

YILDIZ TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

Mikrodenetleyici Destekli Klavye Kartı

Bülent Fidan

Yüksek Lisans Tezi

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

Kot	R 152 : 157
Alındığı Yer	: Fen Bil Ens.
Tarih	: 10.4.95
Fatura	:
Fiyatı	: 186.000.-
Ayniyat No	: 1-6
Kayıt No	: 59986
UDC	:
Ek	:

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

I. İÇ KAPAK

II. İÇİNDEKİLER

III. ÖZET

IV. SUMMARY

GİRİŞ

1. SİSTEMİN ÖZELLİKLERİ VE KULLANILAN TEKNİKLER

1.1. MİKRODENETLEYİCİ

1.1.1 MİKRODENETLEYİCİNİN TEKNİKTEKİ YERİ

1.2. PAL

MİKRODENETLEYİCİ DESTEKLİ KLAVYE KARTI

1.2.1 PAL NEDİR ?

- OPARATRİS CİHAZI -

1.2.2 PAL TEKNOLOJİSİNİN GELİŞİMİ

1.3. RS 232 C SERİ HABERLEŞME

1.3.1 RS 232 C STANDARTI VE TEKNİK

1.3.2 SERİ HABERLEŞME NEDİR KULLANILIR ?

1.4. LCD DISPLAY

1.4.1 LCD DISPLAY'İN KULLANIMI

1.5. AYARLANIRLI MOD

1.5.1 SNPS DEVRELERİ VE İNSTRÜMLERİ

1.5.2 TEMEL SNPS TİPLERİ

2. DEVRENİN İNCELENMESİ

2.1. SAAT DEVRESİNİN OLUŞTURULMASI

2.2. KONTROL LOJİK DEVRESİ

2.3. TUSLARIN TABAN DEVRESİ

2.4. LEDLERİN YAKILMASI VE YAKILMA

2.5. SERİ HABERLEŞME

2.6. DEVRENİN BESLEME KARTI

2.7. DEVREYE TOPLU GİRİŞ BAKIŞ

3. MONİTOR PROGRAMININ İNCELENMESİ -YAZILIM-

3.1. SİSTEMİN İLK ŞARTLARININ VERİLMESİ

3.2. ESAS DÖNGÜ -MAIN LOOP-

3.3. SANTRAL İLE HABERLEŞME

3.4. TUSLARIN BELİRLENMESİ

3.4.1 AYRNE TUSLARI

3.4.2 HEX. KLAVYE TUSLARI

3.4.3 ÖZEL FONKSİYON TUSLARI

İSTANBUL 1990

İ Ç İ N D E K İ L E R

I. İÇ KAPAK	41
II. İÇİNDEKİLER	
III. ÖZET	
IV. SUMMARY	48
GİRİŞ	1
1. SİSTEMİN ÖZELLİKLERİ VE KULLANILAN TEKNİKLER	2
1.1 MİKRODENETLEYİCİ	
1.1.1 MİKRODENETLEYİCİNİN TEKNİKTEKİ ROLÜ	99
1.2 PAL	
1.2.1 PAL NEDİR ?	
1.2.2 PAL TEKNOLOJİSİNİN GETİRDİKLERİ	
1.3 RS 232 C SERİ HABERLEŞME	
1.3.1 RS 232 C STANDARDI VE TEKNİĞİ	
1.3.2 SERİ HABERLEŞME NEDEN KULLANILDI ?	
1.4 LCD DISPLAY	
1.4.1 LCD DISPLAY'İN KULLANILMASI	
1.5. ANAHTARLAMALI MOD GÜÇ KAYNAĞI	
1.5.1 SMPS DEVRELERİ VE ÇALIŞMA PRENSİPLERİ	
1.5.2 TEMEL SMPS TİPLERİ	110
2. DEVRENİN İNCELENMESİ -DONANIM-	171
2.1 SAAT DEVRESİNİN OLUŞTURULMASI	
2.2 KONTROL LOJİK DEVRESİ	
2.3 TUŞLARIN TARANMASI VE KONUMU	
2.4 LEDLERİN TARANMASI VE SÜRÜLMESİ	
2.5 SERİ HABERLEŞME	
2.6 DEVRENİN BESLEME KATI	
2.7 DEVREYE TOPLU BİR BAKIŞ	
3. MONİTÖR PROGRAMININ İNCELENMESİ -YAZILIM-	27
3.1 SİSTEMİN İLK ŞARTLARININ VERİLMESİ	
3.2 ESAS DÖNGÜ -MAIN LOOP-	
3.3 SANTRAL İLE HABERLEŞME	
3.4 TUŞLARIN BELİRLENMESİ	
3.4.1 ABONE TUŞLARI	
3.4.2 HEX. KLAVYE TUŞLARI	
3.4.3 ÖZEL FONKSİYON TUŞLARI	

4. DEVRE ŞEMALARI

41

4.1 ANA KARTIN AÇIK ŞEMASI

4.2 ANA KARTIN BASKILI DEVRESİ

4.3 KLAVYE KARTININ BASKILI DEVRESİ

5. MONİTÖR PROGRAMI

46

6. TARTIŞMA VE SONUÇ

97

6.1. DONANIM DEĞERLENDİRMESİ

6.2. YAZILIM DEĞERLENDİRMESİ

EK- KATALOG BİLGİLERİ

99

A. 8032 MICROCONTROLLER

A.1 FİZİKSEL YAPISI

A.2 ELEKTRONİK YAPISI

B. PAL - 16 L 8 -

B.1 FİZİKSEL VE ELEKTRONİK YAPISI

C. LCD

C.1 FİZİKSEL VE ELEKTRONİK YAPISI

C.2 LCD' NİN ŞARTLANDIRILMASI

D. SMPS

D.1 LM 2575 ENTEGRESİNİN TEKNİK ÖZELLİKLERİ

KAYNAKLAR

110

ÖZGEÇMİŞ

111

ÖZET

Günümüz piyasasında, elektronik kontrol sistemleri, akıllı sistemler olma yolunda bir hayli mesafe katettiler. Yeni teknolojiler üretmenin pek mümkün olamadığı Türkiye konumundaki ülkeler açısından, mevcut teknolojileri kontrol sistemleri üzerinde kullanıp üst düzeyde verim elde etmek oldukça tercih edilir bir yoldur.

Mikrodenetleyici türü elemanların, çevre birimlerinin de bünyesinde toplanıp TUM DEVRE halinde kullanıcıya sunulmuş olması çekiciliklerini daha da arttırmaktadır.

Bugün için, Türkiye'nin haberleşme ile ilgili piyasasında pek çok üretici tarafından kullanılan 8 bitlik bir mikrodenetleyici olan 8032'yi projemde kullanmayı uygun buldum. Seri ve paralel haberleşme için gerekli çevre birimleri ve zamanlayıcı/sayıcı'yı bünyesinde toplamıştır. Daha fazla özellikler içeren mikrodenetleyiciler de vardır, fakat 8031 hem çok yaygın, hemde kolay ve ucuz temin edilebilen bir entegredir.

Proje, 8 harici 32 dahili hat taşıyan digital bir santrale bağlı olarak çalışması düşünülen OPARATRİS devresidir. Devre, bağlı olduğu santralle yansımali olarak çalışmaktadır. Yani sadece çevre etkileri belirleyip, ne yapacağını santraldan gelen bilgiyle anlayabilmektedir. Bu haliyle belki tam bir akıllı devre diyemeyiz, fakat bu eksiklik mikrodenetleyiciden veya yapımdan kaynaklanmamaktadır. Böyle bir cihazın santralle tam bir uyum içinde çalışması en başından belirlendiği için, dolayısıyla santrale tam bir bağımlılık söz konusudur.

Böyle olmakla birlikte, proje esnasında kullanılan herşey, öncelikle bir araştırma açısından, daha sonra ise pratik bir gerçekleştirme açısından oldukça doyurucudur. Değişik alanlarda kullanılan pek çok tekniği üzerinde toplamış ve bu teknikler hakkında özlü araştırma ve açıklamalar yapılmıştır. Sonuç olarak, burada anlatılan bir oparatris cihazının tasarımı olduğundan söz konusu bu tekniklerin ayrıntılarına fazlasıyla girilememiş, bunlar ayrıca okuyucuya bırakılmıştır.

SUMMARY

The electronic control systems which are seen on today's market took some distance on the way to become intelligent systems. In such countries as Turkey, where it is not possible to produce new technologies, it is common to use existing technologies on control systems.

The integration of peripheral devices into the same chip as the microprocessor, resulting in the availability of single chip microcontroller devices increases their attraction.

I found it suitable to use the 8032, 8-bit microcontroller in my project, which is also used by many manufacturers of telecommunication devices in Turkey. The peripherals for parallel and serial communication and also the timer/counter is integrated on this single chip device. There are microcontrollers with more capabilities, too. But 8032 is very common, cheap and everywhere available.

The project is consist of a digital PBX with 8 trunks and 32 extensions. The circuit is working in a echo mode. In other words it executes instructions send by the PBX. So the device seems not to be intelligent. But this is not a result of technical restrictions. The only reason of such a operation is to guarantee full compatibility with the PBX. So it can be treated as a 'slave device'.

Although, the device is not very intelligent, the several block of it is very interesting. In both of theoretical and practical means. The work is containing many techniques used in several areas. These techniques are declared only in topics. Details are not contained.

1.1. SİSTEMİN ÖZELLİKLERİ GİRİŞ VE KULLANILAN TEKNİKLER

Pratik olarak gerçekleştirdiğim bu projeyi önce çok genel amaçlı olarak yapmayı düşünüyordum. Daha sonra özel bir şirketten sağladığım finansman ile belirli bir nitelik kazanmıştır.

Proje, bir telefon santraline bağlanabilecek ve 32 dahili ile 8 harici hatta ait haberleşme bağlantısını kontrol edip, özel fonksiyon tuşları ile pek çok değişik hizmeti gerçekleştirecek bir operatris cihazıdır.

INTEL MCS-51 ailesinden 8032 mikrodenetleyici entegresi üzerine kurulmuş olan bu proje, 8032'nin yaklaşık tüm fonksiyonlarının - yazılım & donanım - kullanıldığı ilginç bir uygulamadır. Bir bitirme projesi olarak ta düşünüldüğünden pek çok değişik teknolojinin bütünleştiği bir sistem meydana getirilmiştir. Bunlardan en belirginleri; lojik ifadeleri bünyesinde toplayıp tüm devre halinde sunabilen PAL tekniği uygulaması, karakter jeneratörü üzerinde tanımlı NOKTA MATRİS LCD uygulaması, seri haberleşme için özel bir RS 232 entegresi kullanılması, besleme kaynağı olarak ANAHTARLAMALI MOD GÜÇ KAYNAĞI kullanılması şeklinde sıralanabilir.

Projede kullanmış olduğum bu teknikleri önce genel hatları ile açıklayıp daha sonra devrenin incelenmesine geçeceğim.

Bu projenin gerçekleşmesinde teknik ve yöntem hususundaki değerli katkılarından dolayı sayın hocam Prof. ŞEFİK SARIKAYALAR'A, ayrıca her konudaki yardımlarıyla bana rahat bir çalışma imkanı sağlayan SENTÜRK TELEKOM. LTD. ŞTİ. sahiplerine ve AR-GE bölümü çalışanlarına teşekkür ederim.

Bu projede kullanılan, esasen ayrık lojik kapılardan oluşmuş devrelere verilen adur. Bu devrelerin en büyük özelliği belirli bir algoritma programlanabilir olmasıdır. Maket programlama da yaygın olarak kullanılmaktadır. Bunların içyapıları AND kapıları, OR kapıları ve

1. SİSTEMİN ÖZELLİKLERİ VE KULLANILAN TEKNİKLER

1.1. MİKRODENETLEYİCİLER

1.1.1 MİKRODENETLEYİCİLERİN TEKNİKTEKİ ROLÜ

Son on yılda mikroişlemcilerin kapasitelerinde heyecan verici değişimler gözlenmiştir. Üreticiler daha güçlü CPU (Central processing unit) ihtiyacına cevap vermek için 16 , 32 bitlik işlemciler üretmektedirler. Bu işlemciler PC'lerin belkemiğini oluştururlar ve adeta mühendisliğin devrim aracı olmuşlardır.

Mikroişlemcilerin, tüm bu kabiliyetlerine rağmen çevre birimleri ile olan yoğun ilişkisi, hız ve yerleşim açısından bazı sorunları da beraberinde getiriyordu. Bu sorunları gidermek ve yaygın bir kullanım kazanmak açısından, mikroişlemcilerin çevre birimleri ile birlikte tüm devre olarak düşünüldüğü mikrodeneleyiciler (Microcontroller) tasarlanmıştır. (Bundan sonra yazımızda mikrodeneleyici kelimesini kullanacağız).

Mikrodeneleyicilerin çıkmasıyla önceden ayrı olarak kullanılan kesme kontrol entegreleri, programlanabilir zamanlayıcı entegreleri, rom, ram entegreleri tek bir entegre içinde birleştirilmiş oluyordu.

Böyle bir entegre geliştirmekle kazancımız yalnızca entegrenin görünen yüzüyle değil, aynı zamanda entegrenin gördüğü alanın artmış olmasıyla da ortaya çıkıyordu. Böylece tek bir entegre ile daha geniş bir alanı kontrol edip, erişme imkanına sahip olduk.

1.2. PAL

1.2.1. PAL NEDİR ?

Programlanabilir lojik devreler, bir devrenin en son lojik konfigürasyonunda kullanılan, esasen ayrı lojik kapılardan müteşekkil devrelere verilen addır. Bu devrelerin en büyük özelliği belirli bir alanının programlanabilir olmasıdır. Maskeli programlama da yaygın olarak kullanılmaktadır.

Bunların içyapıları AND kapıları, OR kapıları ve

registerlerin anahtarlanması ile oluşturulmuştur. Bu devreler, ardışıl devrelerde (sequential) olduğu gibi karışık tasarımların da (combinational) kullanılmasına müsaade eder.

Programlama gösterimi, AND - OR lojigi şeklindedir. Bu gösterim (SUM-OF-PRODUCTS) SOP ailesindeki gösterim şeklidir. Bir açıklama yapmada fayda var ki; programlanabilir lojik, yazılım içermeksizin tamamlanamaz. Programlanabilir lojik devresi ile yardımcı sistem tasarımında, bu yapıların en önemli parçası yazılım ve otomasyon tasarım aletleridir.

İlk programlanabilir lojik devre entegresi 1960 yılı başlarında üretilmiştir ve diyot matris yapısındadır. Satırlar ve sütunlardan oluşan bu yapıda köşe noktaları arası bağlantı, diyot ve alüminyum sigortalardan ibaretti. Bu sigortalar seçilip yakılarak bazı köşeler diğer bağlantılardan ayrılıyorlardı. Sonuçta -diyot OR- lojigi şeklinde bir yapı ortaya çıkıyordu.

Bunun ardından entegre devre tasarımcıları, giriş kot çözücülerini ve çıkış sürücülerini temel matris yapısı içinde toplayıp PROM yapısındaki hafıza elemanlarını keşfetmişlerdir.

Daha sonra ortaya çıkarılan -FPLA- denilen (Field Programmable Lojik Array) yapılar, prom yapısından gelen bazı sınırlamaları ortadan kaldırabilmiştir. Çünkü prom yapısı, kullanılmak istenen pekçok lojik fonksiyonun tanımını içermiyordu. FPLA tekniğinin de bazı sorunları yok değildi. Girişlerin ve fonksiyonların sayısı özel bir durum taşıyordu. Tanımlanabilecek terim sayısı, 2^n (n, giriş ucu sayısı) ile sınırlıydı.

Bunun ardından geliştirilen PAL (Programmable Array Logic) tekniğiyle PROM'larda, FPLA'da görülen sınırlı giriş durumu ve FPLA'daki OR matris yapı kaldırılarak daha yetenekli bir yapı elde edilmiştir. Hemen belirtelim ki Pal yapılarda OR kapıları terimin son halinin belirlenmesinde yeni yollar oluştururlar. Sonuç itibarıyla bunlar da sınırlı

durumlarına rağmen standart olmaları ile avantaj sağlarlar.

1.2.2. PAL TEKNOLOJİSİNİN GETİRDİKLERİ

Programlanabilir lojik devreler kullanmak, sistem tasarımıda pek çok avantajı beraberinde getirir. Programlanabilir lojik devrelerin tüm uygulamalarındaki en genel faydaları aşağıda anlatılmıştır.

A- AZ YER KAPLAMA : Bu gün teknolojisinde programlanabilir lojik devreler 4 e 20 SSI ve MSI teknigiyle tüm devre olarak üretilmektedir. Pc kartın gerçek boyutu, sistem ve programlanabilir lojik devrelerin kart yüzeylerinin küçüklüğü en değerli ve limit özelliğidir.

B- HIZLI DEVRE TASARIMI : Sistem tasarımıda meydana gelen değişimler oldukça önemlidir. Sistem prototipleri mevcut tasarım geliştirme araçlarının yardımıyla kısa sürede oluşturulur. Standart tasarım aletlerinde, döküman ve elle tasarım ihtiyacı daha da azdır. İlk prototip oluşturulduktan sonra lojik değişiklikler ve düzeltmeler, kart üzerinde yeni bağlantılara ve çalışmalara gerek duymaksızın yapılabilir. En sonunda programlanabilir lojik kullanılarak, firma için rekabete yönelik bir avantaj elde edilir ve yeni ürün piyasaya sürülür.

C- TASARIM ESNEKLİĞİ : Sistem tasarımı genellikle , tekrarlı bir oluşumdur. Başlangıçta, fikirler, düşünceler ve o zamana kadar ki gelişmeler seri bir şekilde düzenlenip kaba bir tasarım elde edilir. Elde edilen prosesler içinde en beğenilene ve en zahmetsiz olanı baştan aşağı programlanabilir lojik olarak tanımlanır.

Programlanabilir lojik kullanılması ile, tasarımcı az bir malzemeye oluşturulmuş standardın içine hapis olunmamıştır, standart dışı lojik yapıları da rahatlıkla kullanabilir. Mühendise, şimdi sadece üretimi süren uygun bir PAL entegresi seçmek kalmıştır.

Tasarım esnekliği, programlanabilir lojik bünyesindeki lojik yapının değişimiyle, sistemin çalışma işaretleri-

nin tüm konumlarıyla elde edilebilir olmasıdır. Örneğin, tüm lojik değişiklikler prototip süresince yapılabilir, sistemin test edilmesi süresince yapılabilir, sistemin üretilmesi sırasında yapılabilir. Tüm bu değişimler boyunca sistem temel çalışmasını bozmamalıdır.

D- KASKAT LOJİKTE AZALTMA : Tasarımcı programlanabilir lojik kullanarak, kaskat bağlı lojik devreleri iki kat AND-OR yapısıyla baştan sona kadar tasarlayabilir. Bu azaltma tasarımı ve kullanılan malzemede hız ve/veya az güç harcama imkanı getirir.

E- HARCAMALARDA AZALTMA : Sistem üreticileri programlanabilir lojik kullanarak, değişik faktörlerle harcamada azaltma işini realize edebilirler. Bu faktörlerin başlıca türleri şunlardır ;

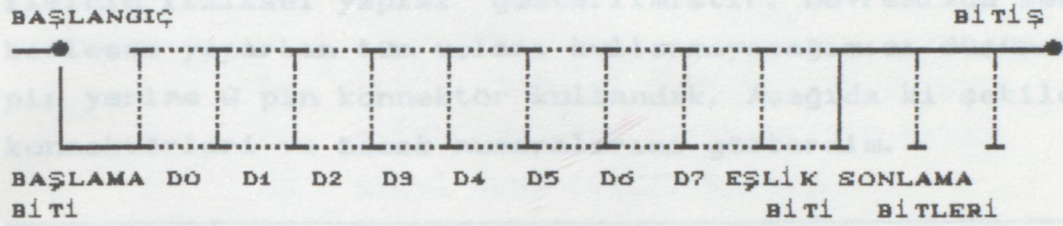
- Düşük parça ücreti
- Düşük tasarım ve geliştirme ücreti
- Düşük imalat ücreti
- Düşük servis ücreti... [7]

1.3. RS 232 C SERİ HABERLEŞME

1.3.1. RS 232 C STANDARDI VE TEKNİĞİ

RS 232 C bir seri haberleşme standardıdır. 1969 yılında (Elektronics Industry Association) EIA bu standardı çıkardı, şimdi bu standart yaygın bir şekilde mikrokomputer sistemlerinde ve benzeri büyük sistemlerde kullanılmaktadır. Bu standart, iki adet çift yönlü bilgi kanalı üzerinde bilginin seri olarak, senkron veya asenkron olarak gönderilmesi işini sağlar. Bu gönderim anlaşma sinyalleşmesi yapılarak veya yapılmadan gerçekleştirilebilir.

Anlaşma sinyalleşmesi bilgilerin alınıp verilme işlemi sırasında herhangi bir hataya sebep olmamak için yapılır. Gönderilen ve alınan bilgiler, tek tel üzerinden seri olarak gönderilir ve alınır. Hat uzarından seri olarak iletilen bilgi belirli bir paketleme düzeni içermektedir. Aşağıda bu paket düzeni şematik olarak gösterilmiştir.



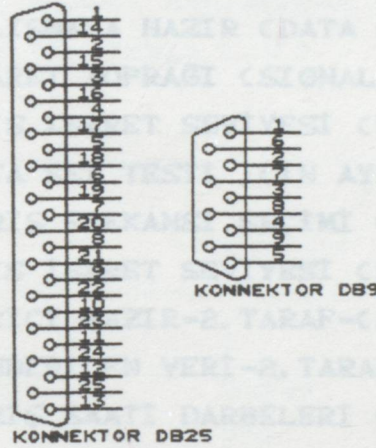
ŞEKİL : 1.9.4. - 1 SERİ BİLGİ PAKETİNİN YAPISI

Her bilgi gönderilmeden önce iki birim arasında ki senkronizasyonun sağlanması ve gönderilen bilginin ne olduğunun anlaşılabilmesi için hatta bir adet BAŞLAMA BİTİ gönderilir. Bu bit aktif lojik 0 dır ve karakterin iletiminin başladığını gösterir. Bundan sonra belirli sayıda ki bilgi bitleri yollanır. Hatta ilk yollanan bit , en az ağırlıklı bittir. Bilgi bitlerinden sonra hattaki iletimden dolayı oluşabilecek hataların anlaşılabilmesi için EŞLİK BİTİ (EVEN/ODD) gönderilir. En son olarak ta bilgi iletiminin bittiğini belirten SONLANDIRMA BİTİ yollanır. SONLANDIRMA BİTİ bir yada iki bit olabilmektedir. Verilerin kaç bit olduğu, EŞLİK BİT'in tipi, EŞLİK BİT olup olmayacağı, SONLANDIRMA BİTİNİN kaç adet olduğu sisteme göre değişir.

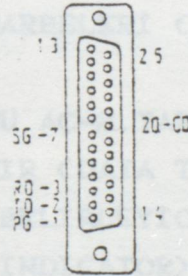
Seri haberleşme belirlenmiş bazı hızlarla yapılmaktadır. Hız birimi bps'tir. (baud per second) iletilen bit miktarıdır. Pratikte kullanılan bazı hızlar aşağıda gösterilmiştir.

150	bps	:	Saniyede	150	bit
300	"	:	"	300	bit
600	"	:	"	600	bit
1200	"	:	"	1200	bit
2400	"	:	"	2400	bit
4800	"	:	"	4800	bit
9600	"	:	"	9600	bit
19200	"	:	"	19200	bit

Şekil 1.3.1 - 2 de DB-25 tipi RS 232 C arabirim fişinin fiziksel yapısı gösterilmiştir. Devremizde seri haberleşme yaparken tüm uçları kullanmayacağımızı düşünerek 25 pin yerine 9 pin konektör kullandık. Aşağıda ki şekilde bu konektörleri ve bacak numaralarını gösterdim.



RS 232 C arabiriminin minyatür D-fiş. PG = koruma toprağı; TD = veri çıkışı; RD = veri girişi; SC = işletme toprağı.



