YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

80C32 MİKROKONTROLÖR TABANLI OTOMATİK TEST CİHAZI TASARIMI

Elektronik.Müh **Kenan ENGİN**

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

hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof.Şefik SARIKAYALAR

İSTANBUL ,1996

İÇİNDEKİLER

TEŞEKKÜR.

Bu tez Marmara Elektronik Limited bünyesinde , uzun ve yoğun mühendislik emeği gerektiren çalışmalar sonucu gerçekleştirilebilmiştir.

Bu meyanda anlayış ve yönlendirmelerini esirgemeyen hocam Prof. Şefik SARIKAYALAR' a , çalışmaların yürütülebilmesi için gerekli tüm imkanları sunan Bilgisayar Y.Mühendisi Sn.Can ÖZKAN ' a ve yazılım esnasında kıymetli katkılarından dolayı Elektronik Y.Mühendisi Sn. Ayhan GÜNEŞDOĞDU' ya teşekkürü borç bilirim.

ÖZET.

Otomatik test cihazları pazarı gittikçe çok daha fazla kullanım imkanına sahip olmaktadır.Bu gelişme dikkate alınarak temel anlamda gerçek bir otomatik test cihazının tasarımı bu tezin amacını oluşturmuştur.

Cihazın ana kartında mikrokontrolör olarak MCS51 ailesinden 80C32 kullanılmıştır.ADC, Real Time Clock ve Seri haberleşme üniteleri gibi temel donanım birimlerine sahip olan ana kart aslında genel amaçlı bir geliştirme kartı hüviyetinde olup piyasada mühendislik bazında yapılacak her türlü araştırma ve geliştirme çalışmalarını tam anlamıyla destekleyebilecek kapasitededir. Bunun için standart bir Seri Haberleşme programı kullanarak ana kartın PC ile haberleşmesi sağlanmış ve araştırma esnasında en hızlı bir şekilde yazılım çalışması yapmaya uygun hale getirilmiştir.

Tez tamamen pratik bir çalışma olduğu için deneysel çalışma da bir 3 fazlı motor kontrol kartının fonksiyonel olarak test edilmesi amaçlanmıştır. Seçilen hedef kartın standartlara uygun bir haberleşme birimi olmaması dolayısıyla detaylı bir arabağlaşım (interface) ünitesinin tasarımını gerçekleştirmek te zorunlu olmuştur.

Ana kart sabit kalmak şartıyla ilave donanım çalışmalarını en aza indirgeyerek sadece interface ünitesinde temel değişiklikler yapılarak bu cihazın farklı kartları test edebilme imkanı mümkündür.Piyasadaki çok kapsamlı otomatik test cihazlarına kıyas edildiğinde servis verebildiği kart sayısının bir adet olmasına karşılık bu cihaz fiyat/performans kriteri açısından doyurucu özelliklere sahiptir.Aynı zamanda yalnız direnç ölçümüne dayalı bir test tekniği kullanılmayıp gerçek anlamda fonksiyonel bir test tekniği uygulandığı için hatasız test sonuçlarına varmak mümkün olmaktadır.

Sistemin donanım yapısı gereği cihaza I/O kartlarını adapte etmek kolaydır.Bu şekilde hizmet sunabildiği kart adedi rahatlıkla artırılabilir.Cihazın aynı zamanda farklı amaçlar için kullanıma uygun olması ve donanım problemini standart gerekler açısından ortadan kaldırmış olması kayda değer özellikleri arasındadır.

ABSTRACT

The market of automatic test equipments is getting more and more bigger and gains wide spread applications. Considering this assesment , the scope and goal of this thesis is to realize a real automatic test equipment.

80C32 microkontroller from the family of MCS51 is used on the main board. Having essential hardware units such as ADC,Real Time Clock and Serial Communications Main Board may be considered as an general purpose emulator and is capable of supporting all research and development requirements. in order to accomplish this feature , communications between main board and PC is provided via a serial com. software and so the feature of functioning at high speed is given to the main bord.

For the thesis is an practical study , the testing of a 3-phase motor control print is selected as an experimental study. Designing of a complex interface unit between main bord and target board becomes necessary because target board has no any standart communications unit.

This equipment can test various prints by redesigning interface unit .Thus additional hardware requirements were significantly reduced. Although the equipment can only service one print , It has satisfying features in respect of cost/ performance criteria compared with current automatic test equipments in the market.

At the same time Since a real functional test technique is applied by not using a passive test approch based on resistive measurement it is possible to get reliable result.

As a consequnt of architecture of the system it is easy to insert I/O modules.Hence the number of serviceable prints may increase.The fact that the equipment can easly suit various purpose and that it can eliminate hardware problem in regard of standart requirements are among the other significant features.

BÖLÜM 1. GİRİŞ

Teknolojinin hızlı değişimine ayak uydurmak , işletmelerin sürekliliğini koruyacak bir zorunluluktur.Aksi taktirde birleşen pazarlar,serbest piyasa ekonomisi değişime uymayan işletmeleri hazin bir sona taşıyacaktır.Bu noktada hedef teknolojinin sunduğu en son yenilikleri bilmek ,teknik elemanları eğiterek bilgi çağına uyum sağlamak , üretkenliklerini artırırken baskım maliyetlerini düşürerek işletmelerin performansını yükseltmektir.

Bakım maliyetlerinin azaltılması sahasında da ilerleyen teknolojik imkanlar sayesinde klasik tamir ve bakım metodlarından çok farklı ve kesin sonuç verebilen anlayışlar gelişmiş bulunmaktadır.Her türlü elektronik kart tamirini mümkün kılabilen arıza tesbit zamanını azaltan, elemanların rezistif karakteristiklerini depolayarak daha sonradan tekrar kullanım ve karşılaştırma imkanı veren cihazlar halihazırda piyasada kullanılmaya başlamıştır.Yapılan istatistikler sonucunda kart üzerinde arızayı bulma zamanı , verimlilikte artışla birlikte % 90 oranında azalmıştır. Servis süresini en aza indiren bu yeni arıza arama ve test sistemi , sağlam kartların orjinal test değerlerini depolayarak ve gerektiği zaman kullanarak devre şeması yada yedek karta ihtiyaç duymaksızın bozuk kartların onarımı imkanını sağlamaktadır.

1.1 GELİŞMİŞ BAKIM SİSTEMLERİ VE TEKNOLOJİK GELİŞMELER

1.1.1 BAKIM YÖNTEMLERİ :

Planlı ve plansız bakım yöntemleri üç alt başlıkta toplanabilir.

1.1.2 Plansız Bakım :

Arıza çıktıkça bakım adı ilede nitelenebilen bu yöntemde bakım arıza ortaya çıktıktan sonra yapılmaktadır.Kendi haline bırakılan bir sistemde çıkabilecek arıza başka arızalarada yol açacağı için ve üretim hattında kesintilere yolaçacağından dolayı ekonomik olarak bu yöntem işlerliğini kaybetmiştir.Ancak işletme açısından önemli olmayan ya da seri üretim hattında yer almayan makinalar için bu yöntem uygulanabilir.

1.1.3 Planlı Bakım Teknikleri :

Koruyucu Bakım (Preventive Maintanence)

Koruyucu Bakım Yöntemi modern bir sistem olarak bir önceki yönteme alternatif olarak ortaya çıkmıştır.Zaman içinde bu tarz sisteme çeşitli bakış tarzları geliştirildiği için iki başlık altında ele alınmaktadır.

Periyodik Bakım :

Bu sistemin ana teması belirlenecek periyotlarda bakım yapılması , parça değiştirilmesi ya da eklemelerin gerçekleştirilmesidir. Sistemin görünen dezavantajı sistemde arıza yapmamış ve daha uzun süre iş görebilecek parçaların önceden gereksiz yere değiştirilmesidir. Bu da haliyle üretim kaybına , gereksiz parça masrafına ve işçilik maliyetlerine yol açacaktır.

Bu noktada diğer bir olasılıkta makinanın tesbit edilen peryodundan önce de arıza çıkarabileceğin varlığıdır. Bu durumda sistem özelliğini kaybederek Arıza Çıktıkça Bakım sistemine dönüşmektedir.Bunu önlemek için bakım peryodu kısa tutulmak istenmekle beraber bu da işçilik/ parça/üretim kaybı maliyetlerini yükseltmektedir.

Ortaya çıkan bu sonuç işletmelerde ikileme yol açmaktadır.Çünkü sonuçta bu yeni sistem bir öncekinden daha masraflı bir hal almaktadır.Bu noktada bir adım daha ilerleme imkanına sahip olmayan işletmeler tekrar eski yöntemi seçmek zorunda kalmaktadırlar.

Bakım peryodunun belirlenmesi olayı oldukça göreceli bir kavram olmakla birlikte çok yoğun bir istatistiksel bilgi eldesi çalışması gerektirdiği için bir işletmede tam anlamıyla yerleşmesi çok uzun süreler almaktadır. Örnek olarak bir seramik fabrikasında bilgisayar destekli periyodik bakım sistemi 10 elemanla ancak 2 ya da 3 yıl zarfında rayına oturtulabilmekte ve sonuçlarıda 5 yıllık bir perspektifle

alınmaktadır.Ancak hiçbir işletmenin bu kadar uzun bir süreyi gözönüne alarak iş yapması ekonomik sayılamaz.Diğer yandan bu sistem için gerekli istatistiksel bilgiler mevcut olmadığı için ve nasıl elde edileceği imkanı olmadığından dolayı tüm yapılabilecek sistem kuruluş çalışmaları tamamen tecrübeye dayanmaktadır.Bundan dolayı elemanların tamamının fabrikayı çok iyi tanıyanlardan seçilmesi zorunluluğu vardır.Bu zorunluluk ise tecrübeli elemanların yalnız bu işe tahsisi yüzünden çok yüksek oranda verim kaybına yol açmaktadır.

1.2 GELİŞMİŞ BAKIM SİSTEMİ :

Gelişmiş bakım planlaması gelecek hakkında bugün ve geçmişten alınan ölçüm değerlerinin , data karşılaştırma yöntemi kullanılarak farkın hesap edilmesi ana temasına dayanmaktadır. Bu yönü ile ölçüme dayalı periyodik bakıma benzemektedir. Bakış açısından peryodik bakım mantığı uygulanmaz.Arıza çıkarabilecek sistem önceden ölçümler ile algılandığından geleceğe yönelik bir bakım onarım programı oluşturulmasını sağladığı gibi , doğabilecek arıza duruşlarına neden olan arızalarıda ortadan kaldırmaktadır.

Bilgisayar teknolojisinin gelişimi BİLGİSAYAR DESTEKLİ BAKIM (CAM) uygulamalarının çok yaygın bir şekilde kullanıma sunulmasına yol açmıştır.Bu sistem 1988 yılından itibaren hızlı bir gelişme göstererek günümüzde oldukça pratik , işletim maliyeti düşük bir konuma gelmiştir. İşletmede 2-3 kişi ile uygulanabilen bu sistem hali hazırdaki bakım yömteminide çok çabuk bir şekilde iyileştirilmesi amacını taşır.

İşletmede uygulanan bakım yöntemi ne olursa olsun gelişmiş bakım yönteminin devreye en az yatırım masrafı ile alınabilmesi en büyük avantajdır.Maxmimum 1 ay içinde sistem oturmakta ilk 6 aylık zaman zarfında tam olarak en yüksek randımanını vermeye başlamaktadır.Kısa zamanda halledilen arızalar ve elde edilen zaman kazancı ile yatırımın geri dönüşü süreci daha ilk uygulamalarda başlamaktadır.

1.2.1 PERYODİK BAKIM PLANINA GEÇİŞ AŞAMALARI

a) Programa alınacak sistemlerin listelenmesi

- Kritik sistem ve elektronik kartlrın tesbiti

- Elektronik kartkarın yerleşim planının ortaya çıkarılması

b) Kartların sistemdeki yerleşim krokisinin çizilmesi

c) Her sistem için ölçüm alınacak noktaların ve yönlerinin belirlenmesi

d) Belirlenen her kart için hangi ölçümün yapılacağının tesbiti

e) Her kart için geçerli gerilim ve frekans seviyelerinin tesbiti

f) Her kart için yalnız o karta mahsus olan karakterlerin çıkarılması

g)Her kartın ve malzemenin temel ölçüm değerlerinin alınması

h) Uygulama elemanlarının eğitilmesi

1.2.2 BİLGİSAYAR DESTEKLİ PERYODİK BAKIM BASAMAKLARI

1) Ölçüm kartlarının organizasyonu

2) Seçilen karta ait tüm bilgilerin bilgisayara aktarılması

3) Karttaki birinci noktadan başlayarak sistem üzerinden ölçümlerin alınması

4) Operatorün gerekli ölçümleri gerçekleştirmesi

5) Yapılan ölçüm sonuçlarının bilgisayar hafızasına kaydedilmesi

6) Bilgisayara yüklenmiş programlar dahilinde verilerin değerlendirilmesi

7) Bilgisayardan alınan rapor ve yapılan ddeğerlendirmeler sonucu gerekli bakım onarım programı yapılır ve onarım gerçekleştirilir.

BÖLÜM 2 . ANA (CPU) KART TANIMLAMALARI

2.1 DONANIM ÖZELLİKLERİ

Otomatik test cihazının ana kartı ve aynı zamanda başlıbaşına bir geliştirme sistemi olarakta kullanılabilen kısmı MCS51 mikroişlemciler ailesinden 80C32 mikrokontroleri üzerine kurulmuştur.. Ana kart ve Back panel'den ibarettir. +5V ile çalışır. Ana kartta bulunan tüm işlem kısımlarının listesi aşağıdaki listede verilmiştir.

27C64 8Kx8 Eprom
62C64 8Kx8 Ram
MC146818 Real Time Clock (RTC)
8255 PIO
ADC0808 ADC (ANALOG-DIGITAL CONVERTER)

tümdevreleri bulunmaktadır. Ayrıca EEPROM/NOVRAM soketi ve Liquid Crystal Display (LCD) konnektörü mevcuttur. Bu ikisi sonradan ilave edilebilir.
MC146818 8032'deki Ext.Int.0 'ı, ADC0808 Ext.Int.1 'I kullanır.
EEPROM/NOVRAM için bağlantı uçları aşağıdaki gibidir:

SK P1.4 (Seri senkronizasyon pini)
CE P1.5 (Chip enable pini)
DIO P1.6 (Seri data hattı)

LCD konnektörünün bağlantıları JP03 olarak verilmiştir.

2.2 ANA KART BELLEK HARİTASI

Ana karta ait program yazılırken MON51 Emülator programı kullanıldığı için bellek haritasıda bu programın gereklerine uygun bir şekilde düzenlenmiştir.

EPROM 2764 : 0000h - 1FFFh

RAM 6264 : 2000h - 3FFFh

REAL TIME CLOCK MC146818 : FE00h - FEFFh

Saniye : FEX0h

Saniye alarmı : FEX1h

Dakika : FEX2h

Dakika alarmı : FEX3h

Saat : FEX4h

Saat alarmı : FEX5h

Haftanın günü : FEX6h

Ayın günü : FEX7h

Ay : FEX8h

Yıl : FEX9h

Register A : FEXAh

Register B : FEXBh

Register C : FEXCh

Register D : FEXDh

LIQUID CRYSTAL DISPLAY : FF00h - FFFFh

Command adres : FFX0h

Read adres : FFX1h

Data adres : FFX2h

PIO 8255 : FD00h - FDFFh

Port A : FDX0h
Port B : FDX1h
Port C : FDX2h
Port K : FDX3h

ADC0808 : FC00h - FCFFh

4000h - FBFFh adresleri genişleme konnektörü üzerinden kullanılabilir. Kart içinde kullanılan adresler dışarıda kullanılamaz.

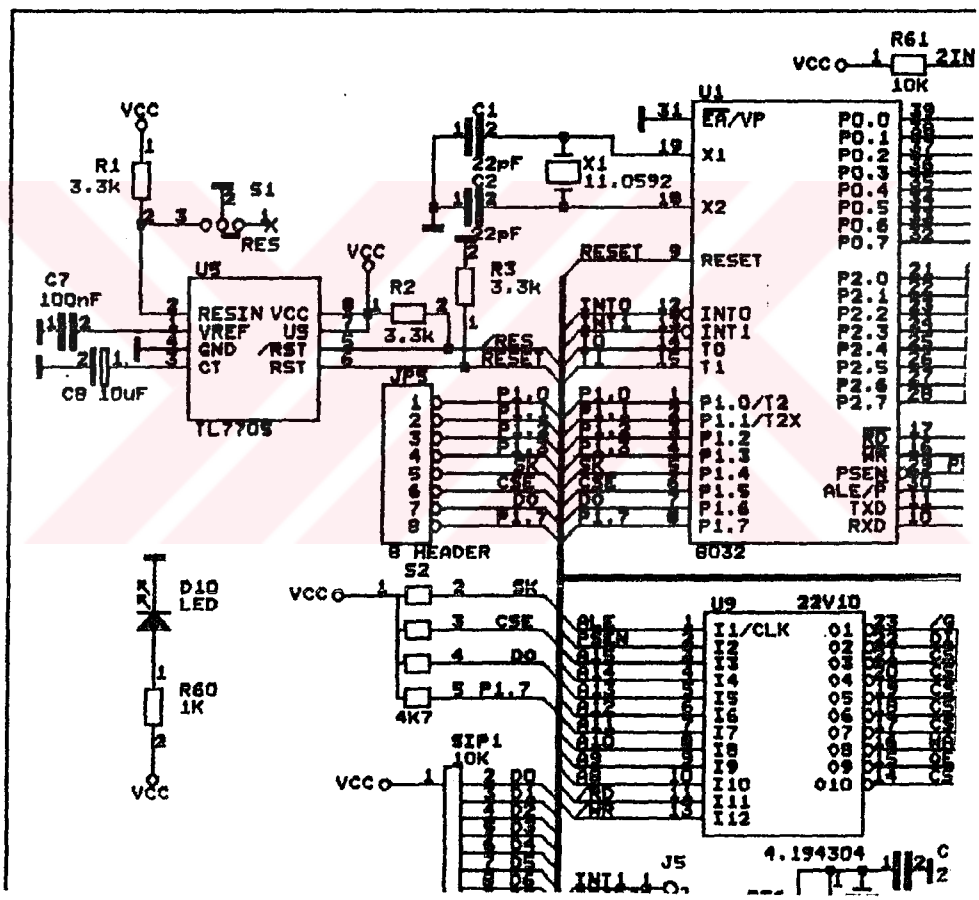
2.3 ANA KART İŞLEM ÜNİTELERİ

2.3.1 Reset Ünitesi

Ana kart reset ünitesinde, her koşulda sisteme güvenli ve doğru zamanlamaya sahip reset sinyali sağlayabilecek olan TL7705 entegresi kullanılmıştır. TL7705 mikroişlemcili tüm sistemlerde sıklıkla kullanılan ve yalnız bu fonksiyon için tahsis edilmiş bir supervisory elemanıdır.Bu gibi entegre elemanlar kullanarak normalde kompleks devre yapısı gerektiren güç kaynağı izleme devrelerden en az yer harcayarak kurtulmak mümkün olmaktadır.Aynı zamanda ayrık devre elemanların sahip olamadıkları sistem güvenilirliğinin artırılması bu gibi elemanlarla mümkün olmaktadır.

TL7705 2 nolu pinden 5 V kaynak gerilimini alır ve CT pinine bağlı kapasite değerine bağlı bir zaman süresince kontrollü olarak RESET sinyali üretir. 2 ayrı reset sinyali çıkış hattı vardır. Bu hatlar birbirinin evriği olarak çıkış verir.Aktif 0 reset çıkışı

için pull -up direnci , aktif 1 reset çıkışı için de pull-down direncinin kullanılması her zaman için düzenli çalışma açısından gerekmektedir.



Şekil 2.3.1 Ana Kart Reset Devresi

2.3.2 Ana Kart Adres Çözücü Devresi

Ana kart adres çözümleme kısmında ayrık elemanlar yerine PAL kullanılmıştır. Pal genel anlamda programlanabilir lojik devre anlamında kullanılmaktadır.Ayrık elemanların tüm sakıncalarını çok basit bir metodla ortadan kaldıran pal cihazları şu anda tamamen kullanım sahalarına hakim olmuş durumdadır.Çoğu standart paller genellikle OR kapısıyla takip edilen AND kapısı lojik devrelerinden oluşmaktadır.Giriş uçları istenen işlemi yapacak tarzda programlanmış AND kapılarıuna uygulanır ve sonuç elde edilir. Elde edilen bu değişik sonuçlar en son çıkışı oluşturmak için OR kapılarına uygulanır.

Mikroişlemcili sistemlerde pal kullanmanın avantajları dizayn kolaylığı, performans,güvenilirlik ve ucuzluk olarak görülebilir.Örneğin hız problemi dizaynçıları pal kullanımına sevkeden en büyük sebeplerden biridir.TTL pal devreleri mümkün ayrık elemanların sağlayabileceği en yüksek hız performansından daha fazlasını kullanıcıya sunabilmektedir.Aynı zamanda yüksek hızla birlikte güç tüketimi artmamaktadır.

Devrede kullanılan PAL22V10 en kullanışlı elemanalardan biridir.İkinci nesil PAL mimarisine sahiptir.22v10 aşağıdaki özelliklere sahiptir.

