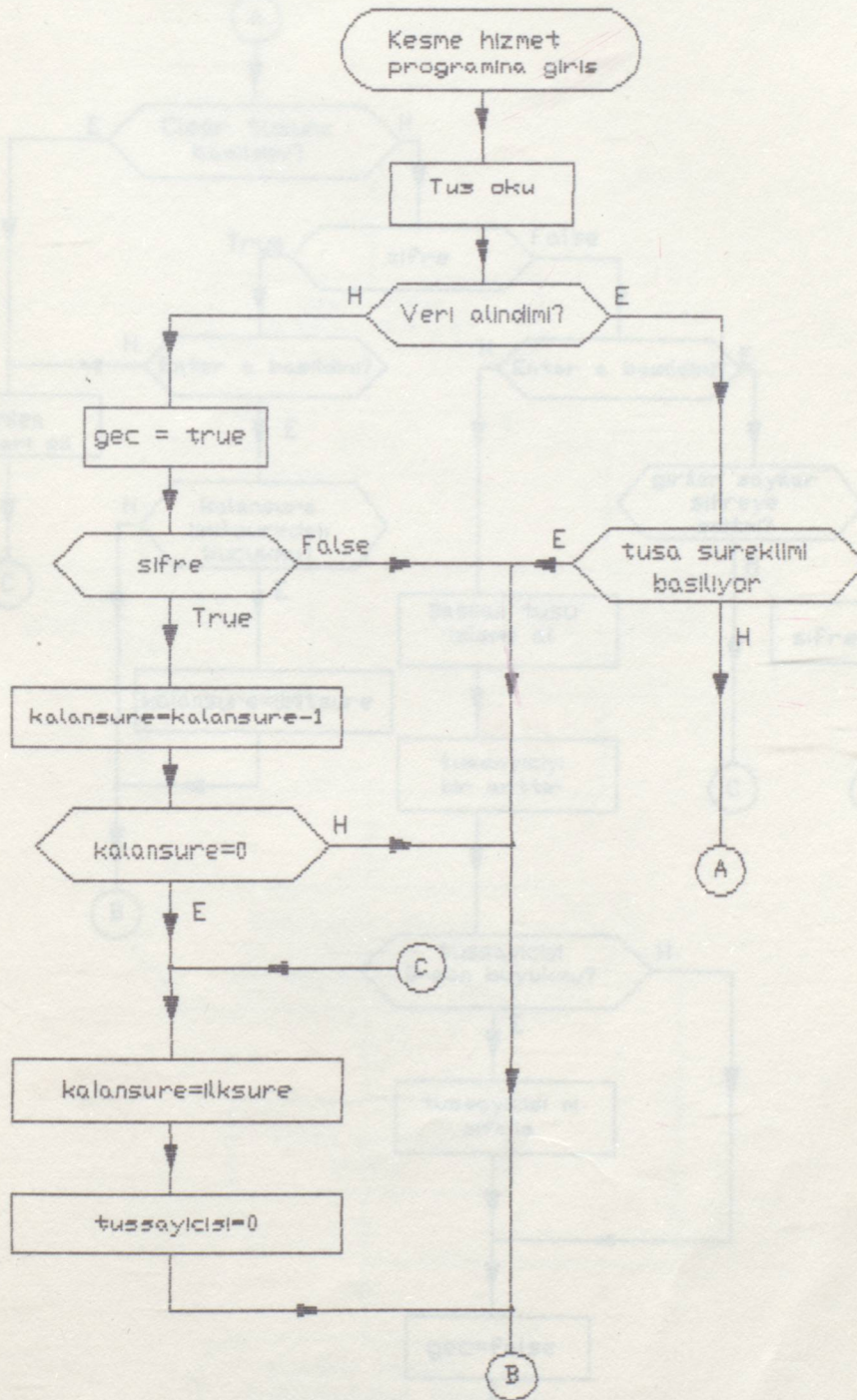
Birincisi ve en kolayı sistem disketinde AUTOEXEC.BAT programının başında, vektör yerleştiren bir COM veya EXE programını koşturarak sorun halledilebilir.

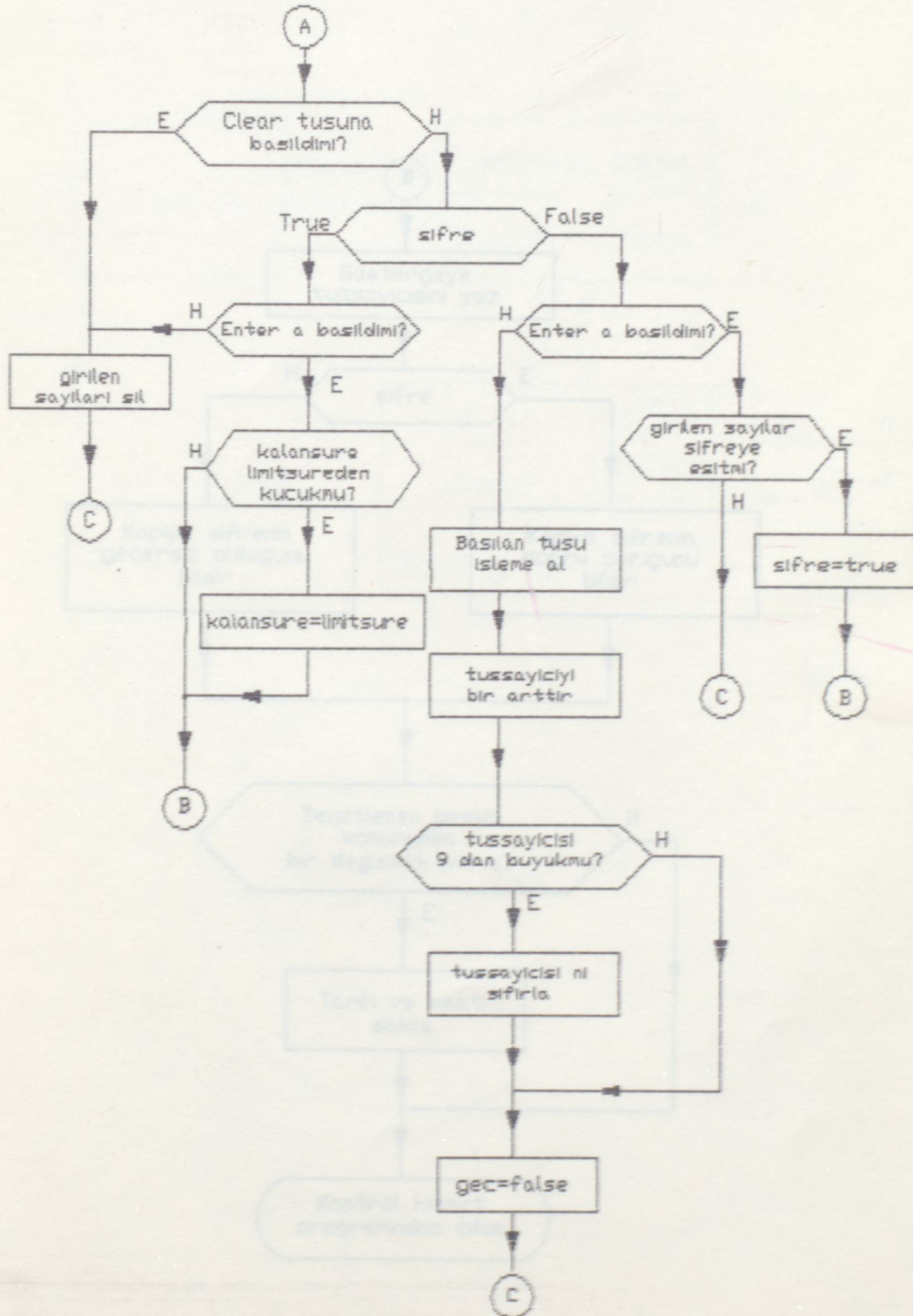
ikincisi ise, bilgisayarın BIOS unda yazılım kesmesi vektörlerinin bulunduğu tabloda ilgili vektör değıştiren, yeni bir EPROM a tekrardan BIOS yüklenip bu EPROM bilgisayara takılabilir.

Burada dikkat edilmesi gereken şey, bu programın artık makinanın donanımının bir parçası olması sonucu, sürekli aktif olmasıdır.