ŞEKİL : 1.3.1. - 2 RS 232 C ARABİRİM FİŞİ

Tablo 1.3.1 -1 de DB-25 (D tip 25 bacak konektör) konektörünün bacak numaraları ve ne amaçla kullanıldıkları gösterildi.

Yön	Bacak no	Açıklama
	1	PG KORUMA TOPRAĞI (PROTECTIVE GROUND)
C	2	TD BİLGİ GÖNDERME (TRANSMITTED DATA)
G	3	RD BİLGİ ALMA (RECEIVED DATA)
C	4	RTS VERİCİ BÖLÜMÜNÜ AÇ (REQUEST TO SEND)
G	5	CTS VERİCİ HAZIR (CLEAR TO SEND)
G	6	DSR ÇALIŞMAYA HAZIR (DATA SET READY)
	7	SG İŞARET TOPRAĞI (SIGNAL GROUND)
G	8	DCD ALIŞ İŞARET SEVİYESİ (REC.LINE SIG.DET.)
	9/10	-- DATA SET TESTİ İÇİN AYRILMIŞTIR
C	11	VERİŞ FREKANSI SEÇİMİ (SEL.TRANSMIT FREQ.)
C	12	ALIŞ İŞARET SEVİYESİ (SEC.CARRIER DET.)
C	13	VERİCİ HAZIR-2.TARAF-(SEC.CLEAR TO SEND)
G	14	GÖNDERİLEN VERİ-2.TARAF- (SEC.TRANS.DATA)
G	15	DCE VERİŞ SAATİ DARBELERİ (CLOCK FROM MODEM)
C	16	ALINAN VERİ-2.TARAF- (SEC.REC.DATA)
C	17	RC ALIŞ SAATİ DARBELERİ (RECEIVE CLOCK)
	18	-- İŞARETSİZ
G	19	VERİCİ BÖLÜMÜ AÇ-2.TARAF-(SEC.REQ. TO SEND)
C	20	DTR TERMİNAL HAZIR (DATA TERMINAL READY)
G	21	ALIŞ KALİTE BELİRLEYİCİ (SIG.QUALITY DET.)
G	22	RI ÇAĞRI (RING INDICATOR)
C	23	DTE İLETİŞİM HIZI (DATA SIGNAL RATE SELECTOR)
C	24	DTE TEMİNALDEN VERİŞ SAATİ (TRANS.SIG.E.TIM.)
	25	-- İŞARETSİZ

TABLO : 1.9.1. - RS 232 C BİRİMİNİN BAĞLANTILARI

RS 232 C standardında iletim yapılmak üzere hatta gönderilen bilginin gerilim seviyesi bilinen TTL seviyesi değildir. İletim sırasında bilginin bozulmaması için gerilim

seviyeleri biraz yüksek tutulmuştur. RS 232 C ile haberleşmede 30m'ye kadar uzaklıklar için bir sorun çıkmamaktadır.

Elektriksel karakteristikleri iki konumda incelenmektedir. RS 232 C negatif lojik olarak çalışır ve -3v ile -25v arası lojik 1, 3v ile 25v arası lojik 0 diye adlandırılır. -3v ve 3v'tan itibaren 2v gürültü bölgesi ayrılmıştır. Hattaki gerilimin hem +, hem - olması nedeni ile hat üzerindeki akımın çift yönlü akması sağlanarak kapasitif etkilerin azaltılması başarılmıştır. Tablo 1.3.1 - 2 de bu voltaj seviyeleri gösterilmiştir.

DURUM	İŞARET VOLTAJI	
	-25v < Vi < -9v	9v < Vi < 25v
İkili Lojik	1	0
İşaret durumu	DARBE	BOŞLUK
Fonksiyon	YOK	VAR

TABLO : 1.3.1. - 2 RS 232 C VOLTAJ SEVİYELERİ

Açık devre halinde veya yükstüz durumda Vo gerilimi 25v'u geçemez. Bu aynı zamanda, RS 232 C hattında + 25 volttan büyük - 25 volttan küçük gerilim olamaz anlamına da gelir.

1.3.2. SERİ HABERLEŞME NEDEN KULLANILDI ?

Gerçekleşmiş olduğum OPORATRİS cihazı, telefon santrali ile sürekli haberleşme durumunda olacağından, herhangi bir yanlışa sebep olmamak için güvenilir bir haberleşme hattına ihtiyaç vardı. Gerçekleştirilecek bu hat, santrale azımsanmayacak bir mesafede olan cihazımızın güvenli bir şekilde haberleşmesini sağlayacak, pek çok kontrol uygulamasına açık olması için standart olacak ve kullanıcı için pratik olacaktır. Butun bunları göz önüne alarak RS 232 C standardında seri bir haberleşme hattının kurulmasını uygun bulduk.

Böylece 50 metrelere kadar varan güvenli bir mesafe gereğinde PC'lere bile bağlayabileceğimiz standart bir yapı ve pek çok kimse için pratik bir kullanım elde etmiş olduk. Buradan da anlaşılacağı gibi seri haberleşme yerine paralel haberleşmeyi seçseydik söz konusu mesafelere bilgi ulaştırmanın çok zor, hatta belkide imkansız olacaktı. Çünkü bilindiği gibi paralel haberleşme kısa mesafeler için oldukça ideal, fakat uzun mesafeler için yetersiz bir donanıma sahiptir. Haberleşme yapılan mesafe arttıkça bilgilerin gürültü kapıp bozulması söz konusudur.

Seri haberleşme yöntemleri içinde neden RS 232 C standardını seçtiğimize gelince; biraz önce de belirttiğim gibi, bu standart pek çok aletin yapısında var. Bu aletlerle bağlantı kurmak istediğimizde hiç bir zorlukla karşılaşmadan birkaç tel bağlayarak işimizi halledebilecektik. Butün bu olumlu görünen yanlarına rağmen RS 232 C bizim için çok önemli bir problemi de beraberinde getiriyordu.

RS 232 C için gerekli voltaj seviyeleri sağlanmıyordu. Kullandığım tüm elemanlar TTL uyumlu olup sadece +5 v bir gerilime ihtiyaç vardı. Gerekli gerilimleri, yeni bir kaynak oluşturarak sağlamak zorundaydım. Bu da yeni bir devre, yeni malzemeler ve ek bir maliyet demekti.

Bu durumdan kurtulmak için, ileride ayrıntılı olarak anlatacağım bir entegre kullandım. Bunun sayesinde hem seri haberleşme yapabiliyor, hem de tek voltaj ile (+5 volt) gerekli voltaj seviyelerini elde edebiliyordum. (-5 volt ile +5 volt RS 232 C haberleşmesi için yeterlidir.) [6]

1.4. LCD DISPLAY

1.4.1. LCD DISPLAY'İN KULLANILMASI

(Liquid Crystal Display) kullanımı mikroislemcili, mikrodenetleyicili sistem uygulamalarında oldukça yaygındır. Bunun başlıca sebepleri ; düşük güç tüketimi, display modeli tasarımı ki serbestlik ve ince kalıp oluşturabilme imkanı şeklinde sıralanabilir. Düşük güç tüketimi ve yüksek sıklık-

ta paket oluřturma bařlı bařına CMOS proses teknięi ve döz plastik paket kullanılması ile ilgilidir.

Devrede kullandığım LCD display HITACHI HD 44780 sürücüsü ile çalışan nokta matris displaydir. Türkçe karakterlerin de tanımlanabildięi bu LCD ekranda, sistemle ilgili olarak kullanıcının görmek istedięi bilgileri yazdırdım. Ayrıca saat ve tarih bilgisi sürekli olarak ekranda görünmektedir.

LCD display, tek satır veya çok satır olarak kullanılmaktadır. Aynı şekilde sütun sayıları da - ki buna karakter sayısı diyoruz - deęişik olabilmektedir.

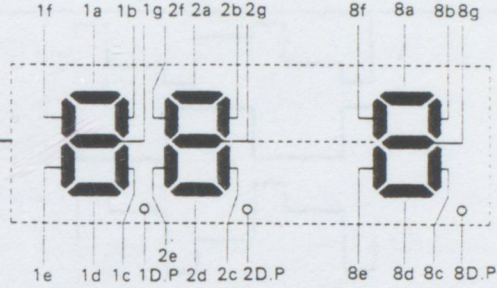
Displayler iki tip yöntemle sürölmektedirler. Bunlar statik ve çoęullamalı sürme yöntemleridir. Statik sürme yöntemi (STATIC DRIVING METHOD), iyi display kalitesi elde edilmesinde kullanılan en temel yöntemdir, bununla beraber pek çok çubuktan oluřan LC'yi display etmeye uygun deęildir, zira bir LCD devresi pek çok çubuk içermektedir. Statik sürme yönteminde, onlarca ve yüzlerce hertz taşıma frekansı ($1/t_f$) kullanılır.

Çoęullamalı sürme yöntemi ise (MULTIPLEX DRIVING METHOD); sürücü devre sayısının azalmasında, devre ile display hücrelerinin arabacıları sayısını ve pek çok display resim elemanının sürölmesindeki maliyeti etkiler. Dinamik sürücülerde, sürücü devre sayısı statik sürücülere göre azdır. Bununla beraber, çoęullama oranının artması sürüş voltaj toleransını daha da azaltmaktadır.

Şekilde Statik ve Çoęullamalı sürme yöntemleri sayısal displayde gösterilmiştir.

statik sürme
yöntemi

ortak uç

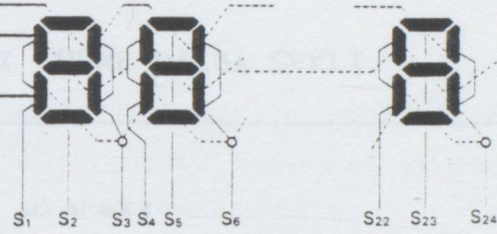


çoğullamalı
sürme yöntemi

o. uç 1

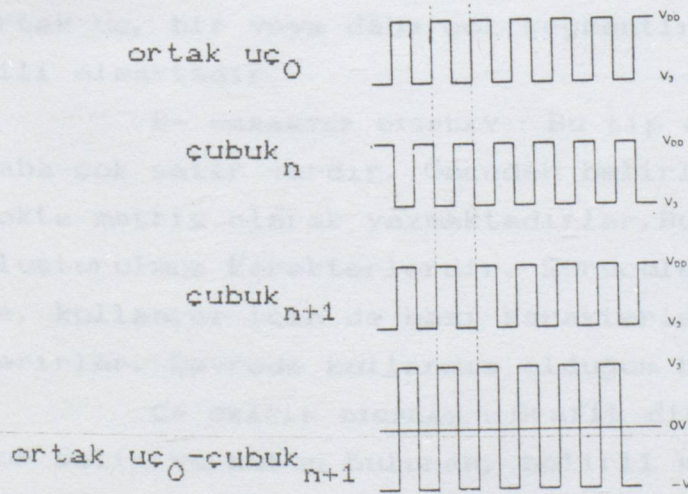
o. uç 2

o. uç 3

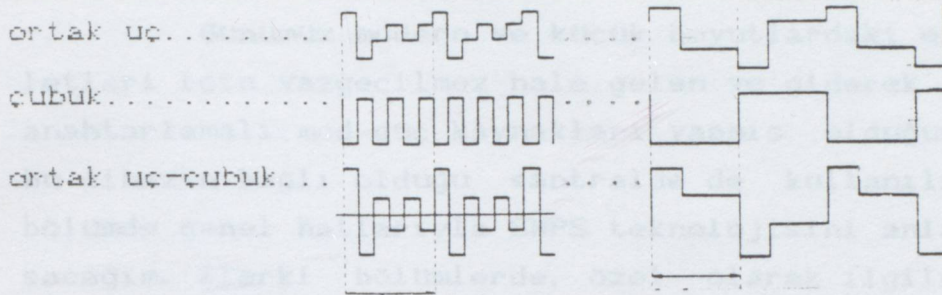


ŞEKİL : 1.4.1. - 1 STATİK VE ÇOĞULLAMALI SÜRME YÖNTEMLERİ

Statik sürme yönteminde sürüş dalga tipi tek olduğu halde Çoğullamalı sürme yönteminde iki tip dalga şekli vardır. Bu dalga şekilleri de aşağıda verilmiştir.



ŞEKİL : 1.4.1. - 2 STATİK SÜRME DALGA ŞEKLİ



ŞEKİL : 1.4.1. - 9 ÇOĞULLAMALI SÜRME DALGA ŞEKLİ

LCD displayler genel olarak;

A- ÇUBUK DISPLAY

B- KARAKTER DISPLAY

C- GRAFİK DISPLAY

diye üçe ayrılırlar. Şimdi kısaca bu üç tür displayi ayrı ayrı inceleyip her birinin özelliklerini ve kullanımını anlamaya çalışacağız.

A- ÇUBUK DISPLAY : Bu tip displayler çubuklarla oluşturulmuştur ve belirli karakterleri yazabilmektedirler. Ortak uç, bir veya daha çok segmentin aktif edilmesinde etkili olmaktadır.

B- KARAKTER DISPLAY : Bu tip displaylerde bir veya daha çok satır vardır. Önceden belirlenmiş bazı karakterleri nokta matris olarak yazmaktadırlar. Bunlar alfanümerik olarak oluşturulmuş karakterlerdir. Sürücülerinin kabiliyetine göre, kullanıcı için de bazı karakterlerin tanımlanmasına izin verirler. Devrede kullanmış olduğum bu tür bir displaydir.

C- GRAFİK DISPLAY : Grafik displayler üzerinde pek çok satır ve sütun bulunan, belirli karakterleri kullanarak bir kısım şekillerin çizilebilmesini sağlayan nokta matris olarak oluşturulmuş displaylerdir. Ayrıca karakter displaylerinde olduğu gibi alfanümerik olarak yazı da yazabilmektedirler. [3]

1.5. ANAHTARLAMALI MOD GÜÇ KAYNAĞI

1.5.1. SMPS DEVRELERİ VE ÇALIŞMA PRENSİPLERİ

Günümüz modern ve küçük boyutlardaki elektronik aletleri için vazgeçilmez hale gelen ve giderek önemi artan anahtarlama mod güç kaynakları yapmış olduğum cihazda ve bu cihazın bağlı olduğu santralde de kullanılmışlardır. Bu bölümde genel hatlarıyla SMPS teknolojisini anlatmaya çalışacağım. İlerki bölümlerde, özel olarak ilgilendiğim SMPS devresini anlatacağım.

Anahtarlama mod güç kaynaklarının önemi her geçen gün artmaktadır. Bu alandaki büyüme, güç çevrimi konusundaki diğer yöntemler arasında, en hızlı büyüme kaydedilen konudur.

Performansları ve lineer kaynaklara oranla daha küçük boyutlarda olmaları, tercih edilme sebeplerindendir.

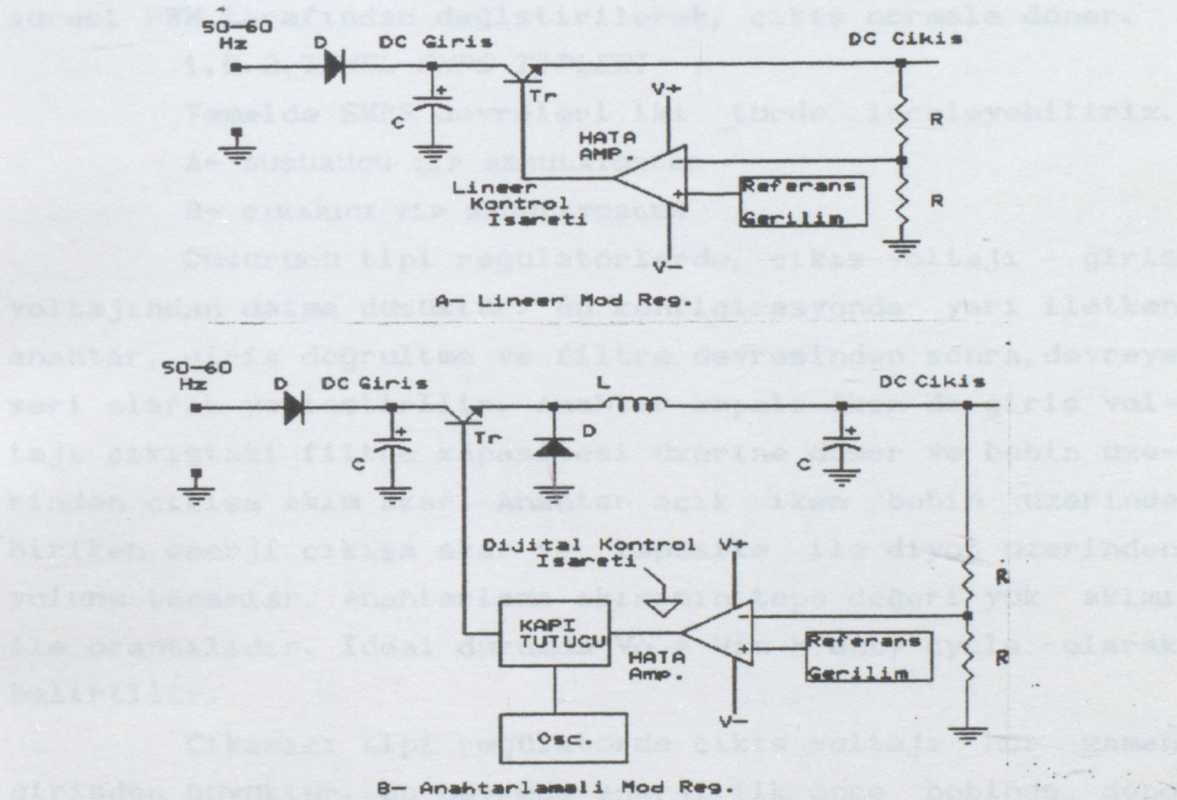
Önde gelen avantajları, verim yönünden iyi olmaları (% 70-80, lineerlerde % 30) ve küçük boyutları, dezavantajı ise önemli miktarda elektriksel gürültü üretmeleridir.

Ayrıca 20W'ın üzerindeki güçler için daha ucuza mal olmaktadır. Aşağıda, tabloda Anahtarlama ve Lineer mod kaynaklar arasındaki belirli özellikler karşılaştırılmıştır.

ÖZELLİK	ANAHTARLAMALI	LINEER
VERİM	% 75	% 30
BOYUT	2W / IN3	0.5W / IN3
AĞIRLIK	40W / Ib.	10W / Ib
HAT VE YÜK REG.	% 0.1	% 0.1
ÇIKIŞ RİPPL	50 mV	5mV
GÜRÜLTÜ	50-200 mV	---
GEÇİŞ CEVABI	1 ms	20 µs
TUTMA ZAMANI	20-30 ms	1-2 ms

TABLO : 1.5.1. - 1 TEMEL ÖZELLİKLER

Anahtarlama mod'da tutma zamanının büyük olması, girişteki filtre kapasitesinin yüksek voltajda kullanılması ve kapasitede depolanan enerjinin üzerindeki gerilimin karesiyle orantılı olmasından dolayıdır. Aşağıda şekil 1.5.1 - 1 de lineer ve anahtarlama regülatörlerin blok şemaları verilmiştir.



ŞEKİL : 1.5.1. - 1 LİNEER VE ANAHTARLAMALI REGÜLATÖRLER

Şekilde görüldüğü gibi ANAHTARLAMALI regülatörlerde önce şebeke voltajı doğrultup filtre edilir. Daha sonra DC voltaj çıkışı, yüksek frekans ile bir transistör tarafından anahtarlanır. Bu anahtarlama transistörü da bir trafoyu sürerek istenen değerde çıkış elde edilir. Trafo çıkışı doğrultulup, filtre edildikten sonra DC çıkış gerilimi elde edilir.

Bu en son çıkış katı seviyesini sabit tutmak için, bir kontrol devresi ile kontrol edilir. Kontrol katında PWM kontrollü osilatör, hata amplifikatörü ve referans gerilimi vardır. Hata amplifikatörü, referans gerilimi ile çıkıştan alınan gerilimi karşılaştırır. Çıkış yükselir veya düşerse hata amplifikatörü bunu sezer ve PWM'ı daha az genişlikte veya daha geniş darbe göndermesi için uyarır. Böylece transistörün ON süresi PWM tarafından değiştirilerek, çıkış normale döner.

1.5.2. TEMEL SMPS TİPLERİ

Temelde SMPS devreleri iki türde inceleyebiliriz.

A- DÜŞÜRÜCÜ TİP REGULATORLER

B- ÇIKARICI TİP REGULATORLER

Düşürücü tipi regülatörlerde, çıkış voltajı giriş voltajından daima düşüktür. Bu konfigürasyonda yarı iletken anahtar, giriş doğrultma ve filtre devresinden sonra, devreye seri olarak yerleştirilir. Anahtar kapalı iken dc giriş voltajı çıkıştaki filtre kapasitesi üzerine düşer ve bobin üzerinden çıkışa akım akar. Anahtar açık iken bobin üzerinde biriken enerji çıkışa akar ve kapasite ile diyot üzerinden yolunu tamamlar. Anahtarlama akımının tepe değeri yük akımı ile orantılıdır. İdeal durumda $V_o = V_{in} \times \text{duty cycle}$ olarak belirtilir.

Çıkarıcı tipi regülatörde çıkış voltajı her zaman girişten büyüktür. Bu devrede enerji ilk önce bobinde depo edilir, sonra bobin bu enerjiyi girişteki enerji (V_{in}) ile birlikte çıkışa aktarır. Anahtar kapalı iken akım bobin ve anahtar üzerinden akar. Bobin böylece enerji yüklenir, fakat çıkışa bu konumda enerji verilmez. Anahtar açık iken çıkıştaki voltaj, giriş voltajı artı bobin üzerindeki voltajdır.

Bu regülatörde tepe anahtarlama akımı, yük akımı ile orantılı değildir.

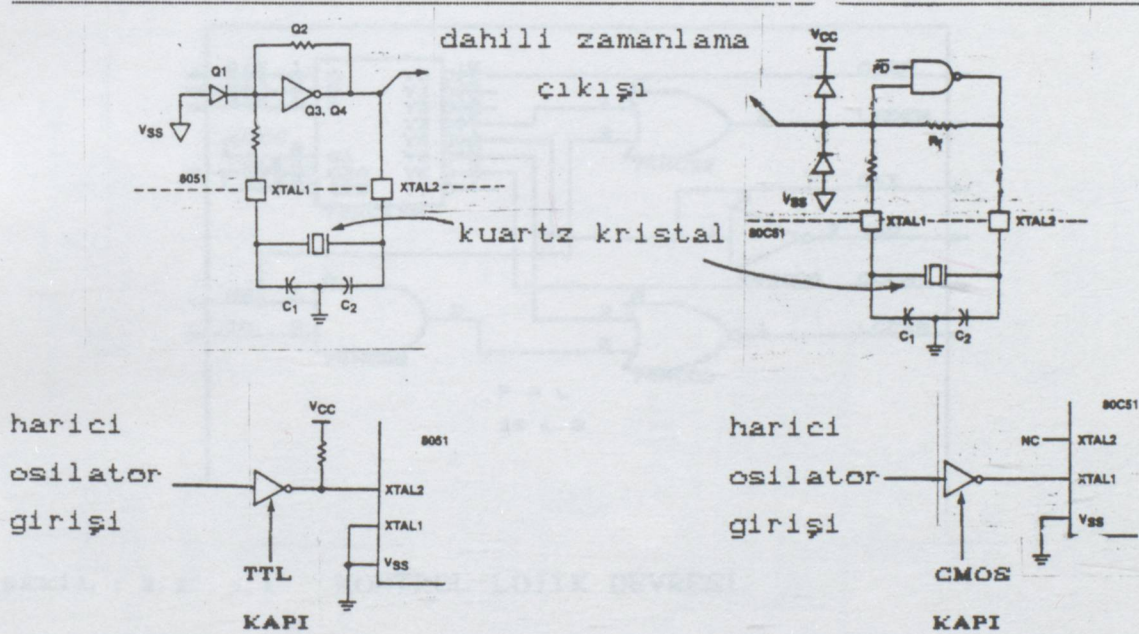
[4]

2. DEVRENİN İNCELENMESİ -DONANIM-

2.1. SAAT DEVRESİNİN OLUŞTURULMASI

Saat (Clock) devresi, akıllı sistemlerin adeta kalp atışı mahiyetindedir. Her bir saat darbesiyle sistemin çalışması adım, adım sağlanmış olur.