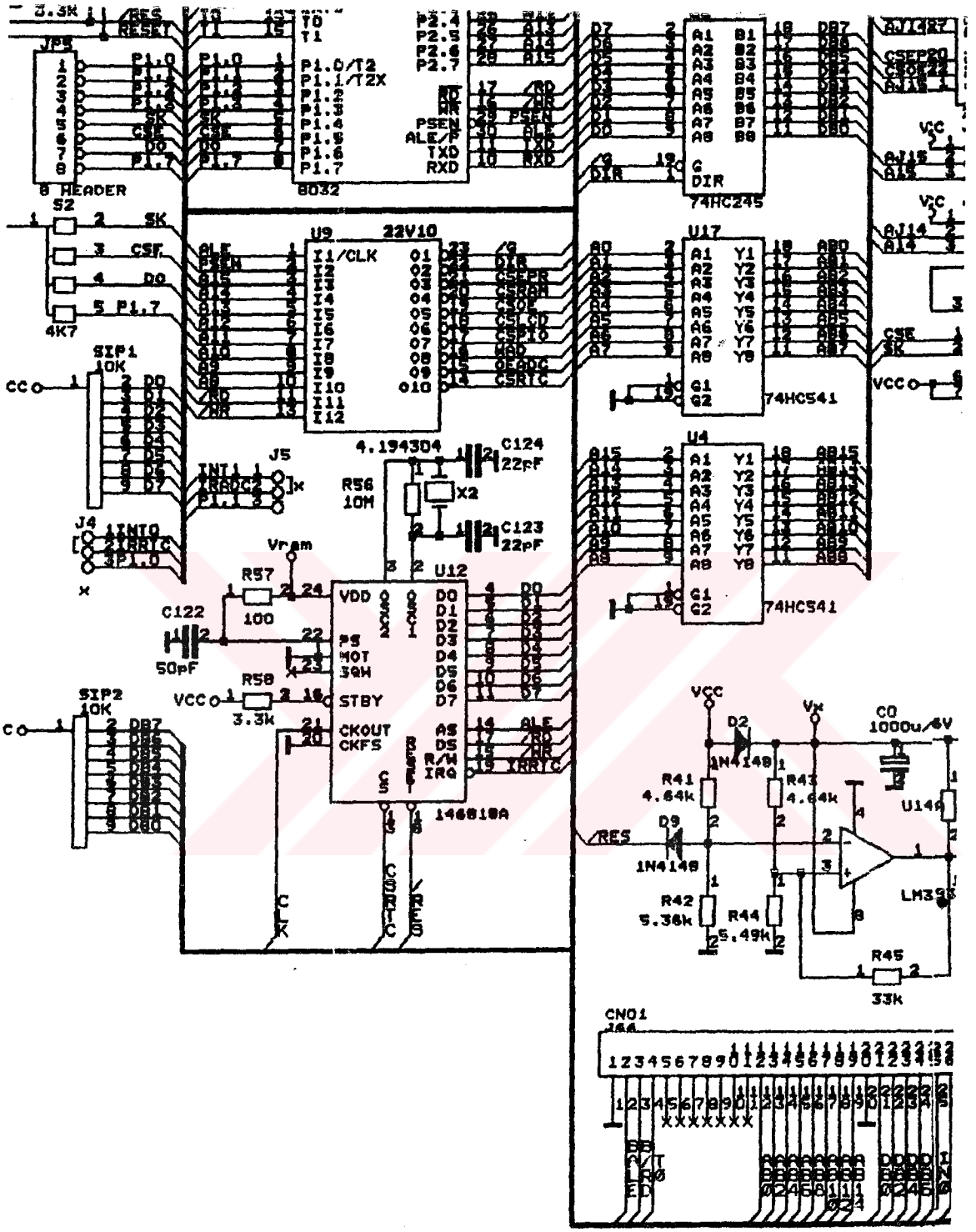
- Her bir çıkış fonksiyonu için 12 girişi kabul edebilecek bir kapasitede olup 24 pinlidir.

-Çıkışlar yalnız klasik lojik ifadelerle sınırlı olmayıp yazmaçlı veya kombinasyonel özelliklerde sunabilmektedir.İlave ifadelerle RESET Ve PRESET özellikleri verebilir.

- 18 ns tipik gecikme süresi ile yüksek hızda çalışma imkanı vardır.

- Özel sigorta yapısı ile izinsiz kullanıma ve okunmasına imkan vermeyerek yüksek güvenlik sağlar.

Ayrı başlık altında sunulmuş olan pal programındaki uygun adres seçimi ile ana kartın emulator modunda ve gerçek zaman çalışma modları arasında geçişi tekrar adres düzenlemesini gerektirmeden sağlanabilmektedir.Yalnız eprom programında basit bir adres değişimi ile gerçek zaman çalışma moduna geçiş sağlanır.



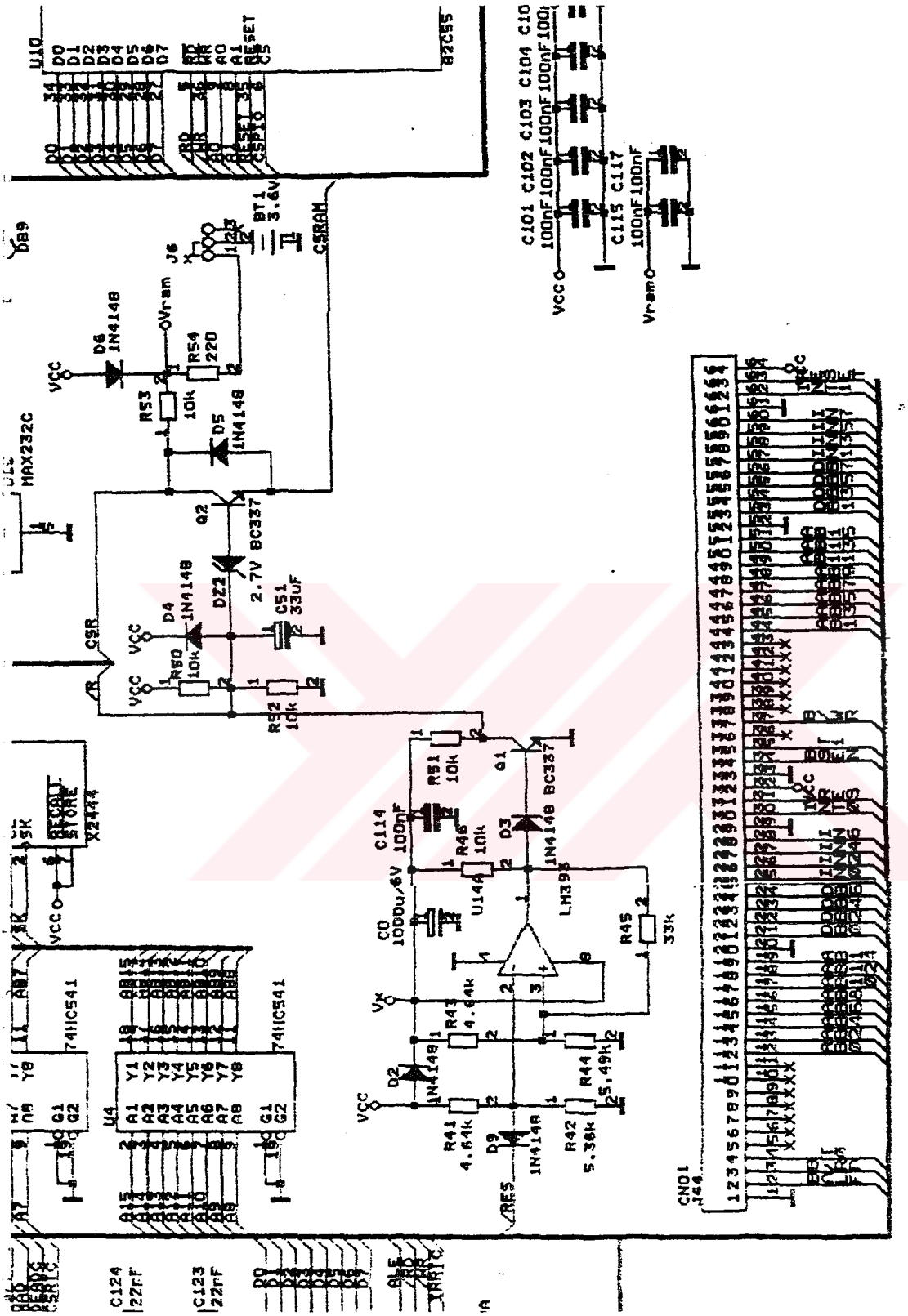
Şekil 2.3.2 Adres Çözücü Devresi

2.3.3 Ram Hafıza Koruma Devresi

Sistemin açılışı ve kapanışı esnasındaki kritik zamanlarda istenmediği halde bir yazma komutunun oluşması durumunda sistem statik Ram hafızasındaki gerçek zaman işlemlerine ait bilgilerin bozulması olasıdır.Bu tehlikenin önlenmesi için sisteme hardware olarak bir koruma devresi ilave edilmiştir. Bu devre sayesinde sistemin reset aldığı herhangi bir anda hafıza entegresi kesinlikle yanlış bir yazma komutu almayacak şekilde çalışma imkanı sağlanmıştır .Özellikle sürekli açma kapama ile suni olarak gürültü yapmakla denenen bu devre tam bir performans göstermiştir.

Devre temel olarak tek bir karşılaştırıcı üzerine kuruludur.bu karşılaştırıcıya Reset sinyali ve kaynak gerilimi sinyalleri giriş olarak verilmektedir.Devrede tam bir 5V olmadığı takdirde Reset sinyalinin transferi önlenerek yanliş yazmalara engel olunabilmektedir.

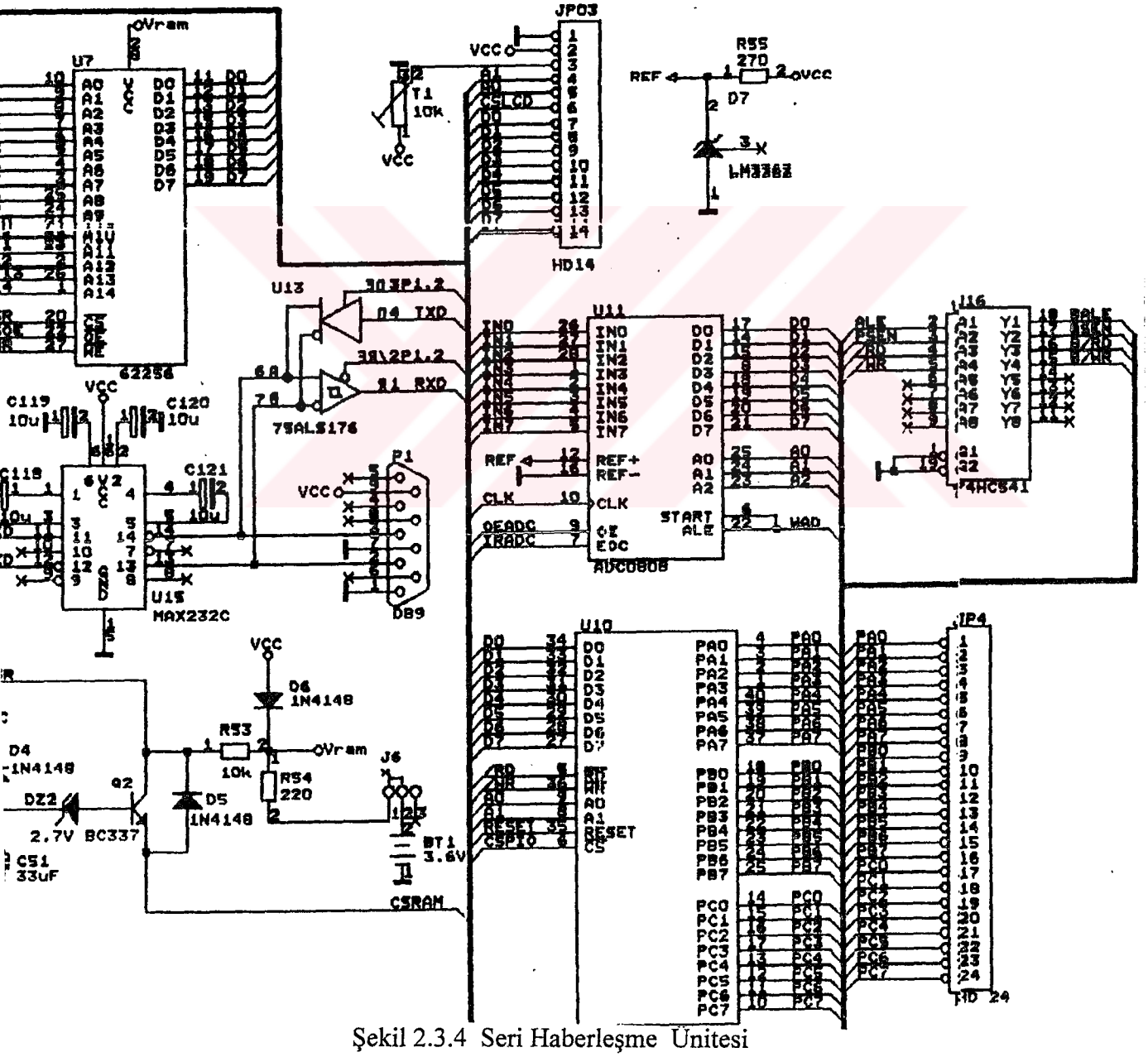
Bu devrenin çıkış sinyali ayrı bir kontrol ucu gibi kullanılarak batarya geriliminin de kontrolü sağlanmıştır.Q2 tranzistörü aracılığı ile ana gerilim kaynağı kesilir kesilmez bataryanın kesintisiz olarak devreye girmesi sağlanmaktadır.


U10
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
RESET
CS
82C55
BT1
3.6V
CSRAM
J6
VCC
D6
1N4148
Vram
RS4
22Ω
RS3
10k
D5
1N4148
MAX232C
Q2
BC337
DZ2
2.7V
D4
1N4148
CS1
33uF
CSR
RS0
10k
RS2
10k
RS1
10k
Q1
BC337
C114
100nF
R46
10k
D3
1N4148
U14A
LM393
CO
1000u/6V
R45
33k
Vx
R43
4.64k
R44
5.49k
D2
1N4148
R41
4.64k
D9
1N4148
R42
5.36k
CRES
X2444
RECALL
STORE
74HC541
U4
C124
22nF
C123
22nF
C101
100nF
C102
100nF
C103
100nF
C104
100nF
C115
C117
100nF
CN01
J64

2.3.4 Seri Haberleşme Ünitesi

Ana kart ile herhangi bir çevre biriminin seri olarak haberleşmesini sağlamak için

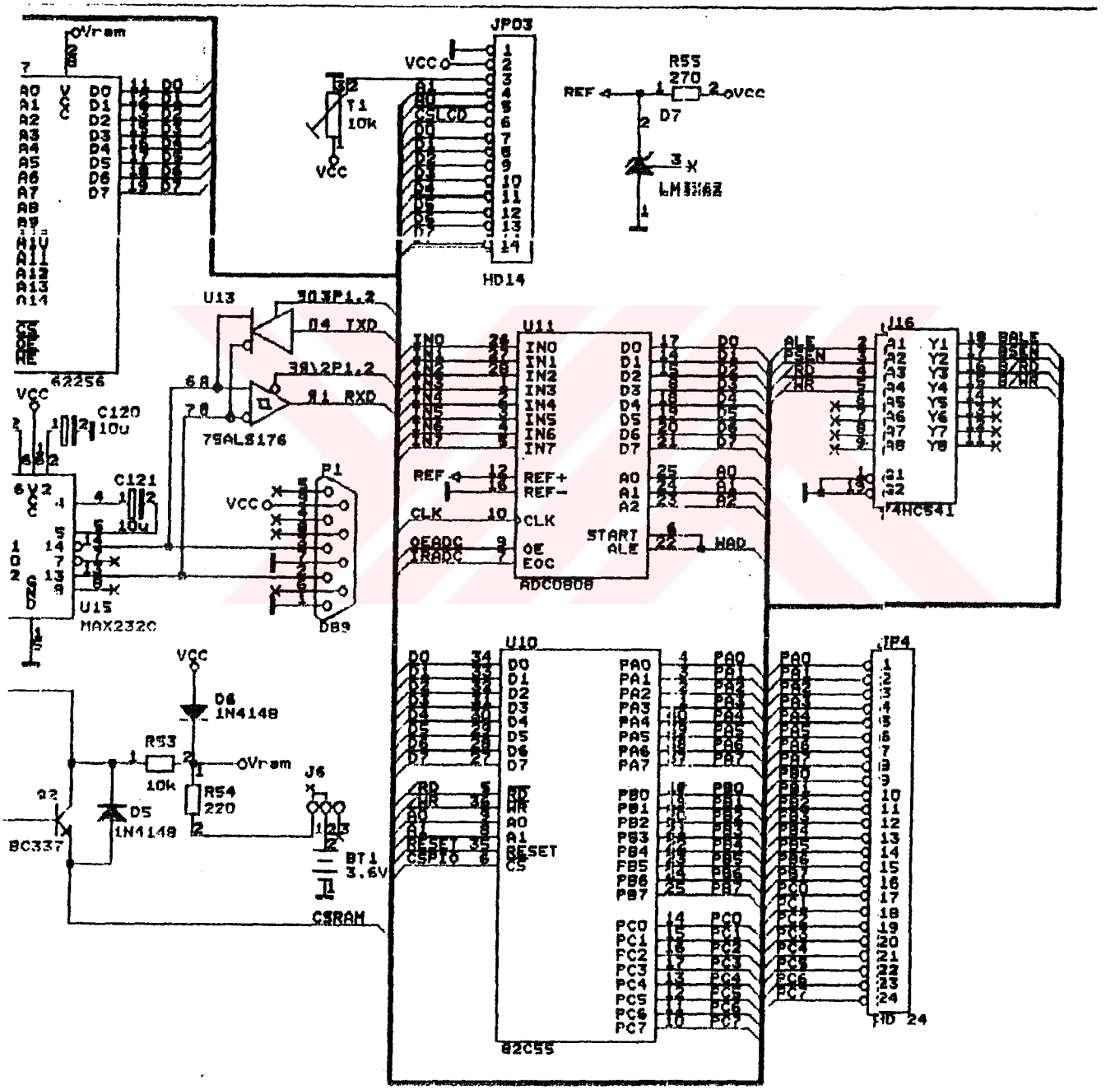
RS-232 veya RS-485 seri haberleşme standartlarını kullanarak gerkli hardware düzenlemesi yapılmıştır. Kart halihazırda RS-232 hattını kullanmakta olup 75176 gibi daha hızlı seri haberleşme elemanları ilavesi ile RS- 485'ün devreye hiçbir düzenleme gerektirmeksizin yalnız yazılımı geliştirerek dahil edilmesi rahatlıkla mümkündür.



Şekil 2.3.4 Seri Haberleşme Ünitesi

2.3.5 Analog- Digital Konverter Devresi

Sistemde 8 girişli bir ADC bulunmaktadır. Yazılım esnasında gerekli tüm hardware düzenlemeleri yapılarak ADC kullanıma direkt olarak hazır konumdadır.ADC nin tüm giriş hatları sistemi dış dünya ile irtibatlandıran CN1 64 lü konnektörüne verilerek hariçteki analog sinyallerin dijitale çevrilmesi mümkün olmaktadır.



