YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MİKROİŞLEMCİ TEMELLİ, ASENKRON/SENKRON VERİ HATTINDA ÖLÇME GÖZLEMLEME VE VERİ SAKLAMA AMAÇLI DÜZENEK TASARIMI VE GERÇEKLENMESİ

Elektronik ve Haberleşme Mühendisi Aydın PARIN
F. B. E. Elektronik Mühendisliği Anabilim Dalı Elektronik Bölümünde
hazırlanan
YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Şefik SARIKAYALAR

İSTANBUL, 1994

İÇİNDEKİLER

KISALTMALAR:

ANSI	American National Standards Institute
ASCII	The American Standart Code for Information Interchange
BIT	BInary digiT (İkili Sayı Tabanı Birim Elemanı)
BYTE	'Sekiz Bit'lik Veri Birimi
CCITT	Consultative Committee on International Telephony and Telegraphy
IEEE	Institute of Electrical and Electronic Engineers
EIA	Electronic Industries Association
EPROM	Erasable Programmable Read Only Memory (Silinebilen -Özel Yöntemle-Yazılabilen Yalnızca Okunabilir Bellek)
ISO	The International Organization for Standardization
LCD	Liquid Crystal Display (Sıvı Kristal Gösterge)
RAM	Random Access Memory (Oku/Yaz Bellek)
UART	Universal Asynchronous Receiver Transmitter

TEŞEKKÜR

Bu tezin hazırlanmasında bana yardımcı olan Sayın Prof. Şefik SARIKAYALAR'a ve STFA SAVRONİK yetkililerine teşekkürlerimi bir borç bilirim.

Elektronik ve Haberleşme Mühendisi
Aydın PARIN

ÖZET

Bilgisayarın her alana hızla yayıldığı günümüzde, veri iletişimi de giderek sayısallaşmaktadır [4]. Sayısal verilerin iletilmesi için tesis edilmiş veri hatlarının sağlıklı çalıştığının ölçülmesi için de düzenekler geliştirilmiştir. Protokol çözümleyici (Protocol Analyzer) olarak adlandırılan bu düzenekler, veri hattındaki bilgi akışını ve durum/el sıkışma işaretlerini değişik şekillerde gözlememizi sağlar. İletişim protokollerini tanıdıkları için, örneğin bir bilgisayar ağı içerisinde oluşan hataların kaynaklarını ararken çok kullanışlı bir araçtır.

Ticari olarak çeşitli şekillerde bulunmaktadır. Fakat tutarları oldukça yüksektir. Eğer yalnızca dar bir çalışma alanında çözümleyici gereksiniminiz varsa, kullanmayacağınız işlevlere de ücret ödeyerek bir protokol çözümleyici almak parasal açıdan olumsuzdur.

Bu düşünceden hareketle, işlevleri kısıtlı, düşük maliyetli, küçük boyutlu bir protokol çözümleyici geliştirilmek istendi. Geliştirme aşamasında işlevlerinin çok kısıtlı olması nedeniyle adı protokol çözümleyici olarak değil, asenkron/senkron veri hattında ölçme,gözlemleme ve veri saklama düzeneği olarak verildi. Devre üzerindeki yazılım geliştirilerek ve/veya daha hızlı bir mikroişlemci kullanılarak, işlevleri protokol çözümleyiciye daha çok yaklaştırılabilir.

ABSTRACT

Nowadays data transmission is getting more digitized. Test and measurement equipments are developed to test operational characteristic of the data links that are established for digital data transmission.

These equipments that are called Protocol Analyzers provide us means to observe data flow and handshake signaling in data links. If we have to give an example, these equipment's provide us means to find transmission errors sources in computer networks, because they are able to distinguish between different transmission protocols.

Commercial types of these equipments are available which have different features. But they are prices are quite high. If you need to buy a protocol analyzer of which restricted features will be enough for you, most probably you will have to pay also the fee of features which you will not use.

Taking into consideration commercial demand which is mentioned in previous paragraph, a protocol analyzer with restricted features small-sized, and so low cost is aimed to be developed -in development phase. Because of its restricted features, it is found to be more appropriate to name it as Asynchronous/ Synchronous data link measurement, observation and data storage equipment. It is possible to add more features to it in order to change it to a protocol analyzer, either by upgrading its software or by using a fast and the powerful microprocessor or both of it.

1. GİRİŞ

1.1. Çalışmanın Tanıtımı

Günümüzde bilgi alış-verişinin hızla yaygınlaştığını görmekteyiz. 'Küreselleşme' olarak da adlandırılan bu gelişme ile birlikte, bilgi üreten, dönüştüren, gönderen, ileten, işleyen ve biriktiren altyapı birimlerinin de gelişmesi gerçekleşmiştir: Yeni işaret işleme yöntemleri, daha yoğun veri sıkıştırma yöntemleri, daha hızlı bilgisayarlar, yüksek sığalı manyetik kayıt ortamları, yüksek güvenlikli iletişim ortamları... ve doğal olarak bu birimlerin sağlıklı ve doğru çalıştıklarını ölçümlemeye yarayan yardımcı araçlar ve donanımlar. Aşağıda iletim hatlarının özel bir türünün çalışması ve bu hatlarda ölçümleme yöntemleri hakkında kısaca bilgi verilmiştir.

İletişim ortamlarında, asenkron veya senkron veri iletimi, yaygın olarak kullanılan bir yöntemdir. Bu yöntemde veriler BİT'lere indirgenir ve yardımcı işaretler kullanılarak iletilir. İşaret seviyeleri ve temel bileşenler için çeşitli yöntemler üretilmiş, bunlardan bazıları CCITT tarafından genel ölçü önerisi olarak değişik zamanlarda yayınlanmıştır. Özellikle bilgisayar dünyasında kullanılan RS232C/V.24 genel ölçü tanımı en bilinenidir. Devre tasarlanırken aynı heyetin yayınladığı V.24 öneri fasikülünün tanımlarının bir kısmından yararlanılmıştır. Bu konu ileri bölümlerde ayrıntılandırılacaktır.

Yukarıda sözü edilen veri hatlarında bilgi ve durum işaretlerini gözlemek istediğinizde bir kaç seçeneğiniz vardır: Led temelli basit gözlem araçları, gerilim-akım-direnç ölçen araçlar, daha ayrıntılı bilgiler edinebileceğiniz osiloskop gibi aletler, genel amaçlı gelişmiş mantıksal çözümleyiciler (Logic Analyzer) ve son olarak yalnızca veri hattı gözlemek amacıyla üretilmiş protokol çözümleyiciler (Protocol Analyzer). Burada 'protokol' bir iletişim şeklinin veri ve durum kuralları bütünü anlamındadır. Ledli gözlem araçlarının yararı, temel durum işaretlerini göstermekten ileri gitmez. Gerilim-akım-direnç ölçümü çoğu kez, diğer araçlar kullanılsa da gerekmektedir. Ancak işaretler zaman içinde değişken olduklarından bu araçlarda yetersizdir. Veri saklayabilen osiloskoplar biraz daha yardımcı olarak işaret zamanlarının ve özel durumlarda verinin bir kısmının sağlığı hakkında bilgi verebilir. Mantıksal çözümleyiciler de, bu konuda çoğunlukla osiloskoptan farksızdır. Protokol çözümleyiciler daha karmaşık yapılardır. Kullanılan hattı tanırlar. İletişim değişken değerlerini girerek ölçümlenen hat hakkında daha geniş bilgiler edinmek mümkündür.

Protokol çözümleyiciler, bir çok ticari marka olmasına karşın, kullanıcılara temel olarak iki ayrı şekilde sunulmuştur: Kişisel bilgisayar üzerine takılan bir kart ve aynı bilgisayarda koşan bir yazılım şeklinde veya tüm donanımın tek gövdede toplandığı adanmış düzenekler şeklinde. Her birinin diğerine üstün olduğu yanlar vardır. Elinizde bulunan kişisel bilgisayarınızı, adanmış düzeneklerin çok altında bir tutarla ve daha önceki kullanımınızdan özveride bulunmaksızın protokol çözümleyici olarak da kullanabilirsiniz. Fakat adanmış bir protokol çözümleyici yalnızca ölçme yapar. Buna karşın adanmış çözümleyicilerde, kişisel bilgisayarların hız veya genişleme kısıtları yoktur, özel yazılım ve donanımları olduğu için elektriksel ölçme yetenekleri genelde daha gelişmiştir.

Bu tezin konusu, protokol çözümleyicilerin işlevlerinden bazılarını, özel kullanım alanlarında, istenen düzeyde gerçekleyebilecek bir ölçüm ve gözlem aracı yapma fikrinden doğmuştur.

1.1.1. Genel Donanım Özellikleri

Devrenin temel amacı, bir iletişim hattındaki verinin ve durum işaretlerinin saklanması ve görüntülenmesidir. Buradan yola çıkıldığında iletişim hattını asenkron veya senkron olarak gözleyebilecek bir iletişim denetleyicisinin, verileri saklayabilecek yeterli büyüklükte bellek alanının, verileri görüntüleyebilecek bir göstergenin, kullanıcının düzeneği denetleyebileceği tuşların ve tüm bu birimlerin üzerinde yazılımın koşacağı mikroişlemcinin gerekliliği ortaya çıkar.

İletişim denetleyicisi, iki bağımsız kanala sahip olmalı, asenkron ve senkron seri iletişim protokollerini desteklemeli, saat darbe kaynağının seçenekli olarak dışarıdan veya içeriden kullanabilmelidir. Bu özellikleri sağlayan AMD firması tarafından üretilmiş SCC AM85C30 (Serial Communication Controller) tümdevresi, seri iletişim denetleyicisi olarak kullanılmıştır.

İletişim hızının üst sınırı 19200 baud olarak düşünülmüştür. Bu hızda ve karakterlerin sekiz bit uzunluğunda olduğu durum için her bir kanaldan saniyede yaklaşık ikibin karakter iletilmektedir. Durum işaretlerinin zaman içerisindeki değişimide saklanmaktadır. Bu değerlerin toplamı 6 Kbyte/saniye etmektedir. Donanım üzerinde 128 Kbyte' lık oku/yaz bellek bulunmaktadır. Verinin en yoğun olduğu durum düşünüldüğünde yaklaşık 20 saniyelik bir zaman dilimi saklanıp gözlenebilecektir. Ortalama iletişim hızları göz önüne alındığında devrenin saklama sığası 1-3 dakika arasındadır.

İletim hattı özellik değişkenlerinin sayısı pek fazla değildir. Bu değerler iletişim protokolüne bağlı olarak birkaç sabit kümesinden oluşmuştur. Değerlerin belirlenmesi için birkaç tuş yeterlidir. Bir menü yapısı altında dallarda hareketi ve değer değişikliklerini sağlayabilecek dört tuş yeterli görülmüştür. Bu tuşlar saklanmış bilgilerin görüntülenmesinde de kullanılmıştır.

Gösterge olarak gereken alan gerçekte, saklanan verilerin türü ve gösterme şekline bağlı olarak birkaç bilgisayar ekranı kadar olmalıdır. Bunun sebebi verilerin çokluğu değil veriler üzerinde yapılabilecek işlemlerin çokluğudur. Böyle bir göstergenin temin edilmesi güç ve maliyeti yüksek olacaktır. Sınırlı alanda kullanılacağı daha önce de belirtilen bu düzenekte, kolay temin edilebilen 2 satır X 16 karakter' lik LCD gösterge kullanılmıştır.

Tüm bu yan birimlerin üzerinde, temel denetleyici olarak bir mikroişlemci gerekir. Devrenin tasarımında mikroişlemci yerine mikrodenetleyici kullanılmasının ana nedeni, tuş ve LCD gösterge gibi giriş/çıkış işlevi gerektiren yan birimler bulunması, zamanlama kesmesi gerekmesidir. Intel MCS51 ailesinden yaygın olarak kullanılan 8031 mikrodenetleyicisi bu özellikleri kendiliğinden sağlayan düşük maliyetli bir mikrodenetleyicidir. Kullanılan LCD göstergenin donanım özellikleri hızlı ulaşmalara izin vermemesi sebebiyle göstergenin bellek haritasında tanımlanması mümkün olmamıştır. Buna karşın tuşlar ve iletişim denetleyicisi bellek haritasına konabilmiştir. 8031' in bellek sığası 64 Kbyte yazılım, 64 Kbyte oku/yaz belleğiyle sınırlıdır. Oysaki veri saklamak için 128 Kbyte oku/yaz bellek kullanılmıştır. Dolayısıyla sayfalanmış bellek yöntemi kullanmak gerekmiştir. İki giriş/çıkış ayağı ile A16 ve A17 adres hatları tanımlanarak 256 Kbyte oku/yaz bellek alanına ulaşılmıştır.

Devre şeması çizimleri ORCAD ile gerçekleştirilmiştir. Baskı devre aşamasında yine ORCAD' in baskı devre hazırlama yordamı kullanılmıştır. Devre elemanlarının yerlerini belirledikten sonra, yordam şemadan aldığı bilgilerle, yolları kendi kendine bağlama durumunda, çok az dış yardımla baskı devre bağlantılarını gerçekleştirmiştir.

1.1.2. Genel Yazılım Özellikleri

Yazılım MCS51 tümdevre ailesi için geliştirilmiş C dili üzerine oturtulmuştur. C dilinde yazılan kaynak yazılımlar derleyiciden geçirildikten sonra, gerekli işlemler yapılarak EPROM'a yüklenip devre üzerinde çalışabilir duruma getirilmektedir. Denemeler sırasında işlemleri kolaylaştırmak ve bir deneme süresini kısaltmak için EPROM gibi davranabilen yardımcı bir donanım kullanılmıştır.