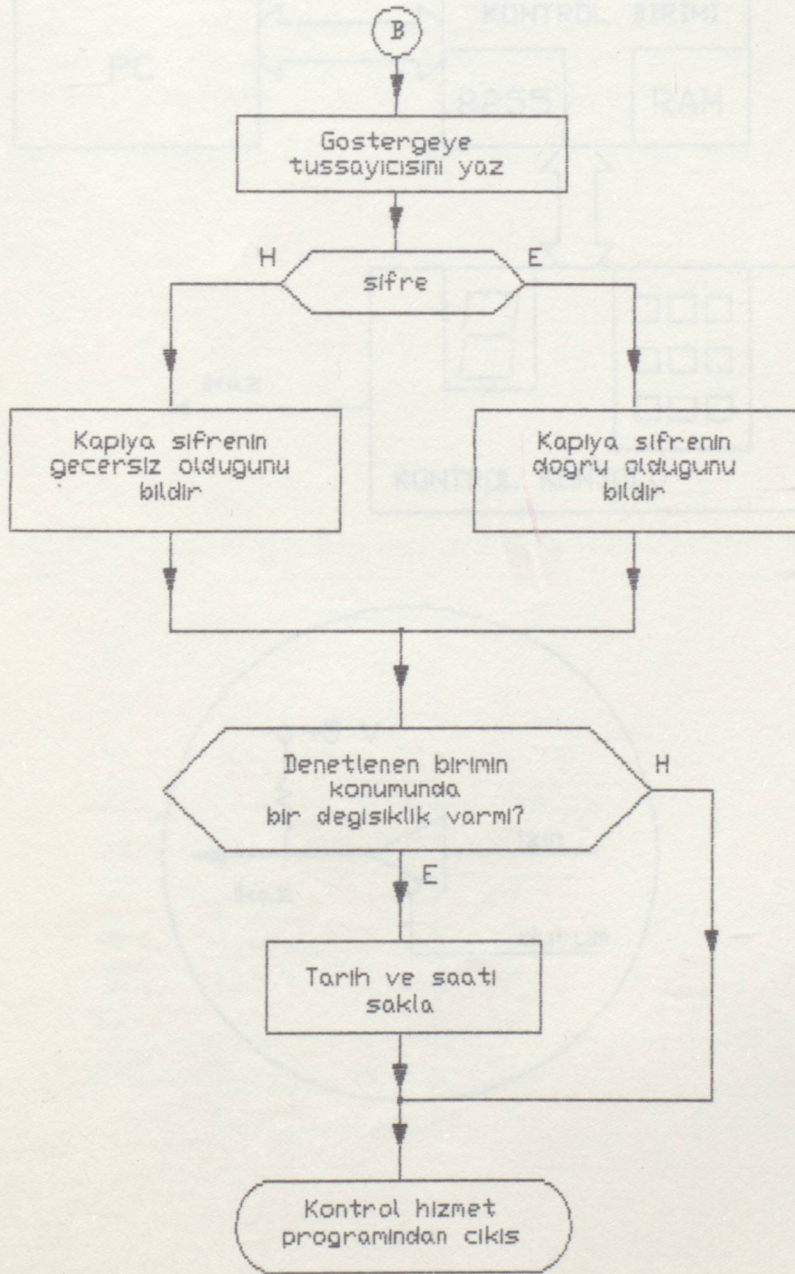
Kısaca bu ikinci yol, programın her zaman aktif olması gereken bir bilgisayarda kullanılması için seçilmelidir.

5 - 3 - Akış Diyagramı

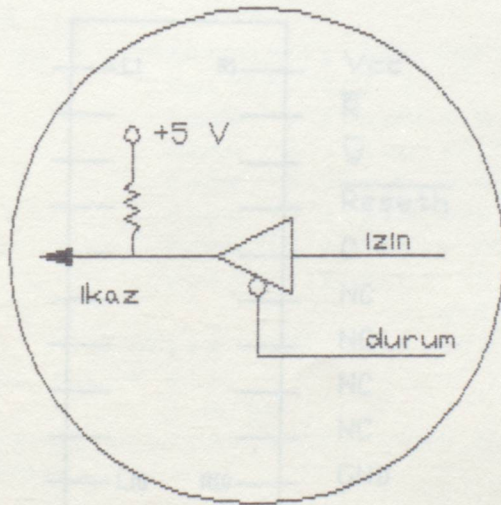
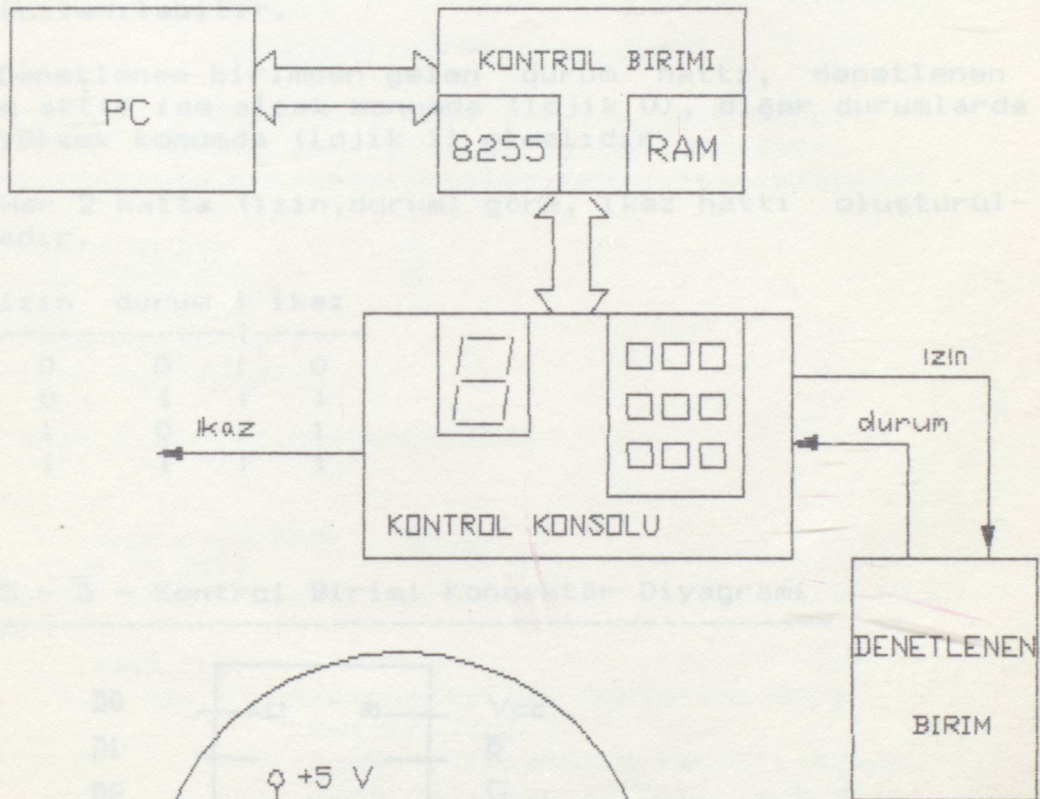




3 - 4 - Kontrol Sistemi Blok Diyagramı



5 - 4 - Kontrol Sistemi Blok Diyagramı



Yukarıdaki blok diyagram, eşanlı gerçek zamanda uygulaması yapılan kontrol birimine aittir.

Kontrol biriminden, yüksek ve alçak konumda aktif olan 2 ayrı izin hattı çıkış olarak verilmiştir.

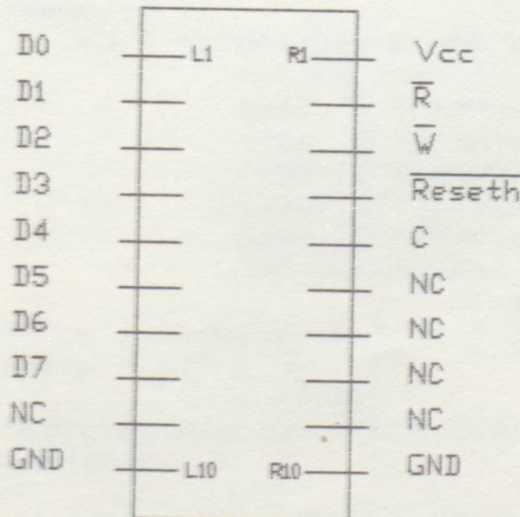
Bunlardan hangisi uygun ise denetlenen birim için o hat kullanılabilir.

Denetlenen birimden gelen durum hattı, denetlenen birim aktif ise alçak konumda (lojik 0), diğer durumlarda ise yüksek konumda (lojik 1) olmalıdır.

Her 2 hatta (izin,durum) göre, ikaz hattı oluşturulmaktadır.