Şekil 2.3.5 ADC Ünitesi

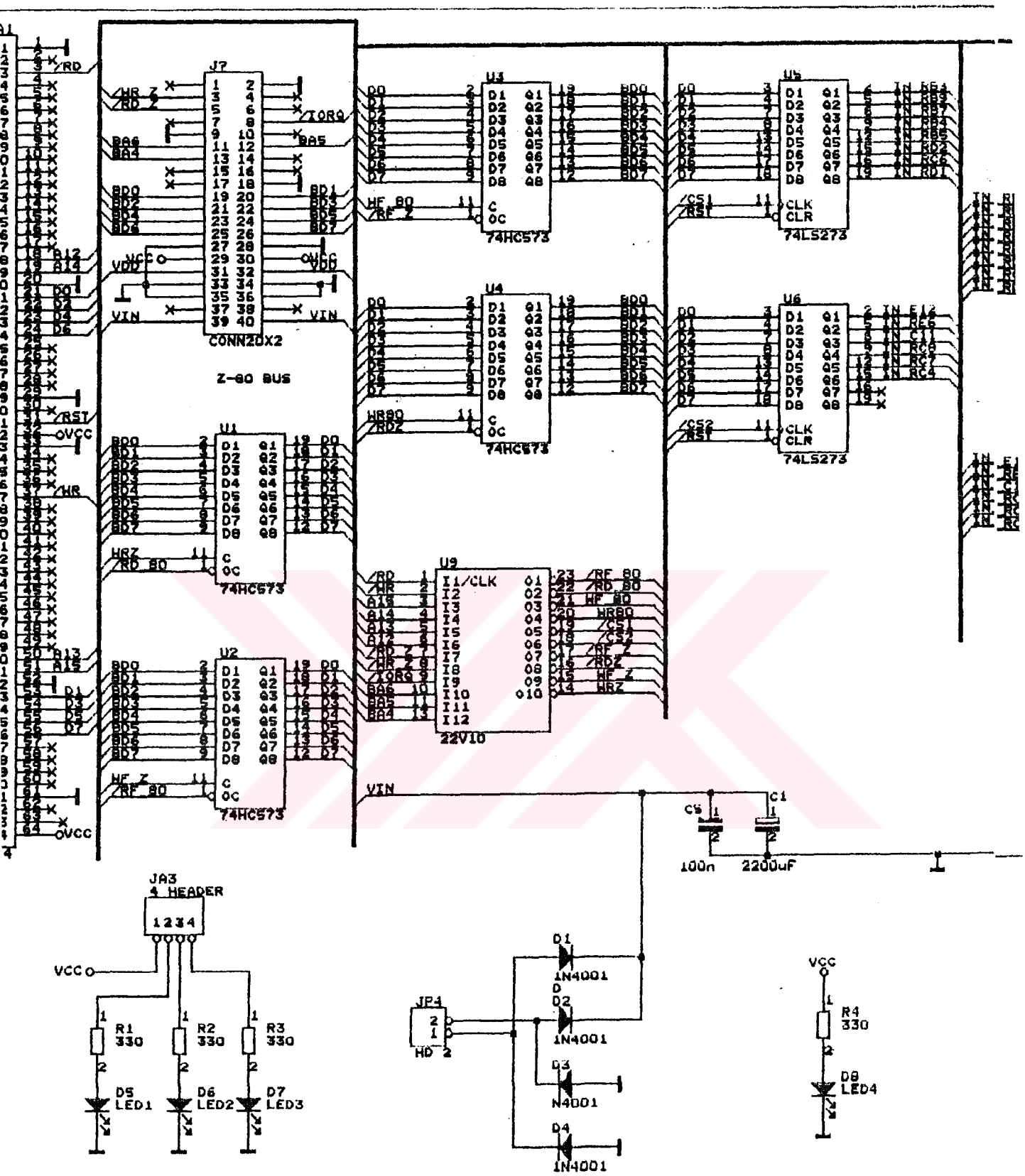
2.3.6 Hedef Kart Özellikleri ve Uygulama Notları

Ana karta bir uygulama örneği olarak 3 fazlı 12kw lık bir AC motor kontrol kartı seçilmiştir.Motor kontrol kartı test açısından analog ve dijital kısımlar başlıkları altında ele alınmış olup analog kısımda ADC devresi ,Counter-Tımer devresi ve Devir Sayacı bağımsız alt başlıkları, dijital kısımda ise ayrık komponent testleri başlıkları bulunmaktadır.

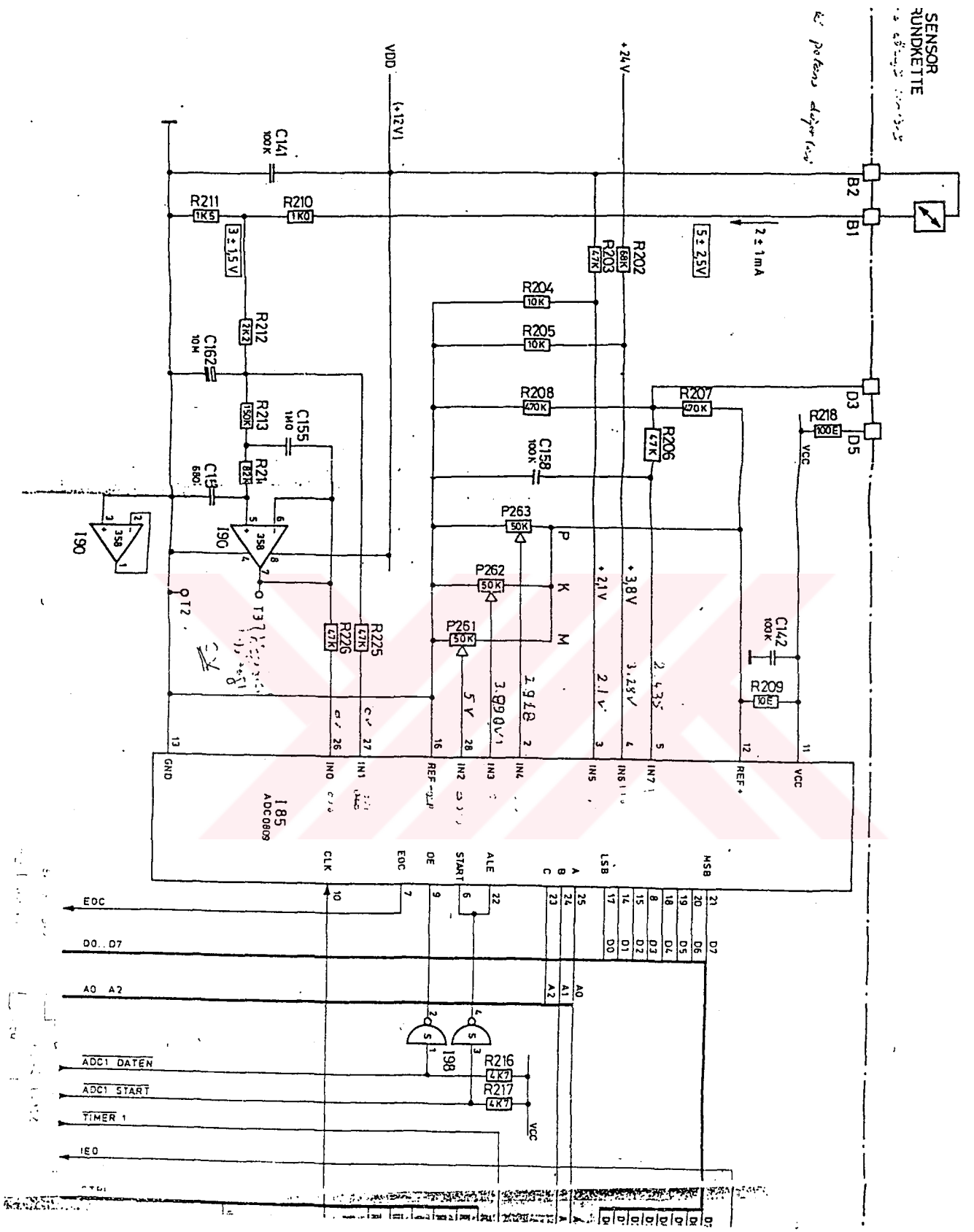
Sağlam olduğu bilinen kontrol kartı üzerinde yazılım çalışmaları yapılmış ve geiştirmeler esnasında suni arızalar çıkarak yazılımın son şekli verilmiştir.

Ana kart ile hedef kartın haberleşmesi ise özel bir interface ünitesi üzerinden gerçekleştirilmiştir. Anakartta standart olarak seri veya paralel olsun her türlü haberleşme için donanım imkanı sağlanmış olmasına karşılık hedef kartın haberleşmeye yönelik bir birimi olmamasımndan dolayı özel ve sırf bu karta mahsus bir interface tasarımına mecburen gidilmiştir. Çünkü hedef karta sistemin dış dünyaya açılabilecek hiçbir imkanı hazır olarak yoktur.

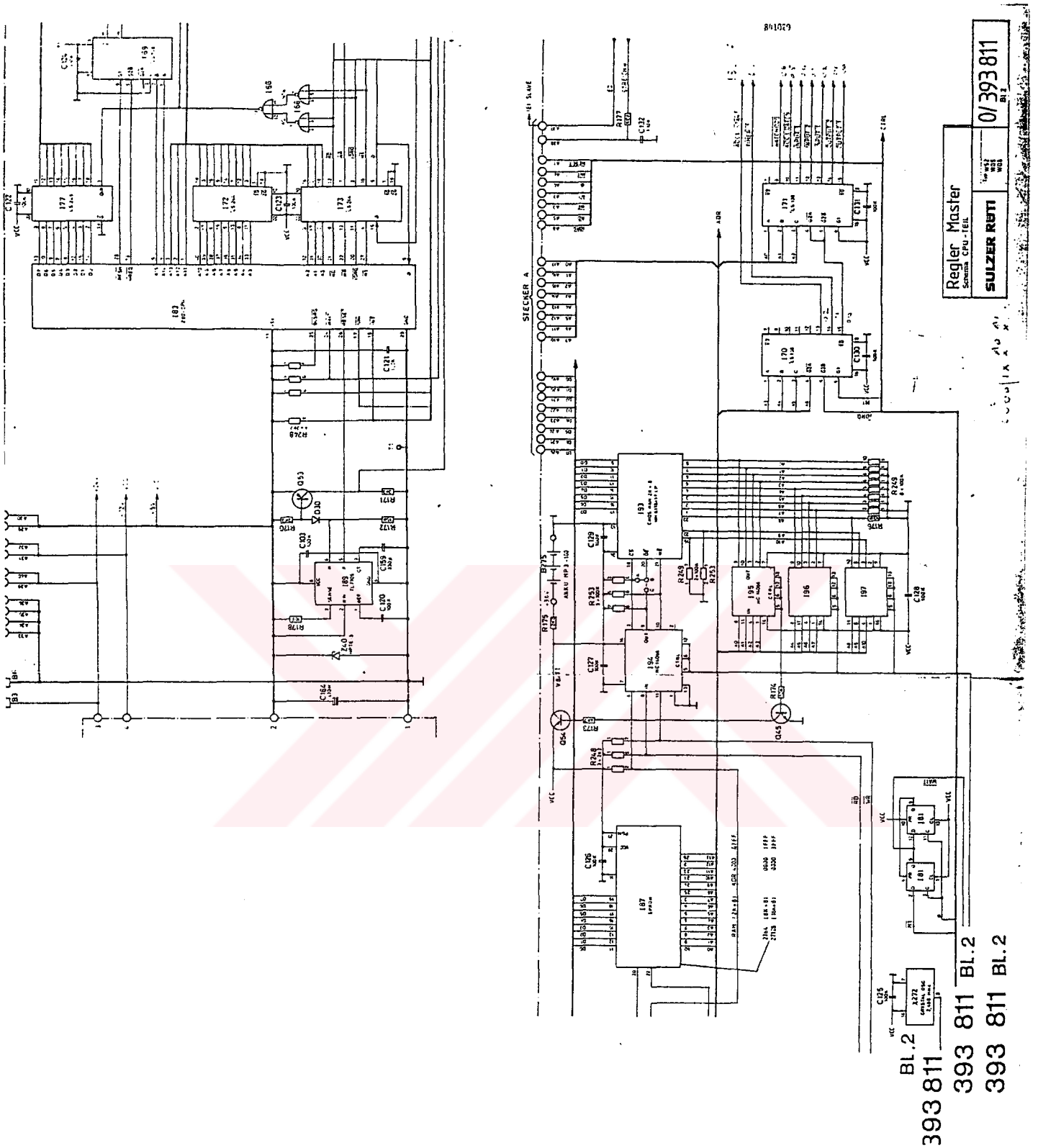
Ayrı olarak interface devresi aşağıda sunulmuştır .Bunun yanısıra hedef kartta bulunan CTC ünitesi,WM interface ünitesi,Dijital giriş kısmı ADC ve CPU devreleri ayrı ayrı verilmektedir.



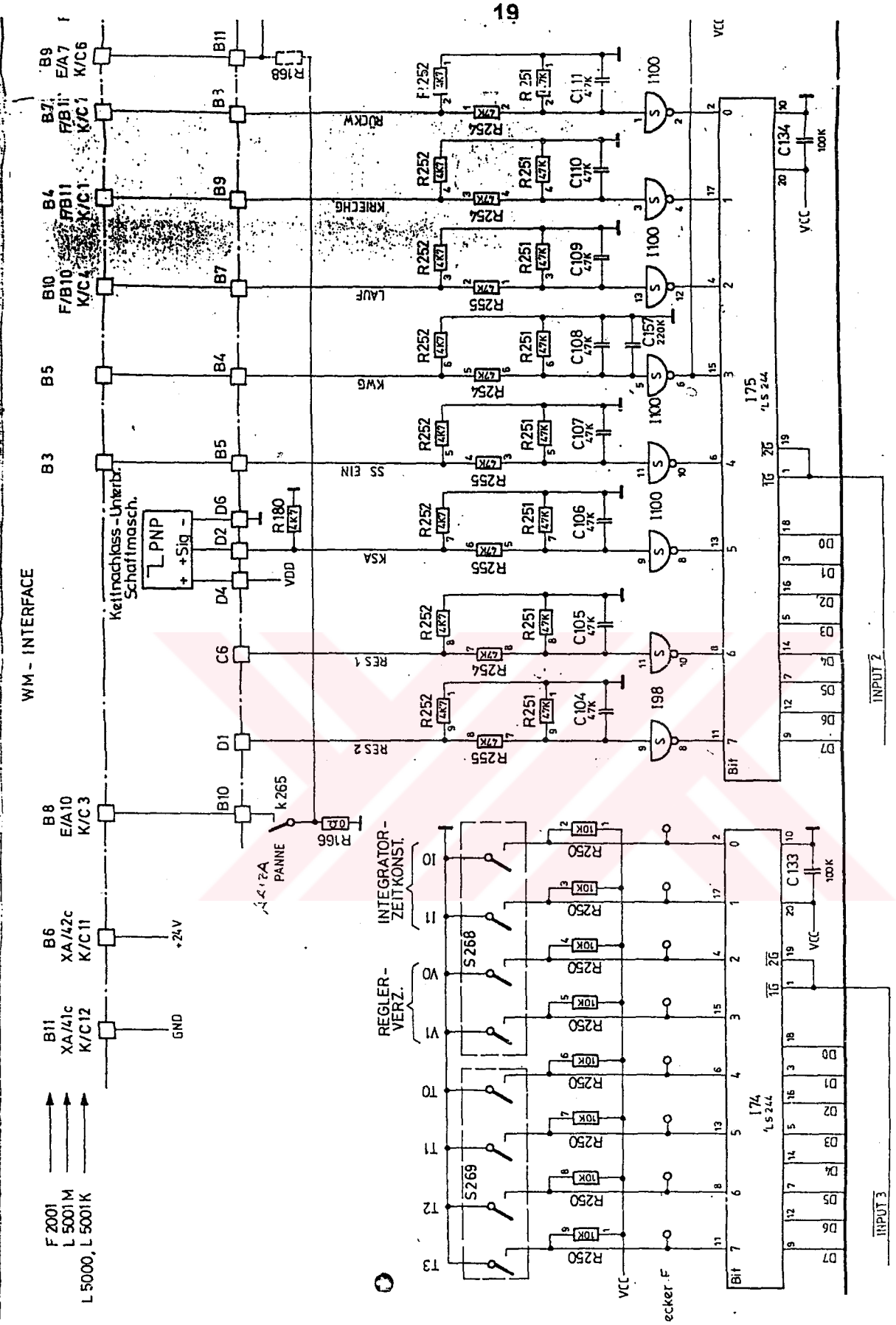
Şekil 2.3.6.1 Ana kart , Hedef kart İnterface Ünitesi



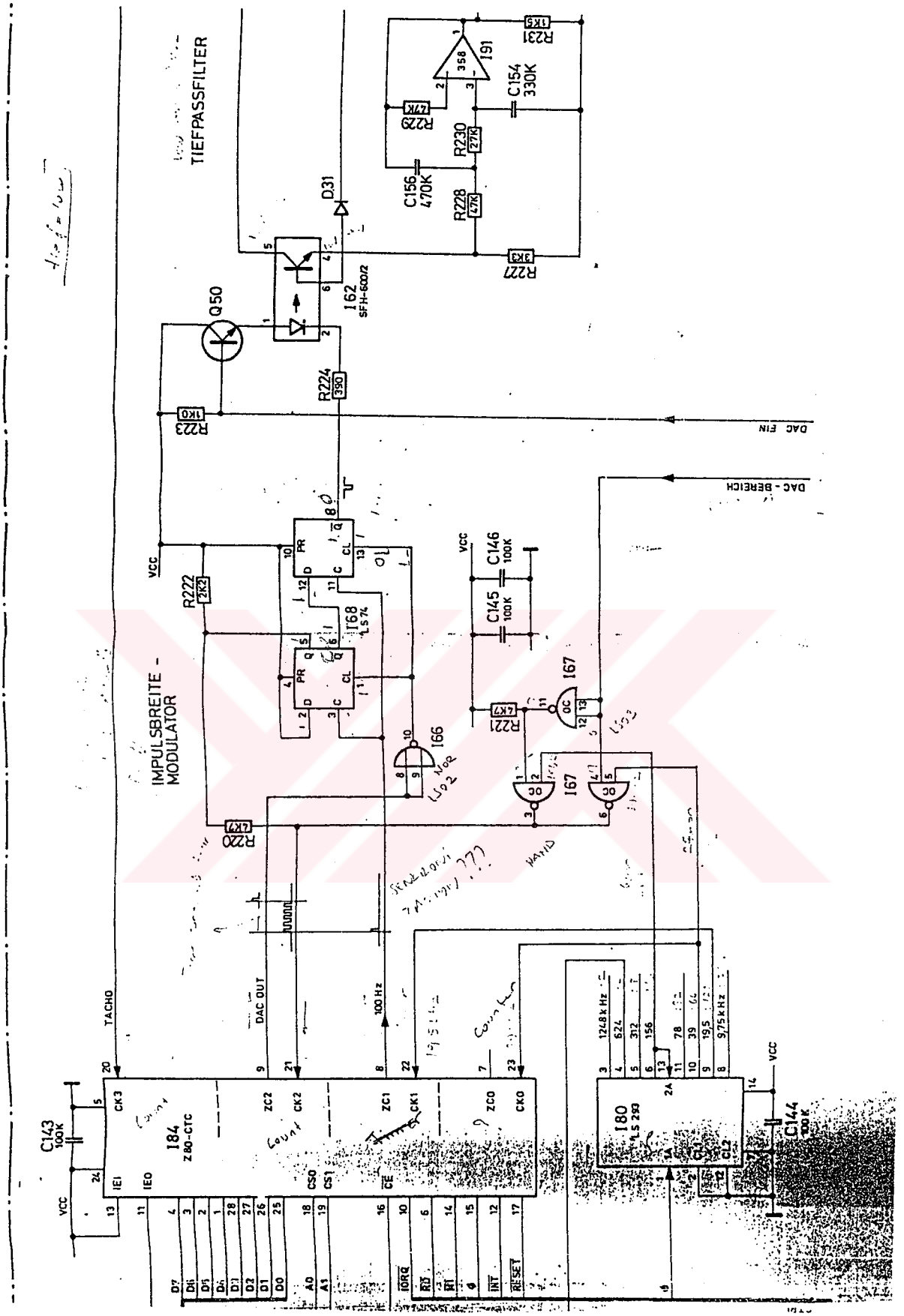
Şekil 2.3.6.2 Hedef kart ADC devresi



Şekil 2.3.6.3 Hedef Kart CPU devresi



Şekil 2.3.6.4 Hedef kart Dijital ınterface devresı



Şekil 2.3.6.5 Hedef kart devir sayacı kısmı

2.4 EMULATOR PROGRAMI ÖZELLİKLERİ

MON51, geliştirme sisteminin epromu üzerinde bulunan hizmet programıdır. 'Internal RAM' içinde Register Bank 3'ü, 71h-7Fh ve F0h-FFh gözlerini, Timer/Counter2'yi, 'External RAM' içinde 2000h-207Fh adreslerini kullanır. Kullanıcı programlarında bu adresler kullanılmamalıdır. Güvenlik açısından kullanıcı kendi programını 2200h adresinden itibaren başlatmalıdır. Programın icrası bittikten sonra MON51'e dönmesi isteniyorsa son satıra

```
lcall 6
```

komutu eklenmelidir.

Kesmet vektörlerinin adresleri değişmiştir.Kullanıcı kesme hizmet programlarını bu adreslerden başlatmalıdır:

IE0	2000h
TF0	2003h
IE1	2006h
TF1	2009h
RI+TI	200Ch
TF2+EXF2	200Fh

Görüldüğü gibi bu adresler arasında 3 byte'lık alan vardır. O yüzden bu adreslere yalnızca ' ljmp ' komutu yazılarak asıl hizmet programına dallanılması sağlanılır.

Mon 51 Komut Seti özeti

Aşağıdaki açıklamalarda ;

"< >" içinde HEX tabanda sayı olduğunu, "[ ]" isteğe bağlı parametre olduğunu gösterir. " / " işareti komut yazımında kullanılmaz, iki yanındaki komutlardan

herhangi birinin kullanılacağını gösterir. Bütün komutlar sadece baş harfleriyle icra edilir. Komutlar büyük yada küçük harfle yazılabilir.

Asm <addr>

: Verilen adresten başlayarak assembler programını yazar . Program yazılımını bitirmek için tekrar <Enter> tuşlanır. Kısa deneme programlarında kullanılır.

Dump [i] [<addr>]/[L<count>]

: Verilen adresten başlayarak bellek gözlerinin içeriklerini gösterir.

L<count> parametresi gösterilecek göz sayısını belirtir. [i] seçeneği internal RAM'i ifade eder. Hiçbir parametre kullanılmadığında son kalınan adresten göstermeye devam eder.

ÖRNEK:

D 3000 ; 3000h RAM adresinden itibaren bellek gözü içeriklerini gösterir.

D i 3F ; 3Fh adresinden itibaren internal RAM içeriklerini gösterir.