Yordamları işlevlerine göre aşağıdaki gibi sınıflayabiliriz:

- Donanım ile ilgili tanımlamaları içeren ve yordamların donanıma ulaşmasını sağlayan, aygıt sürücü tanımına yakın altyordamlar.
    - LCD Göstergeye ulaşan düşük seviyeli komutlar.
    - Bellek okuma/yazma komutları.
    - Tuşları okuyan komutlar.
    - İletişim hattının durumunu -veriakış denetimi / el sıkışma işaretleri- okuyan komutlar.
    - İletişim hattına bağlı seri iletişim denetleyicine ulaşan düşük seviyeli komutlar.
- Donanıma ulaşmayı sağlayan altyordamları kullanan ve anayordamca kullanılan orta seviyeli altyordamlar.
    - Donanımı şartlamaya yönelik altyordamlar.
        - Başlangıç şartlamaları.
        - Kullanıcı tarafından iletişim hattı özelliklerinin değiştirilmesi sonrasındaki şartlamalar.
        - Göstergeye karakter dizisi veya onaltılı sayı tabanında bilgi yazabilen altyordam.
        - Menü içerisinde, kullanıcının bastığı tuşa göre tavır alan altyordam.
        - Donanımın testine yönelik altyordamlar.
- Tüm bu altyordamları kullanan anayordam.
    - Şartlamalar.
    - Kullanıcı arayüzü.
    - Kullanıcı denetiminde donanım testleri.
    - İletişim hattı özelliklerinin belirlenmesi.
    - Başlama tetik şeklinin belirlenmesi.
    - Bellekte saklanmış bilgilerin gösterilmesi.
- Kesmeler.
    - Zamanlayıcı kesmesi.
        - Tuş bilgisinin oluşturulması.
        - Tetik ve saklama anlarının oluştuğunun belirlenmesi.
        - İletişim hattında akan verinin, durum bilgisiyle eşzamanlı olarak 1 ms aralıkla saklanması.
    - İletişim hattına bağlı seri iletişim denetleyicisinden gelen kesme.
        - DTE ve DCE arayüz verilerini saklanmak üzere belleğe aktarılması.

2. İLETİŞİM YAPILARI VE PROTOKOL ÇÖZÜMLEYİCİLER

Genel veri iletişimi örneksel (analog) veya sayısal (dijital) yolla yapılabilmektedir. Olağan bir TV görüntüsü, görüntü algılayıcıdan örneksel olarak çıkar, değişik görüntü etkileri oluşturulmak istenmiyorsa vericiye, oradan da vericiden çıktığı şekliyle örneksel bir işaret olarak ekrana kadar gelir. Eğer teletext işareti gönderiliyorsa, TV üzerinde de teletext alıcısı varsa, sayısal işaretin örneksel'e, örneksel işaretinde sayısal'a bir kaç defa dönüştüğü bir dizge oluşur. Bu tezin kapsamı sayısal veri işaretleri ve bu işaretlerin seri olarak iletildiği hatlarla sınırlıdır.

2.1. Bilgisayarlardaki Veri Yapıları Ve İletimleri

Bilgisayarlarda veriler, genel olarak sekiz adet ikili sayı biriminin (bit) birleşmesinden oluşan birimlerle (byte) saklanır. Bir bit'in alabileceği değerler mantıksal seviyeler olarak da adlandırılan '1 veya 0' dır. Bilgisayar, bilgileri dış dünyaya yine içinde sakladığı gibi bit'lerle veya byte'larla gönderir. Eğer bir byte'ın bit'lerinin birden fazlası aynı anda dış dünyaya veriliyorsa bu iletim yöntemi 'paralel iletim' olarak adlandırılır. Eğer, belirli sabit zaman dilimlerinde, bit'ler birer birer dış dünyaya gönderiliyorsa bu iletim şekli 'seri iletim' adını alır. Zaman dilimleri birbirine eştir ve hassas bir saat darbe kaynağıyla üretilirler. Burada arka arkaya iletilen bit'ler 'seri veri' adını alır. Tez içerisinde 'sayısal seri veri' yerine kısaca 'veri' sözü kullanılacaktır. İletim, ancak karşıda bir alıcı olduğu durumda anlam kazandığından bu durumu da 'iletişim' olarak adlandıracağız.

Saat işareti iletişim şeklinin bir tanımını daha üretir. Senkron ve asenkron iletişim: Saat işareti veri ile birlikte gönderiliyorsa buna 'senkron iletişim' , saat işaretini gönderen ve alan ayrı ayrı üretiyorsa buna da 'asenkron iletişim' denir. Senkron iletişimde, göndericinin gönderme saat işareti, alıcı tarafından alma saat işareti olarak kullanılır. Gönderme ve alma saat kaynakları iki yönde bağımsız olabilirler. Asenkron iletişimde ise veri bitleri içsel saat kaynağı temel alınarak gönderilir. Ancak bitler veriyi oluştururken başlangıç ve bitişleri belirlenir. Bu işlem, gönderici ve alıcı arasındaki eşzamanlamayı sağlar.

İletim veya veri hattı özellikleri, bağlantı noktalarındaki tanımlar, yerel ağlardan başlayan dünya üzerine yayılmış bilgisayar ağları, ağ içi ve ağlar arası protokoller... ve diğer ayrıntılar, herbirine ait genel ölçüm değerleri EIA, CCITT, ANSI, IEEE gibi kuruluşlarca belirlenmiştir.

2.2. Seri Kanallar Ve RS-232C

Seri kanal, bir bilgisayarın dış dünya ile bağlantı kurabildiği yöntemlerden biridir [7]. Modemler yardımıyla bilgisayarın tüm dünyaya ulaşabildiği bir kanaldır. Bu kanal erkek veya dişi bir bağlayıcıdan (konnektör) oluşur. Bağlayıcı, 9 veya 25 elemanlı olabilir. Seri kanallar, bilgisayar ve modem arayüzü (DTE/DCE arayüzü) gibi bağlantı kablolarının işlevleriyle de anılırlar.

RS-232C genel ölçü önerisi DTE (Data Terminal Equipment) ve DCE (Data Communication Equipment) arayüzü bağlantılarının fiziksel ve elektriksel tanımını içerir. Ancak CCITT V.24 ile bu tanımlar genel ölçü birimi haline getirilmiştir. Bu tanımın, tezin kapsamında olan kısmı aşağıdaki tablolarda verilmiştir.

D-tipi Konn. Bağ. No.	**İşaretin İsmi (V.24 Devresi)**	**İşaretin Yönü**	**DCE İşlevi**
2	TxD (103)	Giriş	Transmit Data : Gönderilen veri
3	RxD (104)	Çıkış	Receive Data : Alınan veri
4	RTS (105)	Giriş	Request To Send : Veri akış denetimi/el sıkışma
5	CTS (106)	Çıkış	Clear To Send : Veri akış denetimi/el sıkışma
6	DSR (107)	Çıkış	Data Set Ready: Modem hazır
7	GND (102a)	—	Signal Ground : Veri işaret toprak referansı
	GND (102b)	—	Signal Ground : Veri işaret toprak referansı
8	DCD (109)	Çıkış	Data Carrier Detect : Hat kuruldu
15	TxC (114)	Çıkış	Transmit Signal Element Timing : Gönderme saat işareti
17	RxC (115)	Çıkış	Receive Signal Element Timing : Alma saat işareti
20	DTR (108.2)	Giriş	Data Terminal Ready : Terminal hazır
24	TxC (113)	Giriş	Transmit Signal Element Timing : Gönderme saat işareti

Tablo 2.2.1. DCE Bağlayıcı Bağlantı Tanımı

D-tipi Konn. Bağ. No.	İşaretin İsmi (V.24 Devresi)	İşaretin Yönü	DTE İşlevi
2	TxD (103)	Çıkış	Transmit Data : Gönderilen veri
3	RxD (104)	Giriş	Receive Data : Alınan veri
4	RTS (105)	Çıkış	Request To Send : Veri akış denetimi/el sıkışma
5	CTS (106)	Giriş	Clear To Send : Veri akış denetimi/el sıkışma
6	DSR (107)	Giriş	Data Set Ready: Modem hazır
7	GND (102a)	—	Signal Ground : Veri işaret toprak referansı
	GND (102b)	—	Signal Ground : Veri işaret toprak referansı
8	DCD (109)	Giriş	Data Carrier Detect : Hat kuruldu
15	TxC (114)	Giriş	Transmit Signal Element Timing : Gönderme saat işareti
17	RxC (115)	Giriş	Receive Signal Element Timing : Alma saat işareti
20	DTR (108.2)	Çıkış	Data Terminal Ready : Terminal hazır
24	TxC (113)	Çıkış	Transmit Signal Element Timing : Gönderme saat işareti

Tablo 2.2.2 DTE Bağlayıcı Bağlantı Tanımı

Tablo 2.2.1. ve Tablo 2.2.2.'de DTE/DCE arayüzünün, iletişimin senkron veya asenkron olmasından bağımsız olarak tanımlandığı görülmektedir. Arayüzde veri akış denetimi tek yönlü olarak tanımlıdır. RTS ve CTS veri akış denetimi veya el sıkışma işaretleri olarak şöyle kullanılır: DTE veri göndermek istediğinde RTS ucunu etken duruma getirir, DCE veri alabilir durumda ise CTS ucunu etken duruma getirir. DTE veri gönderme işlemi bittiğinde RTS ucunu etken durumdan çıkarır. Benzer şekilde DCE veri alamaz duruma geldikçe CTS ucunu etken durumdan cıkarır. Bu işlemin bir şekli de şöyledir: RTS ucunun anlamını, CTS gibi kılarak iki yönlü akış denetimi yapılabilir. Yani DTE' nin RTS'i, DCE'nin CTS'i gibi veri alabilmeyi belirleyen bir uç olarak kullanılabilir. Son olarak, RTS-CTS veri akış denetimi uygulanmayabilir. Diğer uçlar genellikle tanımı içinde kullanılır.

Bu uygulamada gerçeklenen düzenek, yukarıdaki DTE tablosundaki tüm uçları, kendine giriş olarak kabul etmektedir. Dolayısıyla DTE/DCE arayüzünün tüm işaretlerini kullanmaktadır.

2.3. Bilgisayarlar Arası İletişim

Donanımda genel ölçü birimlerinin konulması gereği kadar, verinin iletimi sırasında hangi şekilde olacağının da belirlenmesi önemlidir. Genel ölçü birimi konmamış veri biçimlerinin sonucu birbiri ile konuşamayan bilgi birimleri olurdu.

Bilgi alış-verişini genelleştirmek amacıyla ASCII (The American Standart Code for Information Interchange) karakter kümesi geliştirilmiştir. 7 bit veri tabanlı bu kümenin, ofis kullanımına hizmet eden bölümü ANSI X3.4. genel ölçü birimi olarak yayınlanmıştır.

ASCII kümesinin 7 bitlik olması ve yalnızca İngilizce yazı karakterlerini içermesi, farklı dillerdeki bilgilerin iletimini güçleştirmektedir. IBM, Apple ve birkaç UNIX üreticisi bu noktadan hareketle Unicode adında yeni bir karakter kümesi oluşturmuşlardır. Bu küme tüm ulusal (İngilizce) karakterlerle birlikte Doğu Asya alfabe karakterlerini de içermektedir. Bu küme de Haziran 1992'de ISO 10646 olarak genel ölçü birimleri arasına girmiştir. 16 bitlik bu kodlama 16 bitlik dosya aktarımında gerektirmektedir.

Dosya aktarımı bilgisayarlar arasındaki bilgi akışının veri yapısını düzenleyecek protokollerle olmaktadır. Bu protokolleri bilgisayar ağı protokollerinden ayrı düşünmek gerekir. Çünkü bunlar yalnızca dosya aktarımını amaçlayan uygulama yazılımlarıdır. Yaygın olarak kullanılan bazıları XMODEM, YMODEM, ZMODEM ve Kermit yazılımlarıdır.

Bilgisayar ağı protokolleri ise paket temelli altyapılardır. Ethernet, FDDI, SNA ve X.25 örnek olarak verilebilir. Burada değişik üretici kaynaklı donanımların birbiriyle bağlanabilmesi için, veri iletiminin genel kuralları, OSI referans modeli ile konulmuştur. Model yedi katmandan oluşur: Fiziksel, veri hattı, ağ, taşıma, oturum, sunum ve uygulama katmanları. İlk üç katman veri iletimi ve bağlantı için, son üç katman ise kullanıcı uygulamalarına yöneliktir. Dördüncü katman ilk üç ve son üç katmanın arayüzü durmundadır.

Veri iletiminin kiralık veya anahtarlamalı hatlar üzerinden gerçekleşmesini sağlayan ara elemanlar modemlerdir. 1970'lerin başlarında 110-300 baud hızlarında çalışan modemler, günümüzde uygun hatlarda 100 Kbaud'un üzerinde iletişim sağlayabilmekteler. Gelişimi; hız, protokol ve veri sıkıştırma alanında gözlenir. Bell 300 ilk asenkron standart modem olarak çıkmıştır. 1970 sonlarında 1200 baud V.22, 1980 başlarında 2400 baud V.22bis, sonra 9600 baud V.32, 14400 baud V.32bis ve son olarak 19200-28800 baud V.fast genel ölçü birimi olarak çıkmıştır. Akıllılık anlamında bakıldığında, 300 baud

sonrası, Hayes komut kümesini anlayabilen mikroişlemcili-akıllı modemlerdir. Hata bulma ve düzeltme MNP-4 ve MNP-5 (Microcom Network Protokol) özellikli modemlerde vardır. V.42 ve V.42bis ile hatalı veriyi tekrar gönderme özelliği getirildi.

RS-232C veri akış hızını 19200 baud ile sınırlamıştır. Yüksek veri yoğunluğunda, bu hız günümüzde çok düşük kalmaktadır. Ancak, altyapı da yeterince hızlı olmadığı için geçerliliğini bir süre daha sürdürecektir.

2.4. Protokol Çözümleyiciler

Protokol çözümleyiciler, veri hattı ile ilgili geniş bilgiler edinebildiğimiz yapılardır. Aynı düzenekle birden fazla iletişim protokolünü çeşitli arayüz bağlantılarında, kablolamadan bağımsız olarak incelemeyi mümkün kılan yapılar olabilir. Örneğin, CCITT X.25 ağ protokolünü RS232C arayüzünde veya RS422 arayüzünde incelemenize olanak sağlar.