izin	durum	ikaz
0	0	0
0	1	1
1	0	1
1	1	1

5 - 5 - Kontrol Birimi Konnektör Diyagramı



5 - 6 - Kontrol Sistemi Test Programı

```
-----
/*
 *      Bu program
 *
 * KONTROL SİSTEMİNDE Arıza Aranması için yapılmış bir Test
 *
 *      ve Uygulama Programıdır.
 *
 *      Program C dili ile yazılmış olup,
 *      TURBO C derleyicisi ile derlenmiştir.
 */
#define display      6
#define keyboard     5
#include <dos.h>
#include <stdlib.h>
#include <conio.h>

unsigned char  c,buffer[8191];

main()
{
    unsigned char  cc,m;
    m=peekb(0,0x23);
    if (m<8 ! m>= 0xf0) {
        do {
            menu();
            do { cc=getche(); set_kursor(0xf00);
                switch (cc) {
                    case '1':keyboard_test(); break;
                    case '2':display_test(); break;
                    case '3':keyboard_display_test(); break;
                    case '4':izin_kapi_konumu(); break;
                    case '5':ram(); break;
                    default:putch('\b');
                                } set_kursor(0xf);
                } while (cc<'0' ! cc>'5');
            } while (cc!='0');
        } else
            puts("Kontrol sistemi aktif iken bu program çalışamaz.");
        set_kursor(0xf);
    }
}
```



```
menu()
{
    clrscr();
    gotoxy(10,2); printf("ARIZA ARAMA (Diagnostic) PROGRAMI");
    gotoxy(9,3); printf("-----");
    gotoxy(7,6); printf("1- Tuştakımının testi");
    gotoxy(7,8); printf("2- Göstergenin testi");
    gotoxy(7,10);
    printf("3- Tuştakımının ve Göstergenin birlikte testi");
    gotoxy(7,12); printf("4- Izin (LED) testi ve konum kontrolu");
    gotoxy(7,14); printf("5- RAM testi");
    gotoxy(7,20); printf("0- ÇIKIS");
    gotoxy(10,22); printf("Seçiminiz..");
}
```

```
keyboard_test()
{
    char c,i,a,b,b1;
    clrscr();
    gotoxy(15,4); puts("Tuştakımı Test Programı");
    gotoxy(14,5); puts("-----");
    gotoxy(5,9); for (i=0; i<10; i++) printf("%5d",i++);
    puts("Enter Clear");
    gotoxy(10,19); puts("Not : ÇIKIŞ için F1 tuşuna basınız.");
    do {
        a=oku(keyboard); b1=a & 0x10; b=a & 0xf; gotoxy(1,24);
        if (b1 == 16) { gotoxy(15,16); printf("Son basılan tuş : ");
        switch (b) {
            case 0: case 1: case 2: case 3: case 4:
            case 5: case 6: case 7: case 8: case 9:
                printf("%1d",b); gotoxy(5+b*5,9);
                printf(" "); break;
            case 10:
                puts("Enter "); gotoxy(59,9); printf(" "); break;
            case 11:
                puts("Clear "); gotoxy(66,9); printf(" ");
        }
    } while (inkey()!=0x3b00);
}
```

```
display_test()
{
    char c,i;
    clrscr();
    gotoxy(15,4); puts("Gösterge Test Programı");
}
```



```
gotoxy(14,5); puts("-----");
gotoxy(9,19); puts("Not: < + > tuşu ile arttırma,");
gotoxy(15,20); puts("< - > tuşu ile azaltma,");
gotoxy(15,21); puts("< F1 > tuşu ile ÇIKIŞ,");
gotoxy(15,9); puts("Göstergede : 0");
i=0; gotoxy(1,24);
do {
    c=getch();
    switch (c) {
        case '+': i= (i==9 ? 0 : ++i); break;
        case '-': i= (i==0 ? 9 : --i);
    }
    if(c=='-' ! c=='+')
        yaz(display,i);
    gotoxy(29,9); printf("%d",i); gotoxy(1,24); }
    while (c!=0x3b); yaz(display,0xf);
}

keyboard_display_test()
{
    char a,b,b1;
    clrscr();
    gotoxy(15,4); puts("Tuştakımı & Gösterge Test Programı");
    gotoxy(14,5); puts("-----");
    gotoxy(10,19); puts("Not: ÇIKIŞ için F1 tuşuna basınız.");
    do {
        a=oku(keyboard);
        b1=a & 0x10; b=a & 0xf; gotoxy(1,24);
        if(b1==16) { gotoxy(15,16); printf("Son Basılan Tuş : ");
            yaz(display,b);
            switch(b) {
                case 0: case 1: case 2: case 3: case 4:
                case 5: case 6: case 7: case 8: case 9:
                    printf("%1d ",b); break;
                case 10: puts("Enter "); break;
                case 11: puts("Clear ");
            }
        }
        while ( inkey()!=0x3b00); yaz(display,0xf);
    }

izin_kapi_konumu()
{
    char c,i,j; int j1;
    clrscr();
    gotoxy(15,4); puts("Izin (LED) Test Programı");
```



```
gotoxy(14,5);puts("-----");
gotoxy(9,19);
puts("Not: < SPC > tuşu ile konum değiştirme,");
    gotoxy(15,21); puts("< F1 > tuşu ile ÇIKIŞ,");
    gotoxy(15,9); puts("LED :");
    gotoxy(15,11);puts("Konum :");
i=0xf;
do {
    j=oku(keyboard) & 32; gotoxy(23,11);
    switch (j) {
        case 0:putch('0'); break;
        case 32:putch('1');
    }
    j1=getch();
    if (j1==0x20) { i = i ^ 16; yaz(display,i);
        outport(0xe2,0); gotoxy(23,9);
        printf("%3s",i==15 ? "OFF" : "ON ");
        gotoxy(1,24);
    } while (j1!=0x3b); yaz(display,0xf); outp(0xe2,0);
}

void clreol(char coloumn)
{ char i;
for(i=coloumn;i<80;i++) printf(" ");
}

ram_sakla()
{ register int i;
for (i=0;i<=8191;i++) buffer[i]=ram_oku(i);
}

ram_getir()
{ register int i;
for (i=0;i<=8191;i++) ram_yaz(i,buffer[i]);
}

sirali()
{ int i; unsigned char b,c,bm[8191];
gotoxy(1,11); clreol(1);
gotoxy(2,18); puts("8Kb RAM yazılıyor");
for(i=0;i<=8191;i++) {
    b=rand(); bm[i]=b; ram_yaz(i,b);
}
gotoxy(7,15); puts("Not : ÇIKMAK için F1 tuşuna basınız.");
gotoxy(2,18); puts("8Kb RAM test ");
}
```



```
for(i=0;i<=8191;i++) { c=ram_oku(i); b=bm[i];
    if (inkey()==0x3b00) goto terk;
    if (b!=c) { gotoxy(21,18);
        printf("%4X %2X%6X\\a",i,b,c); getch(); }
    } /* end of loop */
gotoxy(1,15); clreol(1); gotoxy(14,18);
puts("i tamam 'RAM sağlamdır.' Herhangi bir tuşa basınız.\\a");
getch(); terk: } /* end of sirali() */

rastgele()
{ unsigned char b,c; unsigned int i;
  gotoxy(1,9); clreol(1);
  gotoxy(7,15); puts("Not : ÇIKMAK için F1 tuşuna basınız.");
  gotoxy(2,18); puts("RAM test ");
  do { i=rand() & 0x1fff; b=rand(); ram_yaz(i,b);
      c=ram_oku(i);
      gotoxy(15,18); printf("%4X %2X",i,b);
      if (b!=c) { printf("%6X\\a",c); getch(); }
    } while(inkey()!=0x3b00);
} /* end of rastgele */

ram() {
gotoxy(1,24); printf("Lütfen Bekleyiniz..."); ram_sakla();
do { clrscr();
  gotoxy(10,4); puts("RAM Test Programı");
  gotoxy(9,5); puts("-----");
  gotoxy(7,9); puts("1- Sirali test");
  gotoxy(7,11); puts("2- Rastgele test");
  gotoxy(7,15); puts("0- Ana menüye dönüş");
  gotoxy(2,22); puts("Dikkat:"); gotoxy(10,22);
  puts("Ram testi yapılırken programı asla kırmayınız.");
  gotoxy(15,23);
  puts("Çünkü RAM de bulunan bütün bilgiler silinir.");
  gotoxy(15,24); printf("Normal yoldan çıkış yapınız.");
  set_kursor(3); gotoxy(10,18); printf("Seçiminiz : ");
  c=getch(); set_kursor(0xf00);
  gotoxy(1,15); clreol(1); gotoxy(1,18); clreol(1);
  switch(c) {
    case '1':sirali(); break;
    case '2':rastgele(); break;
    default :putch('\\b'); }
  set_kursor(1);
    } while(c!='0'); putch('\\a'); ram_getir();
}
```


6 - ARABAĞ SİSTEMİ

6 - 1 - ARABAĞ Sisteminin G/Ç Konnektör Yapısı

ARABAĞ sisteminin giriş/çıkış konnektörünün ayrıntılı açıklaması aşağıda verilmiştir. Bütün hatlar TTL uyumludur.

İŞARET	G/Ç	AÇIKLAMA
A0 - A15	Ç	Adres Hatları : 0 dan 15 e kadar 16 adres hattı bulunmaktadır. Bu hatlar kontrol edilecek birimlerin adreslenmesi için kullanılır. Adres hatları 64 K ' yı adresleyebilir. A0 düşük anlamlı bit ve A15 yüksek anlamlı bitlerdir.
DO - D7	G/Ç	Veri Hatları : 0 dan 7 ye kadar 8 hat bulunmaktadır. Bu hatlar sistem ile birimler arasındaki veri alışverişi için 8 bitlik veri yolunu oluşturur. Sistemden birimlere veri yazılacağına çıkış, PC birimlerden veri okuduğunda ise giriş olarak yönleri değişir. DO düşük anlamlı bit ve D7 ise yüksek anlamlı bitlerdir. Aktif olmadığı durumlarda ise hatların hepsi üçüncü konumda kalırlar.
S0 - S3	Ç	Bölüm Seçme Hatları : 0 dan 3 e kadar 4 hat bulunmaktadır. Bir çeşit adres hattına benzerler. Bölüm seçme işlemi için tanımlanmışlardır. Doğrudan bağlandığında 4 ayrı birimi seçebilirler. Fakat kod çözerek bağlandığında ise 16 ayrı birimi kontrol edebilir. Aktif olma konumu yazılımla seçilip ister 1, ister 0 yapılabilir. 4 hatta da, yazılımla kumanda edilerek istenen çıkışın oluşması sağlanabilir. Altprogramlar çağrılmadığı anda bütün çıkışlar üçüncü konumda kalır. Kullanıcıya ait çıkışlardır.