Enter [i] [<addr>]

: Verilen adresin içeriğini değiştirir. Birtek adresin içeriğini değiştirir. [i] parametresi internal RAM'i ifade eder. Komutu yazdıktan sonra adres ve içeriği görünecektir. Bundan sonra girilecek değer yazılır.

ÖRNEK:

E 3100 ; 3100h adresinin içeriğini değiştirir.

E i 46 ; 46h adresinin içeriğini değiştirir.

Fill [i] <addr> <addr>/L<count> <data>

: İlk verilen adres başlangıç adresi, ikinci adres bitiş adresi olmak üzere bellek gözlerini verilen data ile doldurur. Bitiş adresi yerine L<count> parametresiyle doldurulacak bellek gözü sayısı verilebilir.

[i] seçeneği internal RAM'i ifade eder.

ÖRNEK:

F 3000 3200 55 ; 3000h - 3200h RAM gözleri arasını 55h ile doldurur.

F i 20 L5 1A ; 20h internal RAM adresinden itibaren 5 gözü 1Ah ile doldurur.

Go [=] [<st_addr>] / [<break_addr>]

: Yüklenmiş programın yürütülmesinde kullanılır. <st_addr> başlangıç adresi, <break_addr> programın duracağı adrestir. <break_addr> kullanılmazsa 'lcall 6' görene kadar program yürütülür. Hiçbir parametre kullanılmadığında son kalınan adresten yürütmeye devam eder.

ÖRNEK:

G = 2200 2340 ; 2200h - 2340h arasındaki programı icra eder.

G = 2340 ; 2340h den itibaren programı yürütür.

Load : Yazılmış HEX dosyayı geliştirme sistemine yükler.

Move <source_addr> <source_end_addr>/L<count> <target_addr>

:Blok transferinde kullanılır.

<source_addr> transfer edilecek bloğun başlangıç adresi, <source_end_addr> transfer edilecek bloğun bitiş adresidir. Bitiş adresi yerine L<count> parametresi ile blok uzunluğu verilebilir. <target_addr> hedef adresidir.

ÖRNEK:

M 2800 28FF 3000 ; 2800h - 28FFh adresleri arasındaki bloğu 3000h adresinden itibaren transfer eder.

M 2800 L100 3000 ; 2800h adresinden itibaren 100h uzunluğundaki bloğu 3000h adresinden itibaren transfer eder.

Register [<8052 reg/bit mne.>] [=<hex/bit data>] / R ALL

:Verilen register/bit 'in içeriğini gösterir. [=] parametresiyle birlikte kullanıldığında register/bitin içeriğini değiştirir. Register/bit'in mnemoniklerinin kullanılması zorunludur. Hiçbir parametre kullanılmadığında bütün registerler gösterilir.

R ALL ile SFR'ler (Special Function Register) de gösterilir.

ÖRNEK:

R pc = 2200 ; Program Counter 2200h olarak değiştirilir.

R cy ; Carry flag'inin içeriğini gösterir.

Trace [=] [<st_addr>]/[<addr>]/[L<count>]

: Programı adım adım yürütür. <st_addr> başlangıç adresi <addr> bitiş adresidir. Bitiş adresi yerine L<count > parametresiyle adım sayısıda verilebilir. Hiçbir parametre kullanılmadığında son kalınan adresi icra eder.

ÖRNEK:

T = 2300 2308 ; 2300h - 2308h adresleri arasındaki komutları teker teker icra eder.

T = 2300 L8 ; 2300h adresinden itibaren 8 komutu teker teker icra eder.

Unasm [<addr>]/[L<count>]

: Verilen adresten itibaren makina kodlarını assembler mnemotiklerine çevirir.

L<count> parametresi kod adedini gösterir. Hiçbir parametre kullanılmadığında son kalınan adresten itibaren çevirir.

RESet : Software reset. Bu komut baş harfiyle değil, RES ile icra edilebilir. Bütün dahili registerlere Hardware Reset'teki değerleri yüklenir. Kullanım sırasında ? tuşuna basılarak HELP menüsüne başvurulabilir.

2.5 SİSTEMİN KULLANIMI

Kullanıcı programını herhangi bir editörde ASCII formatta yazar ve .SRC olarak dosyaya kaydeder. Örneğin bu dosyanın ismi DENE.SRC olsun. DOS ortamında

DO51 DENE H L

komutu verilir. Derleyici DENE.SRC dosyasını derleyerek DENE.HEX ve DENE.LST isimli iki dosya üretir. Program yazımı esnasında bir hata varsa bu hata DENE.LST isimli dosyadan incelenerek gerekli düzeltmeler DENE.SRC üzerinde yapılır ve tekrar derlenir. Elde edilen DENE.HEX geliştirme sistemine yüklenecek dosyadır.

Geliştirme sisteminin PC ile haberleşebilmesi için gerekli bağlantı RS232 ara kablosu ile gerçekleştirilir. Herhangi bir veri iletişim programı çalıştırılır. İletişim parametreleri alttaki değerlere tam ve uygun çalışma için cevap verebilmelidir.

4800 baud
No parity
8 data bit
1 stop bit
COM1 / COM2

olarak belirlenir. Burada veri iletişim programı olarak VTERM kullanılmıştır.

2.6 VTERM SERİ HABERLEŞME PROGRAMI

VTERM sıklıkla kullanılan temel bir seri data iletimi yapan bir programdır. < Alt-S > tuşlanarak menüye girilir ve sorulan dosya ismi olarak daha önceden üretilen DENE.HEX isimli HEX dosya verilir . Ardından tekrar < Alt-S > tuşlanarak transfer başlatılır. Daha önceden yazılan programda programın başlangıç adresi 2200h olarak

belirtilmişse HEX dosya da bu adresten itibaren yerletirilir. Bundan sonra geliştirme sistemindeki MON51 hizmet programı kullanılabilir. Kullanıcı programı çalıştırılmadan önce 8032' deki 'Program Counter' programın başlangıç adresine set edilmelidir. Bu işlem 'R' (Register) komutu ile yapılabilir.

ÖRNEK PROGRAMIN AÇIKLAMASI

Örnek programın ismi DENE.SRC 'dir. DENE.HEX geliştirme sistemine yüklenecek programdır.Bu program P3.5 pininden (8032'de T1, 15 nolu pin, 64'lü konnektörde 48 nolu pin) kare dalga çıkışı verir. Kare dalga elde etmek için r0 ve r1 registerleri kullanılmaktadır. Kare dalganın yarı-peryodu r0 ve r1 'e yüklenecek olan SEMI_PERIOD_A ve SEMI_PERIOD_B değerleri ile değiştirilebilir. Yüklenen değer arttıkça kare dalganın peryodu da artar.Program sürekli kare dalga ürettiğinden MON51'e dönmesine gerek yoktur. Bu yüzden programın sonuna 'lcall 6' komutu konulmamıştır. Programın icrası Hardware Reset ile kesilebilir. r0 ve r1 ikisi birden 0'a düştüğünde kare dalga çıkış pininin evriği alınır ve SAYAC bir artırılır.Programdaki 'EQU' (Equal) ifadesi kendinden önceki değişkene sabit bir değer ataDerleyici bu değişkeni gördüğünde yerine sabit değeri koyar. Sabitler HEX (8Eh), Desimal (142), Binary (10001110b) tabanda yazılabilir.

' BIT ' ifadesi kendinden önceki değişkene 8032'deki bitlerden birini atar . Derleyici bu değişkeni gördüğünde yerine 8032' deki biti koyar. Bitin adresi veya mnemotikleri yazılabilir.

' DSEG AT ' ifadesi Data Segment'in başladığı adresi gösterir. Data Segment, değişkenler için kullanılabilecek bellek alanıdır. Internal RAM içinde tanımlı olduğundan kullanılacak değişken sayısı sınırlıdır. ' Stack ' de Internal RAM'de olduğundan buna dikkat edilmelidir. ' DS ' (Define Storage) ifadesi kendinden önceki değişken için Internal RAM'de belirtilen miktarda bellek gözü ayırır.

' CSEG AT ' ifadesi Code Segment'in başladığı adresi gösterir . Code Segment, program kodunun bulunduğu bellek alanıdır. Program kodunun başladığı ilk adrese bu ifade konulur. Geliştirme sisteminde kullanıcı programı External RAM'e (2000h -

3FFFh) yüklendiğinden kullanıcı programları bu aralıkta olmalıdır. Güvenlik açısından kullanıcı programı 2200h adresinden başlamalıdır.

' ORG ' ifadesi program kodunun başladığı adresi gösterir . Farklı adreslerden itibaren program yazılmak istendiğinde (Örneğin kesme vektörleri) bu ifade kullanılır. Program çalıştığında üretilen kare dalganın yarı-peryodu yaklaşık olarak

YARI_PERYOD = r1 * (24 * r0) / Fosc [sn]

formülü ile hesaplanır. r0 ve r1 , registerlere yüklenen değerlerdir. Geliştrme sisteminde Fosc=11.0592MHz'dir.

Kullanıcı editörde programını yazarken son satırına ' END ' ifadesini koymalıdır. Böylece derleyici kaynak programın bittiğini anlar.

2.7 ANA KART KONNEKTÖR BAĞLANTILARI

CN01 : Sistem bağlantılar için ana konnektör olarak kullanılır.

1	GND	33	GND
2	ALE	34	PSEN
3	/RD	35	TIMER1
4	TIMER0	36	
5		37	/WR
6		38	
7		39	
8		40	
9		41	
10		42	
11		43	
12	A0	44	A1

13	A2	45	A3
14	A4	46	A5
15	A6	47	A7
16	A8	48	A9
17	A10	49	A11
18	A12	50	A13
19	A14	51	A15
20	GND	52	GND
21	D0	53	D1
22	D2	54	D3
23	D4	55	D5
24	D6	56	D7
25	INPUT0	57	INPUT1
26	INPUT2	58	INPUT3
27	INPUT4	59	INPUT5
28	INPUT6	60	INPUT7
29	GND	61	GND
30	INT0	62	INT1
31	/RESET	63	RESET
32	+5V	64	+5V

P1 : RS232 bağlantısı için kullanılmaktadır.

1	GND	6	NC
2	RX	7	GND
3	TX	8	NC
4	NC	9	+5V

JP03 : LCD ilavesi i‡in

1	- GND	2-	+5V
3	- Vo	4 -	RS (A1)

5 - R/W (A0)	6 - E
7 -D0	8 - D1
9 - D2	10 - D3
11 - D4	12 -D5
13 -D6	14 -D7

JP4 : PIO bağlantısı için kullanılır.

1 PA0	2 PA1
3 PA2	4 PA3
5 PA4	6 PA5
7 PA6	8 PA7
9 PB0	10 PB1
11 PB2	12 PB3
13 PB4	14 PB5
15 PB6	16 PB7
17 PC0	18 PC1
19 PC2	20 PC3
21 PC4	22 PC5
23 C6	24 PC7
25 +5V	26 GND

JP5 : Port1 bağlantısı için kullanılır.

1 P1.0
2 P1.1
3 P1.2
4 P1.3
5 P1.4
6 P1.
7 P1.6

8 P1.7

A-) EPROM SEÇİMİ

TİP	J2	J3
2764	1-2	1-2
27128	1-2	1-2
27256	2-3	1-2
27512	2-3	2-3

B-) RAM SEÇİMİ

TİP	J1
6264	1-2
62256	2-3

C-) RTC Interrupt Jumper

J4	1-2	2-3
8032 Pin	INT0	P1.0

RTC Interrupt Pini

D-) ADC Interrupt Jumper

J5	1-2	2-3
8032 Pin	INT1	P1.1

ADC Interrupt Pini

E-) BACK-UP BATTERY Jumper

J6	1-2	2-3
	ON	OFF

PAL PROGRAMI

Ana kart tasarımında adres decoding işlemleri gerçekleştirilirken ayrık komponent kullanımı yerine hem daha ekonomik olan ve hemde sistemin kopta edilme olasılığını zorlaştıran PAL kullanımı tercih edilmiştir. PAL 22V10 12 giriş ve 10 çıkış ucuna sahip çok kullanışlı bir entegredir. Sistem tasarımında giriş ve çıkışların tamamı kullanılmaktadır.

İlgili PAL programı SPRINT isimli üniversal eprom programlayıcısına mahsus PAL assembler yazılımı kullanarak derlenmiştir.

```
DEVICE 22V10;
ALE         1
PSEN        2
A15         3
A14         4
A13         5
A12         6
A11         7
A10         8
A9          9
A8          10
RD          11
GND         12
WR          13
CSRTC       14
```

```
OEADC       15
WAD         16
CSPIO       17
CSLCD       18
CSOE        19
CSRAM       20
CSEPR       21
DIR         22
G           23
VCC         24

START
DIR   = RD;
G     = /A15+A14+A13;
CSEPR  = A15+A14+A13+ALE;
CSRAM  = A15+A14+/A13+ALE;
CSOE   = PSEN*RD;
CSLCD  = A15* A14* A13* A12* A11* A10* A9* A8 */ALE*/WR* RD
     + A15* A14* A13* A12* A11* A10* A9* A8 */ALE* WR*/RD;
CSRTC /= A15* A14* A13* A12* A11* A10* A9*/A8*/WR* RD
     + A15* A14* A13* A12* A11* A10* A9*/A8* WR*/RD ;
CSPIO  =/A15+/A14+/A13+/A12+/A11+/A10+ A9+/A8+
       ALE+/WR*/RD+WR*RD;
OEADC /=/A15+/A14+/A13+/A12+/A11+/A10+ A9+ A8+ ALE+ RD;
WAD   /=/A15+/A14+/A13+/A12+/A11+/A10+ A9+ A8+ ALE+ WR;

END;
```

BÖLÜM 3. MCS51 MICROCONTROLLER AİLESİ

"Soft Microcontroller" ailesi 8051 çekirdeği üzerine bina edilmiştir. Birçok işlevi standard 8051 ailesi ile benzerlik göstermekle beraber bazı ilave özelliklere sahiptir. Burada "Soft Microcontroller" ailesinin temel özellikleri anlatılacaktır .

3.1 CPU YAZMAÇLARI :

"Special Function Register" (SFR) olarak isimlendirilen yazmaçlar 8051'dekilerle benzer özelliklere sahiptir.

Accumulator (A): Bütün aritmetik işlemlerde kaynak yada hedef yazmaçtır. Bunun yanında birçok komut tiplerinde kullanılır.

Stack Pointer (SP): "Stack"de depolanan en son değişkeni işaretlemek için kullanılır. Dahili hafızada herhangi bir yeri gösterebilir.

B Register: Çarpma ve bölme işlemlerinde kullanılır. Genel amaçlı olarak da kullanılabilir.

Program Status Word (PSW): Bir önce yürütülen komutun sonucuna göre belirlenen durum bayraklarını içerir. Ayrıca "Register Bank" seçme bitlerini bulundurur.

Data Pointer (DPTR): Veri hafızasına erişmek için kullanılır. 8-bit yada 16-bitlik SFR olarak ulaşılabilir.

3.2 YAZBOZ YAZMAÇLAR

Verilerin doğrudan depolanabildiği yazmaçlardır. 00h ile 7Fh adresleri arasındadır ve MOV komutu ile doğrudan erişilebilir. Dört adet 8-baytlık "register bank"ları da kapsar. "register bank" lar program kolaylığı sağlayan yazmaçlardır.

Seri Giriş/Çıkış:

Gelen veri tamponu, giden veri tamponu ve kontrol yazmaçlarından oluşur. Her iki tampona da SFR adres haritasındaki SBUF registeri ile erişilir. Kontrol yazmacı (SCON) farklı adrestedir. Seri giriş/çıkış işlevlerine izin verildiğinde iki port pini (P3.0, P3.1) seri haberleşmeye ayrılır.

Programlanabilir Zamanlayıcılar:

İki adet 16-bitlik yazmaç sayıcı yada zamanlayıcı olarak programlanabilir. Her yazmacın üst ve alt yarılarına TH1,TL1,TH0,TL0 olarak erişilebilir. TCON kontrol yazmacı zamanlayıcıların çalışma modlarını seçmede kullanılır. İki port pini (P3.4, P3.5) zamanlayıcılara hizmet verecek şekilde programlanabilir.

Paralel Giriş/Çıkış:

Dört SFR dört adet porta erişmeyi sağlar. Bu portlar P0, P1, P2, P3 olarak isimlendirilir. "Byte-wide Bus" kullanıldığında portlar etkilenmez.

Program/Veri Hafızası Arabirimi:

"Soft Microcontroller" dış veri hafızasına erişmek için "Byte-wide Bus"ı kullanır. Dış veri hafızası, yazma-korumalı düzenlenerek program hafızası haline getirilebilir. "Byte-wide Bus" 16 adres yolu, 8 veri yolu ve kontrol yollarından oluşur. İç hafızaya erişildiğinde portlarda değişme olmaz. Dolayısıyla bütün portlar serbestçe kullanılabilir. Port0, Port2, WR(P3.6), RD(P3.7)'den oluşan "Expanded Bus" kullanılarak 64k ROM ve 64k RAM hafızasına erişilebilmekle beraber portlara serbestlik kazandırmak için bu yapı genellikle istenmez.

Koruma Devresi:

Besleme geriliminin kesilmesiyle dahili lityum bataryayı devreye sokarak program/veri hafızasının silinmesini engelleyen devredir. "Power Fail Warning Interrupt", "Automatic Power Down", "Power On Reset" katlarından oluşur. Böylece program/veri hafızası kullanıcı tarafından değiştirilebilir fakat besleme geriliminin kesilmesiyle silinmez.

Yazılım Şifreleme Devresi:

DS5000 ve DS5002 serisi, adres şifreleyici, veri şifreleyici, şifre kelimesinden oluşan yazılım koruma devresine sahiptir. Şifreleme modunda çalışıldığında, dahili adres ve veri bilgileri gerçek değerlerine çevrilerek "Byte-wide Bus" üzerinden iç hafızaya ulaştırılır. Şifreleme kelimesine bağlı olarak adres şifreleyici ve veri şifreleyici farklı algoritmalara sahiptir.

Güvenlik Kilidi Devresi:

Yükleme programı ile program/veri hafızasını değiştirmeyi engeller ve "Expanded Bus" üzerinden komut alma çevrimine izin verir. Yazılımın kopyalanmasını önler

"Timed Acces" devresi:

Yazılım kontrolünün kaybolmasıyla program hafızasının değişmesini engeller. Önemli yazmaçlara erişme sırasında kullanılır.

Yerleşik Yükleyici Program:

Kullanıcı programının seri port yardımıyla program hafızasına yüklenmesini sağlar. Yükleyici progrmına kullanıcı tarafından erişilemez ve sadece programlama modunda aktiftir. Yükleme işlemi bitince etkisiz hale gelir.

Programlama Rehberi

Soft Micro" ailesi program ve veri hafızası için silinmeyen RAM teknolojisi kullanır. Dahili RAM'in bellek haritasında program hafızası olarak yerleştirip yazma-korumalı yapmakla ROM olarak kullanılması sağlanır. Kalan RAM alanı silinmeyen veri depolama alanı olarak kullanılabilir. RAM'in iki bölüme ayrılmasının en önemli avantajı kırmık sayısının azalmasıdır. Örneğin 24kB program hafızası ve 4kB veri hafızası istendiğinde bir adet 32kB RAM kullanılabilir. "Soft Micro", RAM'i program ve veri hafızası olarak ayırabilir. Bu ayırma işlemine bölmelendirme (partitioning) denir.

"Soft Micro" hafıza düzeni

Bütün "Soft Micro" üyeleri standard 8051 gibi üç hafıza alanına sahiptir. Bunlar dahili yazmaçlar, program hafızası ve veri hafızasıdır. Bu alanlara ayrı yollardan erişilir. "Soft Micro" standard 8051 yazmaçları yanında ilave yazmaçlar da içerir. 64kB program ve veri hafızası alanı vardır. Bu alanlara erişmek için değişik yollar kullanılır.

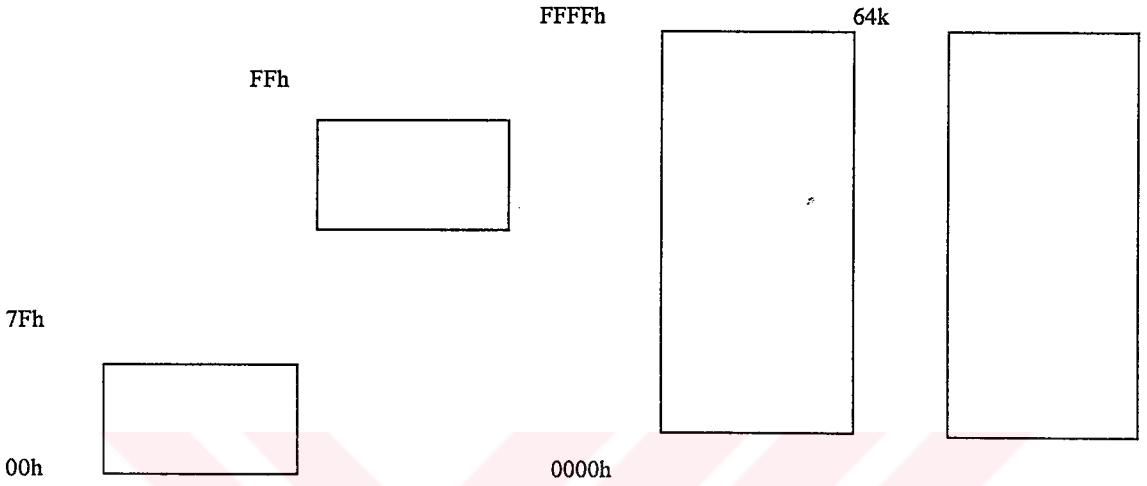
Dahili yazmaçlar

Bu alan iki parçaya ayrılmıştır. Yazboz yazmaçlar ve özel amaçlı yazmaçlar (SFR). 128 adet yazboz yazmaç vardır. Dört adet 8-baytlık grup "register bank" olarak adlandırılır (R0-R7). Bu alan program yada veri hafızasında olmayıp farklı komutlarla erişilir. Özel amaçlı yazmaçlar 80h ile FFh adresleri arasında bulunur ve kırmık içindeki çevre elemanlarını kontrol eder. SFR'lere erişmek için direkt adresleme kullanılmalıdır. Endirekt adresleme kullanılırsa tanımsız değer okunur.