Veri hattında neleri ölçmek ve görmek isteriz? Veri hattındaki bilgi, kullanılan iletişim protokolüne uymalıdır. Benzer şekilde arayüz işaretleri de o arayüzün işaret seviyeleri tanımına ve o protokolün arayüz işaretlerini kullanma yöntemine uymalıdır. Bunların uygunluğunu belirleyebilmek için öncelikle arayüz işaretlerini tanıyıp değerlendirebilmek gerekir. Eğer protokol çözümleyicinin birbirine benzer özellikte arayüzlerde kullanılması söz konusuysa, bu tanımların ortak yönlerinden bir bileşke çıkarıp, çözümleyici üzerine bir tek arayüz konulabilir. RS232C ve RS422 gibi arayüzler bir tek arayüz tanımıyla gözlenebilir. Fakat BNC bağlayıcı ile kablolama yapılmış ise, bu ortam içinde ölçüm ve gözlemleme yapılabilmesi için BNC bağlayıcılı girişi olan bir çözümleyici gereklidir.

Bağlantı noktalarındaki arayüz isterleri karşılandıktan sonra protokol içinde tanımlı veri yapısını tanıyabilme özelliği gerekir. İletişim protokolleri, verilerin paketlenme şekli, paket içi ve paket dışı veri özelliklerinin belirlenmesi üzerine kuruludur. Veriler paketlenirken bilgi, komut, durum paketi gibi isimler alabilir. Çözümleyici tarafından paketin tipinin tanınması, geçerliliğinin denetimi, içindeki bilginin ayrıştırılması ... ancak o iletişim protokolünün genel ölçü birimlerinin yazılıma uygulanmasıyla gerçekleşebilecektir.

Protokol çözümleyicilerin ticari şekillerinin incelenmesi için tanıtıcı kitapçıklar incelendikten sonra, bir süre uygulamalı olarak GN NAVTEL firmasının 9400 serisi kişisel bilgisayar temelli bir çözümleyicisi kullanıldı [1]. Bu çözümleyici, asenkron ve senkron verileri bir protokol altında depolayabiliyor ve RS232C veya RS422 arayüzlerinde çalışabiliyor. Bilgileri depolarken aynı zamanda bilgisayar

ekranında gerçek zamanda gösterebiliyor. Bilgi depolama başlangıcı veya bitişi bir tetiğe bağlanabiliyor. Bu tetik, protokole ait bir paket tipi, arayüze ait bir işaret seviyesi değişimi, belirli bir zaman veya bir tuş olabilir. Örneğin, X.25 hattında DTE'nin özel bir paket göndermesiyle depolamaya başlamasını, yine özel bir paketle durmasını sağlayabiliyorsunuz. Bilgi saklama süresi, kişisel bilgisayarın belleğiyle kısıtlı. Protokole bağlı olarak değişik şekillerde istatistikler tutabilmekte: Bir paketin tekrar sayısı, toplam paket sayısı, toplam veri uzunluğu, toplam bilgi uzunluğu gibi.

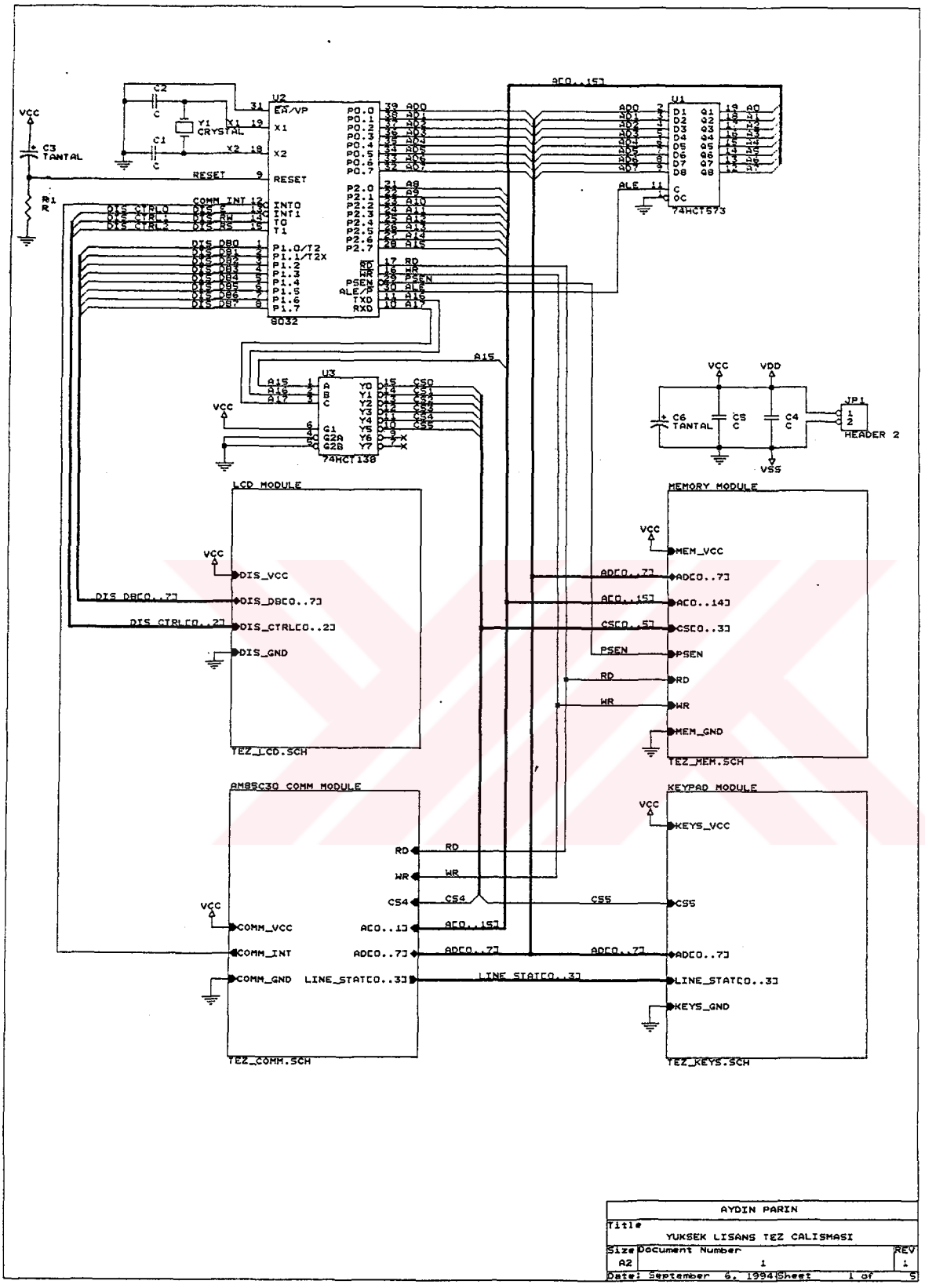
Bu uygulamada, işlevsel özelliklerin bir bölümünde, yukarıda anlatılan çözümleyici örnek alınmıştır.

3. DEVRENİN DONANIM ÖZELLİKLERİ

Devre şemaları, Şekil 3.1.1., Şekil 3.2.1., Şekil 3.3.1. ve Şekil 3.4.1.' de görülmektedir. Ana denetleyici olarak INTEL MCS51 serisinden 80C31 kullanıldı. Veri belleği olarak dört adet 62256 (32 K*8 RAM) sayfalı adresleme ile kullanıldı. Yazılım belleği olarak 27256 (32 K*8 EPROM) kullanıldı. Tuş okuma ve veri hattı durum işaretlerinin bazılarını gözlemek için 74245 (Üç durumlu iki yönlü tampon) kullanıldı. Gösterge olarak, HITACHI HD44780 denetleyici içeren LM052L (16 karakter*2 satır) kullanıldı. Her tümdevrede, gürültü kesmek için bir adet 100 nF' lık kapasite kullanıldı.

3.1. Mikrodenetleyici Birimi

Mikrodenetleyici biriminin devre şeması Şekil 3.1.1.' de verilmiştir. Bu yapı sayfalı adresleme ile 256 Kbyte' lık oku/yaz bellek alanına ve en fazla 64 Kbyte' lık yazılım alanına ulaşabilir. Sayfalı adresleme 74HCT138 kod çözücü tümdevresi yardımıyla gerçekleştirilmiştir. 74HCT574 tümdevresi, 80C31'in çoğullanmış olarak dışarıya verdiği adres ve veri yollarını ayırır.



Şekil 3.1.1. Ana Devre Şeması

3.1.1. MCS51 Ailesinin Donanım Yapısı

Tümdevre	ROM (Byte)	EPROM (Byte)	RAM (Byte)	Zamanlayıcı
80C31BH	-	-	128	2
80C51BH	4K	-	128	2
87C51	-	4K	128	2
80C32T2	-	-	256	3
80C52T2	8K	-	256	3
87C52T2	-	8K	256	3

Tablo 3.1.1.1. MCS51 Ailesi Temel Özellikler

Tablo 3.1.1.1.' de MCS51 ailesinin temel özelliklerini görüyoruz. 8xx1 ailesinin diğer temel özellikleri aşağıdadır [5].

- 8 bit denetleme uygulamaları için geliştirilmiş merkezi işlem birimi
- Full Duplex UART
- İki öncelik seviyeli 5 kesme kaynağı
- Tümdevre üzerinde saat darbe kaynağı
- İkili sayı tabanı işlemcisi
- Bit adreslenebilir bellek
- 64 Kbyte yazılım bellek alanına erişebilme
- 64 Kbyte veri bellek alanına erişebilme

80C31'de sayıcı-zamanlayıcı sayısı ikidir. Bunlar değişik şekillerde şartlanabilir. Beş kesme üreteci vardır. İkisi dışsal, üçü içsel... Üreteçlerin öncelik sırası kullanıcı tarafından belirlenebilir. Kesme üreteçlerinden dışsal olan ikisi düşük seviye tetiklemeli veya düşen kenar tetiklemeli olarak şartlanabilir. 4 Giriş/çıkış birimi (PORT) vardır. Bunlardan ikisi adres ve veri yolu olarak kullanılır. Diğer ikisi genel amaçlı olarak kullanılabildiği gibi özel anlamları da vardır. Mikrodenetleyicinin Güç Kaynağı Kapalı (Power-down) ve Boşta (Idle) olmak üzere iki özel durumu vardır. Güç Kaynağı Kapalı durumunda saat üreteci durur, mikrodenetleyici hiç birşey yapmaz. Bu durumdan RESET işaretiyle kurtulur. Bu durum kalktığında RAM'deki bilgiler dışında herşey başlangıç durumuna döner. Boşta durumu ise mikrodenetleyiciyi işlemsel olarak durdurur fakat kesme, seri giriş/çıkış, zamanlama işlevlerini durdurmaz. Herhangi bir kesme geldiğinde işlem kaldığı yerden devam eder. Tüm değerler saklanmıştır. 80C31'de bir komut evresi 12 saat işareti dönemidir. Bir komutun uygulaması yapılan işlemin özelliğine göre 1-4 komut evresi sürer.

- MCS51 Ailesinin Giriş/Çıkış Biriminin İncelenmesi

80C31'nin dört giriş/çıkış birimi olduğu daha önceki bölümde belirtilmişti.Bu giriş/çıkışların özellikleri daha ayrıntılı olarak incelenirse :

PORT 0

Dışsal bellek veya RD-WR uçlarının kullanımını gerektiren bir çevre birim kullanıldığında, adres ve veri yollarının 0-7 bitleri bu port üzerinden çoğullanmış olarak çıkar. İçsel yollarla o anda adres, veri veya giriş-çıkış bilgisinin yazılıp okunması denetlenir. Port'ta ya adres-veri yada giriş-çıkış bilgisi olacaktır. Bunu EA (External Acsess) ayağının seviyesi belirler. EA düşük seviyede ise dışsal EPROM var demektir. Port yalnızca adres ve veri yolu olur. Yüksek seviyede ise yalnızca giriş-çıkış olur. Port'un içsel yukarı çekme devresi FET'tir ve yalnızca dışsal bellek kullanıldığı zaman etkindir.

PORT 1

8031'de genel giriş-çıkış olarak kullanılan port'un içsel yukarı çekme devresi FET'tir ve her zaman etkindir.

PORT 2

Dışsal bellek kullanıldığında Port 0'ı tamamlar: Adres yolunun 8-15 bitleri bu port üzerinden alınır. Dışsal bellek yoksa genel giriş/çıkış olarak kullanılılır. İçsel yukarı çekme devresi Port 1 gibidir.

PORT 3

Her biti özel anlamlıdır. Giriş/çıkış olarak da şartlanabilen bu bitler şunlardır:

P3.0 RxD Seri bilgi girişi
P3.1 TxD Seri bilgi çıkışı
P3.2 INT0 1. Kesme girişi
P3.3 INT1 2. Kesme girişi
P3.4 T0 1. Sayıcı-zamanlayıcı girişi
P3.5 T1 2. Sayıcı-zamanlayıcı girişi
P3.6 WR Dışsal belleğe yazma belirteci
P3.7 RD Dışsal belleği okuma belirteci

Bu bitlerden 0. ve 1. bitler seri iletişimde etkindir. Seri iletişim TCON, TMOD, SCON ve PCON kontrol kayıtçıları ile şartlanır. Seri iletişim hızı 1 MBaud'dan 110 Baud'a kadar ayarlanabilir. İstenirse seri

iletişim kesmesi kullanılır. 6. ve 7. bitler dışsal belleğe erişim komutlarında etkin edilir: Okuma yapılıyorsa RD, yazma yapılyorsa WR... bu komutlardan bağımsız olarak da seviyeleri değiştirilebilir.