Saat devresinin frekansı, - ki bu osilatör frekansıdır - sistemin çalışma hızını belirler. Bu devre lojik olarak oluşturulabildiği gibi, devrede uyguladığımız şekliyle kristal kullanılarak yapılabilir. Aşağıdaki şekillerde, kullandığımız mikrodenetleyicinin yapım tekniğine de bağlı olarak üretilen osilatör devreleri ve bunlara dışardan ne tip elemanlar bağlandığı gösterilmiştir.



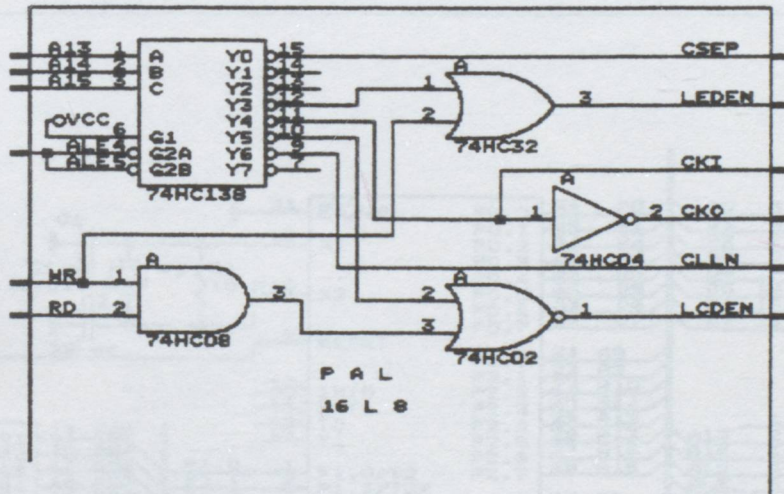
ŞEKİL : 2.1. - 4 HMOS VE CHMOS TEKNİĞİNDE OLUŞTURULMUŞ OSİLATÖR DEVRELERİ

Kartımızda, kristal ve kapasite kullanılarak oluşturulmuş osilatör devresi bulunmaktadır. Bu devreyi, üretici firmanın tavsiyesi üzerine seçtik.

2.2. KONTROL LOJİK DEVRESİ

Kart üzerinde bulunan epromu ve sürücü entegrelerini kontrol edip istenildikleri anda aktif edilmelerini sağlamak için bir lojik devre kurulmuştur. Bu devre belli bir döngü içinde, sırası gelen entegreyi seçip aktif etmektedir. 8032'nin belirli uçlarıyla bu fonksiyonlarını sağlamaktadır. ALE, PSEN, WR, RD, Adres yolu'nun A13, A14, A15 uçlarını kullanarak oluşturulan kontrol lojik devresini daha sonra PAL kullanarak tek bir entegre içine sığdırdım.

Aşağıda, kontrol lojik devresinin ayrıık olarak çizilmiş şekli görünmektedir.



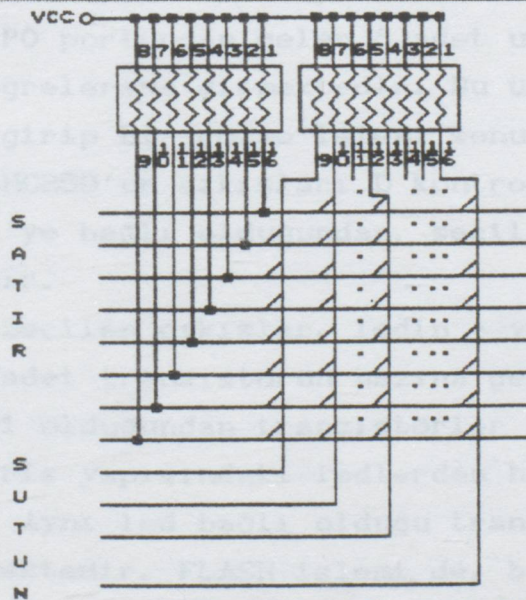
ŞEKİL : 2.2. - 1 KONTROL LOJİK DEVRESİ

Şekilde de görüldüğü gibi Adres Yolu uçları 74HC138 üzerinde adres çözücü uçlarına girip 8 değişik çıkışı belirlerler. Mikrodenetleyiciden gelen ALE ucu ise 74HC138'i seçmektedir. Mikrodenetleyiciden gelen WR, RD uçları da, 138'den gelen uçlarla, AND ve OR kapıları üzerinde ilgili fonksiyonu oluşturmak üzere bağlanırlar. Sonunda oluşan çıkışlar, değişik entegreleri seçmek üzere devreye dağılırlar.

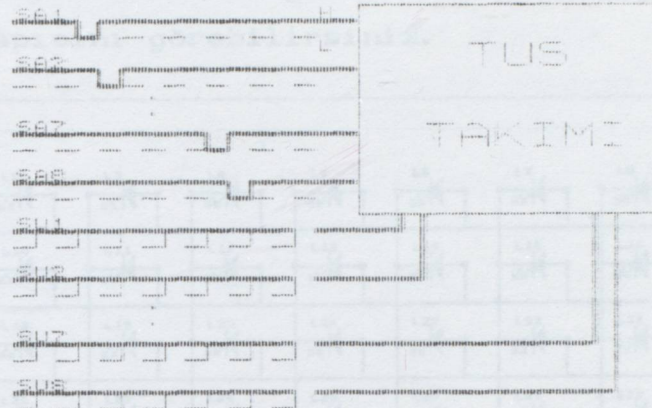
rekt toprağa bağılı olduğundan, biz bilgiyi girdiğimiz anda çıkış oluşmaktadır. 573'un çıkışı birisi lojik 0 diğerleride lojik 1 olacak şekildedir. Ayrıca çıkışlar 3K3 Ohm dirençle yukarı çekilip, normalde lojik 1 olacak şekilde bağlanmıştır.

Lojik 0 seviyesi, sırayla herbir çıkışa verilip herhangi bir tuşa basılmışsa bu seviyenin, tuşların sütunlarını okuyan 74HC244 entegresine iletilmesi sağlanır. 74HC244 daha önce oluşturduğumuz kontrol lojik devresinin CKIN ucuyla seçilmekte ve bu esnada tuşların sütunlarındaki mevcut durumu okumaktadır. Alınan bilgi, aynı anda yine CKIN yardımıyla P0 portuna verilmektedir. Aynı şekilde, P0 portu çıkışları da, tuş takımında meydana gelebilecek herhangi bir kısa devre tehlikesine karşı yukarı çekme dirençleriyle normalde lojik 1 konumuna çekilmiştir.

Tuşlarla ilgili bilginin nasıl değerlendirildiğini ileride, DEVRENİN YAZILIM İNCELEMESİ bölümünde anlatacağım. Ondan Önce tuşların konumunun, gelen ve giden bilginin dalga şeklinin bilinmesi gerekmektedir. Aşağıda ki şekillerde bu ayrıntıları vermeye çalıştım.



ŞEKİL : 2.9. - 2 TUŞLARIN BAĞLANTI KONUMU



ŞEKİL : 2.9. - 9 TUS TARAMA DALGA ŞEKLİ

2.4. LEDLERİN TARANMASI VE SÜRÜLMESİ

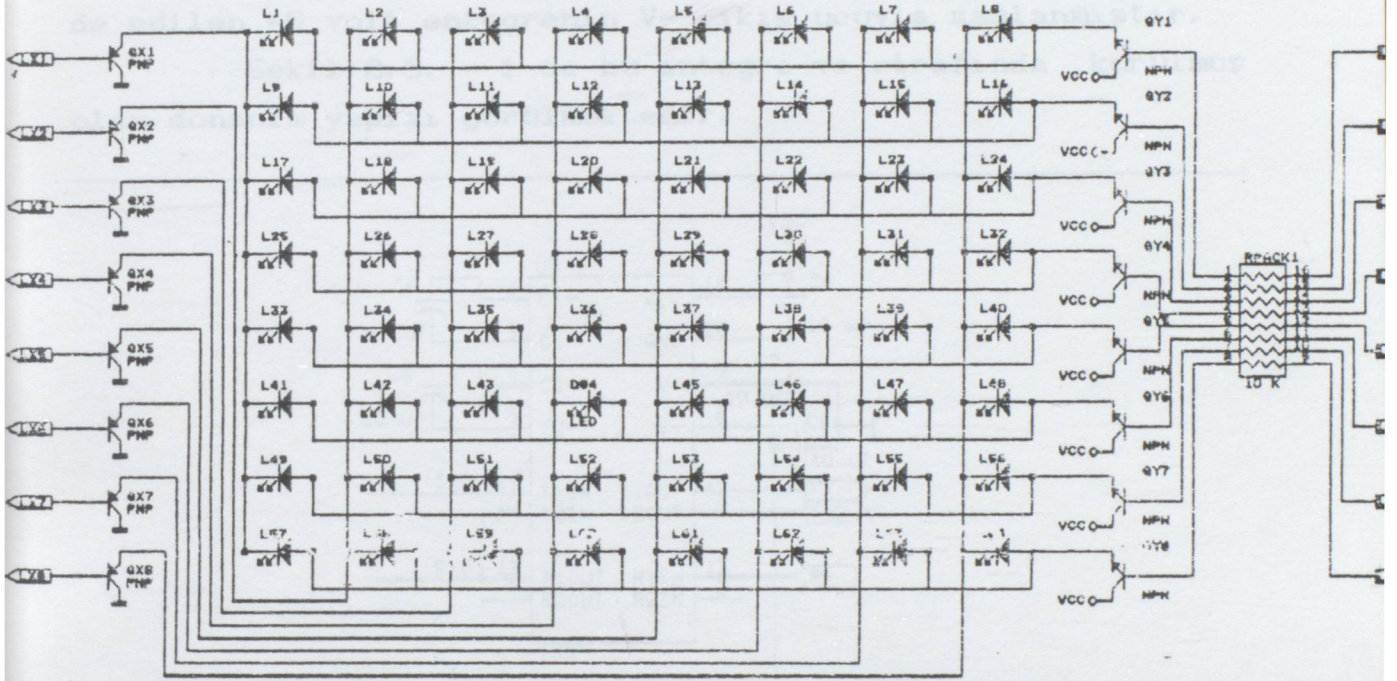
Kullandığımız ledler, fiziki olarak tuşların üzerinde olduğundan bunların konumu da aynen tuşlar gibidir. Herbir led hemen hemen aynı mantıkla sürülüp yakılmakta, söndürülmekte yada hızlı olarak yanıp sönmektedir (FLASH).

Ledlerin + ve - uçlarına bağlı iki adet tranzistörü süren 74HC259 entegresi, kontrol lojik devresinden gelen LDEN ve CLLN seçme uçlarıyla kontrol edilmektedir.

P0 portundan gelen 8 adet uç geçer, üçer ayrılıp sürücü entegrelerine girmektedir. Bu üç bit 74HC259'un seçme uçlarına girip bir çözme işlemi sonunda çıkışa aktarılmaktadır. 74HC259'un çıkışları D kontrol ucuna bağlıdır. Devrede bu üç Vcc'ye bağlı olduğundan, seçilen çıkış lojik 1 olarak çıkmaktadır.

Seçilen çıkışlar, ledin + ve - uçlarına bağlı NPN tipi iki adet tranzistörün bazına gelmektedir. Çıkışın konumu lojik 1 olduğundan tranzistörler aktif hale gelmekte ve 8'e 8 matris yapısındaki ledlerden hangisi seçilmiş ise yanmaktadır. Aynı led bağlı olduğu tranzistörler pasif edilerek söndürülmektedir. FLASH işlemi de, bu iki ayrı durumun kısa süreli tekrarları ile sağlanmaktadır. Led sürme işleminin ayrıntıları ileride YAZILIM İNCELEME bölümünde ele alınacak.

Aşağıda, ledlerin bağlantılarını ve sürülmesi işlemini donanım yapısını görebilirsiniz.



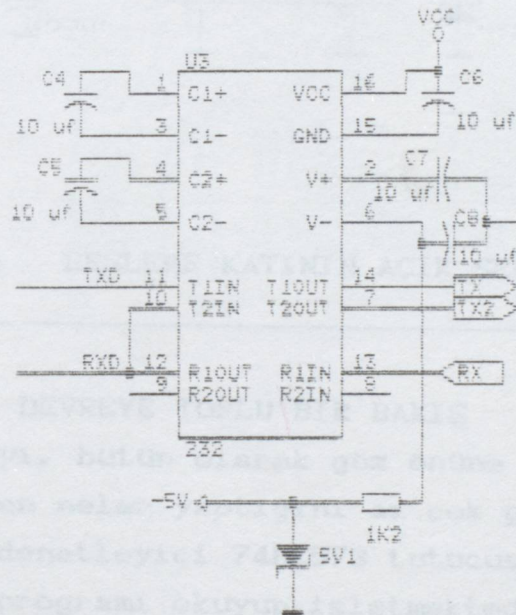
ŞEKİL : 2.4. - 1 LEDLERİN BAĞLANTILARI VE SÜRÜLMESİ

2.5. SERİ HABERLEŞME

Seri haberleşmeyi daha önce belirttiğim gibi, seri haberleşme için oluşturulmuş TSC232CPE entegresi ile gerçekleddim. Entegrenin iç yapısını, EK D kısmında bulabilirsiniz. Genel hatları ile bakarsak 10µf'lık kapasiteler ters ve düz polaritelerde bağlanarak +/- 10v'luk bir gerilim hattı oluşturulmuştur. Entegreye gelen seri bilgi bu +/- 10voltluk gerilim seviyesinin üzerine bindirilerek taşınmakta ve iletilmektedir.

Entegrenin, R1IN ucu santralden gelen bilgiyi alıp R1OUT ucuyla 8032'ye iletmekte ve T1IN ucu da aynı işlemi tersinden yaparak 8032'den gelen bilgiyi santrale iletmektedir. LCD üzerinde kullanmak üzere, bir zener yardımı ile elde edilen -5 volt entegrenin V- çıkış ucuyla sağlanmıştır.

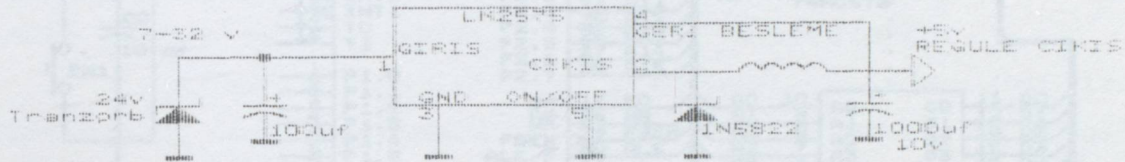
Sekil 2.5. - 1 de bu entegre ve etrafında kurulmuş olan donanım yapısı görülmektedir.



ŞEKİL : 2.5. - 1 SERİ HABERLEŞME ENTEGRESİ VE DONANIMI

2.6. DEVRENİN BESLEME KATI

Devrenin besleme katını, DC'den DC'ye dönüştürme tipi bir SMPS kullanarak yaptım. Giriş olarak, santral tarafından gelen 24v DC gerilim kullanılmakta ve çıkışta 5v DC gerilim elde edilmektedir. Kullandığım bu devre tipini daha önce anlatmıştım. Bu bölümde devrenin açık şemasını vereceğim.



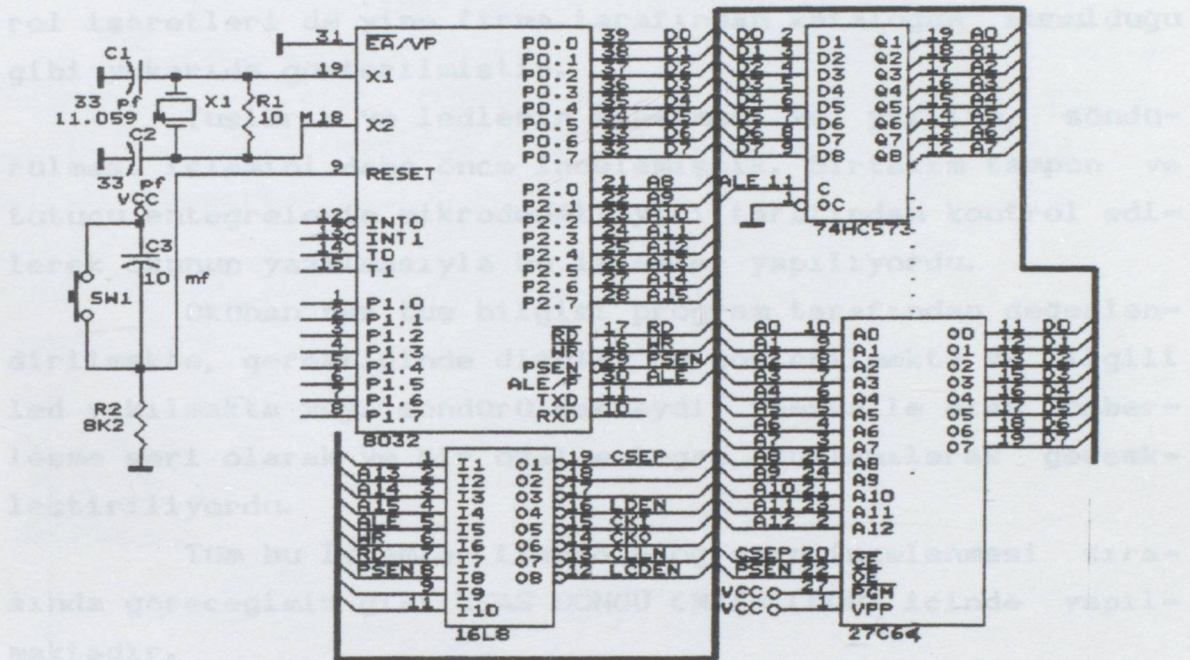
ŞEKİL : 2.6. - 1 BESLEME KATININ AÇIK ŞEMASI

2.7. DEVREYE TOPLU BİR BAKIŞ

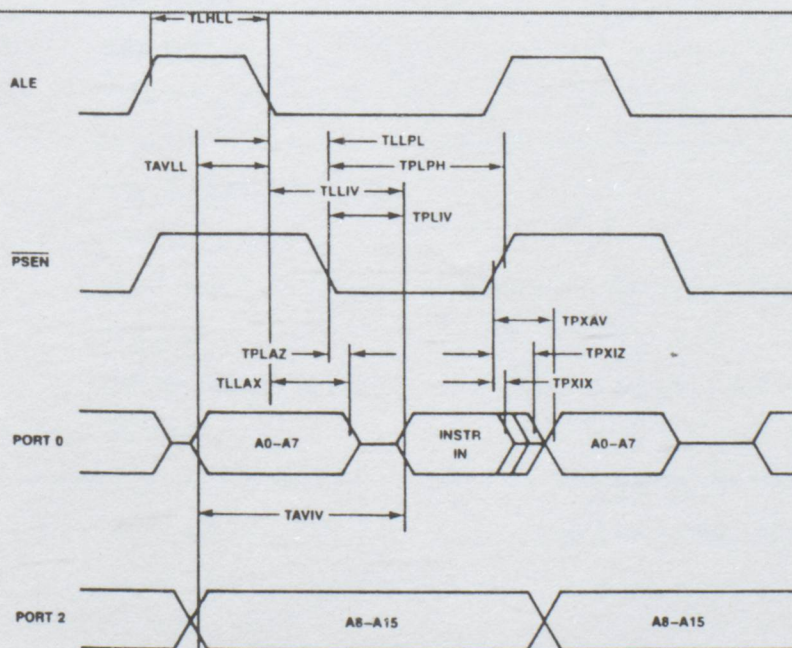
Devreyi, bütün olarak göz önüne getirirsek yazılım kısmına girmeden neler yaptığını az çok görebiliriz.

Mikrodenetleyici 74HC573 tutucusunun yardımıyla Eprom üzerinden programı okuyup işletmektedir. 16 bitlik adres yoluna sahip olan mikrodenetleyici P0 ve P2 portlarını kullanarak epromu okumaktadır. Önce iki portuda adres yolu olarak kullanmakta ve P0 portuna düşen ilk 8 bitlik kısmı o sırada tutucu tarafından tutulmaktadır. Arkasından bilginin alınması işlemi sırasında eproma gelen PSEN kontrol işareti ile epromun çıkışında oluşan bilgi, tutucuda tutulan adresin ilk 8 biti ile P2 portunda tutulan son 8 bitinin, epromda gördüğü adresin içeriğidir. Bu işlemi aşağıda ki şekilde görebiliriz.

ŞEKİL : 2.7. - 2 EPROM OKUMA DALGA ŞEKLİLERİ



ŞEKİL : 2.7. - 4 EPROMUN OKUNMASI



ŞEKİL : 2.7. - 2 EPROM OKUMA DALGA ŞEKİLLERİ

Bu işlemleri mikrodenetleyiciyi üreten firmanın katalogundaki verilerle gerçekleştirdim. Kullandığımız kontrol işaretleri de yine firma tarafından katalogda sunulduğu gibi yukarıda gösterilmiştir.

Tuşların ve ledlerin taranması ve yakılıp, söndürülmesi işlemini daha önce incelemiştik. Birtakım tampon ve tutucu entegrelerin mikrodenetleyici tarafından kontrol edilerek okunup yazılmasıyla bu işlemler yapılıyordu.

Okunan her tuş bilgisi program tarafından değerlendirilmekte, gerektiğinde display'de gösterilmekte ve ilgili led yakılmakta veya söndürülmekteydi. Santralle olan haberleşme seri olarak ve bir özel entegre kullanılarak gerçekleştiriliyordu.

Tüm bu işlemler ilerde programın incelenmesi sırasında göreceğimiz gibi ESAS DÖNGÜ (MAIN LOOP) içinde yapılmaktadır.

```

call    DELAY_L
call    DELAY_L
call    DELAY_L
call    DELAY_L
call    DELAY_L
call    DELAY_L
;-----
MOV     R0,200FH
LD      R0,R0
MOV     R0,400          ;internal R00's temsili
SINC    R0,LP
MOV     DPTH,6000 LINE   ;aslında 100'ler 1000
MOVX    RDPTR,A
;-----
call    DEF_VAL          ;initialisation
call    LCD_INIT
call    TIMER_INIT
call    AUTO_CLR

```


3. MONİTÖR PROGRAMININ İNCELEMESİ -YAZILIM-

3.1. SİSTEMİN İLK ŞARTLARININ VERİLMESİ

Program, öncelikle devrede kullandığımız elemanların ilk değerlerini (DEFAULT VALUE) tanımlamaktadır. Bu değerler her açılış anında veya her reset durumundan sonra yeniden okunmaktadır.

...
CSEG AT OOH ;reset initialization vector
JMP INITIAL

...

...

...INITIAL DEFAULT VALUES

INITIAL:

CLR EA;ALL ;interrupts are disabled

MOV SP,#08 ;D2H

call DELAY_L

call DELAY_L ;init. value for scan k. board

call DELAY_L ;init. value for scan k. board

call DELAY_L

call DELAY_L ;init. value for scan k. board

call DELAY_L ;init. value for scan k. board

MOV RO,#0FFH

LP:

MOV @RO,#00 ;internal RAM'ı temizle

DJNZ RO,LP

mov DPTR,#CLR_LINE ;açılışta led'ler sonuk

movx @DPTR,A

call DEF_VAL ;initialization

call LCD_INIT

call TIMER_INIT

call AUTO_CLK

...

Yukardaki programda, açılış anında sistemin taşınması gereken bazı ilk değerler gösterilmiştir. Sistem belirli bir kararlığa ulaşana kadar kesme (INTERRUPT) girişleri etkisiz kılınmıştır. Ledler sönmük durumdadır ve dahili ram bölgesi boşaltılmıştır. DELAY_L alt programına yapılan atlama komutları, şartlandırmaya başlamadan önce bazı elemanların elektriksel kararlılığa ulaşmalarını beklemek için yazıldı.