Yazboz yazmaçlar genel amaçlı veri saklama gözleridir. Küçük miktarda değişkenlere yüksek hızda erişmek için kullanılırlar. Veri miktarı 128-bayttan fazlaysa

harici veri hafızası kullanılabilir. "Soft Micro"da harici veri hafızası lityum pil ile korunmuştur.

Yazboz yazmaçların iki temel işlevi vardır. 16-baytlık kısmı bit-adreslenebilir alandır. Böylece 128 adet genel amaçlı durum bayrağı elde edilebilir. Yazboz yazmaçlar ikinci olarak "stack" için kullanılırlar.



Şekil 3.1 MCS51 80C32 hafıza haritası

8051 komut seti, "register bank"lar kullanıldığında tek saykıllık komutlar üretebilir. "Register bank" lar dahili hafızada 00h-1Fh adresleri arasındadır ve R0-R7 olarak isimlendirilirler. PSW'deki iki bitle (PSW.4, PSW.3) dört banktan biri seçilebilir. R0-R7'yi kullanan komutlar, seçilen gruptaki 8-baytlık alanı kullanırlar.

R1	R0	Bank adresi
0	0	00h-07h
0	1	08h-0Ah
1	0	10h-17h
1	1	18h-1Fh

Program/veri hafızası

"Soft Micro" hafızayı program ve veri hafızası olarak iki parçaya ayırır. Her parça 64kB uzunluğundadır. program hafızası sadece okunabilir olduğundan bu alana yazma komutu yoktur. 8051'de bu iki alan birbiri içine geçebilir. "Soft Micro"da dahili hafıza içinde bu iki alan birbirinden ayrılmıştır.

"Soft Micro" hafıza erişimi için iki ayrı bus kullanır. Birincisi paket içindeki SRAM'e bağlı bulunan, adres, veri, kontrol hatlarının gittiği "Byte-wide Bus"tır. Bu yapıda program ve veri alanları birbirinden ayrılır. İkincisi Port0 ve Port2'yi kullanan "Expanded Bus"tır. Standard 8051 uyumlu yapı olmakla beraber çoğunlukla kullanılmaz.

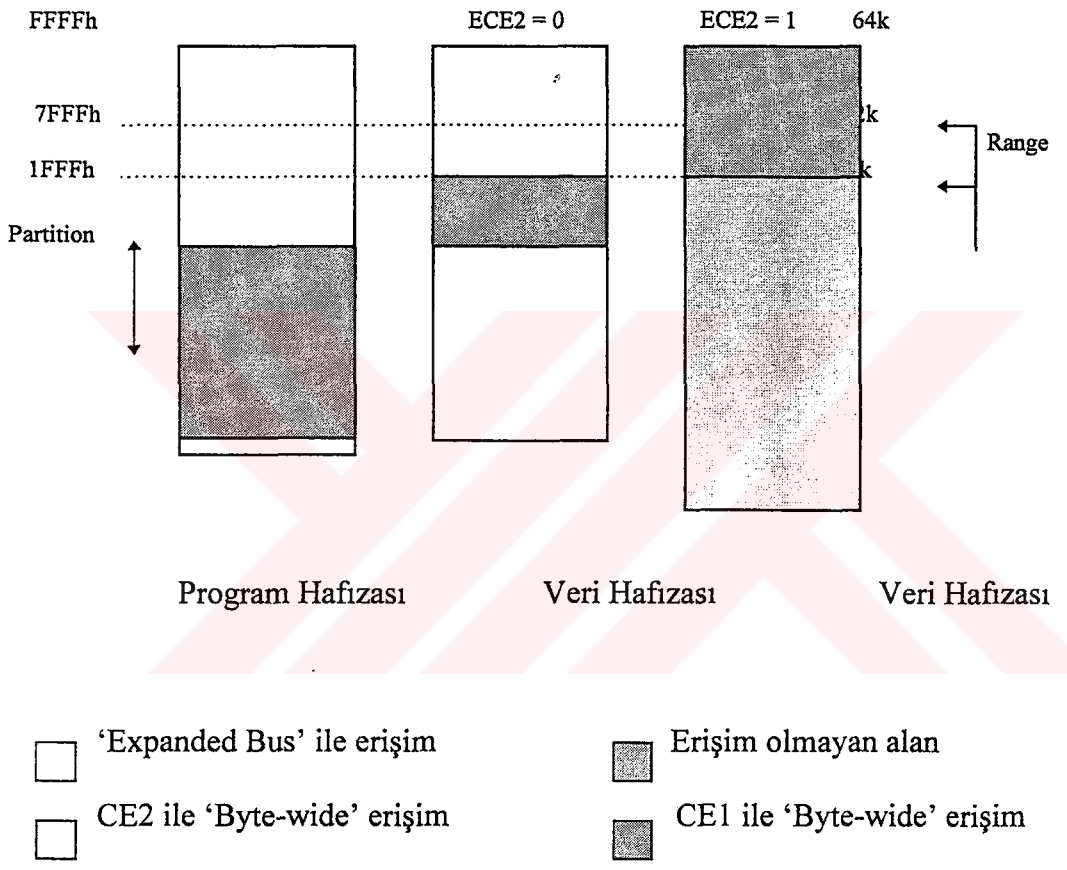
Hafıza adresleri sözkonusu olduğunda iki önemli terim kullanılır: "Partition" ve "Range". "Partition" dahili SRAM'daki veri hafızasının başlangıç adresidir. Yükleyici program tarafından seçilebilir yada kullanıcı programıyla değiştirilebilir. "Range" dahili SRAM'in üst sınırıdır. Yalnızca yükleyici program tarafından değiştirilebilir.

DS5000T hafıza düzeni

DS5000T'de bulunan "Byte-wide Bus", 14 adres, 8 veri, R/W ve iki seçme yollarından oluşur. Bu yollar modül içerisinde bağlanmış olup dahili hafızayı kontrol eder. DS5000T'de 8kB yada 32kB hafıza bulunabilir. Hafıza miktarı MCON.3 biti ile belirlenir ve kullanıcı tarafından değiştirilemez. DS5000T hafıza modüllerine erişmek için iki seçme ucu kulanır. Birincisi (CE1), 8kB/32kB program/veri hafızasına erişir. "Partition" kullanıcı tarafından seçilir ve yükleyici program yardımıyla değiştirilebilir. "Partition", MCON yazmacı ile 2kB'lik adımlarla artırılabilir.

Aşağıdaki şekilde DS5000T'nin hafıza haritası gösterilmektedir. "Partition", "Range", ECE2, DS5000T'nin "Byte-wide Bus" yada "Expanded Bus" kullanmasını belirler. Burada dikkat edilmesi gereken nokta, "Partition" ve "Range" birbirine otomatik olarak bağlanmaz. Bu yüzden "Range"den daha büyük bir "Partition" seçilmemelidir. Hafıza haritasında olmayan herhangi bir adres, Port0 ve Port2'yi kullanarak "Expanded Bus" çalışmasına yol açar. Aşağıdaki hafıza haritası standard çalışma şartlarını gösterir. Bu haritayı değiştirmenin iki yolu vardır; EA ucu ve güvenlik kilidi. EA ucu topraklandığında DS5000T bütün hafıza erişimlerini "Expanded Bus"

üzerinden yapar. Bu durumda "Partition", "Range", ECE2 önemsizdir. Normal çalışmada EA ucu besleme gerilimine çekilmelidir. Güvenlik kilidi aktif edildiğinde, yükleyici programın dahili hafızayı okumasına izin verilmez, "Expended Bus" üzerinden program okuma
engellenir. Dolayısıyla yalnızca paket içindeki program hafızasından okuma yapılabilir. Güvenlik kilidi EA ucundaki seviyeden üstündür.



Şekil A.2 DS5000T hafıza haritası

Hafıza haritası ayarları, sistem tasarımında esneklik sağlar. "Partition" işlevi program/veri hafızası sınırını ayarlar. "Range" işlevi veri hafızasının üst sınırını belirler. Bununla beraber, çelişkili ayarlara karşı da önlemler alınmıştır. Örneğin "Partition" 0000h olarak seçilmişse "Expanded Bus" aktif edilir. Başka bir durum "Range" 8k seçilip "Partition" 8k'dan büyük seçilmesidir. Bu durumda sınırlama faktörü devreye

girer. "Range" üzerindeki adresler "Expanded Bus" üzerinden işlem görür. Dahili hafıza bu durumdan etkilenmez.

Program hafızasını yüklemek

"Soft Micro" paket içindeki seri yükleyici ile programlanabilir. Bu yükleyici aynı zamanda hafıza ayarlarını da gerçekleştirir. Lityum pil kullanıldığından her açılışta önemli ayarlarınyeniden yapılması gerekmez. Bir kez ayarlandığında, yeniden ayarlanana kadar değişmeden kalır. "Soft Micro"nun en önemli avantajı, sistem içinde programlanabilmesidir.

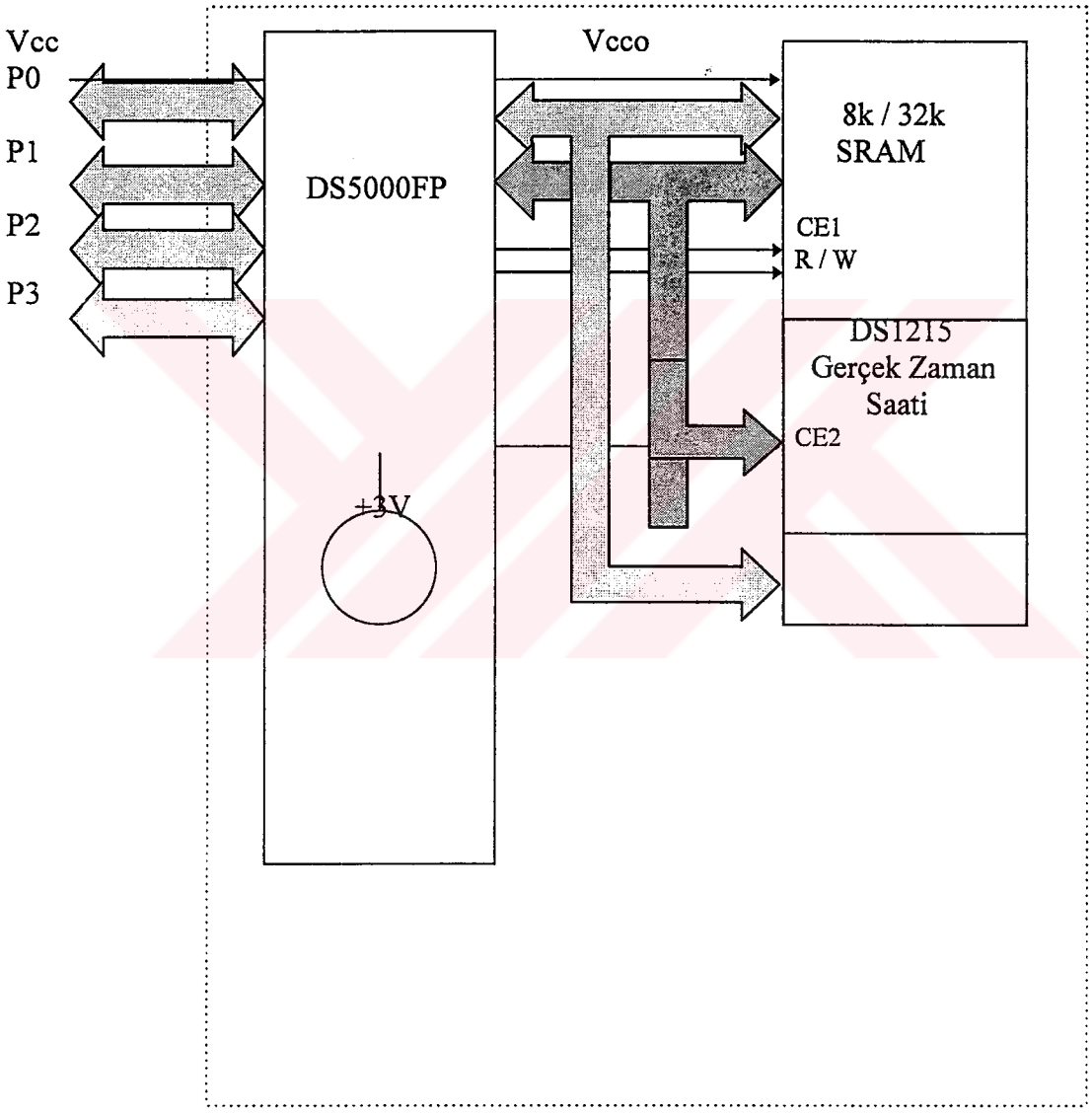
Özel amaçlı yazmaçlar (SFR)

SFR'ler birçok işlevi kontrol etmek için kullanılır. 80h-FFh adreslerinde bulunurlar. Bazıları bit adreslenebilir özelliğe sahiptirler. SFR'ler 8-bit içerdiğinden diğer ayarları etkilemeden değişiklik yapma bakımından bu özellik oldukça kullanışlıdır. Birkaç küçük ilave dışında SFR'ler standart 8051 ile aynıdır.

Tablo 3.1 MCS51 özel amaçlı yazmaçlar (SFR)

Adres	Bit Sembol								SFR
F0h									***B***
E0h									***ACC***
D0h	C	AC	F0	RS1	RS0	OV		P	***PSW***
C7h									TA
C6h	PA3	PA2	PA1	PA0	R32/8	ECE2	PAA	SL	MCON
B8h	RWT			PS	PT1	PX1	PT0	PX0	***IP***
B0h									***P3***
A8h	EA			ES	ET1	EX1	ET0	EX0	***IE***
A0h									***P2***
99h									SBUF
98h	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	***SCON***
90h									***P1***
8Dh									TH1
8Ch									TH0
8Bh									TL1
8Ah									TL0
89h	GATE	C/T	M1	M0	GATE	C/T	M1	M0	TMOD
88h	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	***TCON***
87h	SMOD	POR	PFW	WTR	EPFW	EWT	STOP	IDL	PCON
83h									DPH
82h									DPL
81h									SP
80h									***P0***

DS5000 tamamıyla 8051 uyumlu 8-bit CMOS mikrokontrolördür. Paket içine yerleştirilen lityum pil sayesinde bütün bilgiler sistem besleme geriliminin kaybolmasına karşı korunmuştur. Dahili program/veri hafızası 32kB CMOS SRAM'dir. Ayrıca dahili veri ve kontrol yazmaçları da korumalıdır. Gerçek-zaman saati ile saniyenin yüzde birine kadar zaman bilgisine erişilebilir.



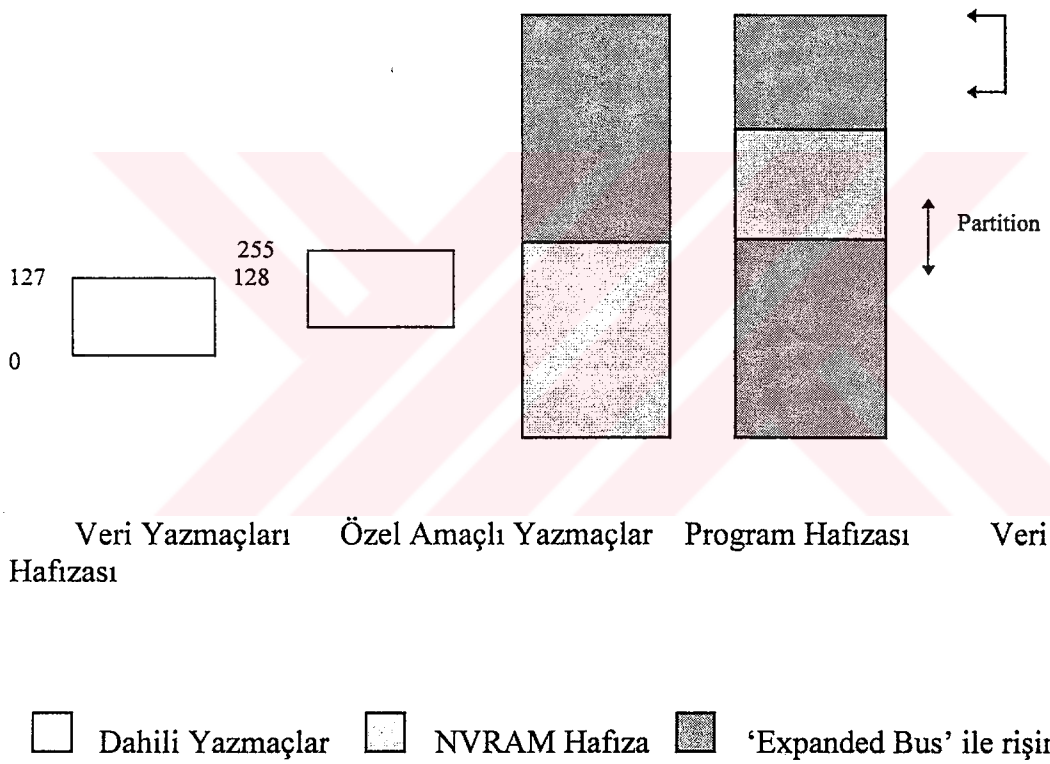
Şekil 3. MCS51 blok diyagramı

Komut seti

DS5000T, standart 8051 komut setiyle uyumludur. 8051 için kullanılan bütün geliştirme yazılımları DS5000T ile kullanılabilir.

Hafıza düzeni

DS5000T'nin adres haritası aşağıdaki şekilde gösterilmiştir. Görüldüğü gibi adres ve veri hafızası ayrı alanlardadır. Adres hatları 16-bit olduğundan, 64kB program hafızası ve 64kB veri hafızasına erişilebilir. DS5000T'nin içinde program yada veri hafızası olarak kullanılabilen 32kB RAM bulunmaktadır. DS5000T'deki gerçek-zaman saati seri kontrollü olup MCON.2 (ECE2) biti ile erişilebilir.



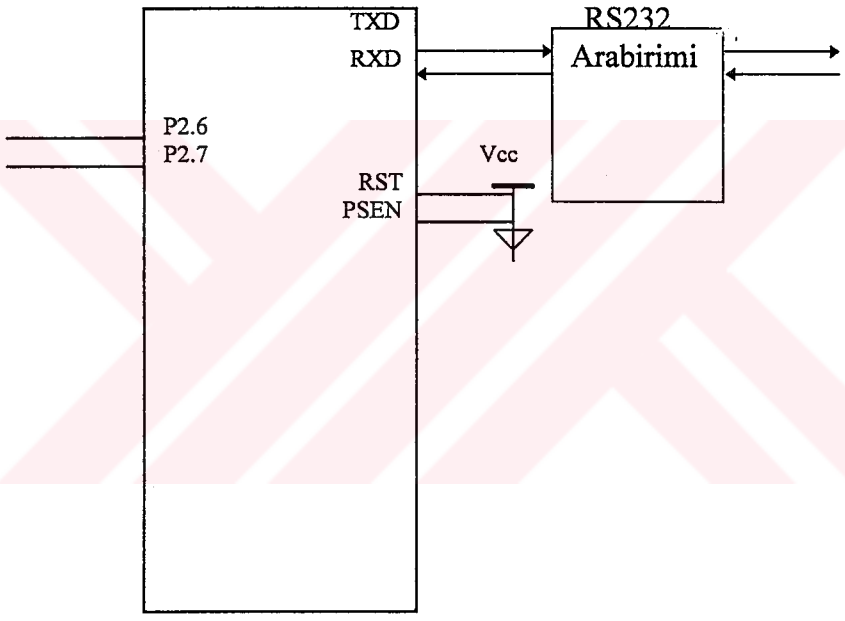
Şekil 3.4 DS5000T mantıksal adres alanları

Program yükleme

Program yükleme modu, program/veri hafızası ayrımını sağlar. Bu işlem için iki yol vardır:

1. DS5000T'nin özelliği olan seri yükleme. Bu özellik sistem içinde programlanabilmeyi sağlar.
2. Giriş/çıkış pinleri yardımıyla paralel yükleme. Bu mod 87C51H tipi ile aynı zamanlama akışına sahiptir.

DS5000T'yi program yükleme moduna getirmek için, aynı anda RST ucu "1" seviyesine, PSEN ucu "0" seviyesine getirilmelidir. Bu işlemin ardından DS5000T paralel program yükleme darbesini yada seri porttan ASCII-0Dh karakterini bekler.