Bir port'a bilgi yazımı sırasında, bilgi önce port bilgi tutucusuna yazılır. İki saat dönemi sonrasında ayağa aktarılır. Okuma sırasında üç farklı yol vardır: Ayak seviyesi okuma, tutucu seviyesi okuma ve Oku-Düzenle-Yaz okuma. Ayak seviyesi okuma işleminde doğrudan ayak gerilim seviyesi okunur. Örneğin anahtarın konumu okunacaksa anahtar uygun donanımla bağlanmalı ve normal okuma işleminden önce port bilgi tutucusuna uygun seviye bilgisi yazılmalıdır. Tutucu seviyesi okuma işleminde port bilgi tutucu seviyesi okunur. İşlem sırasında okunan ayak gerilimi ile tutucu seviyesi arasındaki farklılık nedeniyle bir hata olmaması için, ayak gerilimi okunmaz. Örneğin transistörün beyzini süren bir ayak için; port bilgi tutucu seviyesi yüksek olduğu halde ayak gerilimi okunduğunda alçak seviyeli görülecektir. Oku-Düzenle-Yaz durumunda özel komutlar vardır: ANL, ORL, XRL, JBC, INC, DEC... Bu komut işlenirken portun bilgi tutucusu okunur, işlenir, tekrar tutucuya yazılır. Yanıp sönen ışık uygulamasında bilgi tutucu bitinin evriğini almak bu işlem için doğru yöntemdir.

- MCS51 Ailesinin Kesme Biriminin İncelenmesi

Kesme üreteçleri öncelik sırası ve başlangıç adresleri şöyledir:

1. IE0	1. Kesme girişi	0003H
2. TF0	1. Sayıcı-zamanlayıcı taşma	000BH
3. IE1	2. Kesme girişi	0013H
4. TF1	2. Sayıcı-zamanlayıcı taşma	001BH
5. RI+TI	Seri iletişim isteği	0023H
6. TF2+EXF2	3. Sayıcı-zamanlayıcı taşma	002BH

IE0 ve IE1 girişleri seviye veya kenar tetiklemeli olarak şartlanabilmektedir. TF0 ve TF1 kesmeleri 1. ve 2. sayıcı zamanlayıcı taşma çıkışlarıdır. Yazılımla içsel yollarla çeşitli amaçlarda kullanılabilir. R1 + T1 kesmeleri seri iletişim isteği anında etkindir. TF2 + EXF2 kesmesi de 3. sayıcı zamanlayıcı taşmasında veya EXF2 girişinin etkin olmasında gerçekleşir. Tüm kesmelerin EI -Kesme etkin- kayıtçısıyla etkin hale gelmesi denetlenir. IP -Kesme öncelik- kayıtçısıyla da mikrodenetleyicinin aynı anda gelen kesmelerden hangisinin uygulanacağına karar vermesi sağlanır.

- MCS51 Ailesinin Sayıcı/Zamanlayıcı Yapısının İncelenmesi

80C31'de 16 bitlik iki sayıcı zamanlayıcı vardır. Bunlar TMOD ve TCON isimli kayıtçılarla şartlanır. TMOD içerisinde dört bitlik iki grup vardır. Bu bitler Gate, C/T, M1 ve M0'dır. M1 ve M0 birlikte

sayıcı zamanlayıcının çalışma şeklini belirler. Gate ise dışsal kesmenin ayaktan mı, yoksa içsel olarak zamanlayıcı taşmasından mı olacağını belirler. C/T sayıcı veya zamanlayıcı olarak çalışmasını denetler. TCON kayıtçısında iki sayıcı zamanlayıcıya ait denetlemeler bulunur. Bu kayıtçının bitleri yazılım ve donanım ile denetlenir. Sayıcı zamanlayıcı, tutma şeklinde çalıştığında 8 bitlik olmaktadır.

3.1.2. MCS51 Ailesinin Yazılıma Ait Özellikleri

80C31'in yazılımında kullanılan donanım kayıtçıları ve bayrakları vardır. Tüm kayıtçılar tümdevrenin içsel belleğinin haritasında görülebilir ve sıradan bellek ulaşımı komutlarıyla da ulaşılabilir. İçsel bellek ikiye bölünmüştür. Doğrudan veya dolaylı ulaşılabilen 128 byte, buraya SFR (Special Function Registers -Özel İşlev Kayıtçıları-) alanı da denilmektedir. SFR alanında tüm özel kayıtçılar, ACC, B, kayıtçı bankaları, zamanlayıcı-sayıcılar, giriş-çıkış birimleri bulunmaktadır. Diğer bölümü de yalnızca dolaylı adresleme ile ulaşılabilen 128 byte'lık bellek alanıdır. Komutların ana kayıtçısı ACC veya A olarak kısaltılan akümülatör kayıtçısıdır. Kayıtçı bankası olarak anılan, sekizer kayıtçıdan oluşan dört bellek alanı vardır. Bu kayıtçılar için komut setinde özel komutlar ayrılmıştır. Komut seti ve ilgili kayıtçıların özellikleri aşağıdadır.

- Durum Gösterici

Durum gösterici (PSW- Program status Word) CPU'nun o anki durumunu yansıtan bitler içerir.

PSW.0	P
PSW.1	- -
PSW.2	OV
PSW.3	RS0
PSW.4	RS1
PSW.5	FO
PSW.6	AC
PSW.7	CY

PSW.7 -Carry (Elde) biti- Akümülatör işlemlerinde meydana gelen taşmaları gösteren bittir.

PSW.6 -Auxiliary Carry (Yardımcı Elde) biti- Bu bit akümülatörde BCD işlemler sırasında meydana gelen taşmaları gösterir. BCD işlemlerde LSD taşmalarında yardımcı elde biti, MSD taşmalarında ise elde biti etkilenir.

PSW.5 biti kullanıcı tarafından belirlenen özel amaçlar için ayrılmıştır.

PSW.4 ve PSW.3 bitleri içsel bellek alanında yer alan dört kayıtçı bölgesinin seçimini Tablo 3.1.2.1.' deki gibi sağlar.

PSW.4	PSW.3	Kayıtçı alanı
0	0	0
0	1	1
1	0	2
1	1	3

Tablo 3.1.2.1. Kayıtçı Alanının PSW.3 ve PSW.4 Bitlerine Bağımlılığı

PSW.2 biti matematiksel işlemlerde taşma olduğunu belirtir.
PSW.0 Eşlik biti Akümülatördeki binary değerin parity değerini tek eşlik biti olarak gösterir; yani çift eşlik biti üretir.

- Adresleme şekilleri

- Direkt Adresleme: Bu adresleme modunda komutlar içerisinde veri 8 bitlik adres alanıyla tanımlanır. Bu modda özel işlev kayıtçıları ve içsel bellek alanına ulaşılabilir. Örneğin ADD A,7FH: İçsel bellek alanında yeralan 7FH bellek gözesinin içeriğini akümülatörle topla.

- Dolaylı Adresleme: Dolaylı adreslemede komut icrası kayıtçılar üzerinden sağlanır yani komut verinin adresini içeren bir kayıtçı ile tanımlanır. Dolaylı adreslemede bir başka kısıtlama da kayıtçı alanı içerisinden bu amaç için sadece R0 ve R1'in seçilebilmesidir. Örneğin ADD A,@R0: R0 kayıtçısının gösterdiği adres içeriğini akümülatörle topla

- İvedi Sabitler: Bu modda program hafızasında opcode'u takip eden sabit bir değer yer alır. Örneğin; MOV A, #100: 100 desimal değerini akümülatör'e koy.

- Kayıtçı komutları: Kayıtçı alanı içindeki 8 kayıtçı üzerinde işlem gören komutlardır. Komut içinde yer alan opcode'un 3 biti R0-R7 arasıdaki hangi kayıtçının seçildiğini belirtir. Dolaylı adresleme yapılıyorsa sadece R0 ve R1 sözkonusu olacağından opcode'un tek biti bunun için yeterli olacaktır. Örneğin ADD A,R7: R7 kayıtçısının içeriğini akümülatörle topla.

- Özel Kayıtçı Komutları: Bazı komutlar sadece bir kayıtçıya yöneliktir. Bu komutlar için kayıtçı adresini belirtmeye gerek yoktur. ACC, DPTR... gibi

- İndeks Adresleme: Bu yöntemle sadece program bellek alanına ulaşılabilir ve yalnızca okunabilir. Böylece "Look Up Table" olarak adlandırılan okuma gerçeklenir. Bu yötemde DPTR veya PC, 16 bitlik temel adres kayıtçısı olarak okunacak tablonun başlangıcını belirler akümülatör ise tabloya giriş numarasını içerir. Örneğin MOVC A,@A+DPTR. Bu adreslemenin diğer tipide "Case Jump" komutlarında kullanılır. Bu durumda jump komutunun icra edileceği adres temel gösterici (DPTR, PC) ve akümülatörün toplamlarıyla belirlenir. Örneğin JMP @A+DPTR

- Matematiksel Komutlar

80C31'nin komut setinde yer alan matematiksel deyimler: ADD -Toplama-, SUB -Çıkarma-, INC -Arttırma-, DEC -Azaltma-, MUL-Çarpma-, DIV-Bölme-, DA-Ondalık şartlama- dır. Çarpma ve bölme yapabilmesi üstün yeteneklerindendir. 80C31 bütün matematiksel komutların icrasını 2 µs'de gerçekleştirirken sadece çarpma ve bölme komutlarını 4 µs de gerçekleştirir.

- Mantıksal Komutlar

Mantıksal deyimler AND -Ve-, OR -Veya-, EXOR -Özel veya- ROTATE -Döndürme-, CLEAR -Sıfırlama-, COMPLEMENT -Tümleme- SWAP -Yer değiştirme- dir. Bu komutların icrası 80C31 tarafından en fazla 2 µs de sağlanır.

- Bilgi Aktarım Komutları

Bilgi aktarımı içsel bellek alanı içerisinde veya içsel bellek alanı ile dışsal bellek alanı arasında sağlanmaktadır. İçsel bellek alanı üzerindeki bilgi aktarımı 80X2 için alt 128 byte için hem direkt hem de dolaylı adresleme ile sağlanabilirken üst 128 byte için sadece dolaylı adresleme ile sağlanabilir. Özel işlev kayıtçısı üzerinden yapılan bilgi aktarımları sadece direkt adresleme ile yapılabilir.

- Bit İşlem Komutları

MCS51 Ailesinin Durum gösterici ve bit adreslenebilen içsel bellek alanı üzerinde yapılan işlemler sırasında bit komutları kolaylık sağlamaktadır. Bu komutlar aracılığıyla tek bitlik bilgi için byte kullanmaya gerek kalmamaktadır. Örneğin: CLR C, CLR Bit, CPL C, ANL C,Bit , MOV C,Bit

komutları ile bellek alanı üzerinden istenen bitle işlem yapılabilmektedir. Bütün bit komutları direkt adresleme ile icra edilir.

- Atlama ve Altprogram Çağırma Komutları

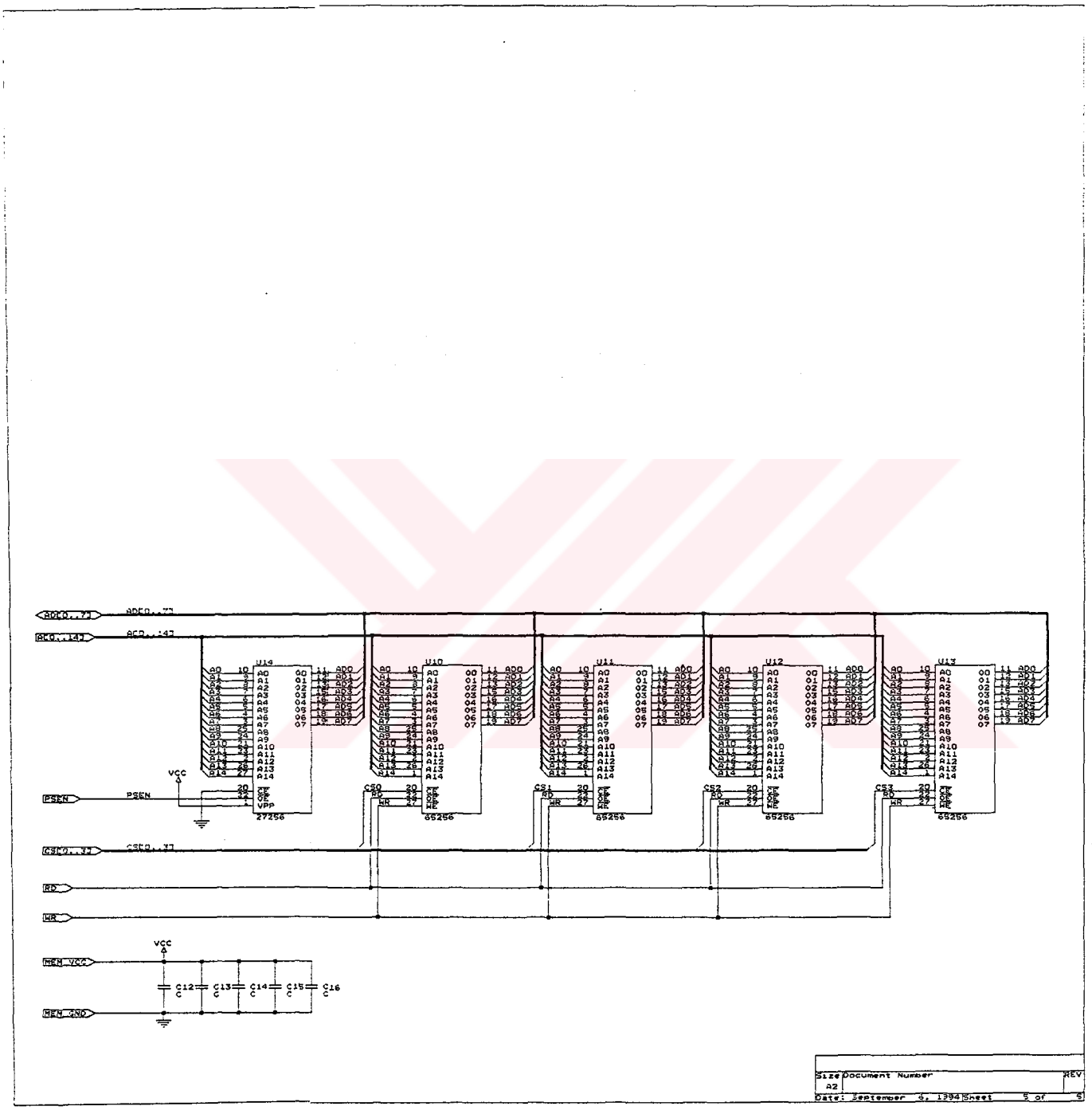
80C31 de atlamalar mutlak, uzun, kısa veya şartlı ve bağıl veya mutlak şekilde olabilir. Şartlar bit veya byte temelli karşılaştırmalardır. 80C31'nin mutlak atlamaları 11 bit adreslidir. Uzun atlamalar 16 bit, kısa atlamalar bağıldır ve 8 bittir. Altprogramlarda atlamalardaki özelliklerin bir kısmı vardır. Altprogramı şartlı veya kısa çağırma olanağı yoktur. Mutlak veya uzun çağırma yapılabilir.