İŞARET	G/Ç	AÇIKLAMA
R/W Ready	Ç	Oku-Yaz Hattı : Bu hat bir birime veri yollanacağında, yazma yani 0, bir birimden veri alınacağında, okuma yani 1 konumuna geçer. Seçilmediği anda çıkış sürekli 1 konumunda, yani okuma durumundadır. Böylece aktif olmadığı durumlarda yazmaya geçip herhangi bir birimde veri değişikliğine sebep olmaz.
Enable INT1 - INT3	Ç	Geçerli Olma Hattı : Bu hat cihazların seçme işleminin geçerli olduğu anı gösterir. Aktif olduğu durum 0 konumudur. Diğer durumlarda lojik 1 konumundadır. Aktif olma halinde iş yapılabilir.
VMA Vcc GND	Ç	Geçerli Bellek Adres Hattı : Bu hat adres hatlarının ve diğer hatların yerleşmesinden sonra aktif hale geçer. Yani bütün hatların geçerli olduğunu belirtir. Enable hattına benzer, fakat bu hattan daha sonra aktif hale geçer. En son aktif duruma geçen kontrol hattıdır. Aktif olduğu anda okuma veya yazma yapılabilir. Aktif olduğu durum lojik -1- konumudur. Diğer durumlarda 0 konumundadır.
Reset	Ç	Sıfırlama Hattı : Bu hat bilgisayara enerji verildiği anda ve yazılımla çağrıldığı anda, ortalama 1ms aktif duruma geçer. Aktif durum, lojik -0- konumudur. Diğer durumlarda lojik -1- konumundadır. Bu hattın aktif duruma geçmesi cihazlardaki bütün bilgilerin silinmesi ile sonuçlanır. Diğer hatların konumlarına hiç bir etkisi yoktur.
E (C2)	Ç	E Hattı : 6800 ailesi çevre birimlerinin kenar ile çalışması gereken yerlerde kullandığı bir benzeri hattır. Aktif olduğu durum lojik -1- konumudur.

4 - 2 - Araba G/C Konnektör Diyagramı

İŞARET	G/C	AÇIKLAMA
Ready	G	Hazır Hattı : Bazı çevre birimlerinin çıkardığı bu hattı ilgili altprogramları kullanarak denetlemek mümkündür. Yaklaşık olarak maksimum 7 saniye içerisinde çevre biriminden cevap gelmediği takdirde icra edilen altprogram terk edilir. Bu hat lojik -0- ise zaman dolana kadar beklenir, lojik -1- olduğunda iş yapılır.
INT1 - INT3	G	Kesme Giriş Hatları : Gerçek bir kesme olmamakla birlikte yazılımla kontrolü yapılmaktadır. En yüksek öncelikli değer 0, en düşük öncelikli değer 7 dir. Kesme isteği olup olmadığı yüksek öncelik seviyesine göre test edilir. Kesme isteği gelmiş ise seviyesine göre değerlendirilir. Kabul edilecek ise sonuç, bir bellek gözüne yerleştirilir. Kullanıcı tarafından bu bellek gözü kontrol edilerek bir istek gelip gelmediği öğrenilebilir.
Vcc		Sistemin besleme gerilimi +5V DC
GND		Sistemin toprak hattı

6 - 2 - Arabağ G/Ç Konnektör Diyagramı

Bu olarak Arabağ donanımının çalışabilmesi için en kontrol eden makine dilindeki programın bilgisayara yüklenmesi gerekir.

Bu yükleniş işlemi için sistem altındaki ARABAG.EXE programı çalıştırılır ve böylece program bilgisayar verisi olarak yüklenir. Ardından kendi- sini başka bir makineye nakletmek için gerekli işlemi alır. Bu makine ile ilgili programın, gerek duyul- duğunda çalıştırılması ile gerçekleştirilir.

İNT8, C6 (INT2) - bu makineye IBM PC/XT de konularak çalıştırılır. Kalıcı RAMC programını çalıştırarak için kullanılır.

Kalıcı RAMC programının çalıştırılması için, normalde bilgisayara disk sürücülerine yazılır, yani, işletim sisteminin uygun olduğu bir makineye yazılır. Bir başka yol ise doğrudan makineye yazılmasıdır. Bu işlem için bir başka makineye yazılması gerekir. Ancak bu işlem için Kalıcı RAMC çalıştırılması gerekir. Fakat bu işlem için çok gereksizdir. Çünkü işletim sistemi bu işlemi yapabilir. Kalıcı RAMC ile çalıştırılması mümkündür.

Sonuç olarak Kalıcı RAMC ile sistem yerleştiril- ten sonra makineye yazılır. Makineye yazılan işletiminde çalıştırılır. Bu makineye yazılan işletiminde çalıştırılır. Bir başka makineye yazılmasıdır. Bu işlem için bir başka makineye yazılması gerekir. Ancak bu işlem için Kalıcı RAMC çalıştırılması gerekir. Fakat bu işlem için çok gereksizdir. Çünkü işletim sistemi bu işlemi yapabilir. Kalıcı RAMC ile çalıştırılması mümkündür.

Birden fazla makineye yazılmasıdır. Bu işlem için bir başka makineye yazılması gerekir. Ancak bu işlem için Kalıcı RAMC çalıştırılması gerekir. Fakat bu işlem için çok gereksizdir. Çünkü işletim sistemi bu işlemi yapabilir. Kalıcı RAMC ile çalıştırılması mümkündür.

6 - 4 - Arabağ G/Ç Konnektör Diyagramı

ARABAG programından çeşitli altprogramlar çağırılarak ARABAG donanımına bağlanarak işletilebilir.

Makine dilinde kullanıcıya hizmet veren altprogramlar şunlardır:

Silme (erase), Okuma, Yazma, Blok Karsılaştırma, Blok Okuma, Blok Yazma, Blok Doldurma ve Hazır Hattı (ready) ile Okuma, Hazır Hattı ile Yazma, Hazır Hattı ile Blok Karsılaştırma, Hazır Hattı ile Blok Okuma, Hazır Hattı ile Blok Yazma, Hazır Hattı ile Blok Doldurma dır.

6 - 3 - Arabağ Yazılımının Yerleştirilmesi

ilk olarak Arabağ donanımının çalışabilmesi için onu kontrol eden makina dilindeki programın bilgisayara yüklenmesi gerekir.

Bu yükleme işide, DOS işletim sistemi altında "ARABAG.EXE" programı çağrılarak yapılır ve böylece program belleğe yerleşerek vektörü koyduktan sonra kendisini başka programların ezmemesi için gerekli önlemi alır.

Bu makina dili ile yazılmış programın, gerek duyulduğunda çağrılması INT18 yazılım kesmesi ile gerçekleşir.

INT18 yazılım kesmesi esasında sadece IBM PC/XT de ROM larda bulunan ' Kalıcı BASIC ' programını çağırmak için kullanılır.

Kalıcı BASIC programının çalışabilmesi için, normalde bilgisayardaki disk sürücülerin olmaması yani, işletim sisteminin bulunmaması gerekmektedir. Bir başka yol ise doğrudan makina dilinden INT18 i çağırmaktır ve geri dönüşü yoktur. Ancak bu koşullar altında Kalıcı BASIC çalışabilmektedir. Fakat bu son yol çok gereksizdir. Çünkü işletim sistemi tarafından yüklenebilen gelişmiş BASIC ile çalışmak daha yaygındır.

Sonuç olarak Kalıcı BASIC, işletim sistemi yerleştikten sonra kullanılmadığından, INT18 tanımlanan işlevinde çalışmamaktadır. Bu nedenle bilgisayar açıldıktan sonra birdaha kullanılmayan bu yazılım kesmesine ARABAG programı kendi başlangıç adresini (vektörünü) yerleştirir.

6 - 4 - Hizmet Veren Altprogramlar

ARABAG programından çeşitli altprogramlar çağırılarak ARABAG donanımına bağlanmış birimlere erişilebilir.

Makina dilinde kullanıcıya hizmet veren altprogramlar şunlardır:

Silme (reset), Okuma, Yazma, Blok Karşılaştırma, Blok Okuma, Blok Yazma, Blok Doldurma ve Hazır Hattı (ready) ile Okuma, Hazır Hattı ile Yazma, Hazır Hattı ile Blok Karşılaştırma, Hazır Hattı ile Blok Okuma, Hazır Hattı ile Blok Yazma, Hazır Hattı ile Blok Doldurma dır.