Şekil A.5 Seri porttan program yükleme

Seri program yükleyici

DS5000T için seri yükleme modu en basit, hızlı ve tercih edilen yöntemdir. Haberleşme için asenkron seri port yeterlidir. RS232C arabirimi DS5000T'nin sistem içinde programlanmasına imkan verir. Paralel program moduna geçmeyi önlemk için P2.6 ve P2.7 "1" seviyesine çekilmelidir. 11.0592Mhz kristal kullnıldığında standart haberleşme hızları seçilebilir. Seri yükleyici standart UART ile 3-hatlı haberleşebilecek şekilde tasarlanmıştır. Seri yükleyici, kullanıcı programını DS5000T'ye yüklemek ve yüklenmiş programı okumak için basit komutlara sahiptir. Aşağıda açıklanan tek karakterlik komutlarla çalışır:

C Dahili RAM'in CRC-16 kodunu hesaplar.
D Intel-Hex formatında program okur.
F Dahili RAM'i verilen sabitle doldurur.
K 40-bit şifre kelimesini yükler.
L Intel-Hex formatındaki programı yükler.
R MCON yazmacını okur.
T Intel-Hex formatında gelen bilgiyi yankılar.
U Güvenlik şifresini siler.
V Intel-Hex formatındaki dosyayı Dahili RAM ile karşılaştırır.
W MCON yazmacına yazar.
Z Güvenlik kilidini aktifleştirir.
P Porta değer gönderir.
G Porttan değer okur.

BÖLÜM 4. MCS51 AİLESİ İÇİN ÖZEL C DİLİ TANIMLAMALARI

C dili verimli kodlar, yapısal programlamaya uyumlu geniş veri yapısı, çeşitli operatörlerden oluşan genel amaçlı bir programlama dilidir. C dili özel uygulamalar için geliştirilmiş yüksek seviyeli bir dil değildir. Bununla birlikte birçok uygulamada diğer özel amaçlı programlardan daha basit olarak kullanılabilir .

8051 mikrokontrolör uygulamaları için C51 derleyicisi geliştirilmiştir. C51 derleyicisi 8051'e uydurulmuş C derleyicisi değildir. Özel olarak 8051 için geliştirilen bir derleyicidir. ANSI standardında C dilini kullanarak 8051 makina kodlarını üretir. C gibi yüksek seviyeli dillerin makina diline göre birçok üstünlükleri vardır:

- İşlemcinin komut setinin bilinmesi gereksizdir. Temelde hafıza yapısını öğrenmek yeterlidir.
- Yazmaç adresleri ve hafıza/veri tipleri gibi detaylar derleyici tarafından yönetilir.
- Program bağımsız fonksiyonlara ayrılarak yapısal hale getirilebilir.
- Farklı modüllerin birleştirilebilmesi programın okunuşunda kolaylık sağlar.
- Program yazım ve test süresi önemli ölçüde kısalarak verim artar.
- C kütüphanelerindeki standard program parçaları kullanılabilir.
- Yapısal programlama yöntemiyle mevcut programa yeni bölümler kolayca eklenebilir.
- C dili oldukça destek bulmuş bir dildir ve bütün sistemlere uyumludur. Mevcut program ihtiyaç duyulduğunda başka bir işlemciye kolayca uyarlanabilir.

4.1 Veri tipleri

C51 derleyicisi aşağıdaki veri tiplerini destekler. Değişkenler yapılar içinde birleştirilebilir. Çok boyutlu diziler oluşturulabilir ve değişkenlere "pointer" yardımıyla erişilebilir. İki özel veri tipiyle ("sbit" ve "sfr") 8051 işlemcisindeki "SFR"lere erişilebilir. "sfr P0 = 0x80" komutu PORT0 yerine P0 değişkenini tanımlar.

Tablo B.1 C51 derleyicisinde veri tipleri

Veri Tipi	Uzunluk	Değer
bit	1 bit	0 veya 1
signed char	1 bayt	-128...+127

unsigned char	1 bayt	0...255
signed int	2 bayt	-32768...+32767
unsigned int	2 bayt	0...65535
signed long	4 bayt	-2147483648...+2147483647
unsigned long	4 bayt	0...4294967295
float	4 bayt	±1.176E-38...±3.40E+38
pointer	1...3 bayt	değişken adresi
SFR erişimi için veri tipleri:		
sbit	1 bit	0 veya 1
sfr	1 bayt	0...255
sfr16	2 bayt	0...65535

4.3 Hafıza tipleri

51 derleyicisi 8051 ve ona uyumlu mikrokontrolör ailesinin mimari yapısını destekler. MCS51 sistemindeki tüm donanım bileşenlerine erişebilir. Her değişkenin hafıza tipi açık olarak belirtilmelidir. IDATA tipli değişkene erişme süresi, XDATA tipli değişkene erişme süresinden daha kısadır. Bu yüzden değişkenlerin hafıza tipinin seçimine dikkat edilmelidir.

Tablo B.2 C51 derleyicisinde hafıza tipleri

Hafıza Tipi	Tanım
data	doğrudan adreslenebilen iç veri hafızası (128 bayt).
bdata	bit-adreslenebilen iç veri hafızası (16 bayt).
idata	dolaylı adreslenebilen iç veri hafızası; bütün iç adresleri içerir.
pdata	256 bayt olarak adreslenebilen dış veri hafızası.
xdata	64k olarak adreslenebilen dış veri hafızası.
code	64k olarak adreslenebilen program hafızası.

Değişken tanımları için örnek:

```
data char var1;
unsigned long xdata array[100];
float idata x,y,z;
unsigned int pdata b[10][4][4];
```

```
sfr P0 = 0x80;
char bdata flags;
sbit flag0 = flags^0;
```

Eğer değişken tanımındaki hafıza tipi belirtilmemişse, o anki kurulu modeline göre hafıza tipi seçilir. Hafıza modelleri üç tiptir ve SMALL, COMPACT, LARGE komutları ile derleyiciye bildirilir. SMALL modelinde hafıza tipi "data", COMPACT modelinde hafıza tipi "pdata", LARGE modelinde hafıza tipi "xdata" olarak belirlenir.

4.3Hafıza modelleri

Hafıza modeli, hafıza tipi verilmemiş değişkenlerin otomatik ayarlanması için hafıza tipini belirler. C51'in özelliği olarak değişkenler sabit adreslerde bulunurlar. SMALL modelinde iç veri hafızası kullanılır. LARGE ve COMPACT modelleri dış veri hafızasını kullanırlar.

Tablo B.3 C51 derleyicisinde hafıza modelleri

Hafıza Modeli	Tanım
SMALL	Değişkenler ve parametreler doğrudan adreslenebilen iç veri hafızasına yerleştirilir. Hafıza tipi "DATA"dır.
COMPACT	Değişkenler ve parametreler 256 bayt olarak adreslenebilen dış veri hafızasına yerleştirilir. Hafıza tipi "PDATA"dır.
LARGE	Değişkenler ve parametreler doğrudan adreslenebilen dış veri hafızasına yerleştirilir. Hafıza tipi "XDATA"dır.

Pointer

C51 derleyicisi "memory spesific" ve "generic" olmak üzere iki pointer tipini destekler. Birincisinin hafıza tipi kaynak kodunda belirtilir ve derleme anında bilinir. 1 veya 2 bayt gerektirir. İkinci pointer tipi 3 bayta ihtiyaç duyar. 1 bayt hafıza için, 2 bayt gerçek adres için kullanılır. Hafıza tipi değişkenin bulunduğu hafıza alanını belirler. Böylece hafıza tipi farklı değişkenler arasında işlemler yapılabilir.

Tablo B.4 C51 derleyicisinde pointer

Tanım	Uzunluk	Gösterdiği değişken
float *p;	3 bayt	Bütün hafıza alanlarındaki "float" değişken adresi.
char data *dp;	1 bayt	"data" alanındaki "char" değişken.
int idata *ip;	1 bayt	"idata" alanındaki "int" değişken.
long pdata *pp;	1 bayt	"pdata" alanındaki "long" değişken.
char xdata *xp;	2 bayt	"xdata" alanındaki "char" değişken.
int code *cp;	2 bayt	"code" alanındaki "int" değişken.

Kesme yapısı

C51 derleyicisi kesme yapılarını ve "register bank" kontrolünü destekler. Böylece etkili kesme programları yazılabilir. Aşağıda zaman ölçümü için örnek bir program verilmiştir:

```
unsigned char cnt , scn ;
timer0 () interrupt 1 using 1   /* 'Register Bank 1'i kullanan timer0 kesme programı */
{
if (++cnt = = 250)  { scn ++ ;  cnt = 0 ; }
}
```

Parametre geçişleri

3 parametreye kadar işlemcinin iç yazmaçları kullanılır. Eğer iç yazmaçlar kullanılıyorsa yada "#pragma NOREGPARMS" komutu kullanıldıysa sabit hafıza alanları parametre geçişinde kullanılır.

Tablo B.5 C51 derleyicisinde parametre geçişleri

Parametre Tipi	char, 1 bayt ptr	int, 2 bayt ptr	long, float	generic ptr
1. Parametre	R7	R6, R7	R4...R7	R1, R2, R3
2. Parametre	R5	R4, R5	R4...R7	R1, R2, R3
3. Parametre	R3	R2, R3		R1, R2, R3

Fonksiyonun dönüş değerleri işlemcinin iç yazmaçları üzerinden olur. Böylece makina dilinde yazılmış programlar ile bağlantı sağlanabilir.

Tablo B.6 C51 derleyicisinde parametre dönüşleri

Dönüş Değeri	Yazmaç	Anlamı
bit	C biti	
(unsigned) char	R7	
(unsigned) int	R6, R7	R6 MSB, R7 LSB
(unsigned) long	R4...R7	R4 MSB, R7 LSB
float	R4...R7	32-bit IEEE formatı
pointer	R1, R2, R3	R3 seçici, R2 MSB, R1 LSB

Kod sadeleştirme

C51 derleyicisi makina dili kodları üretebilir. Böylece yazılan programın makina dilindeki karşılığı gözlenebilir. Kullanıcı için 1...5 arasında beş adet sadeleştirme seviyesi vardır. Her seviye bir önceki seviyeyi de kapsar. Her seviyede uzunluk (SIZE) ve hız (SPEED) olarak iki seçenekten biri seçilmelidir. Derleyiciye "OPTIMIZE(5,SIZE)" komutu ile bildirilir.

Programda hata ayıklama

C51 derleyicisi, Intel Object Format (OMF51) kullanır ve değişken listesini üretir. Intel uyumlu bütün geliştirme sistemleri hata ayıklama için kullanılabilir. Bununla birlikte "OBJECTEXTEND" komutu değişkenlerin tiplerini de ("char", "int" gibi) listeye ekler.

C51 kütüphanesi

C51 derleyicisi, çeşitli fonksiyonlar için geliştirilmiş altı değişik derleme kütüphanesi içerir. Bu kütüphaneler standard ANSI fonksiyon çağırımlarını destekler. C51 derleyicisi, donanıma bağlı uygulamalarda ihtiyaç duyulan bütün kütüphane modüllerini kaynak kod olarak içerir. Kullanıcının kütüphane fonksiyonlarını donanıma uydurabilmesi için sadece giriş/çıkış modüllerinde değişiklik yapması yeterlidir.

C51 derleyicisinde kütüphaneler

Kütüphane	Tanım
C51S.LIB	SMALL model kütüphane.

C51FPS.LIB	SMALL model kayan-nokta-aritmetik kütüphane.
C51C.LIB	COMPACT model kütüphane.
C51FPC.LIB	COMPACT model kayan-nokta-aritmetik kütüphane.
C51L.LIB	LARGE model kütüphane.
C51FPL.LIB	LARGE model kayan-nokta-aritmetik kütüphane.

L51 bağlayıcı (Linker)

L51 bağlayıcı ayrı program modüllerini birtek 8051 program kodunda tümleştirir. İç ve dış değişkenler çözümlenir ve değişkenler mutlak adreslere yerleştirilir. Birleştirilecek modüller C, makina dili yada PL/M51 ile yazılabilir. L51 gerekli kütüphaneleri seçer ve gerekli program parçalarını hedef programa bağlar.

Veri adresi yönetimi

C derleyicisi 8051 işlemcisinin küçük iç veri hafızasını üst üste çakıştırma yöntemiyle yönetir. Bu yöntemle farklı fonksiyonlardaki parametreler ve yerel değişkenler aynı fiziksel adresi kullanır. Böylece hafıza ihtiyacı önemli ölçüde azalır. "OVERLAY" komutu ile L51 bağlayıcısı fonksiyonlardaki değişkenlerin adres paylaşımını sağlar. "NOOVERLAY" komutu adres paylaşımını engeller.

BÖLÜM 5. SONUÇLAR

Yapılan uygulamalar sonucunda hedef kartın tüm devrelerini kapsayan otomatik test cihazının performansı tamamen doyurucu bulunmuştur.Çünkü klasik yöntemlerle herhangi bir motor kontrol kartında arızanın çeşidine göre günlerce sürebilecek test çalışmaları en fazla 10 dakikalık bir zamana indirgenebilmiştir.Aynı zamanda hedef kartın tüm devreleri fonksiyonel yaklaşımla test edildiği için tamirden geçen kartın geri dönme riski sıfırlanabilmektedir. Kart sayesinde komponent bazında arıza tesbiti mümkün olmaktadır.

Sistemin tek dezavantajı büyük mühendislik emeği harcanarak yapılan tasarımın yalnızca tek bir karta hitap edebilmesidir. Farklı kartların test edilmesi gerektiği zaman ana kartta hiçbir çalışmada bulunmadan yalnızca arabağlaşım ünitesi tekrar tasarlanarak
sistemin servis sahasını genişletmek mümkündür.Bu durum halihazırda otomatik test cihazları sektöründe mevcut olan profesyonel çalışmalarla kıyaslandığında sistemin bir zaafı olarak ortaya çıkmaktadır. Şu an için 128 hatta 512 giriş - çıkış hattına sahip otomatik test cihazları mevcuttur. Bu test cihazlarında her bir hat istenen lojik seviyelere set edilerek PC yardımı ile yazılan program ile çok geniş bir sahada farklı kartların fonksiyonel yaklaşımla test edilmesini mümkün kılınmaktadır. Ölçü olarak bu cihazlar ele alındığında bu tez çalışması bir çekirdek yapı teşkil etmektedir.Gerekli I\0 kartları ilavesi yapılarak bu sistemin de giriş çıkış hattı sayısını donanım imkanlarının elverdiği ölçüde artırmak mümkündür.Bu noktada sistemimiz fiyat açısından herbiri onbinler dolar mertebesinde olan bu test cihazlarına karşı bir avantajda sağlamaktadır.

KAYNAKLAR

1) "Data Acquisition IC Handbook " Teledyne Semiconductor, 1993

2) MANO , M.M " Computer System Architectecture " Prentice / Hall International Inc. 1984