3.2. Bellek Birimi

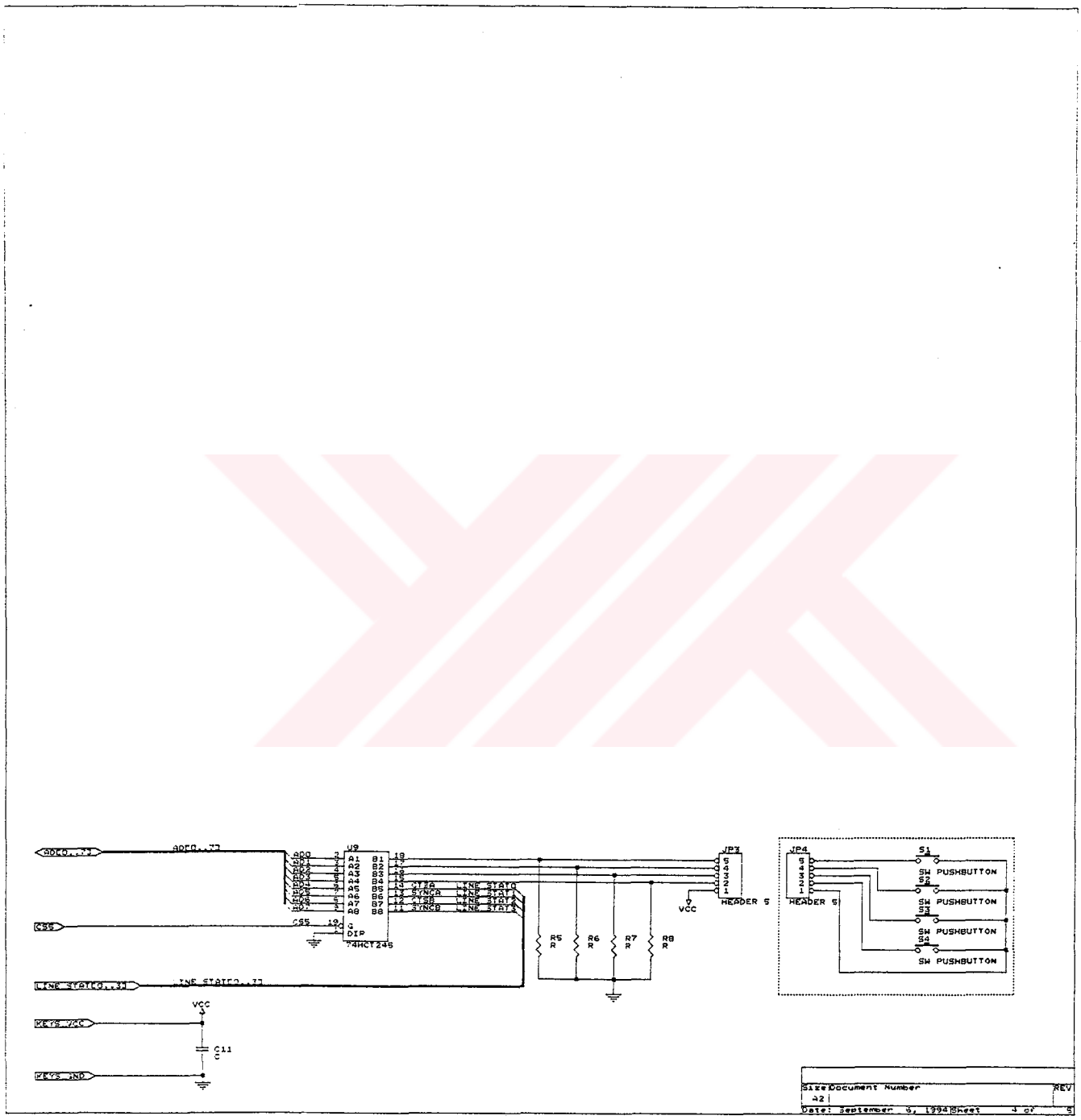
Devre şeması Şekil 3.2.1'de görülen bellek birimi, yazılımın yüklendiği bir adet 32 Kbyte' lık 27256 EPROM ve sistem, veri ve durum bilgilerinin saklandığı, dört adet 32 Kbyte' lık 62256 RAM içerir. RAM' lar sayfalı adreslemeyle işlemciye bağlıdır. Devrenin üzerinde şu anda 32 Kbyte' lık bir adet 62256 RAM bulunmaktadır. Yazılımda bellek boyları tanımlanarak, devre üzerine istenilen miktarda bellek konulabilir.

Bellek, yazılım içerisinde, toplam bellek alanından bağımsız olarak işlevine göre üç bölüme ayrıldı:

- Sistem belleği, yazılım ana işlev belleği olarak da adlandırılabilir. Düzeneğin, mikroişlemcinin koşarken yazılımın gerektirdiği değişkenleri sakladığı dışsal bellek alanı. Bu alan için, bu boyutta kullanılmıyor olmasına karşın 4 Kbyte'lık yer ayrılmıştır.
- Veri belleği, veri hattında akan bilginin konduğu alan olarak yazılım belleğinin hemen ardından kalan alanın yarısı olarak ayrıldı. Düzeneğin üzerinde şu anda bulunan bellek miktarı 32 Kbyte olduğu için veri belleği alanı 14 KByte olarak ayrıldı.
- Durum belleği, veri hattının durumunu ve o zaman diliminde alınan veri bilgisi olup olmadığının saklandığı alan olarak veri belleğiyle aynı boyda ayrıldı. Bu belleğin bir byte'ı veri hattının RTS, CTS, DTR, DSR, DCD işaret seviyelerini ve o zaman diliminde hangi yönlerde veri olduğunu içerir. Belleğe durum bilgisi şu anda 1 ms'lik zamanlayıcı kesmesiyle saklanıyor.



Şekil 3.2.1. Bellek Birimi Devre Şeması



Şekil 3.3.1. Durum ve Tuş Okuma Arabirimi Devre Şeması

3.3. Durum ve Tuş Okuma Arabirimi

Veri hattının özelliklerini girebilmek ve saklanan bilgiyi görebilmek için düzenek üzerine dört adet tuş konuldu. Tuş bilgisi zamanlayıcı kesmelerinde, Şekil 3.3.1' de görülen, 74HCT245 üç konumlu tampon tümdevresinden okunmakta ve uygun yazılımla değerlendirilmektedir. Veri hattının durum bilgisine kolayca ulaşabilmek için, seri iletişim denetçisinde bu bilgi olduğu halde, tuşlarla birlikte aynı üç konumlu tampondan okunmaktadır. Bu tampon bellek haritasına okuma emri ile bağlanmıştır.

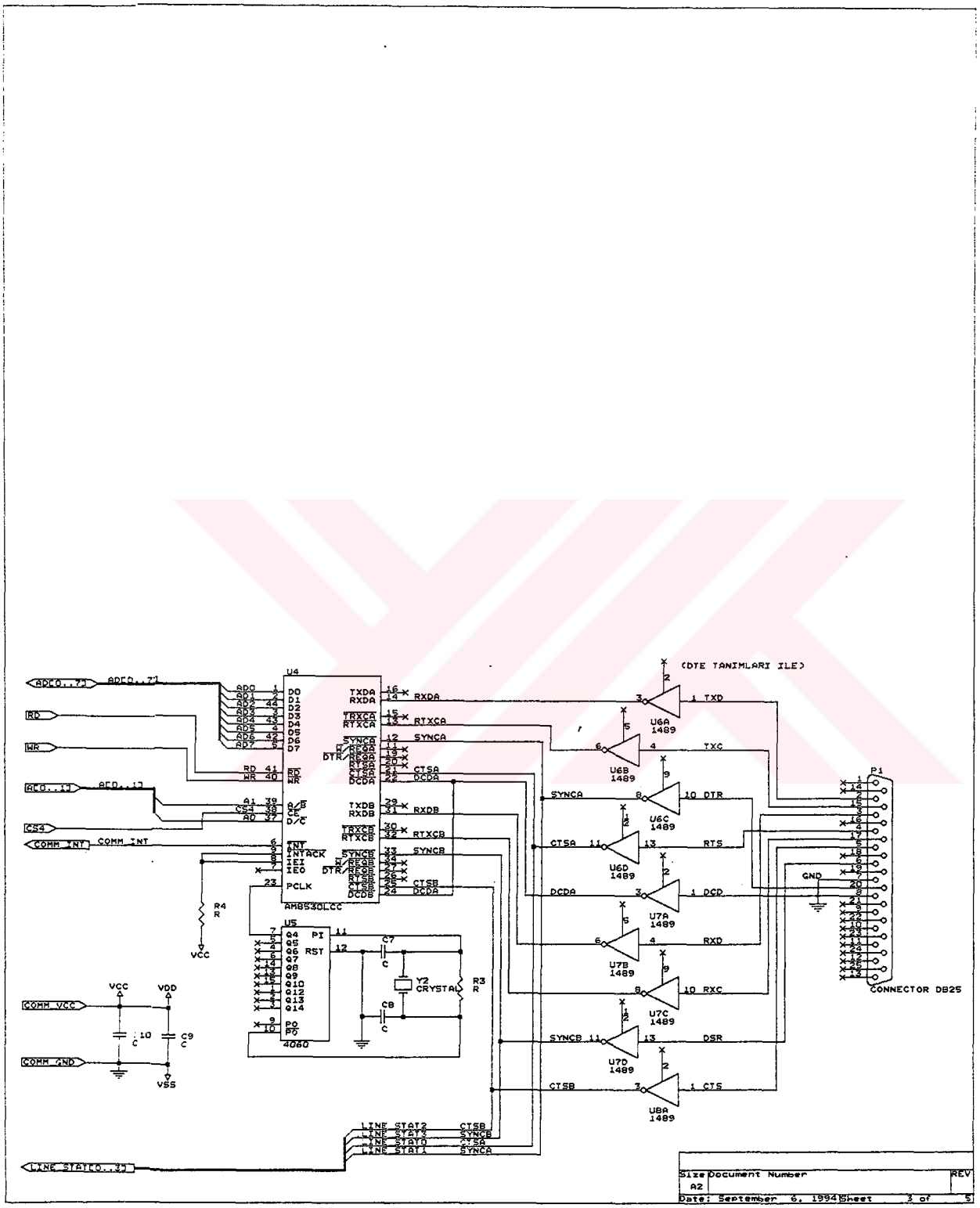
3.4. Seri İletişim Denetleyicisi

Senkron ve asenkron veri iletişimi gözleneceği için, bu veri hatlarında çalışabilecek seri iletişim denetçisi olarak üreticisi AMD olan AM85C30 SCC tümdevresi, Şekil 3.4.1' de görülen bağlantı şekliyle kullanılmıştır [2]. Denetçinin içsel yapısı, çoğullanmamış veri hatları arayüzlerinde gerekli tüm kesme ve denetleme mantık devrelerini sağlar. Arayüz devreleri aynı zamanda modem veya diğer çevre birimlerinin denetleme giriş/çıkışlarını gözlemeyi sağlar. Tüm denetleme işaretleri genel amaçlıdır; modem denetimi için kullanıldığı kadar farklı çevre birim araçlarına da uygulanabilir.

Veri işlevlerinin merkezi, içsel okuma-yazma kayıtçılarıdır. Bu kayıtçıların şartlanması tümdevrenin kişisel işlevselliğini sağlar; kayıtçı değerleri koşullanarak tümdevrenin verilen iletişim protokolünde nasıl davranacağını belirler. Tüm iletişim şekilleri yazma kayıtçılarınındaki bit değerleriyle oluşturulur. Veri alınırken veya gönderilirken okuma kayıtçıları değişebilir. Bu değişimler yazılımın nasıl davranacağını belirlerken aynı zamanda önceki koşullamalara da bağlı olarak tümdevrenin bir sonraki durum değişimini de belirler.

Her bir kanalın kayıtçı kümesi farklı okuma ve yazma kayıtçıları içerir. On adet yazma kayıtçısı denetim için kullanılmıştır. İki adedi senkron iletişimde eşzamanlama karakterini üretir. İki adedi tümdevre üzerinde hız birimini oluşturan bölme sayacının değerini oluşturur. İki yazma kayıtçısı her iki kanal için de ortaktır; biri kesme vektörü olarak diğer biri de ana kesme denetimi için kullanılmıştır. Her iki kayıtçı, her bir kanalda ortak ve her birinden ulaşılabilir durumdadır.

Altı okuma kayıtçısı durum işlevlerini gösterir; ikisi tümdevre üzerinde hız birimini oluşturan bölme sayacının değerini ve biri de alma tamponunu gösterir. Kalan iki okuma kayıtçısı her iki kanal için ortaktır; biri kesme askıya alma durumunu diğeri de kesme kaynağını vektör olarak gösterir.