Tüm sistem için ortak ilk şartların alınmasıyla alt düzeydeki şartlandırmalara geçilmiştir.

```

...      ENTRY_SET
...      CLEAR_DSP
;INITIAL DEFAULT VALUES
DEF_VAL:
mov      TABDAT_S,#01
mov      TABDAT_S+1,#01
mov      TABDAT_S+2,#90
mov      SATIR,#0FEH      ;init.value for scan.K.board
mov      SUTUN_ES,#OFFH
;
clr      KEY_VAL           ;tuşa basıldığını goster
clr      DSPBUF_FUL_FL     ;display bilgisini goster
setb     CLK_FLAG
setb     X_EN_FL           ;Xon
clr      NEW_CLK_FL
clr      RDY_CLK_DSP_FL
clr      K_DEC_FL         ; 1.VE 2.dig.i ayırt eden fl
;
mov      TIMER,#00         ;KLAVYE DELAY İÇİN
mov      TIMER2,#00        ;LED KONTROLU İÇİN
mov      TIMER_ES,#00
mov      TIMER4,#00
mov      TIMER3,#00
mov      TIMER3_ES,#00
mov      POIN_REC,#TABREC_S

```



```

mov     POIN_DSP,#TABDSP_S
mov     DIG_NO,#00;
...
LCD_INIT:
call    LCD_DFLT
call    FUNC_SET
call    DSP_CONT
call    ENTRY_SET
call    CLEAR_DSP
RET                                           ;main loop
...
;INITIALIZE TIMERS and UART
TIMER_INIT:
orl     TMOD,#20H           ;timer1 ve timer0 mode 2 de
orl     87,#80H            ;PCON is at 87h address
mov     SCON,#50H
mov     TH1,#TH1H
...
mov     SBUF,#'>'
;konsol 'initial' oldu işareti (hex 3A)
;-----
setb    TRO                ;run timer0
setb    TR1                ;run timer1
setb    EA                 ;global intr. enable
RET                                           ;return main loop
...

```

Yukarıdaki program parçalarında ilk değerler ve LCD ile zamanlayıcılar için şartlandırma değerleri verilmiştir.

Son olarak ta '>' işareti LCD'ye gönderilip şartlandırmanın bittiği onaylanır.

3.2. ESAS DÖNGÜ -MAIN LOOP-

Sistem ilk şartlarını aldıktan sonra artık kararlı olarak çalışması gereken bir döngüye girmektedir. Bu döngü içinde sistem tüm fonksiyonlarını gerçekleştirir. Aşağıda ki program parçasında da görüleceği gibi tüm program bu ESAS DÖNGÜ içinde çalışmaktadır.

...

MAIN_LOOP:

```
call    LCD_DSP
call    KEY_SCAN
call    LED_CON
call    CLK_DSP
jmp     MAIN_LOOP
```

...

LCD_DSP alt programında display'e yazdıracağımız bilgiler düzenlenmekte ve display edilmektedir. KEY_SCAN alt programında tuş tarama, basılan tuşu okuma işlemleri yapılmaktadır. LED_CON alt programında ledler ile ilgili bilgilerin işleme sokulması, yani yakılacak ve söndürülecek ledler yakılmakta ve söndürülmektedir. CLK_DSP bölümünde saat ve tarih bilgisinin LCD'de gösterilip gösterilmeyeceği karara bağlanır. Daha önce de belirttiğim gibi, display aboneler ile ilgili bir işlem yürütmüyorsa, normalde saat ve tarih bilgisini göstermeye şartlandırılmıştı.

3.3 SANTRAL İLE HABERLEŞME

Santral ile haberleşmenin bir RS232 entegresi üzerinden yapıldığını daha önce belirtmiştim. Haberleşmeyle ilgili alt program aşağıdadır.

...

```
REC_UART:                                ;RECIEVE RECEPTION
push    acc
push    dph
```



```

push        dpl
push        0
push        1
push        2

```

```

;-----

```

```

jnb         RI,RET_INT
clr         RI
;ECHO TEST
mov         A,SBUF
cjne       A,KEY_CHR,ETR1
mov         ECHO,A
jnb         ACC.7,ATF1
mov         KEY_CHR,#ECHO_DEF
call        NUMBER_DSP
clr         CLR_DSP_FL
jmp         RET_INT

```

```

;
ATF1:
mov         KEY_CHR,#ECHO_DEF
jb         X_EN_FL,QD1
jmp         RET_INT
QD1:
call        QUICK_DSP      ;ECHO'yu display et
jmp         RET_INT
;-----

```

```

ETR1:
mov         KEY_CHR,#ECHO_DEF
call        REC_TABLE
RET_INT:
CLR         TI
pop         2
pop         1
pop         0
pop         dpl

```



```

pop          dph
pop          acc
RETI          ;return from serial port INTERRUPT
;-----
TXD_UART:    ;TRANSMIT RECEPTION
clr          TI
mov          SBUF,KEY_CHR
RET          ;return to klavye
...

```

Santral ile bir haberleşme yapılacaksa sistem seri hattan kesme alıp, kesme hizmet programına atlamaktadır. Bu bölümde Alma ve Verme işlemleri yürütülmektedir.

Alma işleminden önce, bazı saklayıcıların bilgileri saklanmakta, ardından seri tampondaki bilgi akümülatöre yüklenmektedir. Alınan bilgi gerekli işleme sokulmaktadır. Ardından saklayıcıların tutulan eski değerleri geri verilir.

Gönderme işlemi de benzer şekilde seri tampona okunan karakter değeri yüklenerek santrale gönderilmektedir.

3.4 TUŞLARIN BELİRLENMESİ

Kullandığım 64 tuşun 40 tanesi aboneleri, 16 tanesi hex. sayıları 8 tanesi de özel fonksiyon tuşlarını belirtmektedir. Bu tuşlardan herhangi birine basılıp basılmadığı aşağıda bir kısmını sunacağım KEY_SCAN alt programıyla anlaşılmaktadır.

```

...
KEY_SCAN:
mov          DPTR,#KLAV_ADR          ;tuş okuma routine'i
mov          P1,SATIR                ;P1'e yaz
movx         A,@DPTR                 ;data bus'tan acc 'le oku
...
...
mov          SUTUN,A                 ;sutunu oku
cjne         A,#OFFH,WHICH_COL        ;tuşa basıldı mı?
jb           KEY_VAL,LOW_HIGH          ;low'dan high'a geçiş var

```



```

;-----
mov          A,SATIR
;satırı sola doğru kaydırma routini
RLA
mov          SATIR,A
RET                      ;return main loop
...

```

Yukarıda ki program parçasından da görüleceği üzere önce klavye adresi atılarak klavyeyi kontrol eden entegreler seçilmiş olur. Ardından satır olarak atılacak değer P1 portuna yazılır ve aynı adresten okuma yapılarak (P0 portuyla) basılmış bir tuşun olup olmadığına karar verilir.

Eğer bir tuşa basılmamış ise satır değeri bir kez sola kaydırılıp bir sonraki tarama için esas döngüye döndürülür. Eğer bir tuşa basılmış ise bu sefer tuşun ne tuşu olduğuna karar verecek alt programlara sıçranır.

```

...
WHICH_COL:
jnb          ACC.0,LB1
jnb          ACC.1,LB2
...
...
jnb          ACC.6,LB7
jnb          ACC.7,LB8
LB1:
mov          SUTUN,#0FEH
jmp          DECISION
LB2:
mov          SUTUN,#0FDH
jmp          DECISION
...
...

```


LB7: geçerliliği hakkında karar verilir.

mov SUTUN,#0BFH

jmp DECISION

LB8: A,SUTUN

mov SUTUN,#07FH tuşun bulunduğu sütun

jmp DECISION

...

Öncelikle tüm tuşları içeren bir işlem olarak basılan tuşun hangi sütunda olduğu tespit edilir. (Tuşun satır nosu daha önceden biliniyor.) Bu tespitten sonra tuş hakkında karar verilen alt programa sıçranır.

...

DECISION:

mov a,SUTUN

cjne A,SUTUN_ES,DIFF_KEY ;tuşun bulunduğu sütunla eşleşmiyorsa

jb KEY_VAL,RETN ;tuşa basılı hal devam ediyor
;karar verilmemiş yeni bir tuş

clr C ;bu durumda karşılaştırmaya devam edilecektir.

mov A,TIMER

subb A,TIMER_ES

jnc DEB1

cpl A ;inversasyonu ;aşağıdaki sayılarda görüldüğü gibi

add A,#1

DEB1: ;A,DEBOUNCE DELAY

clr C ;bu durumda karşılaştırmaya devam edilecektir.

subb A,#DEB_DEL ;A,DEBOUNCE DELAY

jc RETURN ;A< DEBOUNCE DELAY'SA RETURN

setb KEY_VAL ;tuşa basıldı kararı

call TRANS_KEY ;tuşun bulunduğu sütunla karşılaştırma

RET ;main loop

...

Bu bölümde tuşa yeni mi basıldı , yoksa eski tuşa hala basılıyor mu veya tuşa basılıp kaldırıldı mı şeklinde tuşun geçerliliği hakkında karar verilir.

...

TRANS_KEY:

```

mov      A,SATIR
;8*ri+ci+1 formülüne göre tuşun konumu belirlenir
call     KEY_MAP_R
mov      B,#8
mul      AB
mov      B,A          ;B=8*ri
mov      A,SUTUN
call     KEY_MAP_C
add      A,B          ;A=8*ri+ci
add      a,#1         ;A=8*ri+ci+1

```

...

Yukardaki bölümde ise, satır ve sütunu artık kesinlik kazanmış bir tuşun kart üzerindeki durumu, yani hangi tür tuş olduğu belirlenir.

Bu durumda karşımıza 3 türlü olasılık çıkacaktır.

- 1- Abone tuşu
- 2- Hex. klavye tuşu
- 3- Özel fonksiyon tuşu

Bu olasılıkların incelenmesini aşağıdaki sayfalarda göstermeye çalıştım.

3.4.1 ABONE TUŞLARI

Daha önce de bahsettiğim gibi bunların 32 adedi dahili 8 adedi harici hatlar için kullanılmaktadır. Belirleme işlemi için gerçekleştirdiğim alt programın bir kısmını şu şekilde inceleyebiliriz.

```

;----- dahili ve harici abone tuşları -----
ADR24:
cjne     A,#1,ADR25
mov      KEY_CHR,#33          ;H1

```



```

setb      KEY_CHR.7
RET
ADR25:
cjne      A,#2,ADR26
mov       KEY_CHR,#34           ;H2
setb      KEY_CHR.7
RET
ADR26:
cjneA,#3,ADR28
mov       KEY_CHR,#1           ;D1
setb      KEY_CHR.7
RET
ADR28:
cjne      A,#4,ADR29
mov       KEY_CHR,#2           ;D2
setb      KEY_CHR.7
RET
ADR29:
cjne      A,#5,ADR30
mov       KEY_CHR,#3           ;D3
setb      KEY_CHR.7
RET
...
...
ADR33:
cjne      A,#9,ADR34
mov       KEY_CHR,#35           ;H3
setb      KEY_CHR.7
RET
...
ADR50:
cjne      A,#25,ADR51
mov       KEY_CHR,#39           ;H7
setb      KEY_CHR.7
RET

```



```

ADR51:
cjne      A,#58,ADR52
mov       KEY_CHR,#32      ;D32
setb      KEY_CHR.7
RET
ADR52:
cjne      A,#26,ADR53
mov       KEY_CHR,#40      ;H3
setb      KEY_CHR.7
RET
...

```

Oluşturulan tabloya göre, basılan tuşun elde edilen sayısal karşılığı akümülatöre yüklenmiştir. Böylece akümülatör ile karşılaştırmalar yapılarak tuşun hangi dahili veya harici hat olduğu ortaya çıkartılır. Ardından bu tuş değeri santral kısmına gönderilerek yansıma (ECHO) yapılması sağlanır. Böylece hem operatris cihazı için hemde santral için basılan tuş belirlenmiş olur.

```

...
YOLLA:
call      TXD_UART      ;basılan tuş HPC'ye
RET       ;return update routine
...

```

3.4.2 HEX. KLAVYE TUŞLARI

Hex. klavye tuşlarıda aynı mantıkla tablodan karşılaştırma yapıp, santralle yansıma yapılır.

```

...
;----- Hex. klavye tuşları -----
KEY_TRN_TAB:
cjne      A,#35,ADR1
mov       KEY_CHR,#'0'      ;0
mov       KEY_DEC,#0
RET

```



```

ADR1:
cjne A,#36,ADR2
mov KEY_CHR,#'1' ;1
mov KEY_DEC,#1
RET
ADR2:
cjne A,#37,ADR3
mov KEY_CHR,#'2' ;2
mov KEY_DEC,#2
RET
...
ADR13:
cjne A,#60,ADR14
mov KEY_CHR,#'D';D
RET
ADR14:
cjne A,#61,ADR15
mov KEY_CHR,#'E' ;E
RET
ADR15:
cjne A,#62,ADR16
mov KEY_CHR,#'F' ;F
RET
...
YOLLA:
call TXD_UART ;basılan tuş HPC'ye
RET ;return update routine
...

```


3.4.3 ÖZEL FONKSİYON TUŞLARI

Özel fonksiyon tuşları da yine aynı yöntemle belirlenip, santralle yansıma yapılarak işleme konur.

...

;----- Özel fonksiyon tuşları -----

ADR16: *vardır. Böylece ne tür fonksiyonları içerdikini*

cjne A,#47,ADR17

mov KEY_CHR,#'W' ;W

RET

ADR17:

cjne A,#48,ADR18

mov KEY_CHR,#'R' ;R

RET

ADR18:

cjne A,#64,ADR19

mov KEY_CHR,#0DH ;CR carriage return

RET

ADR19:

cjne A,#56,ADR20

mov KEY_CHR,#08H ;BS back space

RET

ADR20:

cjne A,#63,ADR21

mov KEY_CHR,#20H ;SB space bar

RET

ADR21:

cjne A,#39,ADR22

mov KEY_CHR,#1BH ;ESC (CLS)

RET

ADR22:

cjne A,#55,ADR23

mov KEY_CHR,#'T' ;TIME request

RET

ADR23:

```
cjne    A,#40,ADR24
mov     KEY_CHR,#'P'      ;P bekletme park etme
RET
```

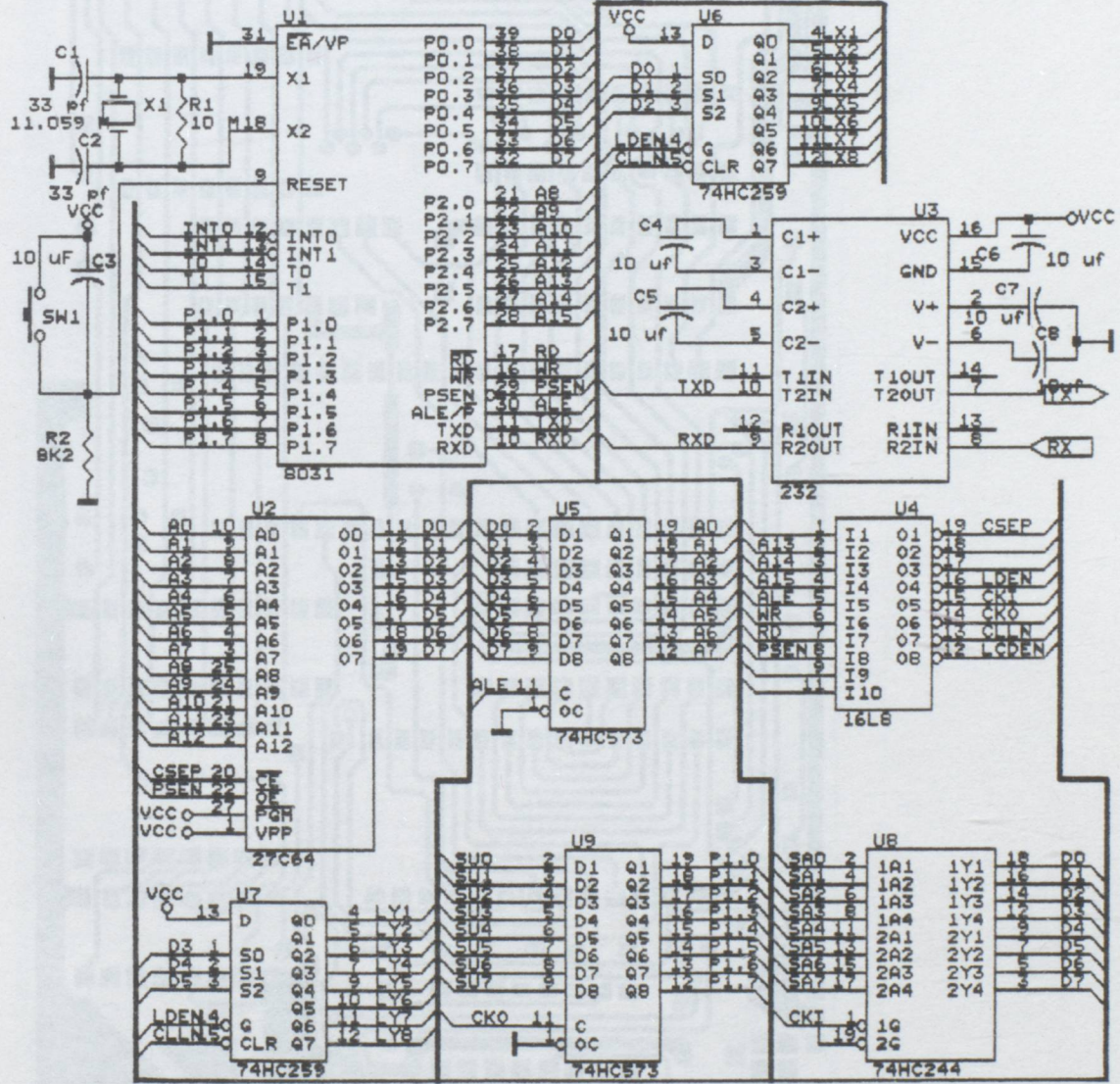
Özel fonksiyon tuşlarının belirlendiği kısmın tamamını verdim. Böylece ne tür fonksiyonları içerdiğini rahatlıkla görebiliriz.



4. DEVRE ŞEMALARI

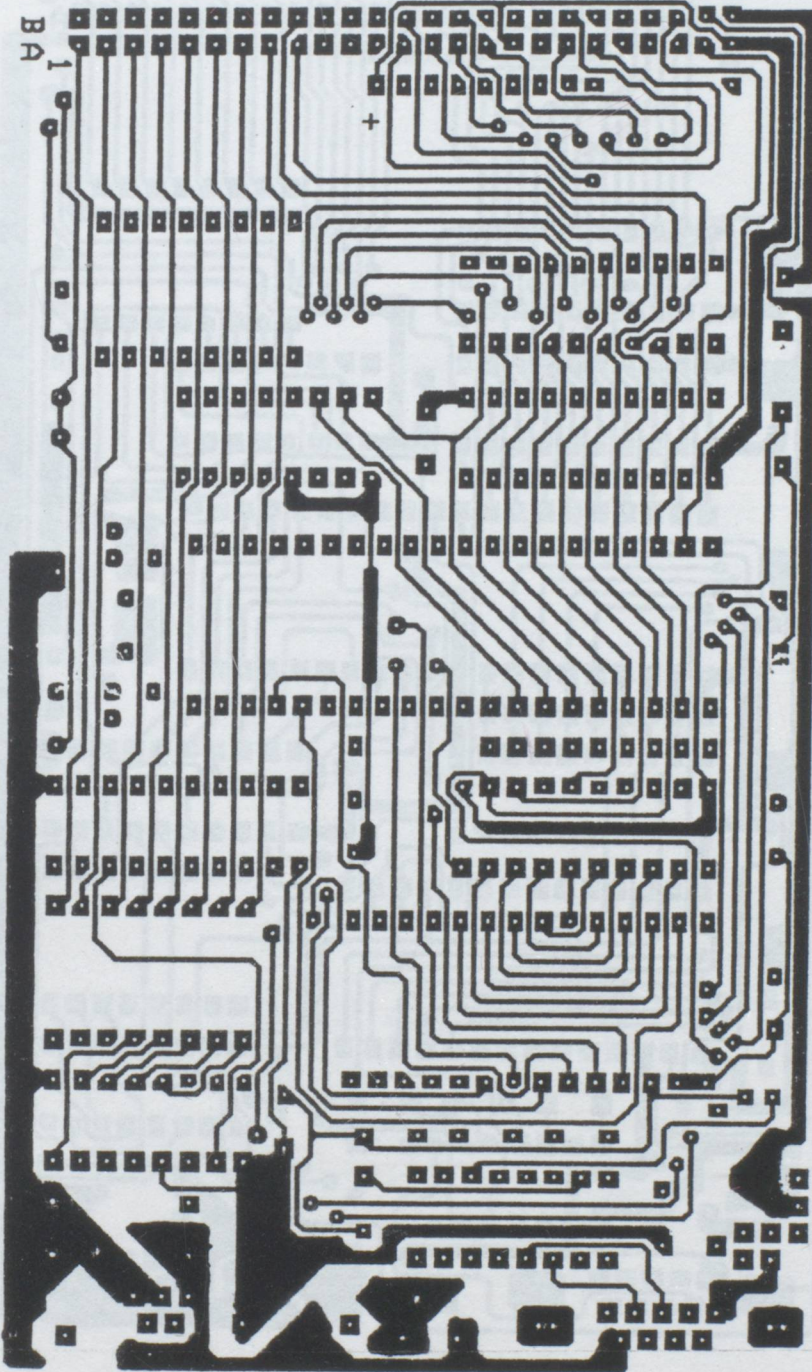
Aşağıdaki devrelerde yaptığım devrelerin, açık şemasını ve baskılı devrelerini bulabilirsiniz.