3) " Soft Microcontroller " , Dallas Semiconductor , 1993

4) " C - Compiler -51" Keil Elektronik GmbH, 1992

5) " PAL Device Data Book " Monolithic Memories Inc. 1988

6) " Otomatik Arıza Bulma Sistemleri " Gate Elektronik 1994

7) "UTS6000 Functional Tester " ATE Electronics Denmark 1996

8) " Yüksek Lisans Tezi " Sakarya Ünv. Ayhan Güneşdoğdu 1996

EK1. ANA KART PROGRAMI

Ana karta ait program kısmen özel bir C dili kullanarak büyük çoğunluklada makina dili düzeyinde yazılmıştır.Aşağıdaki liste ana kart programının tam bir dökümünü vermektedir.

```
;       r7 : Receive
;       r6 : Adc
;       r3 : Display
;       r4 : Delay
;       r5 : Delay
;       r2 :
;       r1 : Sw test
;       r0 : Sw test

;************** THE ADDRESSES ****************************************

ULN1        EQU   0C000h
ULN2            EQU   0D000h
WR80            EQU   0B000h
WF80            EQU   0A000h
RF80            EQU    8000h
RD80            EQU    9000h
RAM_ADR             EQU   2000h          ; 6264 ( 62256 ) start addres

;*********************************************************************

DIG_NUM             EQU   16         ; the number of max.characters for 1.column
LCDCOM_ADR EQU   0FF00h    ; RS(A1) = 0 , R_W(A0) = 0
LCDREAD_ADR         EQU   0FF01h    ; RS(A1) = 0 , R_W(A0) = 1
LCDATA_ADR  EQU   0FF02h    ; RS(A1) = 1 , R_W(A0) = 0

;************** THE CONSTANT *****************************************

OPEN_COD                    EQU   55h
COM_OK_COD                  EQU   20
ESC_COD                     EQU   21

WRONG_COD                   EQU   1
ROBUST_TEST_COD             EQU   2
QUICK_TEST_COD              EQU   3
ROBUST_STRONG_COD           EQU   4
ROBUST_FAIL_COD             EQU   5
QUICK_STRONG_COD            EQU   6
QUICK_FAIL_COD              EQU   7

IC75_TEST_COD               EQU   8
IC75_STRONG_COD             EQU   9
IC75_FAIL_COD               EQU   10

IC76_TEST_COD               EQU   11
IC76_STRONG_COD             EQU   12
IC76_FAIL_COD               EQU   13
```

```
I59_FAIL_COD        EQU   14
I58_FAIL_COD        EQU   15
ALL_OPT_COD         EQU   16

IC78_TEST_COD               EQU   17
ADC_TEST_COD                EQU   18
SWITCH_TEST_COD             EQU   19

CTC_TEST_COD                EQU   20
LS39_COD            EQU   21
VFC_COD                     EQU   22


;************** THE VARIABLES ****************************************

            DSEG AT 8h      ;occupies BANK1
BUF0:       DS    1         ;8
BUF1:       DS    1         ;9
BUF2:       DS    1         ;A
BUF3:       DS    1         ;B
BUF4:       DS    1         ;D
STORE:      DS    1         ;E
TC_BUF:             DS    1         ;F
            DSEG AT 30h
?Load_16?byte: DS     2     ;30h-31h 16-bit data to be operated
?Div_16?byte:  DS     2     ;32h-33h 16-bit divider
?Sub_16?byte:  DS     2     ;34h-35h 16-bit data to be subtracted
?Mul_16?byte:   DS    2     ;36h-37h
OP_0:       DS  1           ;These locations belong to 32-bit pseudo_Acc
OP_1:       DS  1           ;resident in memory
OP_2:       DS  1
OP_3:       DS  1
TMP_0:       DS         1   ;3Ch
TMP_1:       DS         1   ;3Dh
TMP_2:       DS         1   ;3Eh
TMP_3:       DS         1   ;3Fh
H_RSLT:             DS    1         ;40h
L_RSLT:             DS    1         ;41h

;*********************************************************************

            cseg    at      2080h

BEGIN:      clr     EA                      ; MON51 stop!!!
            mov     0C8h,#0                 ; t2con := 0
            clr     ie.5                    ; TF2 := 0
            clr     ES              ; serial port int
            clr     RS0
            clr     RS1                     ; RegisterBank0
            mov     sp,#(127-40)            ; 40 byte stack

;*************************************************

            call COM_INIT
            call LCD_INIT
```

```
                mov a,#50h
                mov dptr,#WR80
                movx @dptr,a     ;so as to make Z-80 wait
                call DELAY
                mov     a,#OPEN_COD
                call      SEND
                call   STROBE
                call   RECEIVE
                cjne   r7,#COM_OK_COD,COM_ERR
                jmp    SKIP
COM_ERR:        jmp    COM_ERROR
SKIP:           call      CLEAR
                mov       dptr,#RAM2K_TEST
                call      WRITE_LCD
                call   SCAN_KEY
                cjne   a,#'*',RUN_RAM
                mov    a,#ESC_COD
        call   SEND
                jmp       ANA_MENU

RUN_RAM:        call   CLEAR
                mov    dptr,#RAM_TEST_OPTION
                call   WRITE_LCD
TEK_RAM:        call   SCAN_KEY
        cjne   a,#'1',QUICK
        mov    a,#ROBUST_TEST_COD
        call   SEND

                call   CLEAR
                mov    dptr,#RAM_TEST
                call   WRITE_LCD

        call   STROBE
        call   RECEIVE

        cjne   r7,#ROBUST_STRONG_COD,RAM_ERROR

                call   CLEAR
                mov    dptr,#RAM_OK_MSG
                call   WRITE_LCD
        call   SCAN_KEY
                jmp       ANA_MENU

QUICK:      cjne   a,#'2',TEK_RAM
                mov    a,#QUICK_TEST_COD
                call   SEND

                call   CLEAR
                mov    dptr,#RAM_TEST
                call   WRITE_LCD

            call   STROBE
        call   RECEIVE
        cjne   r7,#QUICK_STRONG_COD,RAM_ERROR
```

```
                call   CLEAR
                mov    dptr,#RAM_OK_MSG
                call   WRITE_LCD
          call  SCAN_KEY

                jmp      ANA_MENU

COM_ERROR:  call   CLEAR
                mov    dptr,#COM_ERR_MSG
                call   WRITE_LCD
                call   SCAN_KEY
                jmp    ANA_MENU

RAM_ERROR:  call   CLEAR
                mov    dptr,#RAM_FAULT_MSG
                call   WRITE_LCD
                call   SCAN_KEY
                jmp    ANA_MENU
;*****************************************
;*****************************************

ANA_MENU:   call       CLEAR
                mov    dptr,#ANA_MENU_MSG
                call       WRITE_LCD
                call   SCAN_KEY
          cjne  a,#'A',DEVICE_B
;*********************************************
DEVICE_A:     call       CLEAR
                mov    dptr,#IC75_TEST_MSG
                call       WRITE_LCD
                mov    a,#IC75_TEST_COD
                call       SEND
                call       IC_75
                jmp    ANA_MENU
;*********************************************
DEVICE_B:     cjne   a,#'B',DEVICE_C
                call       CLEAR
                mov    dptr,#IC76_TEST_MSG
                call       WRITE_LCD
                mov    a,#IC76_TEST_COD
                call       SEND
                call       IC_76
                jmp    ANA_MENU
;*********************************************
DEVICE_C:     cjne   a,#'C',DEVICE_D
                call       CLEAR
                mov    dptr,#IC78_TEST_MSG
                call       WRITE_LCD
                mov    a,#IC78_TEST_COD
                call       SEND

RUNNING:      call   SCAN_KEY
                cjne   a,#'*',RUNNING
                mov    a,#ESC_COD
                call       SEND
```

```
                jmp     ANA_MENU
;*********************************************
DEVICE_D:       cjne    a,#'D',DEVICE_0
                mov     a,#ADC_TEST_COD

                call    SEND
                call    ADC_TEST
                jmp     ANA_MENU
;*********************************************
DEVICE_0:       cjne    a,#'0',DEVICE_E
                mov     a,#SWITCH_TEST_COD

                call    SEND
                call    CLEAR
                mov     dptr,#SW_MSG
                call    WRITE_LCD
                call    DISPLAY
                jmp     ANA_MENU
;*********************************************

DEVICE_X:       call    CLEAR
                mov     dptr,#UNVALID_KEY
                call    WRITE_LCD
                call    DELAY
                jmp     ANA_MENU
;*********************************************

DEVICE_E:       cjne    a,#'E',DEVICE_X
                mov     a,#CTC_TEST_COD
                call    SEND
                call    CTC_TEST
                jmp     ANA_MENU
;*********************************************
;*********************************************

                lcall   6       ; MON51
;*********************************************
MANUEL:                 ;takes TC_BUF and convert into Decimal then displays the
                ;corresponding DAC output value in format XX,XXX V

                mov     a,TC_BUF
                mov     b,#100
                div     ab
                mov     BUF2,a          ;digit of 100
                mov     a,#30h
                add     a,BUF2
                mov     BUF2,a
                mov     a,b
                mov     b,#10
                div     ab
                mov     BUF1,a          ;digit of 10
                mov     a,#30h
                add     a,BUF1
                mov     BUF1,a
```

```
                mov     BUF0,b          ;1s
                mov     a,#30h
                add     a,BUF0
                mov     BUF0,a
                mov     a,#8Ah
                call    CURSOR_TO

                mov     a,BUF2
                call    WR_DISP_DATA
                mov     a,BUF1
                call    WR_DISP_DATA
                mov     a,BUF0
                call    WR_DISP_DATA

        mov   a,#0Ch
        call  WR_DISP_COM

                mov     a,TC_BUF
                cjne    a,#0,INT_STG_MN
                mov     a,#255
                mov     b,#41           ;( Data X 41 ) = XX,XXX V
                mul   ab
                add     a,#40
                sjmp    EXTNT
INT_STG_MN:     mov     b,#41           ;( Data X 41 ) = XX,XXX V
                mul   ab
EXTNT:          mov     H_RSLT,b
                mov     L_RSLT,a
                call  MULTPLY
                mov     a,#0C9h
                call    CURSOR_TO

                mov     a,BUF4
                call    WR_DISP_DATA
                mov     a,BUF3
                call    WR_DISP_DATA
                mov     a,#','
                call    WR_DISP_DATA
                mov     a,BUF2
                call    WR_DISP_DATA
                mov     a,BUF0
                call    WR_DISP_DATA
                mov     a,BUF1
                call    WR_DISP_DATA

        mov   a,#0Ch
        call  WR_DISP_COM
                ret

;***********************************************
CTC_TEST:       call    CLEAR
                mov   dptr,#VFC_MSG
                call  WRITE_LCD
STRV:           call    SCAN_KEY
                cjne    a,#'B',CROSS_VFC
```

```
                mov     a,#VFC_COD
                call    SEND

                call    CLEAR
                mov   dptr,#ESC_MSG
                call  WRITE_LCD
                call    SCAN_KEY

                mov     dptr,#ULN1
                mov     a,#08h
                movx    @dptr,a
                nop
                nop
                mov     a,#0
                movx    @dptr,a

                call  STROBE
                call    RECEIVE
                cjne    r7,#ESC_COD,MISTAKE
                ret
MISTAKE:        jmp   COM_ERROR

CROSS_VFC:      cjne    a,#'A',STRV
                call    CLEAR
                mov   dptr,#LS239_MSG
                call  WRITE_LCD
INV_KEY:        call    SCAN_KEY
INT_VFC:        cjne    a,#'1',H_FRQ
                mov     a,#LS39_COD
                call    SEND
                sjmp    INT_OK
H_FRQ:          cjne    a,#'2',INV_KEY
                mov     a,#1
                call    SEND
                sjmp  INT_OK
INT_OK:                 mov   a,#128
                mov     dptr,#WR80
                movx    @dptr,a
                mov     TC_BUF,a
AG_TC:          call    CLEAR
                mov   dptr,#CTC_OPT_MSG
                call  WRITE_LCD

AG_TC0:                 call      SCAN_KEY
                cjne    a,#'A',STR
AG_TCX:                 call      CLEAR
                mov   dptr,#CTC0_MSG
                call  WRITE_LCD
                jmp   OPT_A
STR:            cjne    a,#'B',TC_ESC
                mov     p1,#11111110b
                sjmp    TC_DVM
TC_ESC:                 cjne      a,#'*',AG_TC
                mov     a,#ESC_COD
                call    SEND
```

```
                ret

TC_DVM:                 call      CLEAR
                mov    dptr,#OPT_B0_MSG
                call   WRITE_LCD

MNTR:           mov     p1,#11111110b   ; COL0 = 0
MNTR_S:                 jnb       p1.7,AG_TC
                jnb     p1.5,DOWN_TC
                jnb     p1.4,UP_TC
                sjmp    MNTR

DOWN_TC:        dec    TC_BUF
                mov     a,TC_BUF
                mov     dptr,#WR80
                movx    @dptr,a
                call    CLEAR
                mov    dptr,#OPT_B_MSG
                call   WRITE_LCD

                call    MANUEL
                call    DELAY_TC        ;waiting of 110 msn
                jmp     MNTR_S

UP_TC:          inc     TC_BUF
                mov     a,TC_BUF
                mov     dptr,#WR80
                movx    @dptr,a
                call    CLEAR
                mov    dptr,#OPT_B_MSG
                call   WRITE_LCD
                call    MANUEL
                call    DELAY_TC        ;waiting of 110 msn
                jmp     MNTR_S

OPT_A:          call    SCAN_KEY
                mov     BUF2,a
        cjne    a,#'2',GO_DEV_E1
                sjmp    DVM_E_BUF1
GO_DEV_E1:      cjne    a,#'1',GO_DEV_E0
                sjmp    DVM_E_BUF1
GO_DEV_E0:      cjne    a,#'0',NON_OF
                sjmp    DVM_E_BUF1
NON_OF:                 jmp       AG_TC0
DVM_E_BUF1:     call    CLEAR
                mov     dptr,#CTC1_MSG
                call   WRITE_LCD
                mov     a,#8Ah
                call   CURSOR_TO
                mov     a,BUF2
                call    WR_DISP_DATA
RET_0:          call    SCAN_KEY
                mov     BUF1,a
                mov     a,#'2'
        cjne    a,BUF2,STAGE_09
```

```
                mov     a,BUF1                  ;0?....5
                cjne    a,#'0',CHECK_0
                sjmp    CH_EXIT
CHECK_0:        cpl     c
                jc      GO_ON0
                jmp     RET_0
GO_ON0:                 mov     a,BUF1                  ;0....5?
                cjne    a,#'5',CHECK_5
                sjmp    CH_EXIT
CHECK_5:        jc      CH_EXIT
                jmp     RET_0
CH_EXIT:        jmp     CH_EXIT09
STAGE_09:       mov     a,BUF1                  ;0?....9
                cjne    a,#'0',CHECK_0A
                sjmp    CH_EXIT
CHECK_0A:       cpl     c
                jc      GO_ON09
                jmp     RET_0
GO_ON09:        mov     a,BUF1                  ;0....9?
                cjne    a,#'9',CHECK_9
                sjmp    CH_EXIT09
CHECK_9:        jc      CH_EXIT09
                jmp     RET_0
CH_EXIT09:      mov     a,BUF1
                call    WR_DISP_DATA
RET_00:                 call    SCAN_KEY
                mov     BUF0,a

                mov     a,#'2'
                cjne    a,BUF2,NORMAL
                mov     a,#'5'
                cjne    a,BUF1,NORMAL
                mov     a,BUF0          ;0?....5
                cjne    a,#'0',CHECK_00
                sjmp    CH_EXIT0
CHECK_00:       cpl     c
                jc      GO_ON00
                jmp     RET_00
GO_ON00:        mov     a,BUF0                  ;0....5?
                cjne    a,#'5',CHECK_5A
                sjmp    CH_EXIT0
CHECK_5A:       jc      CH_EXIT0
                jmp     RET_00
CH_EXIT0:       jmp     FLT_OK
NORMAL:                 mov     a,BUF0                  ;0?....9
                cjne    a,#'0',CHECK_09A
                sjmp    CH_EXIT0
CHECK_09A:      cpl     c
                jc      GO_ON090
                jmp     RET_00
GO_ON090:       mov     a,BUF0                  ;0....9?
                cjne    a,#'9',CHECK_09
                sjmp    FLT_OK
CHECK_09:       jc      FLT_OK
                jmp     RET_00
```

```
FLT_OK:         call   WR_DISP_DATA

                mov    a,BUF2
                clr    c
                subb   a,#30h
                mov    b,#100
                mul    ab
                mov    BUF2,a
                mov    a,BUF1
                clr    c
                subb   a,#30h
                mov    b,#10
                mul    ab
                mov    BUF1,a
                mov    a,BUF0
                clr    c
                subb   a,#30h
                mov    BUF0,a

                mov    a,BUF2
                add    a,BUF1
                add    a,BUF0
                mov    STORE,a          ;Time constant was acquired

CONFIRM:        call   SCAN_KEY
                cjne   a,#'E',CONFIRM
                call   CLEAR
                mov    dptr,#CTC_RD_MSG
                call   WRITE_LCD
                mov    a,STORE
                mov    dptr,#WR80
                movx   @dptr,a
                mov    TC_BUF,a
                cjne   a,#0,INT_STG
                mov    a,#255
                mov    b,#41            ;( Data X 41 ) = XX,XXX V
                mul    ab
                add    a,#40
                sjmp   EXTENT
INT_STG:        mov    b,#41            ;( Data X 41 ) = XX,XXX V
                mul    ab

EXTENT:         mov    H_RSLT,b
                mov    L_RSLT,a

                call   MULTPLY

                mov    a,#85h
                call   CURSOR_TO

                mov    a,BUF4
                call   WR_DISP_DATA
                mov    a,BUF3
                call   WR_DISP_DATA
                mov    a,#','
```

```
                call    WR_DISP_DATA
                mov     a,BUF2
                call    WR_DISP_DATA
                mov     a,BUF0
                call    WR_DISP_DATA
                mov     a,BUF1
                call    WR_DISP_DATA

        mov     a,#0Ch
        call    WR_DISP_COM

                mov     a,#80h          ; Set DD RAM Adress to 40
                call    CURSOR_TO
JMP_SCAN:       call    SCAN_KEY
                cjne    a,#'*',JMP_AG_TC
                jmp     AG_TC

JMP_AG_TC:      cjne    a,#'0',JMP_SCAN
                jmp     AG_TCX
                ret
;************************************************
MULTPLY:        ;requires a product of multiplying

                mov     ?load_16?byte,b         ;High order
                mov     ?load_16?byte + 1,a     ;low order
                call    LOAD_16                         ;16-bit result was loaded
                                                ;into pseudo Acc
                mov     ?div_16?byte,#27h
                mov     ?div_16?byte + 1,#10h   ;The divisor was loaded

                call    DIV_16                  ;The result stored into OP's
                mov     a,#30h                  ;so as to converse into ASCII
                add     a,op_0
                mov     BUF4,a          ;4.digit means the order of 10000

                mov     ?Mul_16?byte,#27h
                mov     ?Mul_16?byte + 1,#10h
                call    MUL_16                          ;3x10000

                mov     ?Sub_16?byte,op_1
                mov     ?Sub_16?byte + 1,op_0

                mov     ?load_16?byte,H_RSLT            ;High order
                mov     ?load_16?byte + 1,L_RSLT        ;low order
                call    LOAD_16

                call    SUB_16
;-----------------------------
                mov     H_RSLT,op_1
                mov     L_RSLT,op_0

                mov     ?div_16?byte,#3h
                mov     ?div_16?byte + 1,#0E8h  ;The divisor was loaded

                call    DIV_16                  ;The result stored into OP's
```

```
mov     a,#30h
add     a,op_0
mov     BUF3,a              ;3.digit means the order of 1000

mov     ?Mul_16?byte,#3h
mov     ?Mul_16?byte + 1,#0E8h
call    MUL_16                          ;9x1000

mov     ?Sub_16?byte,op_1
mov     ?Sub_16?byte + 1,op_0

mov     ?load_16?byte,H_RSLT            ;High order
mov     ?load_16?byte + 1,L_RSLT        ;low order
call    LOAD_16

call    SUB_16

mov     H_RSLT,op_1
mov     L_RSLT,op_0

mov     ?div_16?byte,#0h
mov     ?div_16?byte + 1,#64h   ;The divisor was loaded

call    DIV_16                  ;The result stored into OP's
mov     a,#30h
add     a,op_0
mov     BUF2,a              ;2.digit means the order of 100

mov     ?Mul_16?byte,#0h
mov     ?Mul_16?byte + 1,#64h
call    MUL_16                          ;7x100

mov     ?Sub_16?byte,op_1
mov     ?Sub_16?byte + 1,op_0

mov     ?load_16?byte,H_RSLT            ;High order
mov     ?load_16?byte + 1,L_RSLT        ;low order
call    LOAD_16

call    SUB_16

mov     a,op_0
mov     b,#10
div     ab

mov     BUF1,b
mov     BUF0,a

mov     a,#30h
add     a,BUF1
mov     BUF1,a
mov     a,#30h
add   a,BUF0
mov     BUF0,a
ret
```

```
;*************************************************
DISPLAY:    mov     p1,#11111110b   ; COL0 = 0
RE_READ:    jnb     p1.7,DSP_EXIT
            mov     a,#84h          ; Set DD RAM Adress to 40
            call    CURSOR_TO
            mov     r0,#8
            mov     dptr,#RD80
            movx    a,@dptr         ;data to display was received.
DSP:        rrc     a               ;bitn 1 or 0
            mov     r1,a
            jc      DISP1
            mov     a,#'0'
            call    WR_DISP_DATA
            mov     a,r1
            djnz    r0,DSP
            sjmp    RE_READ
DISP1:      mov     a,#'1'
            call    WR_DISP_DATA
            mov     a,r1
            djnz    r0,DSP
            sjmp    RE_READ
DSP_EXIT:   ret

;*************************************************
IC_75:      mov     dptr,#ULN1
            mov     a,#0FFh
            movx    @dptr,a
            call    STROBE
            call    RECEIVE
            cjne    r7,#0FFh,IC75_ERR

      mov   dptr,#ULN1
            mov     a,#8h
            movx    @dptr,a
            call    STROBE
            call    RECEIVE
            cjne    r7,#8h,IC75_ERR

      mov   dptr,#ULN1
            mov     a,#0AAh
            movx    @dptr,a
            call    STROBE
            call    RECEIVE
            cjne    r7,#0AAh,IC75_ERR

            call    CLEAR
            mov     dptr,#IC75_OK_MSG
            call    WRITE_LCD
            call    SCAN_KEY
            ret

IC75_ERR:   cjne    r7,#WRONG_COD,COM_FAIL
            call    CLEAR
            mov     dptr,#IC75_FAIL_MSG
```

```
                call   WRITE_LCD
                call   SCAN_KEY
                ret

COM_FAIL:       call   CLEAR
                mov    dptr,#UNDEFINED_MSG
                call   WRITE_LCD
                call   SCAN_KEY
                ret
;*****************************************

IC_76:          mov    dptr,#ULN2
                mov    a,#0FFh
                movx   @dptr,a
                call   STROBE
                call   RECEIVE
                cjne   r7,#0FFh,IC76_ERR

                mov    dptr,#ULN2
                mov    a,#0h
                movx   @dptr,a

                call   STROBE
                call   RECEIVE
                cjne   r7,#0C0h,IC76_ERR

        call   CLEAR
                mov    dptr,#IC76_OK_MSG
                call   WRITE_LCD
                call   SCAN_KEY
                ret

IC76_ERR:       cjne   r7,#I59_FAIL_COD,I58
                call   CLEAR
                mov    dptr,#I59_MSG
                call   WRITE_LCD
                call   SCAN_KEY
                ret

I58:            cjne   r7,#I58_FAIL_COD,ALL0
                call   CLEAR
                mov    dptr,#I58_MSG
                call   WRITE_LCD
                call   SCAN_KEY
                ret

ALL0:           cjne   r7,#ALL_OPT_COD,WRN_OK
                call   CLEAR
                mov    dptr,#ALL_MSG
                call   WRITE_LCD
                call   SCAN_KEY
                ret

WRN_OK:             cjne   r7,#WRONG_COD,COM_FAIL2
                call   CLEAR
```

```
                mov   dptr,#IC76_FAIL_MSG
                call  WRITE_LCD

                mov   dptr,#RD80
                movx  a,@dptr
                mov   dptr,#4000h
                movx  @dptr,a

                call  SCAN_KEY
                ret

COM_FAIL2:      call    CLEAR
                mov   dptr,#UNDEFINED_MSG
                call  WRITE_LCD