Şekil 3.4.1. Seri İletişim Denetleyicisi Devre Şeması

Seri iletişim denetçisi üç ayrı giriş-çıkış şeklinden biriyle çalışır:

- Yoklama veya gözleme olarak adlandırılabilecek ilk yöntem uygulanması kolay olan yöntemdir. Bu yöntemde kesmeler yoktur. Yazılım iletişim denetçisinin durumunu yoklar; veri alınmış mı veya gönderme bitmiş mi... gibi. Yazılım, sıfırıncı okuma kayıtçısını (RR0) okuyarak denetçinin veri alma, veri gönderme ve dışsal işaret konumlarını anlayıp ilgili yordamı çağırmalıdır.

- Kesme kaynaklarını kullanan yöntemde iletişim denetçisi, istenilen durumlarda kesme işareti üretecek şekilde şartlanır. Her bir kanal için üç kesme kaynağı tanımlanabilir: Alma kaynaklı, gönderme kaynaklı ve dışsal işaret değişimi kaynaklı. Her bir kesme kaynağının etkin duruma getirilmesi yazılımın denetimi altındadır.

- Toplu aktarım olarak adlandırılan son yöntem verinin alma ve gönderme sırasında küme halinde okunup yazılabildiği yöntemdir. Bu yöntemde doğrudan bellek ulaşımını sağlayan içsel birimler şartlandığında, kesme kaynaklarında yazılımın karışması söz konusu olmadan denetçi veriyi uygun bellek alanına yazar. Ancak bu durum özel donanım tasarımı gerektirmektedir.

Bu uygulamada denetçi, yalnızca alma kesmesi üretecek şekilde, kesme kaynaklı olarak çalıştırılmaktadır.

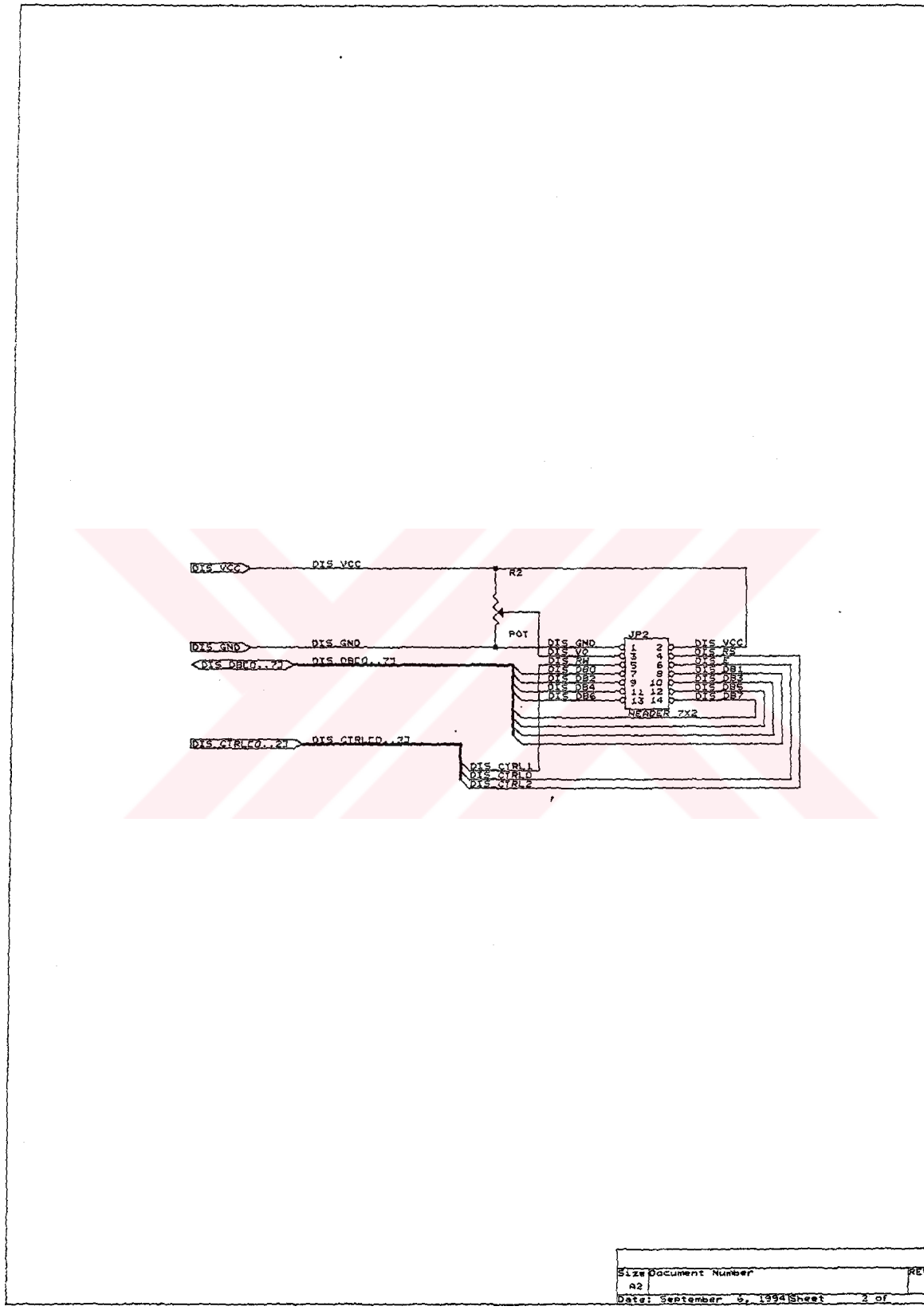
3.5. Gösterge

Gösterge, HD44780 işlemcisi temelli, tüm yazı karakterlerini gösterebilen, nokta matris LCD sürücü çıkışlıdır. Dört veya sekiz bit mikroişlemci arayüzü uyumludur. Tüm LCD yönetim komutları kendi içinde vardır. Düşük güç tüketimi nedeniyle taşınabilir pilli sistemlerde kullanılabilir.

Gösterge ile mikroişlemci arayüz tanımları iki farklı uygulama şekline sahiptir. Dört bitlik mikroişlemci veya sekiz bitlik mikroişlemci ile kullanım. Dört bitlik işlemciyle kullanımda üst dört bit anlamsızlaşır. Gösterge dört bit'lik bir işlemciye bağlanıyorsa bir komut iki evrede gerçeklenir. İlk evrede üst dört veri biti, ikinci evrede alt dört veri biti işlenir. İçsel işlemlerin devam ettiğini bildiren meşgul bayrağının okunması da iki evrede olur.Gösterge sekiz bit'lik işlemciye bağlıysa okuma ve yazma işlemleri bir evrede gerçekleştirilebilir. Gösterge arayüzünün dört veya sekiz bit oluşu yazılım tarafından gösterge

koşullama sırasında bildirilir. Tablo 3.5.1.1.' de gösterge arayüz uçları verilmiştir. Şekil 3.5.1.'de bu uygulamadaki bağlantısı görülmektedir.

Yazılımda, gösterge kullanılırken, göstergeye ilk koşullama sonrasında her ulaşmada meşgul bayrağının okunması gerekmektedir. Aksi durumda gösterge sağlıklı çalışmamaktadır.



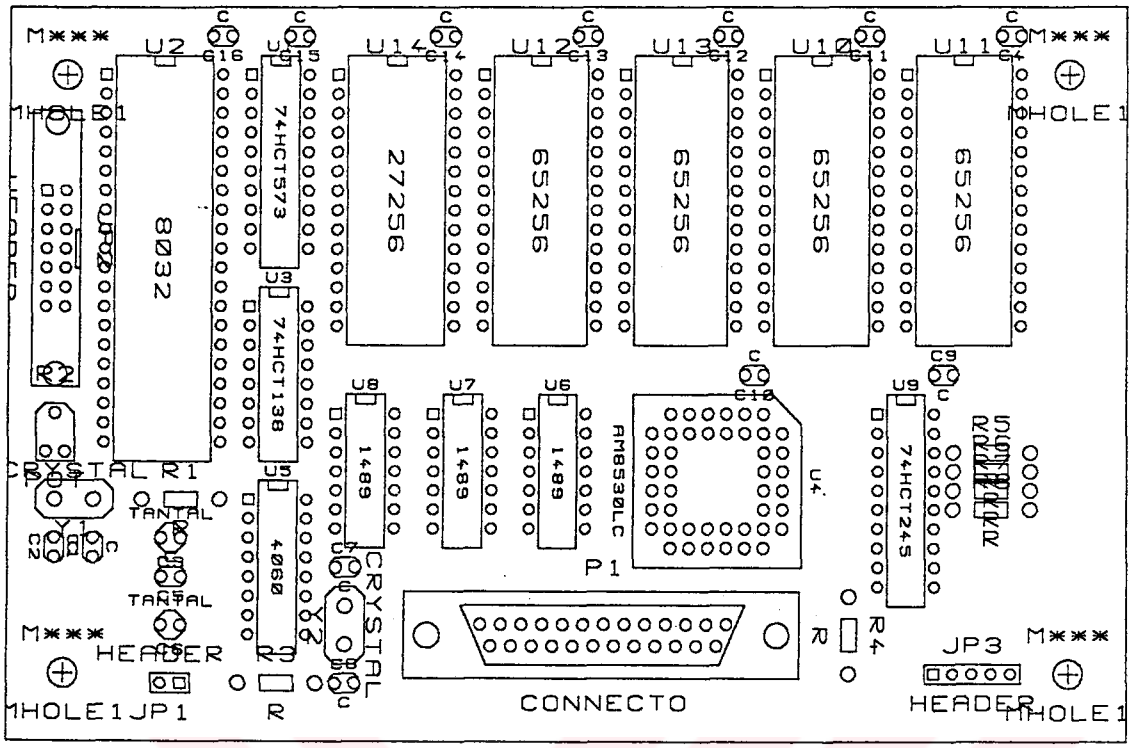
Şekil 3.5.1. Gösterge Bağlantı Devre Şeması

Bağlantı No	Sembol	Seviye	İşlev
1	VSS	-	0V
2	VDD	-	+5V
3	VO	-	LCD parlaklık
4	RS	H/L	L: Komut H: Veri
5	R/W	H/L	H: Veri Oku L: Veri Yaz
6	E	H,H -> L	Etkin İşareti
7	DB0	H/L	Veri Hattı
8	DB1	H/L	Veri Hattı
9	DB2	H/L	Veri Hattı
10	DB3	H/L	Veri Hattı
11	DB4	H/L	Veri Hattı
12	DB5	H/L	Veri Hattı
13	DB6	H/L	Veri Hattı
14	DB7	H/L	Veri Hattı

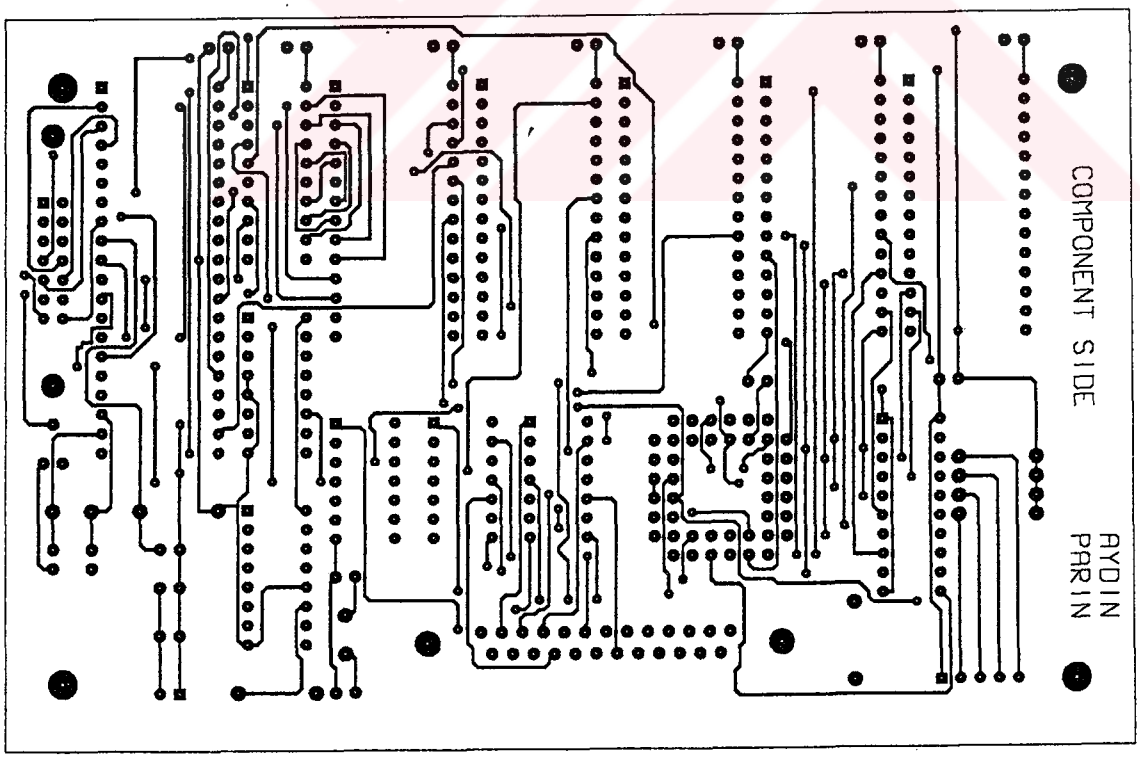
Tablo 3.3.1.1. Gösterge Arayüzü Bağlantıları