4.1. ANA KARTIN AÇIK ŞEMASI

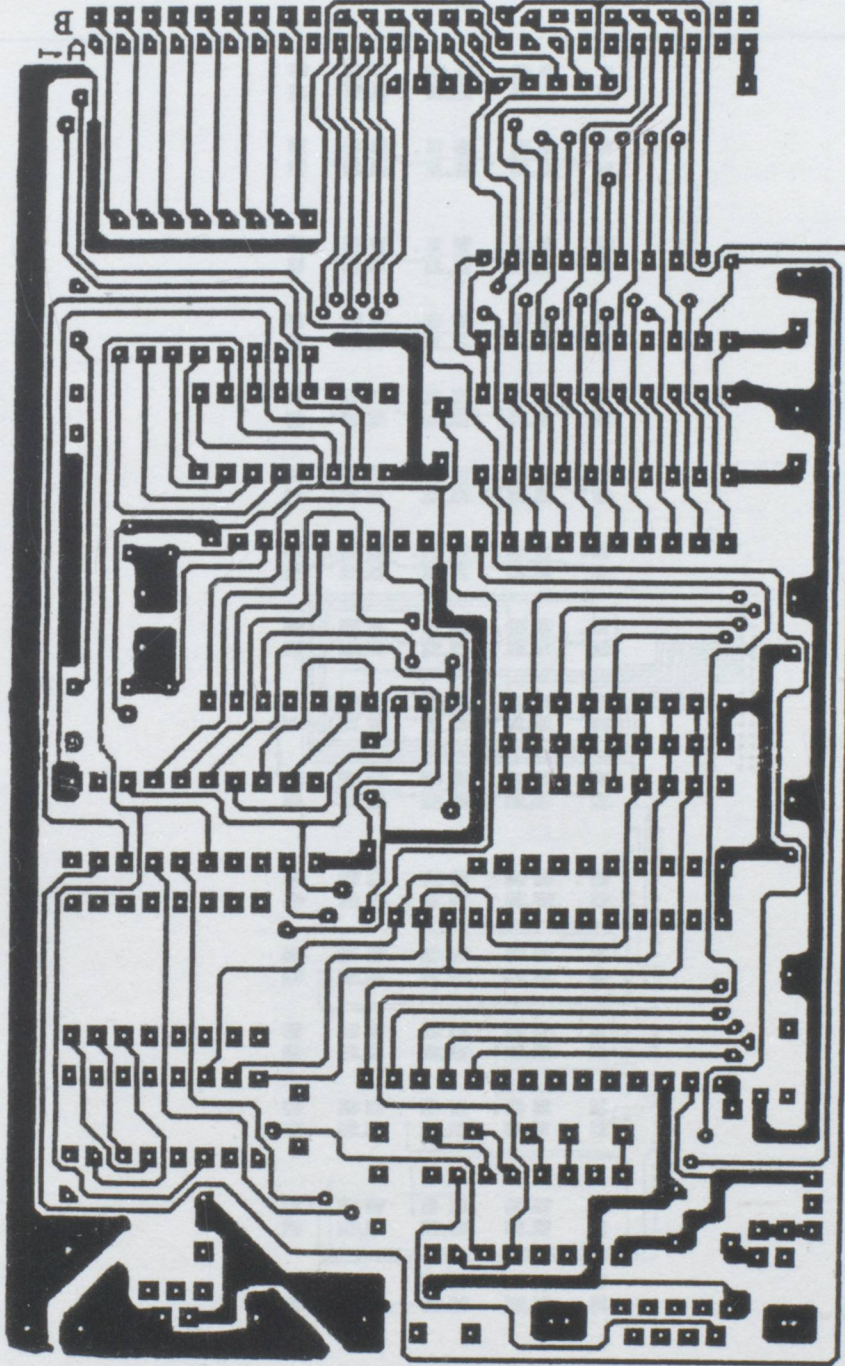


ŞEKİL : 4.1. - 1 ANA KARTIN AÇIK ŞEMASI

4.2. ANA KARTIN BASKILI DEVRESİ

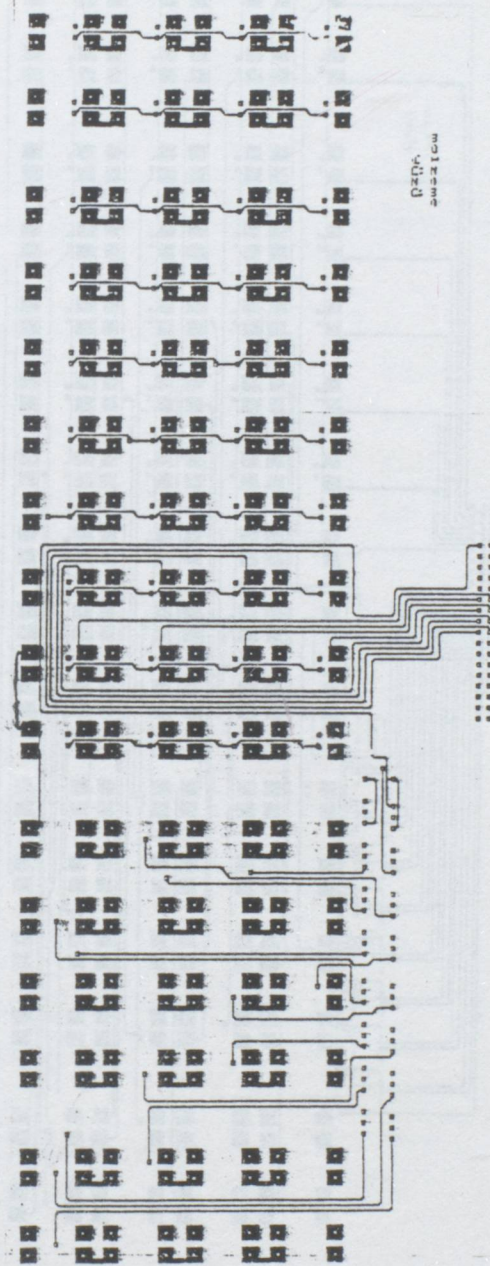


ŞEKİL : 4.2. - 1 ANA KARTIN BASKILI DEVRESİ
ÖN YÜZ -COMPONENT SIDE-

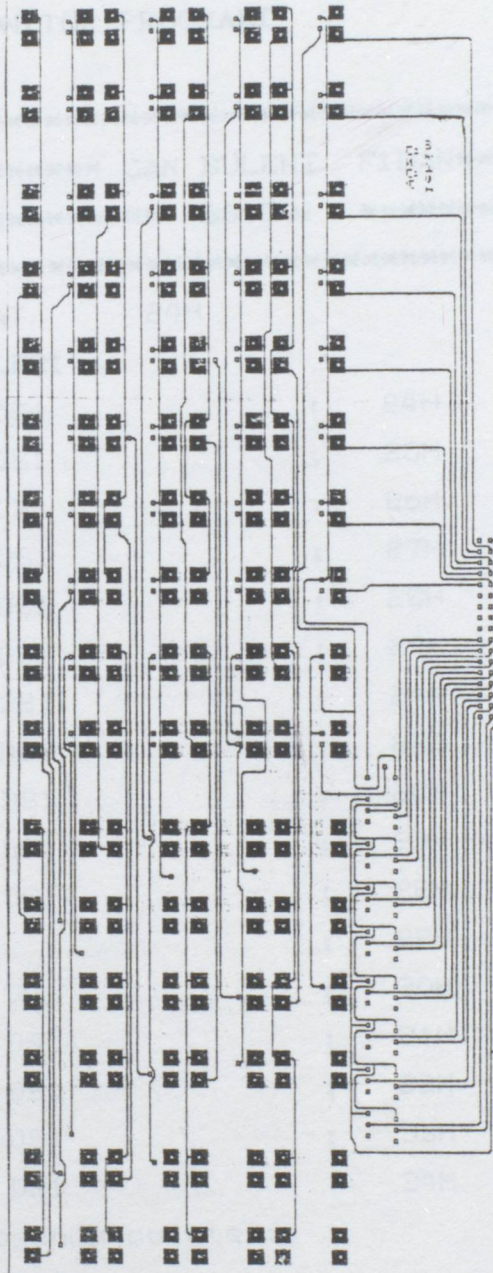


ŞEKİL : 4.2. - 2 ANA KARTIN BASKILI DEVRESİ
ARKA YÜZ -SOLDER SIDE-

4.3. KLAVYE KARTININ BASKILI DEVRESİ



ŞEKİL : 4.3. - 1 KLAVYE KARTININ BASKILI DEVRESİ
ÖN YÜZ -COMPONENT SIDE-



ŞEKİL : 4.9. - 2 KLAVYE KARTININ BASKILI DEVRESİ

ARKA YÜZ -SOLDER SIDE-

5. MONİTÖR PROGRAMI

NAME KONSOL

```

;*****
;***** CAN BÜLENT FİDAN*****
;***** 881501 *****
;*****

```

DSEG AT 24H

;BYTE DEĞİŞKENLERİ

```

TIMER: DS1 ; 24H
TIMER_ES: DS1 ; 25H
SATIR: DS1 ; 26H
SUTUN: DS1 ; 27H
SUTUN_ES: DS1 ; 28H
TIMER2: DS1 ; 29H
KEY_CHR: DS1 ; 2AH
ECHO: DS1 ; 2BH
POIN_DSP: DS1 ; 2CH
DIG_NO: DS1 ; 2DH
TIMER3: DS1 ; 2EH
TIMER3_ES: DS1 ; 2FH
TIMER4: DS1 ; 30H
LSB_DIG: DS1 ; 31H
KEY_DEC: DS1 ; 32H
KEY_DEC_ES: DS1 ; 33H
POIN_REC: DS1 ; 34H

```

;receive tablosunun pointeri

;BIT DEĞİŞKENLERİ

```

KEY_VALBIT 22H.0
DSPBUF_FUL_FLBIT 22H.1
CLK_FLAGBIT 22H.2
CLR_DSP_FLBIT 22H.3
NEW_CLK_FLBIT 22H.4
RDY_CLK_DSP_FLBIT 22H.5
X_EN_FLBIT 22H.6

```



```

TM_DT_SLBIT EQU 22H.7
K_DEC_FLBIT EQU 23H.1
REG_SELBIT EQU P1.1
R_WBIT EQU P1.2
;CURRENT BANK REGISTER=0
;r0=POIN_REC
;r1=POIN_DSP
;r2=COMPARE REG.
;r4 is used for DELAY_L
;r5 is " " DELAY1
;*****
DSEG AT 3BH
TABCLK_S: DS 02 ;clock reg.table (33H-40H)
TABCLK_E:
TABDAT_S: DS 03 ;datetable
TABDAT_E:
TABSUB_S: DS 28H
;subscriber reg.table AT (40H TO 7FH)
TABSUB_E: ;7FH
;EQUATION
TH1H EQU 0E8H
;1200 baudrate sayisi crys.osc.=11.0592 MHz icin
DEB_DEL EQU 05
;debounce delay 40 ms
CLK_DSP_DLY EQU 03
;auto clock delay
LCDIN_AD EQU 0A000H
;LCD init. address
KLAV_ADR EQU 8000H
;klavye enable adresi
LED_ADR EQU 6000H
;LED enable adresi
CLR_LINE EQU 0C000H
;HC259'un clear adresi

```



```

RECBUF_LEN      EQU    42
;Rec. buf'in uzunluđu
TABREC_S        EQU    80H
;recieve buffer'in bařlangıç adresi
TABREC_E        EQU    0AAH
;bitiř adresi
TABDSP_S        EQU    0ABH
;display      "      bařlangıç      "
TABDSP_E        EQU    0D3H
;display      "      bitiř      "
TABNUMBER_S     EQU    0D4H
;Abonelerin izafi noları bařlangıç adr.
TABNUMBER_E     EQU    0FCH
;Abonelerin izafi noları bitiř adr.
ECHO_DEF        EQU    0FFH
;ECHO'nun bařlangıç değeri
;***** HEADER'LAR*****
X_CLK_HEAD      EQU    01      ;HPC'ye giden saat header'i
X_DAT_HEAD      EQU    02      ; " " tarih "
CLK_HEAD        EQU    0F0H    ;saat bilgisi([)
DAT_HEAD        EQU    0F1H    ;tarih "([)
SUBSTA_HEAD     EQU    0F2H    ;abone "(/)
CONN_HEAD       EQU    0F3H    ;connection "(=)
SUBNUM_HEAD     EQU    0F4H    ;Abone nosu (-)
RESET_HEAD      EQU    0F5H
SCREEN_HEAD     EQU    0FEH    ;0AHdisplay "(LF)
;*****
CSEG            AT          00H
JMP              INITIAL      ;reset initialization vector
;
ORG              0BH
JMP              BGS          ;background syst.
;
ORG              23H

```



```

JMP          REC_UART          ;uart interrupt vector
;*****
ORG          30H
;*****
; MAIN      LOOP
;*****
INITIAL:
CLR          EA                ;ALL interrupt are disabled
MOV          SP,#08            ;D2H
call         DELAY_L
call         DELAY_L
call         DELAY_L
call         DELAY_L
call         DELAY_L
call         DELAY_L
call         DELAY_L
call         DELAY_L
;-----
MOV          RO,#OFFH
LP:          MOV    @RO,#00      ;internal RAM'ı temizle
DJNZ        RO,LP
;-----
mov         DPTR,#CLR_LINE      ;açılısta led'ler sonuk
movx        @DPTR,A
;-----
call        DEF_VAL              ;initialization
call        LCD_INIT
call        TIMER_INIT
call        AUTO_CLK
mov         57H,#00
;*****
MAIN_LOOP:
call        LCD_DSP
call        KEY_SCAN
call        LED_CON

```



```

call          CLK_DSP
jmp MAIN_LOOP
;*****
;INITIAL DEFAULT VALUES
DEF_VAL:
mov          TABDAT_S,#01
mov          TABDAT_S+1,#01
mov          TABDAT_S+2,#90
mov          SATIR,#0FEH      ;init.value for scan.K.board
mov          SUTUN_ES,#0FFH
;
clr          KEY_VAL          ;tuşa basıldığını göster
clr          DSPBUF_FUL_FL     ;display bilgisini göster
setb         CLK_FLAG
setb         X_EN_FL          ;Xon
clr          NEW_CLK_FL
clr          RDY_CLK_DSP_FL
clr          K_DEC_FL         ; 1.VE 2.dig.i ayırt eden fl
;
mov          TIMER,#00        ;KLAVYE DELAY İÇİN
mov          TIMER2,#00       ;LED KONTROLU İÇİN
mov          TIMER_ES,#00
mov          TIMER4,#00
mov          TIMER3,#00
mov          TIMER3_ES,#00
mov          POIN_REC,#TABREC_S
mov          POIN_DSP,#TABDSP_S
mov          DIG_NO,#00;
;-----
mov          a,#101           ;default degerler
mov          r1,#TABNUMBER_S
NUM_DAH:
mov          @r1,a
cjne        a,#133,INC_R1

```



```

mov          a,#1
NUM_HAR:
mov          @r1,a
cjne        r1,#TABNUMBER_E,INC_R2
jmp          RET_DEF
INC_R2:
inc          a
inc          r1
jmp          NUM_HAR
INC_R1:
inc          a
inc          r1
jmp          NUM_DAH
;-----
RET_DEF:
RET                                ;main loop
;*****
;INITIALIZE TIMERS and UART
TIMER_INIT:
orl          TMOD,#20H             ;timer1 ve timer0 mode 2 de
orl          87,#80H              ;PCON is at 87h address
mov          SCON,#50H
mov          TH1,#TH1H
;autoreload capture value for timer1
clr          TFO                   ;intr.timer0 flag enable
setb         TF1                   ;intr.timer1disable
setb         ETO                   ;intr. enables timer0
setb         ES                    ;uart intr. enable
setb         PS                    ;uart intr. pr.
mov          SBUF,#'>'
;konsol 'initial' oldu isareti (hex 3A)
;-----
setb         TRO                   ;run timer0
setb         TRI                   ;run timer1

```



```

setb      EA                ;global intr. enable
RET                ;return main loop
;*****
;interrupt service routine
;*****
;register used:acc
;current bank :0
BGS:                ;TIMER REGISTERS
push      ACC
mov       A,TIMER
cjne     A,#40H,DG1
jmp      INC_TIMER4
DG1:
cjne     A,#80H,DG2
jmp      INC_TIMERS
DG2:
cjne     A,#0B0H,DG3
jmp      INC_TIMER4
DG3:
cjne     A,#0FFH,G_RET1
INC_TIMERS:
inc      TIMER2          ;T2=1 s led ring-back durumu
anl      TIMER2,#0FH
mov      a,TIMER2
cjne     a,#0FH,INC_TIMER4
inc      TIMER3          ;T3=15 s CLK_DSP_DLY için
INC_TIMER4:
inc      TIMER4          ;T4=0.5 s " bekletme "
anl      TIMER4,#0FH
G_RET1:
inc      TIMER          ;T=2 ms DEBOUNCE DELAY
pop      ACC
RET

```



```

*****
***** UART SUBROUTINE *****
*****
;register used:acc,r0,r1,dph,dpl
;current PSW is '0'
REC_UART:                                ;RECIEVE RECEPTION
push      acc
push      dph
push      dpl
push      0
push      1
push      2
;-----
jnb      R1,RET_INT
clr      RI
;ECHO TEST
mov      A,SBUF
cjne     A,KEY_CHR,ETR1
mov      ECHO,A
jnb      ACC.7,ATF1
mov      KEY_CHR,#ECHO_DEF
call     NUMBER_DSP
clr      CLR_DSP_FL
jmp      RET_INT
;
ATF1:
mov      KEY_CHR,#ECHO_DEF
jb       X_EN_FL,QD1
jmp      RET_INT
QD1:
call     QUICK_DSP      ;ECHO'yu display et
jmp      RET_INT
;-----

```



```

ETR1:
mov     KEY_CHR,#ECHO_DEF
call    REC_TABLE
RET_INT:
CLR     TI
pop     2
pop     1
pop     0
pop     dpl
pop     dph
pop     acc
RETI                                ;return from serial port INTERRUPT

```

```

;-----
TXD_UART:                                ;TRANSMIT RECEPTION

```

```

clr     TI
mov     SBUF,KEY_CHR
RET     ;return to klavye

```

```

NUMBER_DSP:

```

```

;1 byte'lik hex ab.nosunu 3 digitlik BCD sayıya çevirir

```

```

call    CLEAR_DSP                    ; ve display eder.

```

```

NUM_DSP:

```

```

clr     acc.7

```

```

push    acc

```

```

clr     c

```

```

subb    a,#99

```

```

jc      DAHILI

```

```

mov     ECHO,#'H'

```

```

jmp     DIR1

```

```

DAHILI:

```

```

mov     ECHO,#'D'

```

```

DIR1:

```

```

call    QUICK_DSP

```

```

pop     acc

```

```

add     a,#(TABNUMBER_S-1)

```


;abone numaralarına ulaşılır

```

mov     r1,a
;hex noyu 3 digit BCD'ye çevirir
mov     a,@r1
mov     b,#10
div     ab
mov     LSB_DIG,b
mov     b,#10
div     ab
add     a,#30h
mov     ECHO,A
call    QUICK_DSP
mov     a,b
add     a,#30h
mov     ECHO,a
call    QUICK_DSP
mov     a,LSB_DIG
add     a,#30h
mov     ECHO,a
call    QUICK_DSP
call    AUTO_CLK
RET

```

REC_TABLE:

```

mov     R0,POIN_REC
mov     @R0,A           ;Acc'de SBUF var
cjne    A,#SUBSTA_HEAD,ATLA1
call    SUB_STA
jmp     UPDATE_REC
ATLA1:
cjne    A,#CLK_HEAD,ATLA2
call    SAAT
setb    NEW_CLK_FL
jmp     UPDATE_REC

```



```

ATLA2:
cjne      A,#DAT_HEAD,ATLA3
call      TARIH
jmp       UPDATE_REC
ATLA3:
cjne      A,#SCREEN_HEAD,ATLA4
call      DIAGNOS
jmp       UPDATE_REC
ATLA4:
cjne      A,#SUBNUM_HEAD,ATLA5
call      NUMBER_TRN
jmp       UPDATE_REC
ATLA5:
cjne      A,#CONN_HEAD,ATLA6
call      CONNECT_DSP
jmp       UPDATE_REC
ATLA6:
cjne      A,#RESET_HEAD,INC_TABR
call      SW_RES
;-----
UPDATE_REC:
mov       POIN_REC,#TABREC_S
RET                               ;go receive reception
INC_TABR:
mov       R2,POIN_REC
cjne      r2,#TABREC_E,INC_REC
jmp       UPDATE_REC
INC_REC:
inc       POIN_REC
RET                               ;go receive reception
;-----
SW_RES:
; Bu bolum abone ledlerinin
MOV       R0,#(TABSUB_S-1); tamamını sondurur.

```



```

LOP1:
inc      RO
MOV      @RO,#00
cjne     RO,#(TABSUB_E+1),LOP1
call     DEMO2
RET

;-----
;SUBSCRIBER STATUS
SUB_STA:
dec      RO
mov      A,@RO
anl      A,#00111111B      ;abone nosunu maskele ve al
add      A,#(TABSUB_S)      ;int.ram'deki adrese 39h ekle
; abone noları 1'den 40-dahil-'a kadar
mov      R1,A
;40H eklenmiş abone nosu (TABSUB)
mov      A,@RO
anl      A,#11000000B      ;SS'yi maskele ve al
orl      A,TIMER2          ;update anındaki timer2
mov      @R1,A              ;status reg.ini acc.'ye koy
RET

;*****
SAAT:
mov      R1,#(TABCLK_E-1)
dec      RO
mov      a,@RO
mov      @R1,a
dec      RO
dec      R1
mov      a,@RO
mov      @R1,a
RET

```


TARİH:

;three bytes for date information GUN:AY:YIL(11-6-90)

mov R1,#(TABDAT_E-1)

;TARİH bilgisinin tablodaki yeri

dec RO 00111111

mov a,@RO 00000000

mov @R1,a

dec RO

dec R1

mov a,@RO

mov @R1,a

dec RO

dec R1

mov a,@RO

mov @R1,a

RET

DIAGNOS:

mov RO,#TABREC_S

mov R1,#TABDSP_S

mov A,POIN_REC

add A,#RECBUF_LEN

mov POIN_DSP,A

GO_TRNS: RO

mov a,R1

cjne a,POIN_DSP,INCR1

setb DSPBUF_FUL_FL ;1'SE dsp buffer dolu

RET;rec_table

INCR1:

mov A,@RO

mov @R1,A

inc RO ;receive table pointer

inc R1 ;display table "

jmp GO_TRNS


```
*****
```

```
NUMBER_TRN:
```

```
mov     RO,#TABREC_S
mov     a,@RO
anl     a,#00111111b
add     a,#(TABNUMBER_S-1)
mov     R1,a
inc     RO
mov     a,@RO
mov     @R1,a
RET
```

```
*****
```

```
CONNECT_DSP:
```

```
; Hatların kimlerle goruslugunu display eder.
dec     RO
; HPC'den gelen ilk byte o hattın kiminle goruslugunu;
mov     a,@RO
; 2.byte ta basılan tuşun nosunu gosterir
call    NUMBER_DSP
mov     ECHO,#'<'
call    QUICK_DSP
mov     ECHO,#'>'
call    QUICK_DSP
dec     RO
mov     a,@RO
call    NUM_DSP
clr     CLR_DSP_FL
RET
```



```

;*****
;***** LCD SUBROUTINE*****
;*****
;registered used:acc,dptr,r4,r5,p1.1,p1.2
LCD_INIT:
call      LCD_DFLT
call      FUNC_SET
call      DSP_CONT
call      ENTRY_SET
call      CLEAR_DSP
;callCHR_DEF
call      DEMO1
RET                                ;main loop
;-----
LCD_DFLT:
clr       REG_SEL
clr       R_W
mov       DPTR,#LCDIN_AD
mov       a,#3FH
movx      @DPTR,A
call      DELAY_L                ;50 ms wait
movx      @DPTR,A
call      DELAY_L                ;50 ms wait
movx      @DPTR,A
call      DELAY_L                ;50 ms wait
RET                                ;return LCD_INIT
;-----
FUNC_SET:
call      READ_BS_FL
clr       REG_SEL
clr       R_W
mov       a,#34H                ;func.set
movx      @DPTR,A
RET                                ;return LCD_INIT

```



```

;-----
DSP_CONT:      READ_BS_FL
call          READ_BS_FL
clr          REG_SEL
clr          R_W
mov          a,#0EH          ;display on/off cont.
movx        @DPTR,A
RET          CLR_DSP_FL      ;return LCD_INIT
;-----
ENTRY_SET:
call          READ_BS_FL
clr          REG_SEL
clr          R_W
mov          DPTR,#LCDIN_AD
mov          a,#06
movx        @DPTR,A
RET          ;return
;-----
RET_HOME:      @DPTR,A
call          READ_BS_FL
clr          REG_SEL
clr          R_W
mov          a,#02
movx        @DPTR,A
RET;return
BCK_SHIFT:
call          READ_BS_FL
push         acc
clr          REG_SEL
clr          R_W
mov          a,#10H
movx        @DPTR,A
pop          acc
RET          ;return

```


CLEAR_DSP:

```
call    READ_BS_FL
push    acc
clr     REG_SEL
clr     R_W
mov     a,#01
movx    @DPTR,A
setb    CLR_DSP_FL
pop     acc
RET
```

;

DD_RAM:

```
call    READ_BS_FL
push    acc
clr     REG_SEL
clr     R_W
;mov    a,DIG_NO
orl     a,#80h
movx    @DPTR,A
pop     acc
RET
```

;

DSP_SHIFT:

```
call    READ_BS_FL
clr     REG_SEL
clr     R_W
mov     a,#07
movx    @DPTR,A
RET
```

;

DEMO1:

```
mov     dptr,#YAZI1
jmp     WRITE_LCD
```


DEMO2:

call CLEAR_DSP
mov dptr,#YAZI2

;