- Silme (reset) altprogramı ile bağlanan tümdevreleri veya birimleri reset edilebilir. Aynı zamanda bölüm seçicide kullanılıp gerektiğinde sadece bir çevre birimi- de kod çözerek reset edilebilir.
- Okuma (read) altprogramı ile istenen bölüm ve adresden veri okunur.
- Yazma (write) altprogramı ile istenen bölüm ve adrese veri yazılır.
- Blok karşılaştırma (bcomp) altprogramı ile seçilen birimdeki verilerin bellekteki veriler ile karşılaştırması yapılır.
- Blok okuma (bread) altprogramı ile seçilen birimden belleğe blok olarak veri yüklenir.
- Blok yazma (bwrite) altprogramı ile seçilen birime bellekten blok olarak veri kaydedilir.
- Blok doldurma (bfill) altprogramı ile seçilen birimin belirli adresleri arası verilen bir byte değeri ile doldurulur.
- Ready hattı kullanan altprogramlarının çalışma şekli yukarıdaki altprogramlar gibidir. Sadece burada ready hattı dikkate alınır. Ready hattı maksimum 7 saniye içerisinde aktif olmalıdır. Yoksa yapılan iş terkedilir ve AH saklayıcısının içeriği FFh olarak döner.

Bu altprogramları ingilizce olarak kısaltılmış isimlerle kullanacağız.

6 - 5 - Simgesel Dil ile Çağırma

Altprogramlar, 8086 simgesel dili veya yazılım kesmesi çağırabilen yüksek seviyeli diller tarafından çağırılabilir. Altprogramları çağırma örnekleri aşağıdadır :

RESET için	MOV	AH,0
	MOV	AL,bölüm_saklayıcısı
	INT	1BH
READ için	MOV	AH,1
	MOV	AL,bölüm_saklayıcısı
	MOV	DX,ARABAG_adresi
	INT	1BH

Dönüşde BL saklayıcısı okunan sözcük değerini içerir.


```
WRITE icin  MOV    AH,2
             MOV    AL,bölüm_saklayıcısı
             MOV    BL,yazılacak_byte_c_adresi
             MOV    DX,GÇ_adresi
             INT    18H
```

```
BCOMP icin  MOV    AH,3
             MOV    AL,bölüm_saklayıcısı
             MOV    ES,Bellek_segmenti
             MOV    SI,Bellek_baslangıç_adresi
             MOV    DI,GÇ_adres_baslangıcı
             MOV    DX,GÇ_adres_sonu
             INT    18H
```

Farklılık var ise AL<>0 dır ve dönüşde farklı olan adreslerin değerleri SI ve DI olarak döner.

Dönüşde ES:SI ---- DI ---- değerini içerir.

```
WRITE icin  MOV    PC |   | ARA |
             MOV    AL |   |
             MOV    DX |   |
```

```
BREAD icin  MOV    AH,4
             MOV    AL,bölüm_saklayıcısı
BCOMP icin  MOV    ES,Bellek_segmenti
             MOV    DI,Bellek_baslangıç_adresi
             MOV    SI,GÇ_adres_baslangıcı
             MOV    DX,GÇ_adres_sonu
             INT    18H
```

ES:DI ---- SI ----

```
PC |   | ARA |
PC |   | ARA |
PC |   | ARA |
DX |   |
```

```
BWRITE icin MOV    AH,5
             MOV    AL,bölüm_saklayıcısı
             MOV    ES,Bellek_segmenti
             MOV    SI,Bellek_baslangıç_adresi
             MOV    DX,Bellek_sonu
             MOV    DI,GÇ_adres_baslangıcı
             INT    18H
```

ES:SI ---- DI ----

```
PC |   | ARA |
PC |   | ARA |
PC |   | ARA |
DX |   |
```



```
BFILL icin  MOV AH,6
             MOV AL,bölüm_saklayıcısı
             MOV DI,Bellek_baslangıç_adresi
             MOV DX,Bellek_sonu
             MOV BL,Doldurulacak_byte
             INT 18H
```

```
RWRITE icin MOV AH,11H          BLOCKWRITE with READY
             DI ---- AL,bölüm_saklayıcısı
             |      | ES,Bellek_segmenti
             | ARA | SI,Bellek_baslangıç_adresi
             |      | DX,Bellek_sonu
             DX ---- DI,GÇ_adres_baslangıcı
```

```
RREAD icin  MOV AH,7          READ with READY
             MOV AL,bölüm_saklayıcısı
             MOV DX,GÇ_adresi
             INT 18H
```

Dönüşde BL registeri okunan byte değerini içerir.

```
RWRITE icin MOV AH,8          BLOCKWRITE with READY
             MOV AL,bölüm_saklayıcısı
             MOV BL,yazılacak_byte
             MOV DX,GÇ_adresi
             INT 18H
```

```
RBCOMP icin MOV AH,9          BLOCKCOMPARE with READY
             MOV AL,bölüm_saklayıcısı
             MOV ES,Bellek_segmenti
             MOV SI,Bellek_baslangıç_adresi
             MOV DI,GÇ_adres_baslangıcı
             MOV DX,GÇ_adres_sonu
             INT 18H
```

Farklılık var ise AL<>0 dır ve dönüşde farklı olan adreslerin değerleri SI ve DI olarak döner.

```
ES:SI  ----  DI  ----
       |      |      | |
       | PC  |      | ARA |
       |      |      |
       ----  DX  ----
```

```
RBREAD icin MOV AH,10H        BLOCKREAD with READY
             MOV AL,bölüm_saklayıcısı
             MOV ES,Bellek_segmenti
             MOV DI,Bellek_baslangıç_adresi
             MOV SI,GÇ_adres_baslangıcı
             MOV DX,GÇ_adres_sonu
             INT 18H
```


ES:DI ---- SI ----
PC ARA
DX ----

RBWRITE için MOV AH,11d BLOCKWRITE with READY
MOV AL,bölüm_saklayıcısı
MOV ES,Bellek_segmenti
MOV SI,Bellek_baslangıç_adresi
MOV DX,Bellek_sonu
MOV DI,GÇ_adres_baslangıcı
INT 18H

ES:SI ---- DI ----
PC ARA
DX ----

RBFILL için MOV AH,12d BLOCKFILL with READY
MOV AL,bölüm_saklayıcısı
MOV DI,Bellek_baslangıç_adresi
MOV DX,Bellek_sonu
MOV BL,Doldurulacak_byte
INT 18H

DI ----
ARA
DX ----

Burada kullanılan 8088 işlemcisinin saklayıcıları ;
AH,AL,BL : 8 bitlik saklayıcılardır.
0..255 (0..FFh) e kadar değer alabilirler.
DX,ES,SI,DI : 16 bitlik saklayıcılardır.
0..65535 (0..FFFFh) e kadar değer alabilirler.
dır.

Bütün bu altprogramlarda ;
- AH saklayıcısı altprogramın numarasını belirtir. AH ın içeriğine göre hangi altprogramın çağrılacağına karar verilir.

- AL saklayıcısı bölüm saklayıcısıdır. Bu saklayıcının ilk 4 biti (D0,D1,D2,D3) seçilecek bölüm saklayıcısını belirtir.

AL	-----	1	1	1	1	1	S3	S2	S1	S0	1
Bitler		D7	D6	D5	D4	D3	D2	D1	D0		

Bu bitlerin herbiri altprogram çağrıldıktan sonra, bölüm saklayıcısı hatlarından çıkarak konnektörden dışdünyaya iletilir.

AL saklayıcısının 2. 4 biti (D4,D5,D6,D7) ise gecikme için ayrılmıştır ve sadece hazır hattını kullanmadan yazma işlemlerinin yapıldığı altprogramlarda geçerlidir.

Bunlar ise; WRITE, BWRITE, BFILL altprogramlarıdır. Diğer bütün altprogramlarda 2. 4 bit dikkate alınmaz. Gecikmenin kullanılmasının sebebi ise bazı yavaş çevre birimlerinde, yazma işlemlerinin yavaş yapılması gerektiğindendir.

Ancak hazır ol hattı (ready) bulunan çevre birimlerinde gecikme yapmaya gerek yoktur. Çünkü çevre birimlerinde bu hat, bu amaçla kullanılmaktadır.

Gecikme süresi, basitçe aşağıdaki gibi hesaplanabilir.

CPU hızı 4.77 MHz iken;

Gecikme süresi = $(2^G) * 3.8$ mikrosaniye

G değişkeni G3..G0 bitlerinin verdiği değerdir.

G = 0 iken minimum 3.8 mikrosaniye

G = 15 iken maksimum 125 milisaniye dir.

- BL saklayıcısı veri taşıyan saklayıcı olarak kullanılmıştır.

- ES bellek segmenti saklayıcısıdır. IBM PC/XT de bellek segmentlere bölünmektedir. Her segmentde 64K lık bir bellek bulunmaktadır. ES de hangi segmentde çalışılacağını göstermektedir. (Blok yazma işlemlerinde ES segmenti çok dikkatlice seçilmelidir. Yanlış bir segment seçilir ise o bölgedeki veriler, kodlar, sistem değişkenleri veya işletim sistemi ezilebilir.)

- DX saklayıcısı seçilmiş olan altprogramda erişilmesi istenen enson adresi gösterir. Altprograma göre bellek veya çevre biriminin enson adresi olabilir.

- SI, DI saklayıcısı seçilmiş olan altprogramda erişilmesi istenen ilk adresi gösterir. Altprograma göre bellek veya çevre biriminin ilk adresi olabilir.