                call  SCAN_KEY
                ret
;*****************************************
ADC_TEST:
VOID:           call    CLEAR
                mov     dptr,#ADC_TEST_MSG
                call    WRITE_LCD

VOID1:          call    SCAN_KEY
                mov     r6,a        ;press stored in r6
                cjne    a,#'*',CONT
                jmp   ADC_RTN

CONT:           anl     a,#0F8h
                cjne    a,#30h,VOID1    ;key press between ' 0....7 '

                mov     a,r6        ;retrive acc
                anl     a,#07h              ;to send channel adr.
                mov     r6,a

IN0:            cjne    a,#0h,IN1
                call    CLEAR
                mov     dptr,#INPUT0_MSG
                call  WRITE_LCD

                mov     a,r6
                call    SEND            ;which chan to monitor
                call    ADC_DISP
                jmp     VOID

IN1:            cjne    a,#1h,IN2
                call    CLEAR
                mov     dptr,#INPUT1_MSG
                call  WRITE_LCD

                mov     a,r6
        call    SEND
                call    ADC_DISP
                jmp     VOID

IN2:            cjne    a,#2h,IN3
```

```
                call    CLEAR
                mov     dptr,#INPUT2_MSG
                call    WRITE_LCD

                mov     a,r6
                call    SEND
                call    ADC_DISP
                jmp     VOID

IN3:            cjne    a,#3h,IN4
                call    CLEAR
                mov     dptr,#INPUT3_MSG
                call    WRITE_LCD

                mov     a,r6
                call    SEND
                call    ADC_DISP
                jmp     VOID

IN4:            cjne    a,#4h,IN5
                call    CLEAR
                mov     dptr,#INPUT4_MSG
                call    WRITE_LCD

                mov     a,r6
        call    SEND
                call    ADC_DISP
                jmp     VOID

IN5:            cjne    a,#5h,IN6
                call    CLEAR
                mov     dptr,#INPUT5_MSG
                call    WRITE_LCD

                mov     a,r6
                call    SEND
                call    ADC_DISP
                jmp     VOID

IN6:            cjne    a,#6h,IN7
                call    CLEAR
                mov     dptr,#INPUT6_MSG
                call    WRITE_LCD

                mov     a,r6
                call    SEND
                call    ADC_DISP
                jmp     VOID

IN7:            call    CLEAR
                mov     dptr,#INPUT7_MSG
                call    WRITE_LCD

                mov     a,r6
                call    SEND
```

```
                call    ADC_DISP
                jmp     VOID

ADC_RTN:        mov     a,#ESC_COD
                call    SEND
                ret

;***********************************************

ADC_DISP:       mov     p1,#11111110b   ; COL0 = 0
                clr     a
                mov     STORE,a

DISP_CONT:      mov     dptr,#RD80
                movx    a,@dptr         ;read ADC

                mov     b,#156          ;( Data X 156 ) / 8 = Data x 19.5 mV
                mul     ab
                rr      a
                rr      a
                rr      a
                anl     a,#1Fh

                mov     r0,a            ;000X XXXX
                mov     a,b
                rr      a
                rr      a
                rr      a
                mov     r1,a
                anl     a,#1Fh

                mov     r3,a            ;000X XXXX
                mov     a,r1
                anl     a,#0E0h
                mov     b,r0
                orl     a,b
                mov     r5,a
                mov     a,r3
                mov     r6,a

                mov     H_RSLT,r6
                mov     L_RSLT,r5

                call    MULTPLY

                mov     a,#87h
                call    CURSOR_TO

                mov     a,BUF4
                call    WR_DISP_DATA
                mov     a,BUF3
                call    WR_DISP_DATA
                mov     a,BUF2
                call    WR_DISP_DATA
                mov     a,BUF1
```

```
                call    WR_DISP_DATA
                mov     a,BUF0
                call    WR_DISP_DATA

        mov     a,#0Ch
        call    WR_DISP_COM

                mov     a,#80h          ; Set DD RAM Adress to 40
                call    CURSOR_TO

                jb      p1.7,INTER

ADC_DSP_RTN:    ret

INTER:          jmp     DISP_CONT

;*****************************************
D_DECIMAL:      mov     r0,a
                anl     a,#0Fh
                cjne    a,#10,DECIMAL
OVER_DEC:       subb    a,#10
                add     a,#41h
                mov     r3,a
                sjmp    H_NIBBLE
DECIMAL:        jnc     OVER_DEC
                add     a,#30h
                mov     r3,a
H_NIBBLE:       mov     a,r0
                swap    a
                anl     a,#0Fh
                cjne    a,#10,DECIMAL_H
OVER_DEC_H:     subb    a,#10
                add     a,#41h
                mov     r4,a
                ret
DECIMAL_H:      jnc     OVER_DEC_H
                add     a,#30h
                mov     r4,a
                ret
;*********************************************
COM_INIT:       mov     dptr,#WF80
                mov     a,#02h          ;bit0 = 0 as IBFA of 8032
                movx    @dptr,a                 ;bit1 of WF80 = 1 as OBFA
                ret
;*********************************************

SEND:           mov     dptr,#WR80
                movx    @dptr, a        ;data written

                mov     dptr,#WF80
                mov     a,#0h
                movx    @dptr,a             ;set OBFA active

                mov     dptr,#RF80
WAIT1:          movx    a,@dptr             ;wait for ACK to rise from 0 to 1
```

```
                anl     a,#1
                jnb     acc.0,WAIT1     ;Z-80 has not received STROBE as of yet

WAIT0:          movx    a,@dptr         ;Z-80 received STROBE
                anl     a,#1
                jb      acc.0,WAIT0     ;reading.....
                                        ;Z-80 completed reading
                mov     dptr,#WF80
                mov     a,#02h
                movx    @dptr,a         ;OBFA was released.
                ret
;*****************************************

STROBE:         mov     dptr,#RF80
WT0:            movx    a,@dptr
                jb      acc.1,WT0
                call    DELAY_L

                mov     dptr,#RF80
WT01:           movx    a,@dptr
                jb      acc.1,WT01
                ret
;*****************************************

RECEIVE:        mov     dptr,#WF80
                mov     a,#03h
                movx    @dptr,a         ;set IBFA active

                mov     dptr,#RD80
                movx    a,@dptr
                mov     r7,a            ;data returned back through reg.B

                call    DELAY_L

                mov     a,#02h
                mov     dptr,#WF80
                movx    @dptr,a         ;reset IBFA as an ACK for Z-80

                call    DELAY_L
                ret
;*****************************************
WR_DISP_COM:    call    BUSY
                mov     dptr,#LCDCOM_ADR
        movx    @dptr,a
                ret
;*****************************************
WR_DISP_DATA:   call    BUSY
                mov     dptr,#LCDATA_ADR
        movx    @dptr,a
                ret
;*****************************************
CURSOR_TO:      call    BUSY
                mov     dptr,#LCDCOM_ADR
                movx    @dptr,a
                ret
```

```
;*****************************************
BUSY:       push    Acc
            mov     dptr,#LCDREAD_ADR
LOP:        movx    a,@dptr
            jb      Acc.7,LOP
            pop     Acc
            ret
;*****************************************

CLEAR:      mov     a,#01h          ; Clear Display
            call    WR_DISP_COM
            mov     a,#06h          ; I/D = 1 , S = 0
            call    WR_DISP_COM ; Entry Mode Set
            mov     a,#0Fh          ; D = 1 , C = 1 , B = 1
            call    WR_DISP_COM ; Display ON/OFF Control
            ret
;*****************************************

WRITE_LCD:  nop
AS2:        clr     a
            movc    a,@a+dptr
            cjne    a,#0,SEND_CHR
            mov     r2,#0
            ret
;*****************************************

SEND_CHR:   push    dph
            push    dpl
            inc     r2
            call    WR_DISP_DATA
            cjne    r2,#DIG_NUM,YYY

            mov     a,#0A8h         ; Set DD RAM Adress to 40
            call    CURSOR_TO

YYY:        pop     dpl
            pop     dph
            inc     dptr
            jmp     AS2

;*****************************************

SCAN_KEY:   mov     p1,#11110000b

KDWN:       call    KREAD
            jnz     KDWN
            mov     r4,#13
K_LP2:      mov     r5,#0FFh
            djnz    r5,$
            djnz    r4,K_LP2
KUP:        call    KREAD
            jz      KUP
            mov     r4,#13
K_LP3:      mov     r5,#0FFh
            djnz    r5,$
```

```
                djnz    r4,K_LP3
                mov     p1,#11111110b   ; COL0 = 0
                call    KREAD
                jnz     COL0
                mov     p1,#11111101b   ; COL1 = 0
                call    KREAD
                jnz     COL1
                mov     p1,#11111011b   ; COL2 = 0
                call    KREAD
                jnz     COL2
                mov     p1,#11110111b   ; COL3 = 0
                call    KREAD
                mov     dptr,#COLUM3
                jmp     LKUP

COL0:           mov     dptr,#COLUM0
                jmp     LKUP
COL1:           mov     dptr,#COLUM1
                jmp     LKUP
COL2:           mov     dptr,#COLUM2
LKUP:           inc     dptr
                jb      b.7,YUKLE
                inc     dptr
                jb      b.6,YUKLE
                inc     dptr
                jb      b.5,YUKLE
                inc     dptr
YUKLE:                  clr     a
                movc    a,@a+dptr
                ret     ; of TUS_TARA

KREAD:                  mov     a,p1
                anl     a,#0F0h         ; P1.7 ... P1.4 = DATA
                orl     a,#0Fh          ; P1.3 ... P1.0 = 1
                cpl     a
                mov     b,a
                ret
;*****************************************
DELAY_L:        mov     r4,#2           ; 1mS  if r4=100  then 55 msn
WAIT:           mov     r5,#0FFh
                djnz    r5,$
                djnz    r4,WAIT
                ret
;*****************************************
DELAY_TC:       push    04h
                push    05h
                mov     r4,#200         ; 110 msn
WAIT_TC:        mov     r5,#0FFh
                djnz    r5,$
                djnz    r4,WAIT_TC
                pop     05h
                pop     04h
                ret
;*****************************************
DELAY:                  push    04h
```

```
                push    05h
                push    06h
                mov     r4,#3       ;1sn
D1:             mov     r6,#0ffh
D2:             mov     r5,#0ffh
                djnz    r5,$
                djnz    r6,D2
                djnz    r4,D1
                pop     06h
                pop     05h
                pop     04h
                ret
;*****************************************
M_DELAY:;       mov     r4,#1
D11:            mov     r6,#0ffh
D21:            mov     r5,#0ffh
                djnz    r5,$
                djnz    r6,D21
        ;       djnz    r4,D11
                ret
;*****************************************
Load_16:
    ;Load the lower 16 bits of the OP registers with the value supplied
    MOV   OP_3,#0
    MOV   OP_2,#0
    MOV   OP_1,?Load_16?byte
    MOV   OP_0,?Load_16?byte + 1
    RET
;*****************************************
Low_16:
    ;Return the lower 16 bits of the OP registers
    MOV   R6,OP_1
    MOV   R7,OP_0
    RET
;*****************************************
Mul_16:
    ;Multiply the 32 bit OP with the 16 value supplied
    MOV   TMP_3,#0      ;clear out upper 16 bits
    MOV   TMP_2,#0
    ;Generate the lowest byte of the result
    MOV   B,OP_0
    MOV   A,?Mul_16?byte+1
    MUL   AB
    MOV   TMP_0,A       ;low-order result
    MOV   TMP_1,B       ;high_order result
    ;Now generate the next higher order byte
    MOV   B,OP_1
    MOV   A,?Mul_16?byte+1
    MUL   AB
    ADD   A,TMP_1       ;low-order result
    MOV   TMP_1,A       ; save
    MOV   A,B         ; get high-order result
    ADDC  A,TMP_2       ; include carry from previous operation
    MOV   TMP_2,A       ; save
    JNC   Mul_loop1
```

```
    INC   TMP_3        ; propagate carry into TMP_3
Mul_loop1:
    MOV   B,OP_0
    MOV   A,?Mul_16?byte
    MUL   AB
    ADD   A,TMP_1      ;low-order result
    MOV   TMP_1,A      ; save
    MOV   A,B          ; get high-order result
    ADDC  A,TMP_2      ; include carry from previous operation
    MOV   TMP_2,A      ; save
    JNC   Mul_loop2
    INC   TMP_3        ; propagate carry into TMP_3
Mul_loop2:
    ; Now start working on the 3rd byte
    MOV   B,OP_2
    MOV   A,?Mul_16?byte+1
    MUL   AB
    ADD   A,TMP_2      ;low-order result
    MOV   TMP_2,A      ; save
    MOV   A,B          ; get high-order result
    ADDC  A,TMP_3      ; include carry from previous operation
    MOV   TMP_3,A      ; save
    ; Now the other half
    MOV   B,OP_1
    MOV   A,?Mul_16?byte
    MUL   AB
    ADD   A,TMP_2      ;low-order result
    MOV   TMP_2,A      ; save
    MOV   A,B          ; get high-order result
    ADDC  A,TMP_3      ; include carry from previous operation
    MOV   TMP_3,A      ; save
    ; Now finish off the highest order byte
    MOV   B,OP_3
    MOV   A,?Mul_16?byte+1
    MUL   AB
    ADD   A,TMP_3      ;low-order result
    MOV   TMP_3,A      ; save
    ; Forget about the high-order result, this is only 32 bit math!
    MOV   B,OP_2
    MOV   A,?Mul_16?byte
    MUL   AB
    ADD   A,TMP_3      ;low-order result
    MOV   TMP_3,A      ; save
    ; Now we are all done, move the TMP values back into OP
    MOV   OP_0,TMP_0
    MOV   OP_1,TMP_1
    MOV   OP_2,TMP_2
    MOV   OP_3,TMP_3
    RET
;*****************************************
Sub_16:
    ;Subtract the 16 bits supplied by the caller from the OP registers
    CLR   C
    MOV   A,OP_0
    SUBB  A,?Sub_16?byte + 1    ;low byte first
```

```
    MOV   OP_0,A
    MOV   A,OP_1
    SUBB  A,?Sub_16?byte      ;high byte + carry
    MOV   OP_1,A
    MOV   A,OP_2
    SUBB  A,#0                ;propagate carry only
    MOV   OP_2,A
    MOV   A,OP_3
    SUBB  A,#0                ;propagate carry only
    MOV   OP_3,A
    RET
;*****************************************
 Div_16:
    ;This divides the 32 bit OP register by the value supplied
    MOV   R7,#0
    MOV   R6,#0       ;zero out partial remainder
    MOV   TMP_0,#0
    MOV   TMP_1,#0
    MOV   TMP_2,#0
    MOV   TMP_3,#0
    MOV   R1,?Div_16?byte ;load divisor
    MOV   R0,?Div_16?byte+1
    MOV   R5,#32      ;loop count

    ;This begins the loop

Div_loop:
    CALL  Shift_D     ;shift the dividend and return MSB in C
    MOV   A,R6        ;shift carry into LSB of partial remainder
    RLC   A
    MOV   R6,A
    MOV   A,R7
    RLC   A
    MOV   R7,A
    ;now test to see if R7:R6 >= R1:R0

    CLR   C
    MOV   A,R7        ;subtract R1 from R7 to see if R1 < R7
    SUBB  A,R1        ; A = R7 - R1, carry set if R7 < R1
    JC    Cant_sub
    ;at this point R7>R1 or R7=R1

    JNZ   Can_sub     ;jump if R7>R1
    ;if R7 = R1, test for R6>=R0

    CLR   C
    MOV   A,R6
    SUBB  A,R0        ; A = R6 - R0, carry set if R6 < R0
    JC    Cant_sub

Can_sub:
    ;subtract the divisor from the partial remainder
    CLR   C
    MOV   A,R6
    SUBB  A,R0        ; A = R6 - R0
```

```
        MOV     R6,A
        MOV     A,R7
        SUBB    A,R1            ; A = R7 - R1 - Borrow
        MOV     R7,A
        SETB    C               ; shift a 1 into the quotient
        JMP     Quot

Cant_sub:
        ;shift a 0 into the quotient
        CLR     C

Quot:
        ;shift the carry bit into the quotient
        CALL    Shift_Q
        ; Test for competion

        DJNZ    R5,Div_loop

        ; Now we are all done, move the TMP values back into OP
        MOV     OP_0,TMP_0
        MOV     OP_1,TMP_1
        MOV     OP_2,TMP_2
        MOV     OP_3,TMP_3
        RET
;*****************************************

Shift_D:
        ;shift the dividend one bit to the left and return the MSB in C
        CLR     C
        MOV     A,OP_0
        RLC     A
        MOV     OP_0,A
        MOV     A,OP_1
        RLC     A
        MOV     OP_1,A
        MOV     A,OP_2
        RLC     A
        MOV     OP_2,A
        MOV     A,OP_3
        RLC     A
        MOV     OP_3,A
        RET
;*****************************************

Shift_Q:
        ;shift the quotent one bit to the left and shift the C into LSB
        MOV     A,TMP_0
        RLC     A
        MOV     TMP_0,A
        MOV     A,TMP_1
        RLC     A
        MOV     TMP_1,A
        MOV     A,TMP_2
        RLC     A
        MOV     TMP_2,A
```

```
    MOV    A,TMP_3
    RLC    A
    MOV    TMP_3,A
    RET

;*****************************************

COLUM0:             DB      0 ,'*','7','4','1'
COLUM1:             DB      0 ,'0','8','5','2'
COLUM2:             DB      0 ,'E','9','6','3'
COLUM3:             DB      0 ,'D','C','B','A'

;*****************************************
;************** LCD Initialize ******************
LCD_INIT:
                    mov     r3,#0
                    call    DELAY_L
                    call    DELAY_L
                    call    DELAY_L
                    mov     dptr,#LCDCOM_ADR
                    mov     a,#3Fh
          movx      @dptr,a
                    call    DELAY_L                 ; 55 ms
                    movx    @dptr,a
                    call    DELAY_L                 ; 55 ms
                    movx    @dptr,a
                    call    DELAY_L                 ; 55 ms
                    mov     a,#38h          ; DL = 1 , N = 1 , F = 0
                    movx    @dptr,a         ; Function Set
                    mov     a,#08h          ; D = 0 , C = 0 , B = 0
                    call    WR_DISP_COM; Display ON/OFF Control
                    mov     a,#01h          ; Clear Display
                    call    WR_DISP_COM
                    mov     a,#06h          ; I/D = 1 , S = 0
                    call    WR_DISP_COM; Entry Mode Set
                    mov     a,#0Fh          ; D = 1 , C = 1 , B = 1
                    call    WR_DISP_COM; Display ON/OFF Control
                    ret
;*********************************************
                    DB      '1234567890123456'

ANA_MENU_MSG:       DB      ' A-75,B-76,C-78 '
                    DB      'D-ADC,E-CTC,0-SW',0

;---------------------------------------

COM_ERR_MSG:    DB      'dedected COM_ERR'
                    DB      'in int.face unit',0
;---------------------------------------
                    DB      '1234567890123456'

UNVALID_KEY:    DB      ' have pressed '
                    DB      ' unvalid key   ',0
;---------------------------------------
```

```
RAM2K_TEST:    DB     ' 2K Ram Test ? '
               DB     'press * for quit',0
;-----------------------------------------

RAM_TEST:      DB     'testing RAM.... '
               DB     'wait for result ',0
;-----------------------------------------

RAM_TEST_OPTION: DB    ' robust ? ( 1 ) '
               DB     ' quick  ? ( 2 ) ',0
;-----------------------------------------

ERR_KOD_MSG:         DB      ' Wrong Code... '
               DB     'Press any key..',0
;-----------------------------------------
RAM_FAULT_MSG: DB     'Ram test FAIL...'
               DB     ' Press any key  ',0
;-----------------------------------------
RAM_OK_MSG:    DB     'Ram test O.K ...'
               DB     ' Press any key  ',0
;-----------------------------------------

IC75_TEST_MSG: DB     'testing IC75... '
               DB     'wait for result ',0
;-----------------------------------------

IC75_OK_MSG:   DB     'IC_75 PASS..... '
               DB     'press any key   ',0
;-----------------------------------------

IC75_FAIL_MSG: DB     ' IC75 FAIL !!! '
               DB     'press any key   ',0
;-----------------------------------------
IC76_TEST_MSG: DB     'testing IC76... '
               DB     'wait....       ',0
;-----------------------------------------

IC76_FAIL_MSG: DB     ' IC76 FAIL !!! '
               DB     'press any key   ',0
;-----------------------------------------

IC76_OK_MSG:   DB     'IC_76 PASS..... '
               DB     'press any key   ',0
;-----------------------------------------

UNDEFINED_MSG: DB     'Undefined code!!'
               DB     'press any key   ',0
;-----------------------------------------

I59_MSG:       DB     ' I59  FAIL !!! '
               DB     ' I58  PASS *** ',0
;-----------------------------------------

I58_MSG:       DB     ' I58  FAIL !!! '
               DB     ' I59  PASS *** ',0
```

```
;-----------------------------------------

ALL_MSG:        DB      ' I59  FAIL !!! '
                DB      ' I58  FAIL !!! ',0
;-----------------------------------------
IC78_TEST_MSG:  DB      'testing IC78... '
                DB      'Look at LEDs... ',0
;-----------------------------------------
ADC_TEST_MSG:   DB      'Select channel: '
                DB      '   0...7      ',0
;-----------------------------------------
INPUT7_MSG:     DB      'read       mV '
                DB      'on pin5 of ADC  ',0
;-----------------------------------------
INPUT6_MSG:     DB      'read       mV '
                DB      'on pin4 of ADC  ',0
;-----------------------------------------
INPUT5_MSG:     DB      'read       mV '
                DB      'on pin3 of ADC  ',0
;-----------------------------------------
INPUT4_MSG:     DB      'read       mV '
                DB      'on pin2 of ADC  ',0
;-----------------------------------------
INPUT3_MSG:     DB      'read       mV '
                DB      'on pin1 of ADC  ',0
;-----------------------------------------
INPUT2_MSG:     DB      'read       mV '
                DB      'on pin28 of ADC ',0
;-----------------------------------------
INPUT1_MSG:     DB      'read       mV '
                DB      'on pin27 of ADC ',0
;-----------------------------------------
INPUT0_MSG:     DB      'read       mV '
                DB      'on pin26 of ADC ',0
;-----------------------------------------
SW_MSG:                 DB      '               '
                DB      'press * to quit ',0
;-----------------------------------------
CTC_OPT_MSG:            DB      '  CTC TEST     '
                DB      'OPT_A  ,  OPT_B ',0
;-----------------------------------------
OPT_B_MSG:      DB      'Time cons:      '
                DB      'DAC out:      V',0
;-----------------------------------------
OPT_B0_MSG:     DB      '1:inc , 4:dec   '
                DB      '* : esc look E10',0
;-----------------------------------------
CTC0_MSG:       DB      '  CTC TEST     '
                DB      'Enter time cons.',0
;-----------------------------------------
CTC1_MSG:       DB      'Time Cons:      '
                DB      'then press E    ',0
;-----------------------------------------
CTC_RD_MSG:     DB      'read      V E10'
                DB      '* esc , 0 go on ',0
```

```
;-----------------------------------------
LS239_MSG:      DB      'Frequency to run'
                DB      '1-39kHz 2-156kHz',0
;-----------------------------------------
VFC_MSG:        DB      'CTC TEST : A   '
                DB      'VFC TEST : B   ',0
;-----------------------------------------
ESC_MSG:        DB      'PRESS ANY KEY  '
                DB      'TO ESC FROM VFC ',0
;-----------------------------------------
END
```

ÖZGEÇMİŞ

Kenan Engin 1969 yılında İzmit' te doğdu.İlk ve orta öğrenimini İzmit' te tamamladıktan sonra 1987 yılında Yıldız Teknik Üniversitesi Elektronik ve Haberleşme bölümünde lisans eğitimine başladı.1992 yılında mezun oldu ve aynı yıl Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Ana bilim dalında yüksek lisans eğitimine başladı. Şu an Yüksek lisans çalışmalarını tamamlamak üzere olup özel bir şirkette servis mühendisi olarak çalışmaktadır.