WRITE_LCD:

clra
mov ca,@a+dptr
cjne a,#00,SEND_CHR
clr CLR_DSP_FL
RET

;

SEND_CHR:

push dph
push dpl
call PUT_LCD
;
pop dpl
pop dph
inc dptr
jmp WRITE_LCD

;

YAZI1: DB 'C. BULENT FIDAN 881501
YAZI2: DB '*** System Initialized ***',0
SA_DA: DB 'Saat/Dakika=?',0
GUN: DB 'Gun/Ay/Yil=? ',0

;

CHR_DEF:

call READ_BS_FL
mov a,#01
mov R5,#40h
setb REG_SEL
clr R_W
movx @DPTR,A
mov DPTR,#C_PTRN


```

CHR_CHG:
call    CG_RAM
clr     a
movc    a,@a+dptr
call    CG_DATA
cjne    R5,#47,INC_DATA
RET

```

```

INC_DATA:
inc     R5
inc     dptr
jmp     CHR_CHG

```

```

CG_RAM:
call    READ_BS_FL
push    dph
push    dpl
clr     REG_SEL
clr     R_W
mov     a,R5
movx    @DPTR,A
pop     dpl
pop     dph
RET

```

```

CG_DATA:
push    dph
push    dpl
call    PUT_LCD
pop     dpl
pop     dph
RET

```

```

;-----
C_PTRN:
DB      0EH,11H,10H,10H,11H,0EH,04H,00H
;-----

```


DELAY_L: ;~ 50 ms

mov R4,#100

WAIT: mov R5,#0FFH

djnz R5,\$

djnz R4,WAIT

RET

DELAY1: ;500 us

mov R5,#0FFH

djnz R5,\$

RET

READ_BS_FL:

;DB7'den busy flag'i okur 0'sa subr.'den çıkarır

push acc

clr REG_SEL

setb R_W

mov DPTR,#LCDIN_AD

LOOK_AGA:

movx A,@DPTR

jb ACC.7,LOOK_AGA

;LCD busy flag bit is 7.of acc

pop acc

RET

PUT_LCD:

call READ_BS_FL

setb REG_SEL

clr R_W

movx @DPTR,A

RET

QUICK_DSP:

;SFK'ya basılan ve echo'su gelen karakteri yazar

setb CLK_FLAG

clr RDY_CLK_DSP_FL

jb CLR_DSP_FL,DEV_QUICK

call CLEAR_DSP

DEV_QUICK:

mov a,ECHO

cjne a,#0DH,CLS_R ;CR mu?

mov DIG_NO,#00

call RET_HOME

call AUTO_CLK

RET

CLS_R:

cjne a,#1BH,BCK_S ;ESC mi?

mov DIG_NO,#00

;call ENTRY_SET

call CLEAR_DSP

RET

BCK_S:

cjne a,#08,YAZ

call BCK_SHIFT ;back space

RET

YAZ: call PUT_LCD

DUS1: mov a,DIG_NO

clr c

subb a,#40

jc RET_GO

mov DIG_NO,#00

call CLEAR_DSP

RET

RET_GO:

inc DIG_NO

RET


```
;receive reception
```

```
;
```

```
LCD_DSP:
```

```
jb CLK: DSPBUF_FUL_FL,FULL_DSP
```

```
RET
```

```
;MAIN LOOP
```

```
;
```

```
FULL_DSP:
```

```
setb CLK_FLAG
```

```
clr RDY_CLK_DSP_FL
```

```
call CLEAR_DSP
```

```
mov R1,#TABDSP_S
```

```
GO1:
```

```
mov A,R1
```

```
cjne A,POIN_DSP,TEKRIR
```

```
clr CLK: DSPBUF_FUL_FL
```

```
;boşalttıktan sonra flag 0'lanır
```

```
mov POIN_DSP,#TABDSP_S
```

```
call AUTO_CLK
```

```
clr CLR_DSP_FL
```

```
RET
```

```
;main loop
```

```
TEKRIR:
```

```
mov a,@R1
```

```
call PUT_LCD
```

```
inc R1
```

```
jmp GO1
```

```
;
```

```
AUTO_CLK:
```

```
mov TIMER3_ES,TIMER3
```

```
clr CLK_FLAG
```

```
RET
```

```
CLK_DSP:
```

```
jnb CLK_FLAG,DUM_CLK
```

```
jb RDY_CLK_DSP_FL,JNC2
```

```
RET
```

```
;main loop
```



```

JNC2:
    jb     NEW_CLK_FL,PUT_CLK
    RET
DUM_CLK:
    mov    a,TIMER3
    clr    c
    subb   a,TIMER3_ES
    jnc    JNC1
    cpl    a
    add    a,#1
    clr    c
JNC1:
    subb   a,#CLK_DSP_DLY
    jc     RET_SBR
PUT_CLK:
    call   CLK_DEV
    clr    CLR_DSP_FL
    clr    NEW_CLK_FL
    setb   CLK_FLAG
    ;boşalttıktan sonra flag 0'lanır
    setb   RDY_CLK_DSP_FL
RET_SBR:
    RET                                ;MAIN LOOP
CLK_DEV:
    call   CLEAR_DSP
    mov    r1,#TABCLK_S
C_GO1:
    cjne   R1,#TABCLK_E,CLK_YAZ
    call   BCK_SHIFT
    call   BOSLUK
    call   DAT_DSP
    RET

```



```

CLK_YAZ:
    call    PUT_ASC
    mov     a,#' ':
    call    PUT_LCD
    jmp     C_GO1
;-----
BOSLUK:
    mov     R7,#28
DEC_R7:
    djnz    R7,GNDR1
    RET
GNDR1:
    mov     a,#20h
    call    PUT_LCD
    jmp     DEC_R7
;-----
T_DSP:
    mov     R1,#TABDAT_S
D_GO1:
    cjne    R1,#(TABDAT_E+1),DATE_YAZ
    RET                                           ;main loop
DATE_YAZ:
    call    PUT_ASC
    mov     a,#'.'.
    call    PUT_LCD
    jmp     D_GO1
;-----
PUT_ASC:
    mov     a,@r1
    mov     B,#10
    ;HPC'den gelen hex bilgiyi BCD formatına donusturur
    div     ab
    add     a,#30h
    call    PUT_LCD

```



```

mov      a,b
add      a,#30h          ;ASCII karsiligi
call     PUT_LCD
inc      R1
RET

;*****
STORE_TIME:
mov      R7,KEY_CHR
cjne     R7,#1Bh,TM_ADJ
;ESC basılırsa routine'den çık
setb     X_EN_FL
call     TXD_UART        ;ESC'i HPC'ye bildir.
RET
TM_ADJ:
cjne     R7,#'A',DT_ADJ
clr      TM_DT_SL        ;SAAT degisikligi
call     CLEAR_DSP
mov      DPTR,#SA_DA
call     WRITE_LCD
setb     CLR_DSP_FL
clr      K_DEC_FL
RET
DT_ADJ:
cjne     R7,#'B',BK_SP
setb     TM_DT_SL        ;TARİH degisikligi
call     CLEAR_DSP
mov      DPTR,#GUN
call     WRITE_LCD
setb     CLR_DSP_FL
clr      K_DEC_FL
RET
;-----

```



```

BK_SP:
cjne      R7,#0BH,S_T1      ;BACKSPACE CHR.
;call TXD_UART
cpl       K_DEC_FL
mov       ECHO,KEY_CHR
call      QUICK_DSP
RET
S_T1:
cjne      R7,#' ',S_T2      ;SPACE BAR
mov       ECHO,KEY_CHR
call      QUICK_DSP
call      PUT_HEX
RET
S_T2:
cjne      R7,#0Dh,S_T3
call      PUT_HEX
jnb       TM_DT_SL,S_T4
mov       KEY_CHR,#X_DAT_HEAD ;DATE header '1' gonder
jmp       EXIT_ADJ
S_T4:
;CLK header '1' gonder
mov       KEY_CHR,#X_CLK_HEAD
EXIT_ADJ:
setb      X_EN_FL           ;Xon
call      DELAY_L
call      TXD_UART
clr       CLR_DSP_FL
clr       K_DEC_FL
JMP       AUTO_CLK
RET
;-----
S_T3:
jb        K_DEC_FL,EX_ST3
mov       KEY_DEC_ES,KEY_DEC

```


EX_ST3:

```

cpl      K_DEC_FL
mov      ECHO,KEY_CHR
call     QUICK_DSP
RET

```

PUT_HEX:

```

mov      A,KEY_DEC_ES;BCD      sayi      KEY_DEC_ES'dedir.
mov      b,#0AH
;bu bölüm BCD sayıyı hex'e çevirir
mul      ab
;a'da bölüm ,b'de kalan değer bulunur.
add      a,KEY_DEC
mov      KEY_CHR,a
call     TXD_UART
RET

```

```

;*****
;***** KLAVYE CONTROL *****
;*****

```

KEY_SCAN:

```

mov      DPTR,#KLAV_ADR      ;tuş okuma routine'i
mov      P1,SATIR            ;P1'e yaz
movx     A,@DPTR              ;data bus'tan acc 'le oku
nop
nop
nop
nop
nop
nop
nop
nop
nop
movx     A,@DPTR              ;data bus'tan acc 'le oku
mov      SUTUN,A              ;sutunu oku
cjne     A,#OFFH,WHICH_COL     ;tuşa basıldı mı?
jb       KEY_VAL,LOW_HIGH      ;low'dan high'a geçiş var

```



```

mov          A,SATIR
;satırı sola doğru kaydırma rutini
RLA
mov          SATIR,A
RET                                     ;return main loop
;-----

```

```

WHICH_COL:

```

```

jnb          ACC.0,LB1
jnb          ACC.1,LB2
jnb          ACC.2,LB3
jnb          ACC.3,LB4
jnb          ACC.4,LB5
jnb          ACC.5,LB6
jnb          ACC.6,LB7
jnb          ACC.7,LB8

```

```

LB1:

```

```

mov          SUTUN,#0FEH
jmp          DECISION

```

```

LB2:

```

```

mov          SUTUN,#0FDH
jmp          DECISION

```

```

LB3:

```

```

mov          SUTUN,#0FBH
jmp          DECISION

```

```

LB4:

```

```

mov          SUTUN,#0F7H
jmp          DECISION

```

```

LB5:

```

```

mov          SUTUN,#0EFH
jmp          DECISION

```

```

LB6:

```

```

mov          SUTUN,#0DFH
jmp          DECISION

```



```

LB7:
mov     HIGH:      SUTUN,#0BFH      ;transition low to high
jmp     DECISION
LB8:
mov     SUTUN,#07FH
jmp     DECISION
;-----
DECISION:
mov     a,SUTUN
cjne    A,SUTUN_ES,DIFF_KEY
jb      KEY_VAL,RETN      ;tuşa basılı hal devam ediyor
;karar verilmemiş yeni bir tuş
clr     C
mov     A,TIMER
subb    A,TIMER_ES
jnc     DEB1
cpl     KEY:      A
add     A,#1
DEB1:
clr     C
subb    A,#DEB_DEL
jc      RETURN          ;A< DEBOUNCE DELAY'SA RETURN
setb    KEY_VAL        ;tuşa basıldı kararı
call    TRANS_KEY
RET     KEY_VAL:      ;main loop
;-----
RETN:
mov     TIMER_ES,TIMER
RET     ;main loop
;-----
DIFF_KEY:
;farkli yeni bir tus
mov     TIMER_ES,TIMER
mov     SUTUN_ES,SUTUN
RET     ;main loop

```



```

;-----
LOW_HIGH:                                ;transition low to high
clr                                     C
mov                                    A,TIMER
subb                                  A,TIMER_ES
jnc                                   DEB2
cpl                                    A
add                                   A,#1
DEB2:
clr                                    C
subb                                  A,#DEB_DEL
jc                                   RETURN      ;AK DEBOUNCE DELAY'SA RETURN
clr                                    KEY_VAL
mov                                   SUTUN_ES,#OFFH
RETURN:                               RET          ;main loop
;-----

```

```

TRANS_KEY:
mov                                    A,SATIR
;8*ri+ci+1   formülüne göre tuşun konumu belirlenir
call                                   KEY_MAP_R
mov                                    B,#8
mul                                    AB
mov                                    B,A          ;B=8*ri
mov                                    A,SUTUN
call                                   KEY_MAP_C
add                                    A,B          ;A=8*ri+ci
add                                    A,#1         ;A=8*ri+ci+1
cjne                                   A,#55,GO_TAB
jnb                                    RDY_CLK_DSP_FL,T_K1  ;display'de saat varsa
clr                                    X_EN_FL        ;Xoff
RET
T_K1:
call                                   PUT_CLK        ;saati hemen goster
RET

```


GO_TAB:

```
call      KEY_TRN_TAB
jb        X_EN_FL,YOLLA      ;Xon
call      STORE_TIME
RET
```

YOLLA:

```
call      TXD_UART      ;basılan tuş HPC'ye
RET      KEY_CHR,#0      ;return update routine
;CHEX KLAVYE) KEY_DEC,#0
```

KEY_TRN_TAB:

```
cjne      A,#35,ADR1
mov       KEY_CHR,#'0'      ;0
mov       KEY_DEC,#0
RET      KEY_DEC,#0
```

ADR1:

```
cjne      A,#36,ADR2
mov       KEY_CHR,#'1'      ;1
mov       KEY_DEC,#1
RET      KEY_DEC,#0
```

ADR2:

```
cjne      A,#37,ADR3
mov       KEY_CHR,#'2'      ;2
mov       KEY_DEC,#2
RET      KEY_DEC,#0
```

ADR3:

```
cjne      A,#38,ADR4
mov       KEY_CHR,#'3'      ;3
mov       KEY_DEC,#3
RET
```

ADR4:

```
cjne      A,#43,ADR5
mov       KEY_CHR,#'4'      ;4
mov       KEY_DEC,#4
RET
```



```

ADR5:
  cjne      A,#44,ADR6
  mov       KEY_CHR,#'5'           ;5
  mov       KEY_DEC,#5
  RET
ADR6:
  cjne      A,#45,ADR7
  mov       KEY_CHR,#'6'           ;6
  mov       KEY_DEC,#06
  RET
ADR7:
  cjne      A,#46,ADR8
  mov       KEY_CHR,#'7'           ;7
  mov       KEY_DEC,#07
  RET
ADR8:
  cjne      A,#51,ADR9
  mov       KEY_CHR,#'8'           ;8
  mov       KEY_DEC,#08
  RET
ADR9:
  cjne      A,#52,ADR10
  mov       KEY_CHR,#'9'           ;9
  mov       KEY_DEC,#09
  RET
ADR10:
  cjne      A,#53,ADR11
  mov       KEY_CHR,#'A'           ;A
  RET
ADR11:
  cjne      A,#54,ADR12
  mov       KEY_CHR,#'B'           ;B
  RET

```


ADR12:

```

cjne      A,#59,ADR13
mov       KEY_CHR,#'C'      ;C
RET

```

ADR13:

```

cjne      A,#60,ADR14
mov       KEY_CHR,#'D' ;D
RET

```

ADR14:

```

cjne      A,#61,ADR15
mov       KEY_CHR,#'E'      ;E
RET

```

ADR15:

```

cjne      A,#62,ADR16
mov       KEY_CHR,#'F'      ;F
RET

```

;

ADR16:

```

cjne      A,#47,ADR17
mov       KEY_CHR,#'W'      ;W
RET

```

ADR17:

```

cjne      A,#48,ADR18
mov       KEY_CHR,#'R'      ;R
RET

```

ADR18:

```

cjne      A,#64,ADR19
mov       KEY_CHR,#0DH      ;CR carriage return
RET

```

ADR19:

```

cjne      A,#56,ADR20
mov       KEY_CHR,#0BH      ;BS back space
RET

```



```

ADR20:
  cjne      A,#63,ADR21
  mov       KEY_CHR,#20H          ;SB space bar
  RET
ADR21:
  cjne      A,#39,ADR22
  mov       KEY_CHR,#1BH          ;ESC (CLS)
  RET
ADR22:
  cjne      A,#55,ADR23
  mov       KEY_CHR,#'T'          ;TIME request
  RET
ADR23:
  cjne      A,#40,ADR24
  mov       KEY_CHR,#'P'          ;P bekletme park etme
  RET
;***** dahili ve harici hatların tablosu *****
ADR24:
  cjne      A,#1,ADR25
  mov       KEY_CHR,#33           ;H1
  setb      KEY_CHR.7
  RET
ADR25:
  cjne      A,#2,ADR26
  mov       KEY_CHR,#34           ;H2
  setb      KEY_CHR.7
  RET
ADR26:
  cjne      A,#3,ADR28
  mov       KEY_CHR,#1           ;D1
  setb      KEY_CHR.7
  RET

```



```
ADR28:
  cjne      A,#4,ADR29
  mov       KEY_CHR,#2                ;D2
  setb      KEY_CHR.7
  RET

ADR29:
  cjne      A,#5,ADR30
  mov       KEY_CHR,#3                ;D3
  setb      KEY_CHR.7
  RET

ADR30:
  cjne      A,#6,ADR31
  mov       KEY_CHR,#4
  setb      KEY_CHR.7
  RET

ADR31:
  cjne      A,#7,ADR32
  mov       KEY_CHR,#5
  setb      KEY_CHR.7
  RET

ADR32:
  cjne      A,#8,ADR33
  mov       KEY_CHR,#6
  setb      KEY_CHR.7
  RET

ADR33:
  cjne      A,#9,ADR34
  mov       KEY_CHR,#35;H3
  setb      KEY_CHR.7
  RET
```


ADR34:

```
cjne      A,#57,ADR35
mov       KEY_CHR,#31           ;D31
setb     KEY_CHR.7
RET
```

ADR35:

```
cjne      A,#10,ADR36
mov       KEY_CHR,#36           ;H4
setb     KEY_CHR.7
RET
```

ADR36:

```
cjne      A,#11,ADR37
mov       KEY_CHR,#9
setb     KEY_CHR.7
RET
```

ADR37:

```
cjne      A,#12,ADR38
mov       KEY_CHR,#10
setb     KEY_CHR.7
RET
```

ADR38:

```
cjne      A,#13,ADR39
mov       KEY_CHR,#11
setb     KEY_CHR.7
RET
```

ADR39:

```
cjne      A,#14,ADR40
mov       KEY_CHR,#12
setb     KEY_CHR.7
RET
```


ADR40:

```
cjne    A,#15,ADR41
mov     KEY_CHR,#13
setb    KEY_CHR.7
RET
```

ADR41:

```
cjne    A,#16,ADR42
mov     KEY_CHR,#14
setb    KEY_CHR.7
RET
```

ADR42:

```
cjne    A,#17,ADR43
mov     KEY_CHR,#37           ;H5
setb    KEY_CHR.7
RET
```

ADR43:

```
cjne    A,#18,ADR44
mov     KEY_CHR,#38           ;H6
setb    KEY_CHR.7
RET
```

ADR44:

```
cjne    A,#19,ADR45
mov     KEY_CHR,#17           ;D17
setb    KEY_CHR.7
RET
```

ADR45:

```
cjne    A,#20,ADR46
mov     KEY_CHR,#18
setb    KEY_CHR.7
RET
```


ADR46:

cjne A,#21,ADR47

mov KEY_CHR,#19

setb KEY_CHR.7

RET

ADR47:

cjne A,#22,ADR48

mov KEY_CHR,#20

setb KEY_CHR.7

RET

ADR48:

cjne A,#23,ADR49

mov KEY_CHR,#21

setb KEY_CHR.7

RET

ADR49:

cjne A,#24,ADR50

mov KEY_CHR,#22

setb KEY_CHR.7

RET

ADR50:

cjne A,#25,ADR51

mov KEY_CHR,#39 ;H7

setb KEY_CHR.7

RET

ADR51:

cjne A,#58,ADR52

mov KEY_CHR,#32 ;D32

setb KEY_CHR.7

RET

ADR52:

cjne A,#26,ADR53

mov KEY_CHR,#40

;H8

setb KEY_CHR.7

RET

ADR53:

cjne A,#27,ADR54

mov KEY_CHR,#25

setb KEY_CHR.7

RET

ADR54:

cjne A,#28,ADR55

mov KEY_CHR,#26

setb KEY_CHR.7

RET

ADR55:

cjne A,#29,ADR56

mov KEY_CHR,#27

setb KEY_CHR.7

RET

ADR56:

cjne A,#30,ADR57

mov KEY_CHR,#28

setb KEY_CHR.7

RET

ADR57:

cjne A,#31,ADR58

mov KEY_CHR,#29

setb KEY_CHR.7

RET

ADR58:

cjne A,#32,ADR59

mov KEY_CHR,#30

setb KEY_CHR.7

ADR58:

cjne A,#32,ADR59

mov KEY_CHR,#30

setb KEY_CHR.7

RET

ADR59:

cjne A,#33,ADR60

mov KEY_CHR,#7 ;D7

setb KEY_CHR.7

RET

ADR60:

cjne A,#34,ADR61

mov KEY_CHR,#8

setb KEY_CHR.7

RET

ADR61:

cjne A,#41,ADR62

mov KEY_CHR,#15

setb KEY_CHR.7

RET

ADR62:

cjne A,#42,ADR63

mov KEY_CHR,#16

setb KEY_CHR.7

RET

ADR63:

cjne A,#49,ADR64

mov KEY_CHR,#23

setb KEY_CHR.7

RET

ADR64:

cjne A,#50,ADR65

mov KEY_CHR,#24

setb KEY_CHR.7


```

ADR65:
RET
KEY_MAP_R:                                ;SATIR SIRASI
cjne      A,#0FEH,KM1
mov        A,#00                          ;0.satir
RET
KM1:
cjne      A,#0FDH,KM2
mov        A,#01;1.satir
RET
KM2:
cjne      A,#0FBH,KM3
mov        A,#02;2.satir
RET
KM3:
cjne      A,#0F7H,KM4
mov        A,#03;3.satir
RET
KM4:
cjne      A,#0EFH,KM5
mov        A,#04;4.satir
RET
KM5:
cjne      A,#0DFH,KM6
mov        A,#05;5.satir
RET
KM6:
cjne      A,#0BFH,KM7
mov        A,#06;6.satir
RET
KM7:
cjne      A,#07FH,KM8
mov        A,#07;7.satir
KM8:      RET

```



```

KEY_MAP_C:                                ;SUTUN SIRASI
cjne      A,#0FEH,KMA1
mov       A,#0                               ;07
RET
KMA1:
cjne      A,#0FDH,KMA2
mov       A,#1                               ;06
RET
KMA2:
cjne      A,#0FBH,KMA3
mov       A,#2                               ;05
RET
KMA3:
cjne      A,#0F7H,KMA4
mov       A,#3                               ;04
RET
KMA4:
cjne      A,#0EFH,KMA5
mov       A,#4                               ;03
RET
KMA5:
cjne      A,#0DFH,KMA6
mov       A,#5                               ;02
RET
KMA6:
cjne      A,#0BFH,KMA7
mov       A,#6                               ;01
RET
KMA7:
cjne      A,#07FH,KMA8
mov       A,#7                               ;00
KMA8:
RET