6 - 6 - Yüksek Seviyeli Dillerden Çağırma

Yüksek seviyeli diller ile program yazmak daha kolay olduğundan, bundan sonra yüksek seviyeli dillerin bazı-
larında (BASIC, PASCAL ve C dilleri ile) bu altprogram-
ların nasıl kullanılacağına ve dolayısı ile nasıl çağrı-
lacağına bir bakalım.

Bütün bu dillerde de altprogramlara erişmek INT 18H
yazılım kesmesini çağırarak gerçekleşmektedir. Daha önce-
den verilen altprogramların kullanımı aynen geçerlidir.

INT 18H yazılım kesmesine ARABAG programının başlangıç
adresi vektör olarak konmuş olmalıdır. (Vektörü koymak için
ARABAG.EXE programının bir kez çalıştırılması yeterlidir.)

6 - 7 - BASIC Dili ile Çağırma

Basic tarafından INTerrupt ları çağırarak mümkün olma-
dığından bu işi makina dili ile gerçekleştirmek gereklidir.
Bunun içinde PREBAS.EXE adlı programın birkez çalıştırılması
gereklidir.

Daha sonra BASIC den bu altprogramların çağırılması
mümkün olmaktadır.

Aşağıda BASIC den altprogramları çağırabilmek için
ARABAG.BAS programının dökümü verilmiştir.

Değişkenlerin açıklaması :

cihaz%	Cihaz no (0..15)
kontr%	Gecikmenin dikkate alındığı cihaz no 0..15 (cihaz% için) + 16 * 0..15 (gecikme için) (gecikme; 0 ise 3.8 µs , 15 ise 125 ms. dir.)
adres%	Adres (0..65535)
veri%	Veri (0..255)
bel.seg%	Bellek segmenti (kısmı)
bel.adr.ilk%	Belleğin ilk adresi
bel.adr.son%	Belleğin son adresi
gc.adr.ilk%	G/Ç biriminin ilk adresi
gc.adr.son%	G/Ç biriminin son adresi
bel%	Farklı bellek adresi
gc%	Farklı G/Ç birimi adresi
dol.ver%	Doldurulacak veri


```
10 DEF SEG=0
20 IP = PEEK(&H4F0)+256*PEEK(&H4F1)
30 SG = PEEK(&H4F2)+256*PEEK(&H4F3):DEF SEG = SG
40 Rem -----
50   SIL =ip   : OKU   =ip+3 : YAZ   =ip+6
60   BCOMP =ip+9 : BREAD =ip+12: BWRITE =ip+15: BFILL =ip+18
70           ROKU  =ip+21: RYAZ   =ip+24
80   RBCOMP =ip+27: RBREAD=ip+30: RBWRITE=ip+33: RBFILL =ip+36
90           ISTEK1=ip+39: MASKE  =ip+42: ISTEK2 =ip+45
95 Rem -----
100 CALL SIL(cihaz%)           :Rem  RESET altprogramı
110 CALL OKU(cihaz%,adres%,veri%)
120 CALL YAZ(kontr%,adres%,veri%)
130 CALL BCOMP(cihaz%,bel.seg%,bel.adr.ilk%,gc.adr.ilk%,gc.adr.son%,bel
140 CALL BREAD(cihaz%,bel.seg%,bel.adr.ilk%,gc.adr.ilk%,gc.adr.son%)
150 CALL BWRITE(kontr%,bel.seg%,bel.adr.ilk%,bel.adr.son%,gc.adr.ilk%)
160 CALL BFILL(kontr%,gc.adr.ilk%,gc.adr.son%,dol.ver%)
200 Rem ----- Ready hattı ile ilgili altprogramlar -----
210 CALL ROKU(cihaz%,adres%,veri%)
220 CALL RYAZ(cihaz%,adres%,veri%)
230 CALL RBCOMP(cihaz%,bel.seg%,bel.adr.ilk%,gc.adr.ilk%,gc.adr.son%,be
231 Rem -----
232 Rem   BCOMP() ve RBCOMP() için dönüşde,
233 Rem   if cihaz%(>0) then "bellek ile G/Ç birimi arasında veri
234 Rem   farklılığı vardır. bel% ve gc% geçerlidir."
235 Rem -----
240 CALL RBREAD(cihaz%,bel.seg%,bel.adr.ilk%,gc.adr.ilk%,gc.adr.son%)
250 CALL RBWRITE(cihaz%,bel.seg%,bel.adr.ilk%,bel.adr.son%,gc.adr.ilk%)
260 CALL RBFILL(cihaz%,gc.adr.ilk%,gc.adr.son%,dol.ver%)
270 Rem ----- Düzeyli Kesme altprogramı -----
280 Rem   kabul% dönüş parametresi 255 ise kesme isteği yok
290 Rem   düzey% istenen düzey giriş parametresi.Küçük olanlar kabul ed
300 Rem   seviye% kabul edilmiş ise bundan sonraki yeni düzey bilgisi gi
310 Rem -----
311 Rem *****
312 Rem *** maske altprogramı ile maske yerleştirilir. ***
313 Rem *** istek1 altprogramı ile o andaki istek bilgisi alınır. ***
314 Rem *** maske ve istek1 altprogramları birlikte kullanılır. ***
315 Rem *** ----- ***
316 Rem *** INTO - INT2 girişleri saniyede 18 kez kontrol edilerek ***
317 Rem *** bilgi alınır. Bu bilgiler arasında değer olarak en ***
318 Rem *** küçük sayı tutulur. Istek2 altprogramı ile, bu sayı ***
319 Rem *** düzey ile karşılaştırılır, küçük veya eşit ise kabul ***
320 Rem *** e aktarılır ve seviye yeni sayı değeri olarak belleğe ***
321 Rem *** alınır. Kabul edilmemiş ise kabulun değeri 255 dir. ***
322 Rem *****
323 Rem -----
330 CALL MASKE (düzey%)
340 CALL ISTEK1(kabul%)
350 Rem -----
360 CALL ISTEK2(kabul%,düzey%,seviye%)
370 Rem -----
```



```
*****
* Bu altprogramlar BASIC tarafından çağrılmaktadır. *
* PREBAS.EXE programının INT 18H i nasıl çağırdığını *
* göstermek üzere, programın esas kısmı aşağıdaki gibidir. *
* BASIC de bu altprogramları kullanabilmek için daha önce *
* DOS da iken PREBAS.EXE programı çalıştırılmalıdır. *
*****
```

```
code    segment
xx      equ    18h                ; INT 18H
;-----
cntrl   proc   far
        assume cs:code,ds:code,ss:st_seg
reset_opt:
        jmp    reset
read_opt:
        jmp    read
write_opt:
        jmp    write
bcomp_opt:
        jmp    bcomp
bread_opt:
        jmp    bread
bwrite_opt:
        jmp    bwrite
bfill_opt:
        jmp    bfill
rread_opt:
        jmp    rread
rwrite_opt:
        jmp    rwrite
rbcomp_opt:
        jmp    rbcomp
rbread_opt:
        jmp    rbread
rbwrite_opt:
        jmp    rbwrite
rbfill_opt:
        jmp    rbfill
interrupt1_opt:
        jmp    interrupt1
maske_opt:
        jmp    maskeprg
interrupt_opt:
        jmp    interrupt
;-----
```


reset:

```
push    bp
mov     bp,sp
push    ax
push    bx
mov     bx,[bp]+6
mov     ax,[bx]      ; al,cihaz_no
xor     ah,ah        ; ah,0
int     xx
pop     bx
pop     ax
pop     bp
ret     2
```

;-----

read:

```
push    bp
mov     bp,sp
push    ax
push    bx
push    dx
mov     bx,[bp]+10
mov     ax,[bx]      ; al,cihaz_no
mov     ah,1         ; ah,1
jmp     read_routine
```

;-----

write:

```
push    bp
mov     bp,sp
push    ax
push    bx
push    dx
mov     bx,[bp]+10
mov     ax,[bx]      ; al,cihaz_no
mov     ah,2         ; ah,2
jmp     write_routine
```

;-----

bcomp:

```
push    bp
mov     bp,sp
push    ax
push    bx
mov     bx,[bp]+18
mov     ax,[bx]      ; al,cihaz_no
mov     ah,3         ; ah,3
jmp     bcomp_routine
```

;-----

bread:

```
push    bp
mov     bp,sp
push    ax
push    bx
```



```

;-----
;rwriter:
        mov     bx,[bp]+14
        mov     ax,[bx]           ; al,cihaz_no
        mov     ah,4             ; ah,4
        jmp     bread_routine
;-----
bwrite:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        mov     bx,[bp]+14
        mov     ax,[bx]           ; al,cihaz_no
        mov     ah,5             ; ah,5
        jmp     bwrite_routine
;-----
;-----
bfill:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        mov     bx,[bp]+12
        mov     ax,[bx]           ; al,cihaz_no
        mov     ah,6             ; ah,6
        jmp     bfill_routine
;-----
;-----
rread:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        push    dx
        mov     bx,[bp]+10
        mov     ax,[bx]           ; al,cihaz_no
        mov     ah,7             ; ah,7
;-----
read_routine:
        mov     bx,[bp]+8
        mov     dx,[bx]           ; dx,GCadres
        int     xx
        mov     al,bl             ; al,veri
        mov     bx,[bp]+6
        mov     [bx],al
        pop     dx
        pop     bx
        pop     ax
        pop     bp
        ret     6
;-----