3.6. Baskı Devre

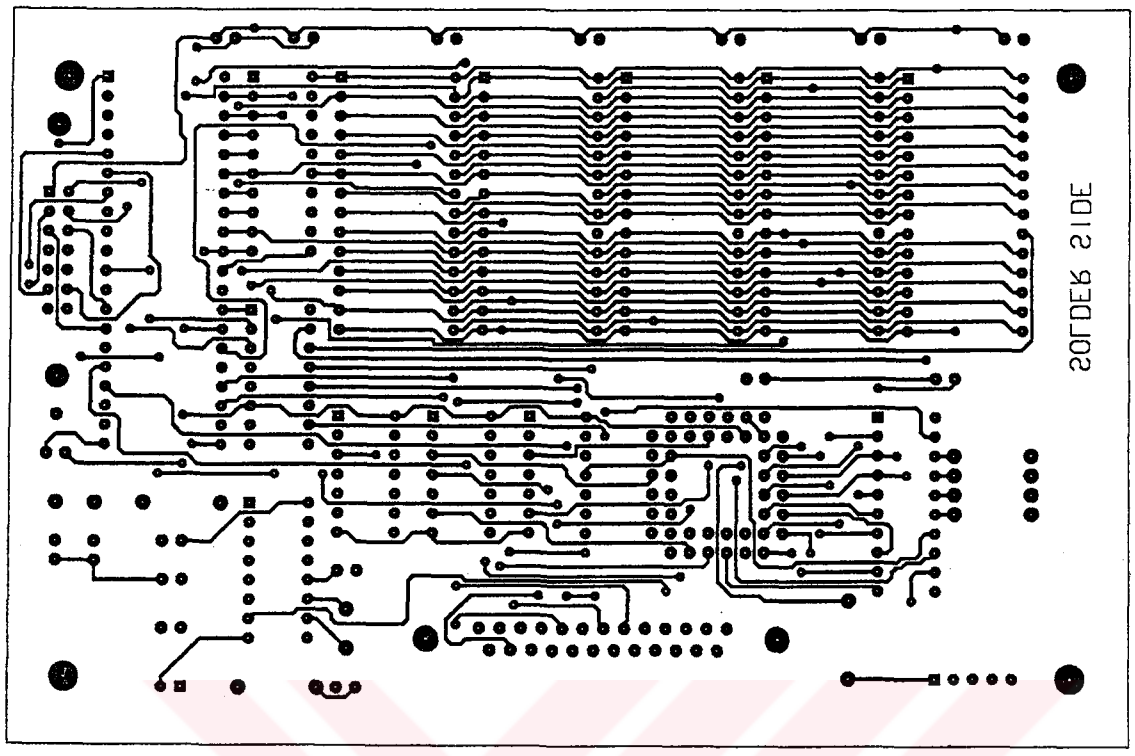
Düzeneğin baskı devrelerinin, devre şemalarının da çizildiği ORCAD ile çizildiği giriş bölümünde anlatılmıştı. Bu bölümde baskı devre çıktıları gerçek boyutlarında verilecektir. Şekil 3.6.1.' de yerleşim düzeni, Şekil 3.6.2.' de üst bakır yüzünün ve Şekil 3.6.3' de de alt bakır yüzünün çıktısı vardır.



Şekil 3.6.1. Baskı Devre Eleman Yerleşim Düzeni



Şekil 3.6.2. Üst Bakır Yüzü Baskı Devre Şeması



Şekil 3.6.3. Alt Bakır Yüzü Baskı Devre Şeması

4. DEVRENİN YAZILIM ÖZELLİKLERİ

4.1. Ana Yordam

Devre çalıştığında ilk değerlere şartlamaları yapar ve 'Menu' ye geçer [3]. Menu yapısı için kullanılan dört tuş sırasıyla 'Menu', 'Enter' veya 'Evet', 'Aşağı' veya 'Hayır' ve 'Yukarı' tuşlarıdır.
Menu alt dalları aşağıdaki gibidir.

"ANA MENU"
- "iLETiSiM"
  - "ASENKRON"
    - "HIZ"
      - "300"
      - "600"
      - "1200"
      - "2400"
      - "4800"
      - "7200"
      - "9600"
      - "12000"
      - "14400"
      - "19200"
    - "VERi UZUNLUGU"
      - "5 VERi BiTi"
      - "6 VERi BiTi"
      - "7 VERi BiTi"
      - "8 VERi BiTi"
    - "DURAK BiTi"
      - "1 DURAK BiTi"
      - "1.5 DURAK BiTi"
      - "2 DURAK BiTi"
    - "ESLiK BiTi"
      - "ESLiK BiTi YOK"
      - "ESLiK BiTi TEK"

"ESLiK BiTi CiFT"

"SENKRON"

"SDLC / HDLC"

"SDLC"

"HDLC"

"MONO / BI SYNC"

"MONO SYNC"

"BI SYNC"

"HIZ"

"300"

"600"

"1200"

"2400"

"4800"

"7200"

"9600"

"12000"

"14400"

"19200"

"SAKLAMA"

"TETiKLEME TiPi"

"DTE iLK KARAKTER"

"DCE iLK KARAKTER"

"DTE OZEL KAR."

"DCE OZEL KAR."

"ENTER TUSU"

"DTE RTS ETKEN"

"DTE CTS ETKEN"

"DTE DTR ETKEN"

"DTE DSR ETKEN"

"DTE DCD ETKEN"

"OZEL KAR. GiRiSi"

"TETiK YERi"

"BELLEK BASI"

"BELLEK ORTASI"

```
                "BELLEK SONU"
        "CALISTIR"
                "TETiK BEKLiYOR"
        "DURDUR"
                "SAKLAMA DURDU"
        "BELLEK SiL"
                "BELLEK SiLiNDi"
        "GOSTERME SEKLi"
                "DURUM KAYNAGI"
                        "RTS"
                        "CTS"
                        "DTR"
                        "DSR"
                        "DCD"
                "VERi KAYNAGI"
                        "DTE"
                        "DCE"
                "HEX / TEXT"
                        "HEXADECIMAL"
                        "TEXT"
        "BELLEK GOSTER"
"TEST"
        "RAM TESTi"
        "TUS TESTi"
        "GOSTERGE TESTi"
```

Menu yardımıyla şu işlevler gerçeklenmektedir:

Test seçeneği altında üç test vardır: Düzenek belleği test edebilmektedir. Ana işlev belleği, veri belleği ve durum belleğine sağlıklı yazılıp okunduğu denetlenmektedir. Tuş testi ile, kullanıcı tarafından tuşların doğru okundukları denetlenmektedir. Gösterge testi göstergeye ulaşımın sorunsuz olduğunu denetlemektedir.

İletişim seçeneği altında veri hattının özellikleri belirtilir. Asenkron iletişim şekli altında hız, veri biti, eşlik biti, durak biti değerleri ayarlanabilir. Senkron iletişim şekli altında senkron iletişim protokolü ve hız belirlenir.

Saklama seçeneği ile asıl ölçüm ve gözlem yapılabilen alt seçimlere ulaşılır. Burada tetikleme yöntemi belirlenir. Tetik etkin duruma getirilip saklama başlatılır. Saklama bittiğinde gösterme şekli belirlenerek saklanmış veri ve durum bilgileri izlenir. Üst satırda gösterilen durum kaynağını ve onaltılı sayı tabanında milisaniye olarak zamanı görmek mümkündür. Alt satırda ise veri kaynağını ve o anda gösterilen verinin sıra numarasını yine onaltılı sayı tabanında izlemek mümkündür. Bellek kullanıcı tarafından silinebildiği gibi, tetik etkin duruma geldiğinde yazılım tarafından da silinmektedir. Saklama bellek tükendiğinde kendiliğinden durduğu gibi istenirse kullanıcı tarafından herhangi bir anda durdurulabilir.

4.2. Dışsal Kesme

80C31'in dışsal kesmesi bu uygulamada seri iletişim denetçisinin kesme kaynağına bağlanmıştır [6]. Seri iletişim denetçisinin yalnızca alma kesmesi etkin duruma şartlanmıştır. Ancak yine de hata kaynaklarını yok etmek için kesme kaynaklarının tamamı denetlenerek gerekli işlemler yapılmaktadır. Örneğin kesme, gönderme kaynaklı ise denetçiye gönderme kesmesini askıya alan komut yazılır.

Veri hattının DTE alma işareti AM85C30'un A kanalına, DCE alma işareti B kanalına bağlıdır. Herhangibir veri akışı olduğunda kesme üretilir. Yazılım bu kesmeye girdiğinde kesme kaynaklarını yoklar ve gerekli işlemleri yapar. Örneğin, kesme A kanalının almasından kaynaklanmışsa, alınan veri okunarak geçici bir yere yazılır. Saklama etkin ise, aynı anda A kanalından alma gerçekleştiğini belirten bir bayrak kaldırılır. Geçici veri ve bayrak, zamanlama kesmesi tarafından saklama işlevlerinde kullanılır.

4.3. Sayıcı Zamanlayıcı Kesmesi

Kesme her 1 milisaniyede oluşur. 80C31'in sıfırıncı sayıcı-zamanlayıcısı (T0),sekiz bit kendiliğinden yüklemeli zamanlayıcı işlevinde koşullanır. Zamanlayıcı kesme kaynağı etkin duruma getirilir. Yazılım koşmaya başladığı andan itibaren, 1 milisaniyelik zaman aralıklarıyla kesme çalışır.

Kesmenin iki ayrı işlevi vardır. İlk işlevi tuş bilgisinin oluşturulmasıdır. Bu amaçla üç durumlu bir altyordam koşmaktadır. Birinci durumda ana yordamın tuş bilgisini okuması beklenir. İkinci durum tuşa basılmasını bekler. Tuşa basıldığında bu bilgiyi saklar ve üçüncü duruma geçer. Üçüncü durum ise tuşun bırakılmasını bekler. Tuş bırakıldığında, eğer geçerli bir tuş bilgisi ise bu bilgiyi anayordama iletir. Aynı anda bir kaç tuşa basılmış ise görmezden gelerek ikinci duruma geri döner.

Kesmenin ikinci işlevi saklama ile ilgili olan işlemlerdir. Bu alt yordam da üç durumlu olarak yazılmıştır. Durum bilgisi tetik durumuna bağlanmıştır: Tetik beklemede, saklama etkin, saklama durdu... Tetik beklenen durumda, kullanıcının verdiği tetik şeklini değerlendirerek saklamanın başlama noktasını belirler. Saklama başladıktan sonra veri ve durum bilgilerini belleğe yazar. Bellek dolduğunda saklama işlemini sonlandırır.

Veri akış hızının arttığı durumda zamanlayıcı kesme aralıklarını sıklaştırmak gerekir. Her karakterde en az bir zamanlayıcı kesmesi gelmesi sağlanmalıdır. Kesme aralıklarını, kesme tanım dosyasındaki TH1 değerini azaltarak sıklaştırmak mümkündür.

SONUÇ

Devre tasarımı 'kısıtlı uygulama alanlarında kullanılabilir' düşüncesiyle gerçekleştirildi. Uygulamanın gerektirdiği kadar işlevi olan, basit bir ölçme ve gözlemleme düzeneği olarak, istenen yetenekte çalıştı. Devrenin veri hattının hızına göre üst yetenek sınırı belirlenmedi. Ancak, bir kesme içinde harcanan zaman ölçülerek kuramsal üst sınır bulunabilir.

Bu düzenek, yazılımının geliştirilmesiyle, özellikle geçerliliği giderek artan, dünya üzerine yayılmış bilgisayar ağı protokollerini tanır duruma getirilebilir. Örneğin, ek bir yazılımla, Türkiye'de de yaygınlaşan X.25 iletişim protokolünü tanıması sağlanabilir.

KAYNAKLAR

1 - 9400 Series Protocol Analyzers Programming Guide, 1991, GN Navtel

2 - Am8530/Am85C30 Serial Communication Controller Technical Manual, 1992, Advanced Micro Devices.

3 - AMP NetConnect Open Wiring Systems, 1992, Catalog 82164

4 - Data Structures Usıng C, 1990, Aaron M. Tanenbaum, Yedidah Langsam, Moshe J. Augenstein, Prentice Hall, ISBN: 0-13-200411-9

5 - Data Networks Concepts Theory and Practice, Uyless D. Black, 1989, Prentice Hall, ISBN: 0-13-198599-x

6 - Eight-Bit 80C51 Embedded Processors, 1990, Advanced Micro Devices.

7 - Serial Communications: A C++ Developer's Guide, 1992, Mark Nelson, M&T Books, ISBN 1-55851-281-0

8 - The Modem Reference, 1992, Michael A. Banks,Brady Publishing, ISBN: 1-56686-027-x.