```



```
***** LED CONTROL *****
```

```
***** SUBSCRIBER REGISTER FORM *****
```

```
***** ISS1|SS0|LS|DT|TIM3|TIM2|TIM1|TIM0 *****
```

```
LED_CON:
```

```
mov RO,#low TABSUB_S
```

```
LED_AGAIN:
```

```
mov OUT: A,@RO
```

```
anl A,#11000000B
```

```
cjne A,#00,L_RING
```

```
jmp NO_CHG ;idle state
```

```
L_RING:
```

```
cjne A,#01000000B,L_BEK
```

```
mov A,@RO
```

```
anl A,#0FH
```

```
cjne A,TIMER2,CHG
```

```
jmp CHG: LED_OUT
```

```
CHG:
```

```
mov A,@RO
```

```
anl A,#0F0H
```

```
add A,TIMER2
```

```
cpl ACC.5 ;LS
```

```
mov @RO,A
```

```
jmp LED_OUT
```

```
L_BEK:
```

```
cjne A,#10000000B,L_ON
```

```
mov A,@RO ;bekletme state'i
```

```
anl A,#0FH
```

```
cjne A,TIMER4,CHG_BEK
```

```
jmp LED_OUT
```

```
CHG_BEK:
```

```
mov A,@RO
```

```
anl A,#0F0H
```

```
add A,TIMER4
```

```
cpl ACC.5 ;LS
```



```

mov     @R0,A
jmp     LED_OUT
L_ON:
mov     A,@R0
setb    ACC.5
mov     @R0,A
LED_OUT:
mov     A,@R0
jnb     ACC.5,NO_CHG
mov     A,R0
call    LED_TRN_TAB
mov     DPTR,#CLR_LINE
movx    @DPTR,A
mov     DPTR,#LED_ADR           ;CS3
movx    @DPTR,A               ;yeni degerin atilmasi
NO_CHG:
mov     R7,#40
djnz    R7,$
mov     DPTR,#CLR_LINE ;(CS6) oncedki cıkışı sıfırlar
movx    @DPTR,A
cjne    R0,#low TABSUB_E,LED_AGA
RET
LED_AGA:
inc     R0
jmp     LED_AGAIN
LED_TRN_TAB:
mov     R3,#20
clr     C
subb    A,#TABSUB_S
push    acc
subb    a,#20
jnc     GREAT_20
pop     acc
call    LDO

```



```
djnz      R3,$
```

```
RET      ;#03,LD6
```

```
GREAT_20: ;#20H
```

```
pop      acc
```

```
call     LD21
```

```
djnz      R3,$
```

```
RET      ;#06,LD7
```

```
;-----;INC|C2|C1|C0|INC|R2|R1|R0|-----
```

```
LD0:     R3
```

```
cjne     a,#00,LD1           ;D1
```

```
mov      a,#20H             ;02
```

```
dec      R3 ;#07,LD8
```

```
RET      ;#21H
```

```
LD1:     R3
```

```
cjne     a,#01,LD2           ;D2
```

```
mov      a,#30H
```

```
dec      R3 ;#09,LD9
```

```
RET      ;#22H
```

```
LD2:     R3
```

```
cjne     a,#02,LD3           ;D3
```

```
mov      a,#40H
```

```
dec      R3 ;#11,LD10
```

```
RET      ;#23H
```

```
LD3:     R3
```

```
cjne     a,#03,LD4           ;D4
```

```
mov      a,#50H
```

```
dec      R3 ;#13,LD11
```

```
RET      ;#24H
```

```
LD4:     R3
```

```
cjne     a,#04,LD5           ;D5
```

```
mov      a,#60H
```

```
dec      R3 ;#15,LD12
```

```
RET      ;#25H
```



```

LD5:
  cjne    a,#05,LD6
  mov     a,#70H
  dec     R3
  RET
LD6:
  cjne    a,#06,LD7
  mov     a,#04H
  dec     R3
  RET
LD7:
  cjne    a,#07,LD8
  mov     a,#14H
  dec     R3
  RET
LD8:
  cjne    a,#08,LD9
  mov     a,#21H
  dec     R3
  RET
LD9:
  cjne    a,#09,LD10          ;D10
  mov     a,#31H
  dec     R3
  RET
LD10:
  cjne    a,#10,LD11
  mov     a,#41H
  dec     R3
  RET
LD11:
  cjne    a,#11,LD12
  mov     a,#51H
  dec     R3

```



```

RET
LD12:
cjne      a,#12,LD13
mov       a,#61H
dec       R3
RET
LD13:
cjne      a,#13,LD14
mov       a,#71H
dec       R3
RET
LD14:
cjne      a,#14,LD15
mov       a,#05H
dec       R3
RET
LD15:
cjne      a,#15,LD16          ;D16
mov       a,#15H
dec       R3
RET
LD16:
cjne      a,#16,LD17
mov       a,#22H
dec       R3
RET
LD17:
cjne      a,#17,LD18
mov       a,#32H
dec       R3
RET
LD18:
cjne      a,#18,LD19
mov       a,#42H

```



```

dec      R3
RET
LD19:
cjne     a,#19,LD20
mov      a,#52H
dec      R3
RET
LD20:
cjne     a,#20,LD21
mov      a,#62H
dec      R3
RET
LD21:
cjne     a,#21,LD22           ;D22
mov      a,#72H
dec      R3
RET
LD22:
cjne     a,#22,LD23
mov      a,#06H
dec      R3
RET
LD23:
cjne     a,#23,LD24
mov      a,#16H
dec      R3
RET
LD24:
cjne     a,#24,LD26           ;D25
mov      a,#23H
dec      R3
RET
LD26:
cjne     a,#25,LD27

```



```

mov      a,#33H
dec      R3
RET
LD27:
cjne     a,#26,LD28
mov      a,#43H
dec      R3
RET
LD28:
cjne     a,#27,LD29
mov      a,#53H
dec      R3
RET
LD29:
cjne     a,#28,LD30
mov      a,#63H
dec      R3
RET
LD30:
cjne     a,#29,LD31
mov      a,#73H
dec      R3
RET
LD31:
cjne     a,#30,LD32
mov      a,#07H
dec      R3
RET
LD32:
cjne     a,#31,LD33           ;D32
mov      a,#17H
dec      R3
RET

```


-----harici hat ledleri-----

LD33:

```
cjne      a,#32,LD34      ;H1
mov       a,#00
dec       R3
RET
```

LD34:

```
cjne      a,#33,LD35      ;H2
mov       a,#10H
dec       R3
RET
```

LD35:

```
cjne      a,#34,LD36      ;H3
mov       a,#01H
dec       R3
RET
```

LD36:

```
cjne      a,#35,LD37      ;H4
mov       a,#11H
dec       R3
RET
```

LD37:

```
cjne      a,#36,LD38      ;H5
mov       a,#02H
dec       R3
RET
```

LD38:

```
cjne      a,#37,LD39      ;H6
mov       a,#12H
dec       R3
RET
```

LD39:

```
cjne      a,#38,LD40      ;H7
mov       a,#03H
```



```
dec      R3
```

```
RET
```

```
LD40:
```

```
cjne    a,#39,LD41      ;H8
```

```
mov      a,#13H
```

```
dec      R3
```

```
LD41:    RET
```

```
END
```

Devrenin kaplayacağı alan mümkün olduğunca küçük tutulmuş ve bu amaçla kontrol lojik kısımlar tek bir entegre olarak tasarlanmıştır. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir.

Entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir.

Entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir.

Entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir. Bu şekilde entegrelerin bir kısmı, bir kısmı ise diğer entegrelerin üzerine monte edilmiştir.

6. TARTIŞMA VE SONUÇ

6.1. DONANIM DEĞERLENDİRMESİ

Devremiz hardware olarak oldukça yoğun bir şekilde tasarlanmıştır. Mikrodenetleyici hemen hemen tüm portlarıyla kullanılmış, açma anından itibaren sürekli olarak çalışma durumunda tutulmuştur.

Devrenin kaplıyacağı alan mümkün olduğunca küçük tutulmuş ve bu amaçla kontrol lojik kısmı tek bir entegre olarak tanımlanmıştır. Pal tekniğiyle oluşturduğum bu yapı en az üç entegrenin yerini tutmuştur.

Mikrodenetleyici kullanmam, bir mikroişlemcinin ihtiyac duyacağı çevre birimleri ihtiyacını da ayrıca yer kaplamadan karşılamış oldu. Tuş ve led taramada kullandığım tampon ve tutma entegreleri 8032'nin portlarını herhangi bir akım zorlamasına karşı korumak ve değişik amaçla birkaç yerde kullanılan bir portun işlevlerini birbirinden ayırmak için kullanılmıştır.

Daha öncede belirttiğim gibi 8032 mikrodenetleyici entegresini yaygın bir kullanıma sahip olduğu, kolay ve ucuz temin edilebildiği için seçtim. Ayrıca 8 bitlik oluşu, temel bir yapıda kolay bir kullanım sağlıyordu. 16 veya 32 bitlik mikrodenetleyiciler kullanmak, böyle bir devre için hem fazla hemde lüks olurdu.

Devrenin yapabilirlikleri açısından bakarsak ve bir oparatris cihazı olarak düşünersek; bazı kontrol uçlarını kullanarak kaydırma (SHIFT) görevi görececek bir tuş daha ekleyip kontrol edilen tuş sayısını arttırabilirdik. Özel fonksiyon olarak iş yapan tuşları arttırabilir ve değişik fonksiyonlar da kullanabilirdik. Ayrıca devreye şu anda üzerinde olmayan bir ram entegresi eklersek santral kısmına yüklediğimiz bazı işlemleri yazılımın da yardımıyla bu devre üzerinde yapabilirdik.

6.2. YAZILIM DEĞERLENDİRMESİ

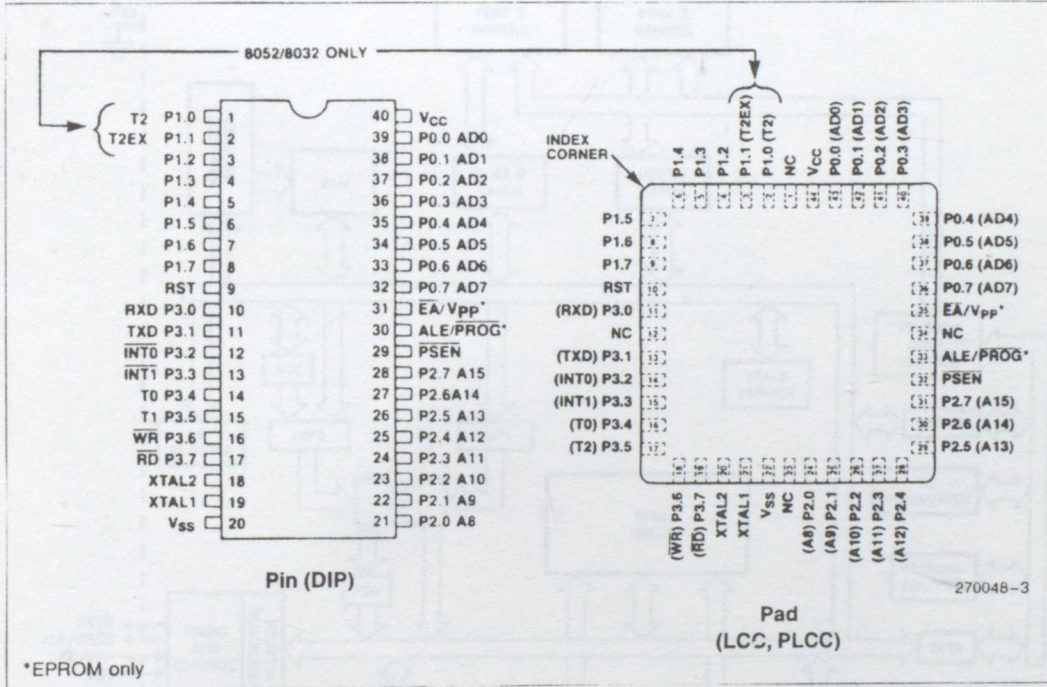
Yazdığım program 8032 yazılım dilinin tüm komutlarını kullanmayı amaçlamıştır. Program olarak mümkün olan en kısa mantıkla sonuca ulaşmaya çalıştım. Özellikle tuş tarama, basılan tuşu tespit etme ve değerlendirme işlemleri için değişik konfigürasyonlar kullanılabilir. Mikrodenetleyicinin 8 bit olması programlama açısından bir zorluk getirmemiştir. Daha ileri düzeyde bir denetleyici için, ki 16 veya 32 bitlik olabilir daha ileri düzeyde bir programlama bilgisi gerekecektir. Böyle bir devre açısından, kullandığımız elemanlar ve yazılım, hız olarak ve doğru değerlendirebilme açısından yeterlidir.

Özel fonksiyon tuşlarına verdiğimiz tanımları değişik ihtiyaçlara göre değiştirebiliriz. Donanım yapısını değerlendirme kısmında bahsettiğim gibi, bir ram entegresi kullanmış olsaydık, santral kısmına yüklediğimiz bazı fonksiyonları devremiz üzerinde gerçekleyebilirdik. Bunların arasında Gerçek zaman (REAL TIME) olarak çalışacak saat bilgisini sağlamak, abonelere ait bazı bilgileri kendi devremiz üzerinde saklayarak direkt ulaşma imkanı elde etmek gibi şeyleri sayabiliriz.

EK- KATALOG BİLGİLERİ

A. 8032 MİKRODENETLEYİCİ

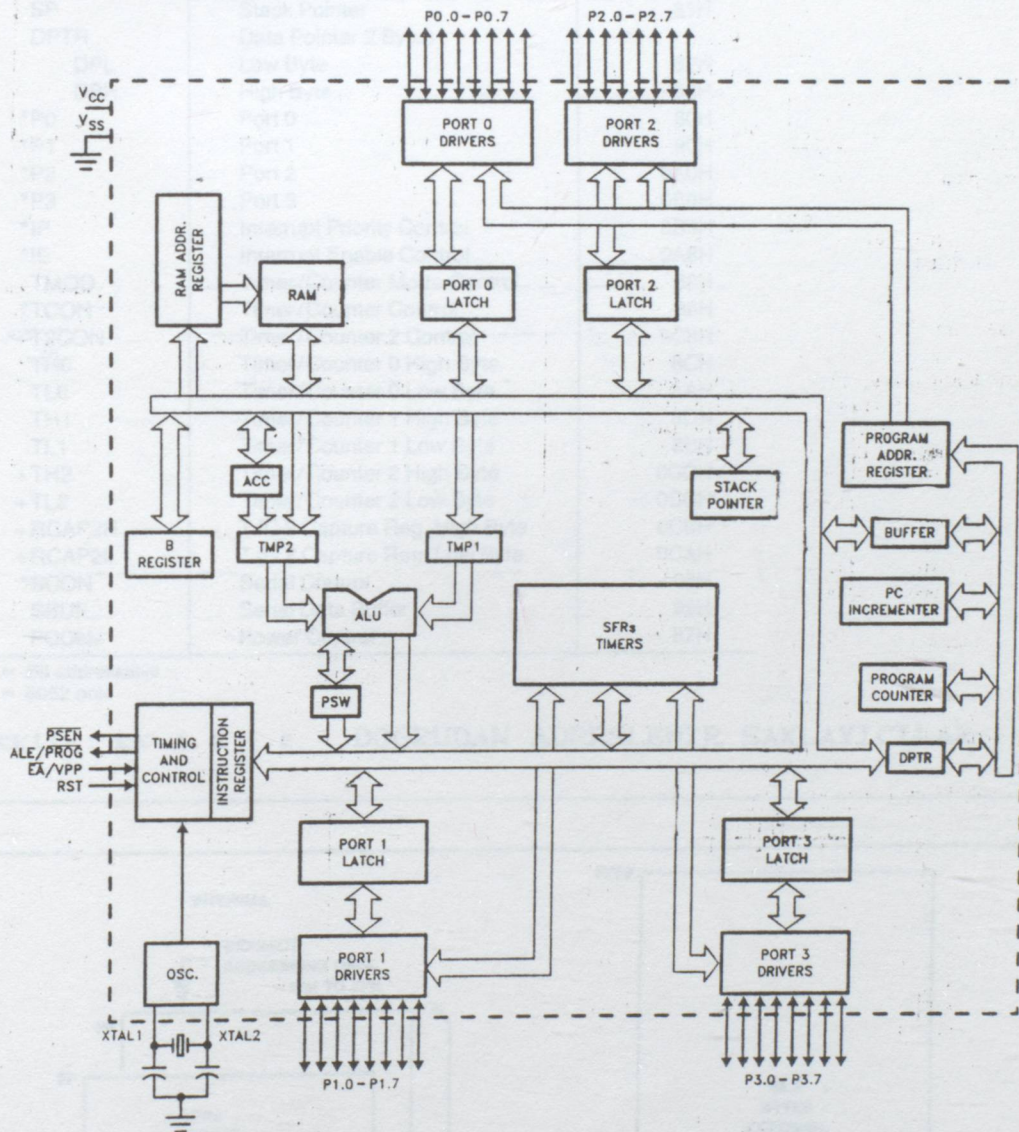
A.1. FİZİKSEL YAPISI



ŞEKİL : EK A.1 - 1

8032'NİN FİZİKSEL GÖRÜNÜMÜ VE
BACAK BAĞLANTILARI

A. 2. ELEKTRONİK YAPISI

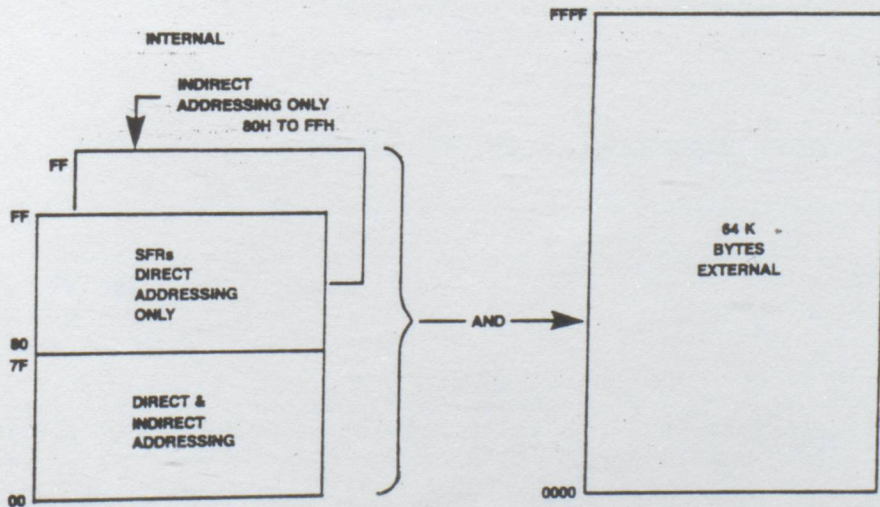


ŞEKİL : EK A.2 - 1 8032'NİN İÇ YAPISI

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

* = Bit addressable
+ = 8052 only

ŞEKİL : EK A.2 - 2 DOĞRUDAN ADRESLENİR SAKLAYICILAR

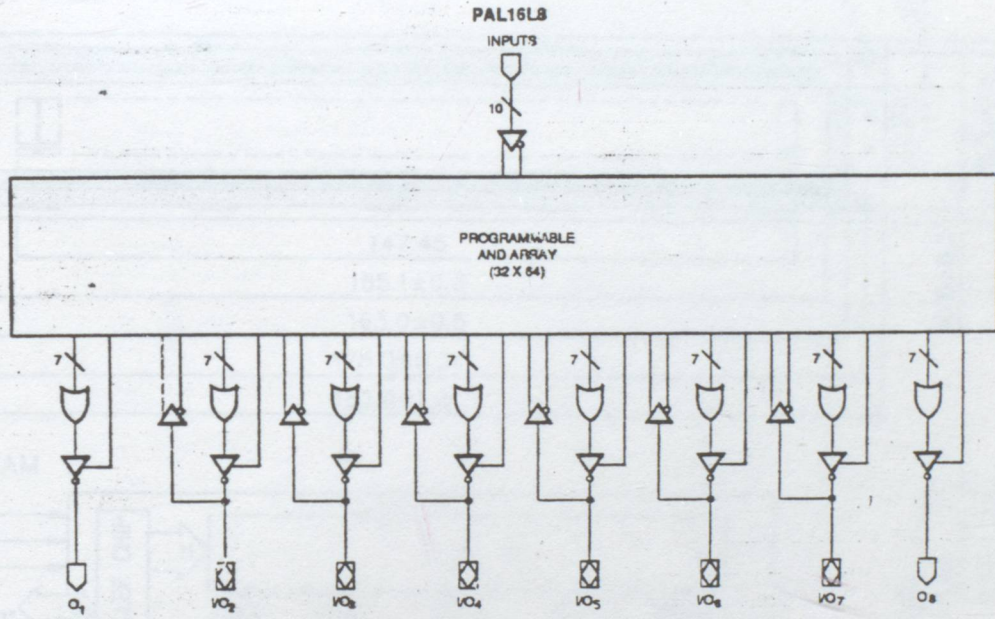


270249-4

ŞEKİL : EK A.2 - 3 8032*NIN MEMORY ORGANİZASYONU

B. PAL - 16 L 8 -

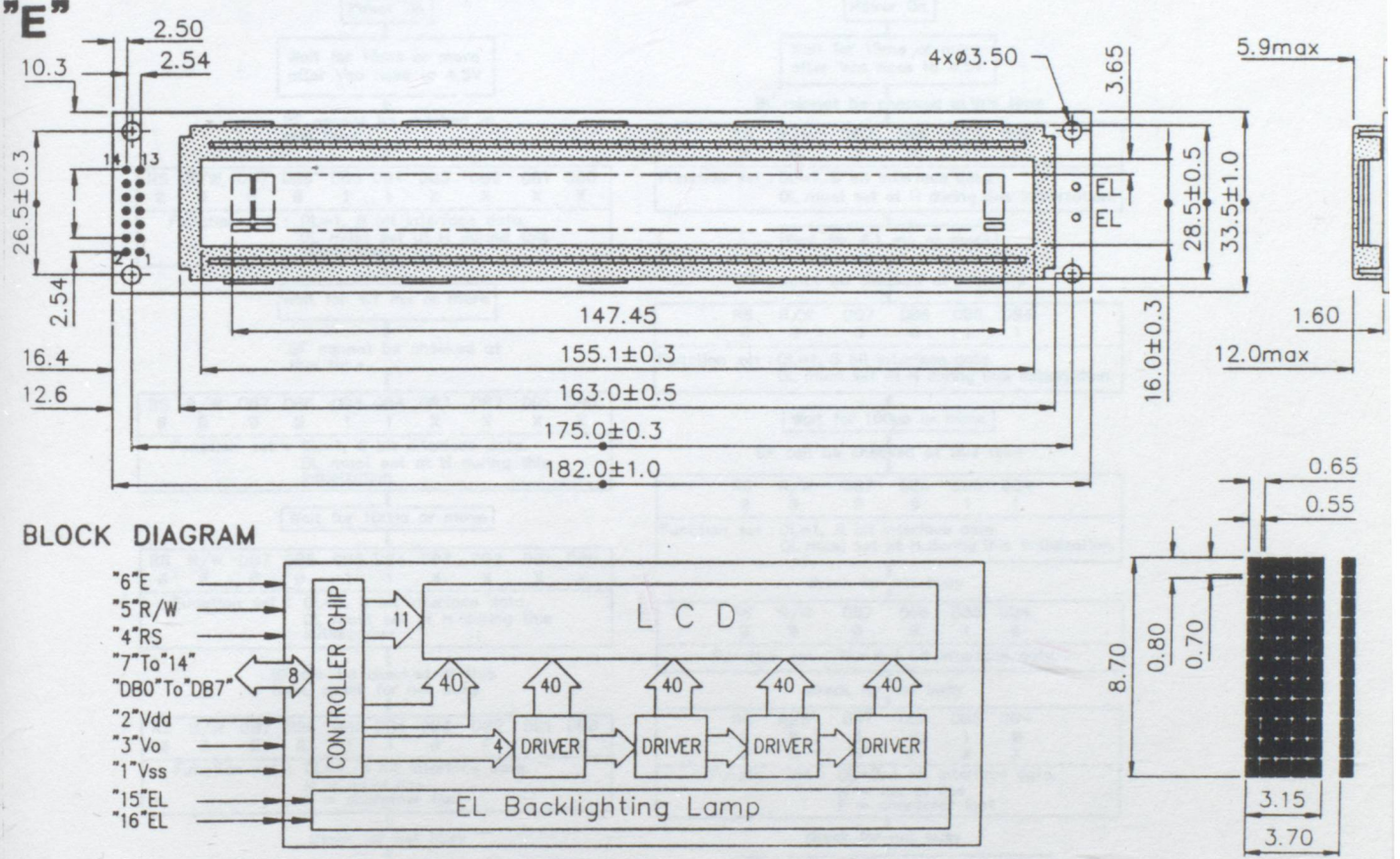
B.1. FİZİKSEL VE ELEKTRONİK YAPISI



ŞEKİL : EK B.1 - 1 FİZİKSEL VE ELEKTRONİK YAPISI

C. LCD NIN SARTLANDIRILMIŞI
C.1 FİZİKSEL VE ELEKTRONİK YAPISI

VK2140 (1x40 DMM, 1/11 MUX)