```



```
rwrite:
    push    bp
    mov     bp,sp
    push    ax
    push    bx
    push    dx
    mov     bx,[bp]+10
    mov     ax,[bx]
    mov     ah,8
; -----
write_rutin:
    mov     bx,[bp]+8
    push    [bx]
    mov     bx,[bp]+6
    mov     dx,[bx]
    pop     bx
    int     xx
    pop     dx
    pop     bx
    pop     ax
    pop     bp
    ret     6
; -----
rbcomp:
    push    bp
    mov     bp,sp
    push    ax
    push    bx
    mov     bx,[bp]+18
    mov     ax,[bx]
    mov     ah,9
; -----
bcomp_rutin:
    push    dx
    push    es
    push    si
    push    di
    mov     bx,[bp]+16
    mov     es,[bx]
    mov     bx,[bp]+14
    mov     si,[bx]
    mov     bx,[bp]+12
    mov     di,[bx]
    mov     bx,[bp]+10
    mov     dx,[bx]
    int     xx
    mov     bx,[bp]+18
    mov     [bx],al
    mov     bx,[bp]+8
    mov     [bx],si
```

; GC adresi

; al,cihaz_no

; ah,8

; dx,GCadres

; bl,veri

; al,cihaz_no

; ah,10d

; es,bellek segmenti

; al,cihaz_no'nun adresi

; ah,9

; al,ilk GCadres

; dx,son GCadres

; es,bellek segmenti

; si,ilk bellek adresi

; di,ilk GCadres

; dx,son GCadres

; compare sonucu

; bellek adresi


```
mov     bx,[bp]+6      ; al,cihaz_no
mov     [bx],di        ; GC adresi
pop     di
pop     si
pop     es
pop     dx
pop     bx
pop     ax
pop     bp,[bp]+12
ret     14,[bx]        ; es,bellek segmenti
;-----
rbread:  mov     si,[bx] ; si,ilk bellek adresi
        push    bp      ; di,ilk GCadres
        mov     bp,sp
        push    ax,[bp]+8 ; dx,son bellek adresi
        push    bx,[bx]
        mov     bx,[bp]+14
        mov     ax,[bx]   ; al,cihaz_no
        mov     ah,10     ; ah,10d
;-----
bread_routine:
        push    dx
        push    es
        push    si
        push    di
        mov     bx,[bp]+12
        mov     es,[bx]   ; es,bellek segmenti
        mov     bx,[bp]+10
        mov     di,[bx]   ; di,ilk bellek adresi
        mov     bx,[bp]+8
        mov     si,[bx]   ; si,ilk GCadres
        mov     bx,[bp]+6
        mov     dx,[bx]   ; dx,son GCadres
        int     xx,[bx]   ; ah,12d
        pop     di
        pop     si
        pop     es
        pop     dx
        pop     bx,[bp]+10
        pop     ax,[bx]   ; di,ilk GCadres
        pop     bp,[bp]+8
        ret     10,[bx]   ; dx,son GCadres
;-----
rbwrite:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        mov     bx,[bp]+14
```



```

        mov     ax,[bx]           ; al,cihaz_no
        mov     ah,11            ; ah,11d
;-----
bwrite_routine:
        push    dx
        push    es
        push    si
        push    di
        mov     bx,[bp]+12
        mov     es,[bx]          ; es,bellek segmenti
        mov     bx,[bp]+10
        mov     si,[bx]          ; si,ilk bellek adresi
        mov     bx,[bp]+6
        mov     di,[bx]          ; di,ilk GCadresesi
        mov     bx,[bp]+8
        mov     dx,[bx]          ; dx,son bellek adresi
        int     xx
        pop     di
        pop     si
        pop     es
        pop     dx
        pop     bx
        pop     ax
        pop     bp
        ret     10
;-----
rbfill:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        mov     bx,[bp]+12
        mov     ax,[bx]          ; al,cihaz_no
        mov     ah,12            ; ah,12d
;-----
bfill_routine:
        push    dx
        push    di
        mov     bx,[bp]+10
        mov     di,[bx]          ; di,ilk GCadresesi
        mov     bx,[bp]+8
        mov     dx,[bx]          ; dx,son GCadresesi
        mov     bx,[bp]+6
        mov     bl,[bx]          ; bl,doldurulan veri
        int     xx
        pop     di
        pop     dx
        pop     bx
        pop     ax
```



```

        pop     bp
        ret     8
;-----
maske    db      7
;-----
maskeprg:
        push    bp
        mov     bp,sp
        push    bx
        push    ax
        mov     bx,[bp]+6
        mov     al,[bx]
        mov     cs:maske,al
        pop     ax
        pop     bx
        pop     bp
        ret     2
;-----
interrupt1:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        push    cx
        in      al,0E6H
        mov     cl,5
        shr     al,cl
        cmp     al,cs:maske
        jae     inret
        mov     al,0FFH
inret:
        mov     bx,[bp]+6
        mov     [bx],al
        pop     cx
        pop     bx
        pop     ax
        pop     bp
        ret     2
;-----
interrupt:
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        mov     bx,[bp]+8
        mov     al,[bx]
        mov     bx,[bp]+10
        mov     byte ptr [bx],255
; istek kabul edilmedi
```



```

    cmp     al,cs:seviye
    jb      kescik
; al > seviye ise
    mov     al,cs:seviye
    mov     [bx],al
; istek kabul edildi
    mov     bx,[bp]+6
    mov     al,[bx]
    mov     cs:seviye,al
kescik:
    pop     bx
    pop     ax
    pop     bp
    ret     6
;-----
seviye     db      7
kesic:
; Bu altprogram
; INT1-INT3 uçlarını
; sürekli olarak
; kontrol eder.
    push    ax
    push    cx
    in      al,0E6H
    mov     cl,5
    shr     al,cl
    cmp     al,cs:seviye
    jae     kesret
    mov     cs:seviye,al
kesret:
    pop     cx
    pop     ax
    iret
;-----
cntrl     endp
code      ends
;-----
st_seg    segment
          db      40 dup ('*')
st_seg    ends
;-----
end
```


6 - 8 - PASCAL Dili ile Çağırma

PASCAL da INTerrupt lara ,

- external komutu ile,
- intr " " ,
- inline " " erişmek mümkündür.

Aşağıda external komutu kullanılmışdır.

PROGRAM

Giris_Cikis_Birimlerini_Kontrol_Eden_Altprogramlar;

procedure kontrol;

external 'PREPAS.COM';

(*****)
(* Kullanılacak altprogramlar buradan itibaren başlamaktadır. *)
(* Kullanılan her altprogramın başına s harfi eklenmiştir. *)
(*****)

procedure sreset (cihaz : byte);
external kontrol[0];

function sread(cihaz : byte;
adres : integer): byte;
external kontrol[3];

procedure swrite(kontrol_degeri : byte;
adres : integer;
veri : byte);
external kontrol[6];

procedure sbcomp(cihaz : byte;
bel_seg,
bel_adr_ilk,
gc_adr_ilk,
gc_adr_son : integer;
var farkli_bel,
farkli_gc : integer);
external kontrol[9];


```
procedure sbread( cihaz : byte;  
    bel_seg,  
    bel_adr_ilk,  
    gc_adr_ilk,  
    gc_adr_son : integer);  
    external kontrol[12];
```

```
procedure sbwrite( kontrol_degeri : byte;  
    bel_seg,  
    bel_adr_ilk,  
    bel_adr_son, veri : integer);  
    gc_adr_ilk : integer);  
    external kontrol[15];
```

```
procedure sbfill( kontrol_degeri : byte;  
    gc_adr_ilk,  
    gc_adr_son,  
    doldurulacak_veri : integer);  
    external kontrol[18];
```

```
(*****  
(* Ready hattı ile ilgili altprogramlar *)  
*****)
```

```
function srread( cihaz : byte;  
    adres : integer):byte;  
    external kontrol[21];
```

```
procedure srwrite( cihaz : byte;  
    adres : integer;  
    veri : byte);  
    external kontrol[24];
```

```
procedure srbcomp( cihaz : byte;  
    bel_seg,  
    bel_adr_ilk,  
    gc_adr_ilk,  
    gc_adr_son : integer;  
    var farkli_bel,  
    farkli_gc :integer );  
    external kontrol[27];
```

```
procedure srbread( cihaz : byte;  
    bel_seg,  
    bel_adr_ilk,  
    gc_adr_ilk,  
    gc_adr_son : integer);  
    external kontrol[30];
```



```
procedure srbwrite( cihaz : byte;
                  bel_seg,
                  bel_adr_ilk,
                  bel_adr_son,
                  gc_adr_ilk : integer);
    external kontrol[33];
begin
procedure srbfill( cihaz : byte;
                  gc_adr_ilk,
                  gc_adr_son,
                  doldurulacak_veri : integer);
    external kontrol[36];

    (*****
    { ----- Düzeyli Kesme altprogramı ----- }
    { kabul% dönüş parametresi 255 ise kesme isteği yok }
    { duzey% istenen düzey giriş parametresi.Küçük olanlar kabul edilir }
    { seviye% kabul edilmiş ise bundan sonraki yeni düzey bilgisi girişi }
    {*****

    (*****
    {*** SMASK altprogramı ile maske yerleştirilir. ***}
    {*** SIRQ1 altprogramı ile o andaki istek bilgisi alınır. ***}
    {*** maske ve istekli altprogramları birlikte kullanılır. ***}
    {*** ----- ***}
    {*** INTO - INT2 girişleri saniyede 18 kez kontrol edilerek ***}
    {*** bilgi alınır. Bu bilgiler arasında değer olarak en ***}
    {*** küçük sayı tutulur. SIRQ2 altprogramı ile, bu sayı ***}
    {*** düzey ile karşılaştırılır, küçük veya eşit ise kabul ***}
    {*** e aktarılır ve seviye yeni sayı değeri olarak belleğe ***}
    {*** alınır. Kabul edilmemiş ise kabulün değeri 255 dir. ***}
    {*****

procedure smask ( duzey : byte);
    external kontrol[39];

procedure SIRQ1( kabul : byte);
    external kontrol[42];

procedure SIRQ2 ( kabul,
                  duzey,
                  seviye : byte);
    external kontrol[45];
```



```
procedure interrupt;
{ Bu procedure, SIRQ2 procedure'u
  kullanılacak ise programın en başında
  birkez çağrılır.
  }
var ip:integer;
begin
  ip:=ofs(kontrol)+48;
  memw[0:$70]:=ip;
  memw[0:$72]:=cseg;
end;
```

```
{ Ana program }
BEGIN
```

```
{ Yukarıdaki altprogramlar,
  burada yazılacak programlar
  içerisinde kullanılabilir.
}
```

```
END.
```

```
{ Program sonu }
```

```
Secikinin kullanıma göre bölümler WRITE, WRITE ve
BFill altprogramlarıdır.
```

```
Secikinin süresi aşağıdaki gibi ayarlanır:
```

```
{ Secikis = ( 2 * S ) * 3.8 us }
```

```
S sayısı, 63..80 değeridir.
```

```
4.77Mhz'lik CPU da minimum 3.2 mikrosaniye,
maksimum 125 mikrosaniye dir.
```

```
void reset(char Cihaz_no);
```

```
unsigned char read(char Cihaz_no,
int BCadres);
```

```
void write(char Cihaz_no,
int BCadres,
char Yazilacak_byte);
```