ŞEKİL : EK C.1 - 1 FİZİKSEL VE ELEKTRONİK YAPISI

1x40 DMM, 1/11 MUX

N=0 : 1-LINE DISPLAY

F=1 : 5X10 DOTS

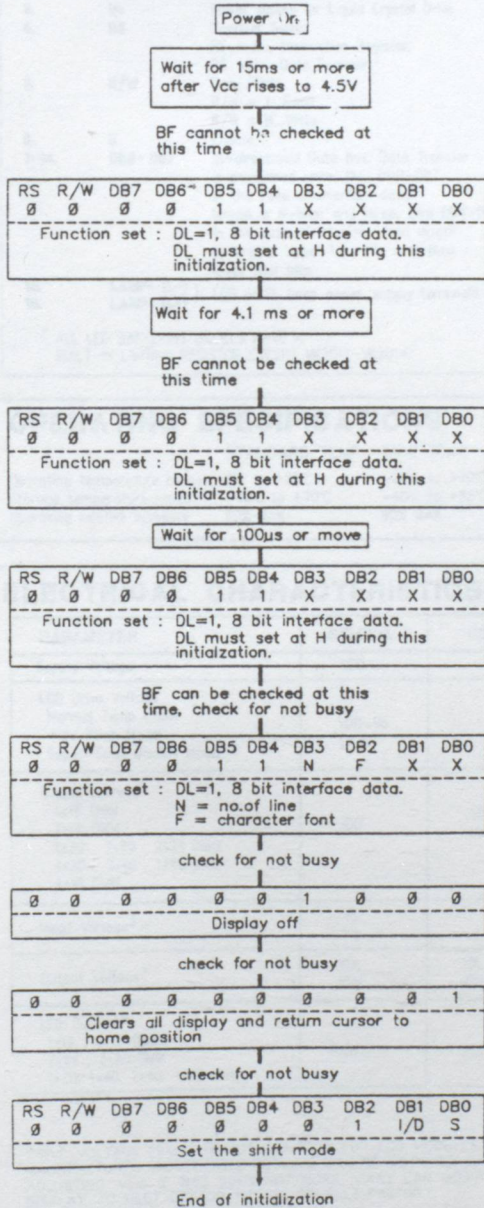
	1	2	3	4	5	6	7	8	9	10	11	12	13		33	34	35	36	37	38	39	40	← DISPLAY POSITI
FIRST LINE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	---	20	21	22	23	24	25	26	27	← DD RAM ADDRE

ŞEKİL : EK C.1 - 2 KARAKTER POZİSYONLARI & DD RAM ADRES

C.2 LCD'NİN ŞARTLANDIRILMASI

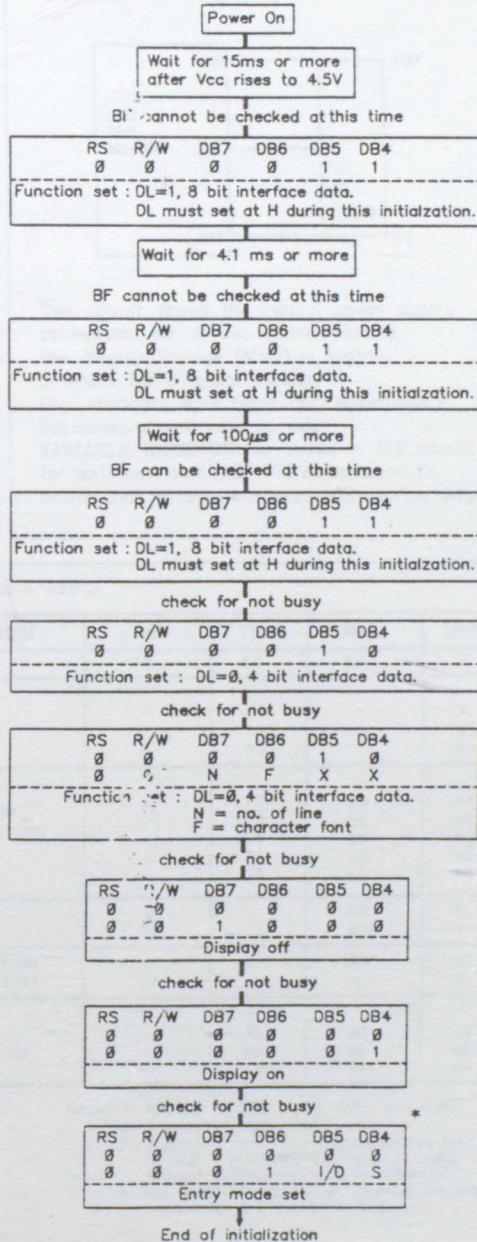
INITIALIZATION

For 8 bit data interfacing



* NOTE : IN NORMAL OPERATION, SET S TO 0

For 4 bit data interfacing



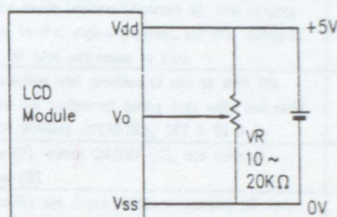
DEFINITION OF TERMINALS

PIN NO.	SYMBOL	FUNCTION
1.	V _{ee}	Ground terminal of module.
2.	V _{dd}	Supply terminal of module, +5V.
3.	V _o	Power supply for Liquid Crystal Drive
4.	RS	Register Select RS = 0... Instruction Register. RS = 1... Data Register.
5.	R/W	Read/Write R/W = 1... Read R/W = 0... Write
6.	E	Enable
7-14.	DB0 - DB7	Bi-directional Data Bus. Data Transfer is performed once, thru DB0-DB7, in the case of interface data length is 8-bits; and twice, thru DB4-DB7, in the case of interface data length is 4-bits. Upper four bits first then lower four bits.
15.	LAMP- (L ₋)	LED or EL lamp power supply terminals
16.	LAMP+ (L ₊)	

* ALL LED BACKLIGHT MODELS HAVE A BUILT-IN LIMITING RESISTOR EXCEPT VK2011-VK2014.

POWER SUPPLY REQUIREMENTS

- Wide Temperature Range Version
- Standard
- Super-Twist Display Version



This circuit shows the typical power supply connection for all dot matrix module. The display Voltage (V_{Lcd}) is slightly different for different version (eg. standard, wide temp and supertwist.) Recommend end user to use VARIABLE RESISTOR as shows in the circuit for optimum V_{Lcd} ($V_{dd}-V_{ee}$) adjustment to obtain best display contrast and viewing angle.

OPERATING SPECIFICATIONS

	STANDARD TEMP	WIDE TEMP
Operating temperature range	0°C to +50°C	-20°C to +70°C
Storage temperature range	-40°C to +70°C	-40°C to +85°C
Operating relative humidity	90% MAX	90% MAX

ELECTRICAL CHARACTERISTICS ($T_a = +25^\circ\text{C}$)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNIT
Supply Voltage	VDD		4.5	5.0	5.5	V
LCD Drive Voltage						
Normal Temp Model	$V_{DD}-V_o$ (V_{Lcd})		-	4.0*	-	V
Wide Temp Model			-	4.5*	-	V
Super-Twist Display Model			-	4.3*	-	V
Supply Current ¹						
1x16 DMM	IDD	VDD = 5V	-	1.0	2.0	mA
2x16 DMM		V _o = 0V MIN	-	1.0	3.0	mA
1x20, 2x20, 2x24 DMM			-	1.5	3.0	mA
4x20, 2x40, 1x40 DMM			-	2.5	4.0	mA
4x16 DMM			-	4.5	5.5	mA
Input Voltage ²	VIL		0	-	0.6	V
	VOH		2.7	-	VDD	V
Output Voltage ³	VOL	IOL = 1.6 mA	-	-	0.4	V
	VOH	IOH = 0.2 mA	2.4	-	-	V
LED Current ⁴						
1x16, 2x16 DMM	ILED	L+ - L- = 5V	-	40	60	mA
2x24, 2x20 DMM			-	60	80	mA
1x20, 1x40, 2x40, 4x20 DMM			-	150	250	mA

* DRIVE VOLTAGE (V_{Lcd}) IS NOT IDENTICAL FOR LCD MODULES. MANUFACTURES ACCEPTABLE RESULTS CAN BE OBTAINED BY ADJUSTING V_{Lcd} . IF THIS DOES NOT WORK, VIKAY CAN MODIFY DISPLAY TO MEET CUSTOM NEEDS. CONSULT FACTORY.

Note: 1. Applies to DB0 - DB7, RS and R/W
2. Applies to DB0 - DB7.
3. Supply current may slightly exceed MAX. Rating if SAMSUNG controller is used without pull-up resistor for DB0-DB7.
4. For VK2011-VK2014, an external limiting resistor of 6.8~10Ω is required.

INSTRUCTION SET

INSTRUCTION	CODE										DESCRIPTION	TYPICAL EXECUTION TIME	
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns the cursor to home position (Address 0). Sets I/D=1 of Entry Mode.	1.64 ms	
Return home	0	0	0	0	0	0	0	0	0	1	Return the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged. Set DD RAM addresses to zero.	1.64 ms	
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Set the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read of DD RAM/CG RAM. FOR NORMAL OPERATION, SET S TO 0	40 μ s	
Display ON/Off control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF all display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40 μ s	
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	●	●	Moves the cursor and shifts the display without changing DD RAM contents.	40 μ s	
Function set	0	0	0	0	1	DL	N	F	●	●	Sets interface data length (DL) number of display lines (N) and character font (F).	40 μ s	
Set the CG RAM address	0	0	0	1	MSB			ACG		LSB	Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μ s	
Set the DD RAM address	0	0	1	MSB			ADD		LSB	Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μ s		
Read busy flag & address	0	1	BF	MSB			AC		LSB	Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	40 μ s		
Write data to CG or DD RAM	1	0	MSB								LSB	Writes data into DD RAM or CG RAM.	40 μ s
Read data from CG or DD RAM	1	1	MSB								LSB	Reads data from DD RAM or CG RAM.	40 μ s
S = 1: Accompanies display shift when data is written, for normal operation, set to 0 I/D=1: Increment DL=1: 8 bits I/D=0: Decrement DL=0: 4 bits S/C=1: Display shift N=1: 2 (1) line S/C=0: Cursor move N=0: 1 line R/L=1: Shift to the right F=1: 5x10 dots R/L=0: Shift to the left F=0: 5x7 dots BF=1: Internally operating BF=0: Can accept instruction											DD RAM : Display data RAM CG RAM : Character generator RAM ACG : CG RAM address ADD : DD RAM address corresponds to cursor address AC : Address counter used for both DD and CG RAM address B : 1=ON 0=OFF (Blinking cursor) C : 1=ON 0=OFF (Cursor) D : 1=ON 0=OFF (Display) ● Don't Care		

INITIALIZATION

The module automatically performed initialization when powered on (using internal reset circuit). The following instructions are executed during initialization :-

1. CLEAR DISPLAY

The Busy Flag is kept in the Busy State (BF=1) until initialization ends. The time is 15 ms.

2. Function Set

DL = 1: 8-bits long interface data
N = 0: 1 Line display

3. DISPLAY ON/OFF CONTROL

D = 0: Display OFF

C = 0: Cursor OFF

B = 0: Blink OFF

4. ENTRY MODE SET

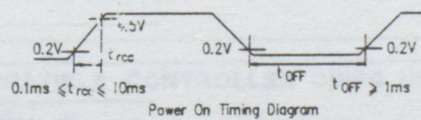
I/D = 1: +1 (INCREMENT)

S = 0: NO SHIFT

5. DD RAM IS SELECTED

Power On Initialization depends on rise time of the supply when it is turned on. The following time relationship must be satisfied.

ITEM	SYMBOL	STANDARD TIME		UNIT
		MIN	TYP	
Power Supply Rise Time	t_{RCC}	0.1	-	ms
Power Supply Off Time	t_{OFF}	1.0	-	ms



Power On Timing Diagram

NOTE :

When the above power supply condition is not satisfied, the internal reset circuitry does not operate correctly. In this case, perform the needed initialization by sending function set instructions thrice from MPU after turning the power on. For example, to designate a 8-bits data length, send the following instructions thrice.

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	●	●	●	●
0	0	0	0	1	1	●	●	●	●
0	0	0	0	1	1	●	●	●	●

When this ends, the module enters 8-bits data length mode without fail, then enter 4-bits data length instruction for 4-bits data length interface.

DOT CHARACTER PATTERNS

For 5x7 Dot Character Patterns

Character Codes (CG RAM Data)	CG RAM Address	Character Patterns (CG RAM Data)
7 6 5 4 3 2 1 0 Higher Lower	5 4 3 2 1 0 Higher Lower	7 6 5 4 3 2 1 0 Higher Lower
0 0 0 0 • 0 0 0	0 0 0	Character Pattern Example (1) Cursor Position
0 0 0 0 • 0 0 1	0 0 1	Character Pattern Example (2)
0 0 0 0 • 1 1 1	1 1 1	No Effect

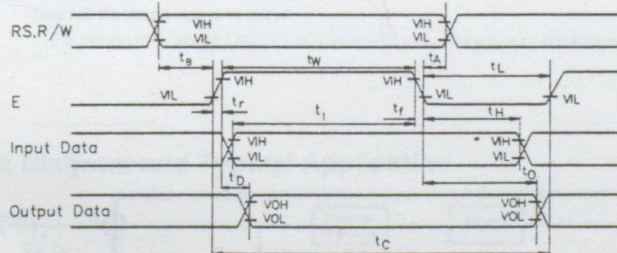
NOTE:
Character code bits 0,2
correspond to CG RAM
address bits 3,5
(3 bits : 8 types)

For 5x10 Dot Character Patterns

Character Codes (CG RAM Data)	CG RAM Address	Character Patterns (CG RAM Data)
7 6 5 4 3 2 1 0 Higher Lower	5 4 3 2 1 0 Higher Lower	7 6 5 4 3 2 1 0 Higher Lower
0 0 0 0 • 0 0 •	0 0 0 0	Character Pattern Example Cursor Position
0 0 0 0 • 1 1 •	1 1 1 1	No Effect

NOTE:
Character code bits 1,2
correspond to CG RAM
address bits 4,5
(2 bits : 4 types)

TIMING DIAGRAM



TIMING CHARACTERISTICS FOR ALL COMPATIBLE CONTROLLER CHIPS USED.

CONTROLLER CHIP		SAMSUNG KS0062	HITACHI HD44780	SANYO LC7985	EPSON SED1278	TOSHIBA T7934	OKI MSM6222	UNIT
PARAMETERS								
Enable Cycle Time	tC (min)	1000	1000	1000	500	500	667	nS
Enable Pulse Width								
High Level	tW (min)	450	450	450	220	220	280	nS
Low Level	tL (min)	450	450	450	220	220	280	nS
E Rise Time	tR (max)	25	25	25	25	20	25	nS
E Fall Time	tF (max)	25	25	25	25	20	25	nS
Set-up Time	tB (min)	140	140	140	40	40	140	nS
Data Set-up Time	tI (min)	195	195	195	60	60	180	nS
Data Delay Time	tD (max)	320	320	320	120	120	220	nS
Address Hold Time	tA (max)	10	10	25	10	10	10	nS
Read Time	tH (min)	10	10	75	10	10	10	nS
Input Data	tO (min)	20	20	20	20	20	20	nS
Output Data								

NOTE :

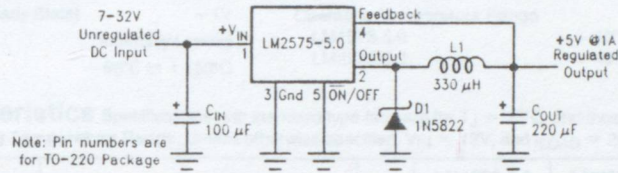
- INITIALIZATION BY POWER-ON RESET INVOLVES MANY UNSTABLE FACTORS CAUSED BY POWER SUPPLY FLUCTUATIONS. THEREFORE, INITIALISING BY INSTRUCTIONS IS STRONGLY RECOMMENDED.
- MODULE INITIALIZATION DOES NOT AFFECT BY USING HD44100, OR KS0061, OR LC7930, OR MSM5259, OR MSM5839, OR MSM5260 DRIVER CHIPS.

ŞEKİL : EK C.2 - 4 KARAKTER OLUŞTURMA VE
ZAMAN DİYAGRAMLARI

D. SMPS

D.1 LM 2575 ENTEGRESİNİN TEKNİK ÖZELLİKLERİ

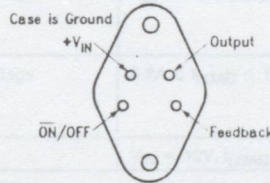
Typical Application



TL/H/10527-1

Connection Diagram and Order Information

4-Lead TO-3 (K)



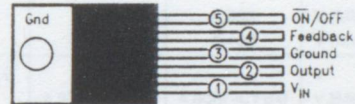
Bottom View

Order Number LM1575K-5.0, LM2575K-5.0
See NS Package Number K04A

For information about LM2575 in dual-in-line or surface mount packages, contact the factory.

TL/H/10527-2

5-Lead TO-220 (T)

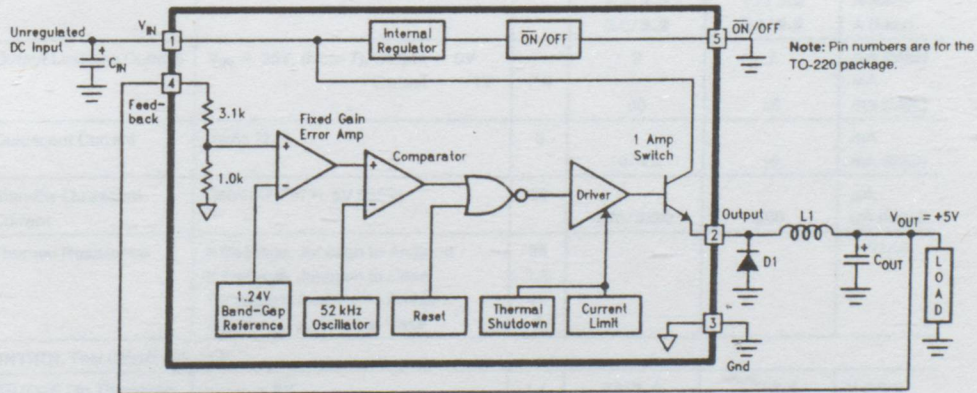


Top View

Order Number LM2575T-5.0
See NS Package Number T05A

TL/H/10527-3

Block Diagram and Typical Application



Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Supply Voltage (see Figure 5)	40V	Minimum ESD Rating (C = 100 pF, R = 1.5 k Ω)	2 kV
$\overline{\text{ON}}$ /OFF Pin Input Voltage	-1 $\leq$ V $\leq$ 15V	Lead Temperature (Soldering, 10 sec.)	260°C
Output Voltage to Ground (Steady State)	-1V	Maximum Junction Temperature	150°C
Power Dissipation	Internally Limited	Operating Temperature Range	
Storage Temperature Range	-65°C to +150°C	LM1575-5.0	-55°C $\leq$ T _J $\leq$ +150°C
		LM2575-5.0	-40°C $\leq$ T _J $\leq$ +125°C

Electrical Characteristics Specifications with standard type face are for T_J = 25°C, and those with boldface type apply over full Operating Temperature Range. Unless otherwise specified, V_{IN} = 12V, and I_{LOAD} = 200 mA.

Symbol	Parameter	Conditions	Typ	LM1575-5.0 Limit (Note 2)	LM2575-5.0 Limit (Note 3)	Units (Limits)
SYSTEM PARAMETERS (Note 4) Test Circuit Figure 1						
V _{OUT}	Output Voltage	V _{IN} = 12V, I _{LOAD} = 0.2A	5.0	4.950 5.050	4.900 5.100	V V (Min) V (Max)
V _{OUT}	Output Voltage	0.2A $\leq$ I _{LOAD} $\leq$ 1A, 8V $\leq$ V _{IN} $\leq$ 35V	5.0	4.850/ 4.800 5.150/ 5.200	4.800/ 4.750 5.200/ 5.250	V V (Min) V (Max)
η	Efficiency	V _{IN} = 12V, I _{LOAD} = 1A, V _{OUT} = 5V	82			%
DEVICE PARAMETERS						
f _O	Oscillator Frequency		52	47/ 43 58/ 62	47/ 42 58/ 63	kHz kHz (Min) kHz (Max)
V _{SAT}	Saturation Voltage	I _{OUT} = 1A (Note 5)	0.9	1.2/ 1.4	1.2/ 1.4	V V (Max)
DC	Max Duty Cycle (ON)	(Note 6)	98	93	93	% % (Min)
I _{CL}	Current Limit	Peak Current, t _{ON} $\leq$ 3 μ s (Note 5)	2.2	1.7/ 1.3 3.0/ 3.2	1.7/ 1.3 3.0/ 3.2	A A (Min) A (Max)
I _L	Output Leakage Current	V _{IN} = 35V, (Note 7), Output = 0V Output = -1V	7.5	2 30	2 30	mA (Max) mA mA (Max)
I _Q	Quiescent Current	(Note 7)	5	10/ 12	10	mA mA (Max)
I _{STBY}	Standby Quiescent Current	$\overline{\text{ON}}$ /OFF Pin = 5V (OFF)	50	200/ 500	200	μ A μ A (Max)
θ_{JA} θ_{JC} θ_{JA} θ_{JC}	Thermal Resistance	K Package, Junction to Ambient K Package, Junction to Case T Package, Junction to Ambient T Package, Junction to Case	35 1.5 40 2			°C/W
ON/OFF CONTROL Test Circuit Figure 1						
V _{IH} V _{IL}	$\overline{\text{ON}}$ /OFF Pin Threshold Voltage	V _{OUT} = 5V V _{OUT} = 0V	1.4 1.2	22/ 2.4 1.0/ 0.8	2.2/ 2.4 1.0/ 0.8	V (Min) V (Max)
I _{IH}	$\overline{\text{ON}}$ /OFF Pin Input Current	$\overline{\text{ON}}$ /OFF Pin = 5V (OFF)	12	30	30	μ A μ A (Max)
I _{IL}		$\overline{\text{ON}}$ /OFF Pin = 0V (ON)	0	10	10	μ A μ A (Max)

ŞEKİL : EK D.1 - 2 ELEKTRİKSEL KARAKTERİSTİKLERİ

KAYNAKLAR

1. INTEL 8 - BIT EMBEDDED CONTROLLERS 1990
2. INTEL EMBEDDED CONTROL APPLICATION 1988
3. HITACHI LCD DRIVER DATA BOOK 1987
4. NATIONAL GENERAL PURPOSE LINEAR DEVICES 1989
5. SGS-THOMSON HIGH SPEED CMOS DATABOOK 1988
6. DATA COMMUNICATIONS FOR MICROCOMPUTERS 1982
7. NATIONAL PROGRAMMABLE LOGIC DESIGN GUIDE 1986
8. HITACHI IC MEMORY PRODUCTS 1986
9. MOTOROLA LINEAR/SWITCHMODE VOLTAGE REGULATOR
HANDBOOK 1989

ÖZGEÇMİŞ

28 Temmuz 1966 Edirne ili Meriç ilçesi doğumluyum. İlk, Orta ve Lise öğrenimimi Uzunköprü'de tamamladım. 1983-84 eğitim, öğretim yılında Yıldız Üni. Elektronik ve Hab. Müh. bölümünü kazandım. Mühendislik öğrenimimi tamamlayıp, Master öğrenimi için 1988-89 eğitim, öğretim yılında Elektronik bilim dalına kayıt yaptırdım. Hazırlamış olduğum bu tezi bitirme tezim olarak sunacağım.