6 - 9 - C Dili ile Çağırma

/* Aşağıda C dili altprogramlarının kullanım şekli görülmektedir. Altprogramlar C dili içerisinde kullanılabilir. Ancak yazılan program, derleme sırasında ARABAGC.OBJ programı ile birlikte derlemeye sokulmalıdır.

Birbaşka yol ise C dilinden DOS.H ı kullanarak interruptları çağırma dır. Bunun içinde daha önceden verilen simgesel dilden çağırmadaki altprogramlar kullanılabilir.

Cihaz_no	G3	G2	G1	G0	S3	S2	S1	S0
Bitler	D7	D6	D5	D4	D3	D2	D1	D0

S3..S0 : Bu 4 bit 0 ile 15 arasındaki cihaz biriminin numarasıdır.

G3..G0 : Bu 4 bit yazma işleminin yapıldığı modlarda geçerli olan gecikme süresini tayin eder.

Gecikmenin kullanıldığı bölümler WRITE, BWRITE ve BFILL altprogramlarıdır.

Gecikmenin süresi aşağıdaki gibi ayarlanır.

{ Gecikme = (2 ^ G) * 3.8 µs }

G sayısı, G3..G0 değeridir.

4.77Mhz lik CPU da minimum 3.8 mikrosaniye, maksimum 125 milisaniye dir.

*/

```
void reset(char Cihaz_no);

unsigned char read(char Cihaz_no,
int GCadres);

void write(char Cihaz_no,
int GCadres,
char Yazilacak_byte);
```



```
int      bcomp(char      Cihaz_no,
                    int      Bellek_segmenti,
                    int      Bellek_baslangic_adresi,
                    int      GCadres_baslangici,
                    int      GCadres_sonu);
/*
Blok karşılaştırma fonksiyonun sonucu 0 dan farklı ise,
Bellek ile GCbirimi verileri arasında fark vardır.

Dönüşde farklı olan adresler ise,
Bellek_baslangic_adresi ile GCadres_baslangici
değişkenlerinde bulunur.
*/
```

```
void      bread(char      Cihaz_no,
                    int      Bellek_segmenti,
                    int      Bellek_baslangic_adresi,
                    int      GCadres_baslangici,
                    int      GCadres_sonu);
```

```
void      bwrite(char      Cihaz_no,
                    int      Bellek_segmenti,
                    int      Bellek_baslangic_adresi,
                    int      Bellek_sonu,
                    int      GCadres_baslangici);
```

```
void      bfill(char      Cihaz_no,
                    int      Bellek_baslangic_adresi,
                    int      Bellek_sonu,
                    int      Doldurulacak_byte);
```

```
unsigned char      rread(char      Cihaz_no,
                    int      GCadres);
```

```
/*
READ with READY
Dönüşde BL registeri okunan byte değerini içerir.
*/
```



```
void rwrite(char Cihaz_no,
              int   Yazilacak_byte,
              int   GCadres);
/*
    WRITE with READY
*/

void rbcomp(char Cihaz_no,
              int   Bellek_segmenti,
              int   Bellek_baslangic_adresi,
              int   GCadres_baslangici,
              int   GCadres_sonu);
/*
    Blok karşılaştırma fonksiyonun sonucu 0 dan farklı ise
    Bellek ile GCbirimi verileri arasında fark vardır.

    Dönüşde farklı olan adresler ise,
    Bellek_baslangic_adresi ile GCadres_baslangici
    değişkenlerinde bulunur.
*/

void rbread(char Cihaz_no,
             int   Bellek_segmenti,
             int   Bellek_baslangic_adresi,
             int   GCadres_baslangici,
             int   GCadres_sonu);
/*
    BLOCKREAD with READY
*/

void rbwrite(char Cihaz_no,
              int   Bellek_segmenti,
              int   Bellek_baslangic_adresi,
              int   Bellek_sonu,
              int   GCadres_baslangici);
/*
    BLOCKWRITE with READY
*/

void rbfill(char Cihaz_no,
            int   Bellek_baslangic_adresi,
            int   Bellek_sonu,
            int   Doldurulacak_byte);
```


7 - SONUÇ

Tez çalışmasında, kişisel bilgisayarın iç yapısını, ek ünitelerini, yazılımını ve çalışmasını araştırıp buna göre tasarlanacak kartın özelliklerini belirlemek oldukça uzun bir süreyi almıştır.

Elektronik devrenin gerçekleştirilmesinde, ilk olarak protoboard üzerinde sürekli çalışmalar yapıldığından, iletken tellerin uzunlukları, çalışma frekansı, gürültü gibi iç ve dış etkenler devrenin çalışmasına en büyük engeli oluşturdu.

Daha sonra devre tasarımının bitirilip, çift taraflı delik içi kaplamalı baskılı devrenin dizayn edilip piyasada kartın bastırılması bir başka zorluk olarak karşımıza çıktı.

— Semiconductor Data Book HD24LS Series

Sonuçta bütün bu sorunlar halledilip, pratikte de bu çalışmanın gerçekleştirilmesi sağlandı.

Fakat bundan sonra yapılacak çalışmaların bu tez çalışmasının bir devamı olacağı düşünülür ise, eklenecek yeni devrelerin ve uygulamaların yapımında böyle-sine hem donanım ve hem de yazılım aşamasında en azından bu tip sorunlarla karşılaşılmayacaktır.

Çalışmada adı geçen Kontrol Birimi ve ARABAĞ Sistemi gerektiğinde bir uygulamada, yazılım ile birleştirilip, istenir ise biri veya her ikisi de hem gerçek zamanlı olarak hem de gerçek zamanlı olamayan bir çalışmaya iyi bir yazılımla yönlendirilebilir.

Uygulama alanları düşünülür ise bir bilgisayara yazılabilecek programların sayısı kadar çoktur. Böylece bir kişisel bilgisayarı, artık bilgisayar kontrollu uygulamalar alanına yönlendirmek mümkün olabilmektedir.

8 - YARARLANILAN KAYNAKLAR

- IBM PC/XT Technical Reference Guide
- IBM Dos 3.2 Reference
- Microsoft Dos 2.00
- Multi I/O Card,
Installation and Operation Guide for The IBM PC
- Multi I/O Card User's Manual
- Monochrome-Graphic/Printer
- Semiconductor Data Book HD74LS Series
- Mos Data Book
- Turbo C User's Guide
- Turbo C Reference Guide

9 - ÖZGEÇMİŞ

1963 yılında İstanbul Bebek de doğdu. Bebek Tevfik Fikret ilkokulu ve Bostancı Ortaokulundan sonra Maçka Endüstri Meslek Lisesi Elektronik Bölümünden mezun oldu. 1981 yılında Yıldız Üniversitesi Mühendislik Fakültesi Elektrik bölümüne girdi ve 1985 yılında bu bölümden mezun olduktan sonra aynı yıl Yıldız Üniversitesi Fen Bilimleri Fakültesi Bilgisayar Bilimlerinde yüksek lisans eğitimine başladı.