EK

C DİLİ YAZILIMLARI

```
/*****************************************************************************/
/*                                                                           */
/* TIME0INT.C                                                                */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
#define _TIME0INT_C

/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\time0int.h>
#include <c:\project\tez\software\mcs51\include\ext0int.h>

/*****************************************************************************/
typedef enum    {
                e_kbd_state1,
                e_kbd_state2,
                e_kbd_state3
                } t_kbd_state;

/*****************************************************************************/
/*  Timer Interrupt var's                                                    */
/*****************************************************************************/
static   byte                    idata          t;      /* for macro activation*/
static   word                    idata          ti;     /* for macro activation*/

/*****************************************************************************/
extern          bit                                     CHA_rx_char_ready;
extern          bit                                     CHB_rx_char_ready;

t_key                                   idata   pressed_key=e_key_none;
static          t_kbd_state     idata   kbd_state=e_kbd_state1;
static          t_key           idata   old_key=e_key_none;
static          byte            idata   key_int_time=0;
static          byte            idata   am8530_status;
static          byte            idata   temp_key;
static          byte            idata   pressed_count = 0;
/*****************************************************************************/
#pragma         OPTIMIZE(5,SPEED)

/*****************************************************************************/
/*      timerint: timer interrupt                                            */
/*****************************************************************************/
#ifdef CPU_8051
        void i_timer_int0() interrupt 1         /*  using 1 use registerbank 1 for interrupt  */
```

```
#endif
        {
        TR0 = 0;
        TH0 = TH0_initial_value;
        TL0 = TL0_initial_value;
        TR0 = 1;

        m_read_status_port(temp_key);

        m_rd_reg_e_RR0(CHA,am8530_status);

        am8530_status = (am8530_status & (byte)e_dcd) | ((temp_key^0xFF) & 0xF0); /* add to status
DCD */

        key_int_time++;
        if (key_int_time _EQU_ m_key_int_period)
                {
                temp_key = (temp_key & 0x0F);
                key_int_time = 0;
                if (kbd_state _EQU_ e_kbd_state1)
                        {
                        if      (_NOT_ pressed_key)             /* main read pressed_key */
                                {
                                if      (_NOT_ temp_key)        /* no key pressed */
                                        {
                                        kbd_state = e_kbd_state2;
                                        }
                                else                                    /* a key pressed */
                                        {
                                        old_key = (t_key)temp_key;
                                        kbd_state = e_kbd_state3;
                                        }
                                }
                        }
                else if (kbd_state _EQU_ e_kbd_state2)
                        {
                        if      (temp_key)                              /* a key pressed */
                                {
                                old_key = (t_key)temp_key;
                                kbd_state = e_kbd_state3;
                                }
                        }
                else if (kbd_state _EQU_ e_kbd_state3)
                        {
                        if      (_NOT_ temp_key)                /* key released */
                                {                                                       /*      check
validity */
                                pressed_count = 0;
                                if (old_key _EQU_ e_key_up_code)
                                        {
                                        pressed_key = e_key_up;
                                        kbd_state = e_kbd_state1;
                                        }
                                else if (old_key _EQU_ e_key_down_code)
                                        {
```

```
                        pressed_key = e_key_down;
                        kbd_state = e_kbd_state1;
                        }
                    else if (old_key _EQU_ e_key_enter_code)
                        {
                        pressed_key = e_key_enter;
                        kbd_state = e_kbd_state1;
                        }
                    else if (old_key _EQU_ e_key_menu_code)
                        {
                        pressed_key = e_key_menu;
                        kbd_state = e_kbd_state1;
                        }
                    else                                    /* key invalid (multikey)
*/
                        {
                        kbd_state = e_kbd_state2;
                        }
                    }
                else                /* still key not released */
                    {
                    pressed_count++;
                    if (pressed_count > 10)
                        {
                        pressed_count = 11;
                        if (old_key _EQU_ e_key_up_code)
                            {
                            pressed_key = e_key_up;
                            }
                        else if (old_key _EQU_ e_key_down_code)
                            {
                            pressed_key = e_key_down;
                            }
                        else if (old_key _EQU_ e_key_enter_code)
                            {
                            pressed_key = e_key_enter;
                            }
                        else if (old_key _EQU_ e_key_menu_code)
                            {
                            pressed_key = e_key_menu;
                            }
                        }
                    }
                }
            }

        /* store trigger waiting */
        if (system_parameters.store_stat _EQU_ e_store_stat_waiting)
            {
            if (system_parameters.trigger_type _EQU_ e_trigger_dte_first_char)
                {
                if (CHA_rx_char_ready)
                    {
                    system_parameters.store_stat = e_store_stat_runing;
```

```
                }
        }
    else if (system_parameters.trigger_type _EQU_ e_trigger_dce_first_char)
        {
        if (CHB_rx_char_ready)
                {
                system_parameters.store_stat = e_store_stat_runing;
                }
        }
    else if (system_parameters.trigger_type _EQU_ e_trigger_dte_special_char)
        {
        if (CHA_rx_char_ready)
                {
                if (CHA_last_received_byte _EQU_
system_parameters.trigger_special_char)
                        {
                        system_parameters.store_stat = e_store_stat_runing;
                        }
                }
        }
    else if (system_parameters.trigger_type _EQU_ e_trigger_dce_special_char)
        {
        if (CHB_rx_char_ready)
                {
                if (CHB_last_received_byte _EQU_
system_parameters.trigger_special_char)
                        {
                        system_parameters.store_stat = e_store_stat_runing;
                        }
                }
        }
    else if (system_parameters.trigger_type _EQU_ e_trigger_enter_key)
        {
        if (pressed_key _EQU_ e_key_enter)
                {
                system_parameters.store_stat = e_store_stat_runing;
                }
        }
    else if (system_parameters.trigger_type _EQU_ e_trigger_rts)
        {
        if (am8530_status & e_stat_rts_mask)
                {
                system_parameters.store_stat = e_store_stat_runing;
                }
        }
    else if (system_parameters.trigger_type _EQU_ e_trigger_cts)
        {
        if (am8530_status & e_stat_cts_mask)
                {
                system_parameters.store_stat = e_store_stat_runing;
                }
        }
    else if (system_parameters.trigger_type _EQU_ e_trigger_dtr)
        {
        if (am8530_status & e_stat_dtr_mask)
```

```
                        {
                        system_parameters.store_stat = e_store_stat_runing;
                        }
                }
        else if (system_parameters.trigger_type _EQU_ e_trigger_dsr)
                {
                if (am8530_status & e_stat_dsr_mask)
                        {
                        system_parameters.store_stat = e_store_stat_runing;
                        }
                }
        else if (system_parameters.trigger_type _EQU_ e_trigger_dcd)
                {
                if (am8530_status & e_stat_dcd_mask)
                        {
                        system_parameters.store_stat = e_store_stat_runing;
                        }
                }
        }
/* store runing */
else if (system_parameters.store_stat _EQU_ e_store_stat_runing)
        {
        if (CHA_rx_char_ready)
                {
                CHA_rx_char_ready = false;

m_write_data_ram(system_parameters.data_end_index,CHA_last_received_byte);

m_inc_circular_buffer_index(system_parameters.data_end_index,DATA_RAM_LENGTH);
                am8530_status = (am8530_status | e_stat_CHA_rx_char_mask);
                }
        if (CHB_rx_char_ready)
                {
                CHB_rx_char_ready = false;

m_write_data_ram(system_parameters.data_end_index,CHB_last_received_byte);

m_inc_circular_buffer_index(system_parameters.data_end_index,DATA_RAM_LENGTH);
                am8530_status = (am8530_status | e_stat_CHB_rx_char_mask);
                }
        m_write_status_ram(system_parameters.status_end_index,am8530_status);

m_inc_circular_buffer_index(system_parameters.status_end_index,STATUS_RAM_LENGTH);


m_inc_circular_buffer_index(system_parameters.status_end_index,STATUS_RAM_LENGTH);
        if (system_parameters.status_end_index _EQU_
system_parameters.status_start_index)
                {
                system_parameters.store_stat = e_store_stat_stoped;
                }

m_dec_circular_buffer_index(system_parameters.status_end_index,STATUS_RAM_LENGTH)
;
        }
```

```
        }

/*****************************************************************************/
/*      timer_init: initialize timer0                                        */
/*****************************************************************************/
void p_timer0_init()
        {
        TH0 = TH0_initial_value;
        TMOD |= T0_mode;                        /* timer 0 mode 2: 8-Bit reload      */
        TR0 = 1;
        ET0 = 1;                                /* enable T0 interrupts */
        }

/*****************************************************************************/
#pragma         OPTIMIZING_LEVEL

/*****************************************************************************/
/*                                                                           */
/* MENU.C                                                                    */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
#define  _MENU_C

/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
#include <c:\project\tez\software\mcs51\include\menu.h>
#include <c:\project\tez\software\mcs51\include\sccutil.h>

/*****************************************************************************/
void p_menu_state_menu_main(void);
void p_menu_state_menu_comm(void);
void p_menu_state_menu_async(void);
void p_menu_state_menu_sync(void);
void p_menu_state_menu_baud_rate(void);
void p_menu_state_menu_data_bits(void);
void p_menu_state_menu_stop_bits(void);
void p_menu_state_menu_parity_bit(void);
void p_menu_state_menu_sync_protocol(void);
void p_menu_state_menu_sync_type(void);
void p_menu_state_menu_store(void);
void p_menu_state_menu_store_runing(void);
void p_menu_state_menu_trigger_type(void);
void p_menu_state_menu_special_char(void);
void p_menu_state_menu_trigger_time(void);
void p_menu_state_menu_display_type(void);
void p_menu_state_menu_status_source(void);
void p_menu_state_menu_data_source(void);
void p_menu_state_menu_data_char_type(void);
void p_menu_state_menu_display_buffer(void);
void p_menu_state_menu_test(void);
void p_menu_state_menu_test_ram(void);
```

```
void p_menu_state_menu_test_key(void);
void p_menu_state_menu_test_display(void);
byte f_choose_menu_item(t_menu_str      *head_str,
                                        t_menu_str      *item_array,
                                        byte            total_item,
                                        byte            active_item);
void p_call_menu_state(t_void_arg_proc_ptr              new_state_ptr);
void p_return_from_menu_state(void);
void p_return_to_menu_main_state(void);
void p_display_ram_data(void);
void p_inc_indexes(void);
void p_dec_indexes(void);


/*****************************************************************************/
static  byte                    idata           t;      /* for macro activation*/
static  word                    idata           ti;     /* for macro activation*/

/*****************************************************************************/
t_void_arg_proc_ptr             idata           menu_state_ptr;
byte                            idata           choosed_item;
byte                            idata           ram_actual_byte;
byte                            idata           test_val;
word                            idata           index;

t_menu_stack                    xdata           menu_stack = {{0},

        {p_menu_state_menu_main}};
t_system_parameters             xdata           system_parameters={
                                                (byte)e_comm_type_async,
                                                (byte)e_9600_baud,
                                                (byte)e_8_data_bits,
                                                {(byte)e_1_stop_bit,(byte)e_none_parity},
                                                (byte)e_trigger_dte_first_char,
                                                (byte)e_time_start_of_buffer,
                                                (byte)e_store_stat_stoped,
                                                (byte)e_display_rts,
                                                (byte)e_display_DTE_data,
                                                (byte)e_display_data_hex,
                                                (byte)0x00,
                                                0x0000,
                                                0x0000,
                                                0x0000,
                                                0x0000
                                                };

/*****************************************************************************/
code t_menu_str  menu_main_menu[1] = {"ANA MENU"};

code t_menu_str  menu_main_sub[3] = {{"iLETiSiM"},
                                                        {"SAKLAMA"},
                                                        {"TEST"}};

code t_menu_str  menu_comm_sub[2] = {{"ASENKRON"},          /* set system_parameters.protocol
*/
```

```
                                                                {"SENKRON"}};

code t_menu_str  menu_async_sub[4] = {{"HIZ"},
                                                                {"VERi UZUNLUGU"},
                                                                {"DURAK BiTi"},
                                                                {"ESLiK BiTi"}};

code t_menu_str  menu_sync_sub[3] = {{"SDLC / HDLC"},
                                                                {"MONO / BI SYNC"},
                                                                {"HIZ"}};

code t_menu_str  menu_baud_rate[s_max_baud] = { {"300"},        /*                  set
system_parameters.baud_rate */

        {"600"},

        {"1200"},

        {"2400"},

        {"4800"},

        {"7200"},

        {"9600"},

        {"12000"},

        {"14400"},

        {"19200"}};

code t_menu_str  menu_data_bits[4] = {    {"5 VERi BiTi"},      /*                  set
system_parameters.data_bits */
                                                                {"6     VERi
BiTi"},
                                                                {"7     VERi
BiTi"},
                                                                {"8     VERi
BiTi"}};

code t_menu_str  menu_stop_bits[3] = {    {"1   DURAK BiTi"},   /*                  set
system_parameters.comm_setup.async.stop_bits */
                                                                {"1.5   DURAK
BiTi"},
                                                                {"2     DURAK
BiTi"}};

code t_menu_str  menu_parity_bit[3] = {   {"ESLiK BiTi YOK"},   /*                  set
system_parameters.comm_setup.async.parity */
                                                                {"ESLiK BiTi TEK"},
                        /* odd */
                                                                {"ESLiK BiTi CiFT"}};
                /* even */
```

```
code t_menu_str  menu_sync_protocol[2] = {{"SDLC"},                    /*        set
system_parameters.comm_setup.sync.sync_protocol */
                                                                       {"HDLC"}};

code t_menu_str  menu_sync_type[2] = {{"MONO SYNC"},                   /*        set
system_parameters.comm_setup.sync.sync_type */
                                                                {"BI SYNC"}};

code t_menu_str  menu_store_sub[8] = {     {"TETiKLEME TiPi"},
                                                                       {"OZEL          KAR.
GiRiSi"},
                                                                       {"TETiK YERi"},
                                                                       {"CALISTIR"},
                                                                       {"DURDUR"},
                                                                       {"BELLEK SiL"},
                                                                       {"GOSTERME
SEKLi"},
                                                                       {"BELLEK
GOSTER"}};

code t_menu_str  store_wait[1] = {"TETiK BEKLiYOR"};
code t_menu_str  store_run[1] = {"SAKLAMA AKTiF"};
code t_menu_str  store_stop[1] = {"SAKLAMA DURDU"};
code t_menu_str  store_clear[1] = {"BELLEK SiLiNDi"};

code t_menu_str  menu_trigger_type[10] = {         {"DTE iLK KARAKTER"},
                                                                       {"DCE           iLK
KARAKTER"},
                                                                       {"DTE OZEL KAR."},
                                                                       {"DCE OZEL KAR."},
                                                                       {"ENTER TUSU"},
                                                                       {"DTE RTS ETKEN"},
                                                                       {"DTE CTS ETKEN"},
                                                                       {"DTE DTR ETKEN"},
                                                                       {"DTE DSR ETKEN"},
                                                                       {"DTE           DCD
ETKEN"}};

code t_menu_str  menu_trigger_time[3] = { {"BELLEK BASI"},
                                                                       {"BELLEK ORTASI"},
                                                                       {"BELLEK SONU"}};

code t_menu_str  menu_display_sub[3] = {{"DURUM KAYNAGI"},
                                                                              {"VERi
KAYNAGI"},
                                                                              {"HEX      /
TEXT"}};

code t_menu_str  menu_display_status[5] = {{"RTS"},
        {"CTS"},
        {"DTR"},
        {"DSR"},
```

```
        {"DCD"}};

code t_menu_str  menu_display_data[2] = {{"DTE"},
                                                                {"DCE"}};

code t_menu_str  menu_display_data_char_type[2] = {{"HEXADECIMAL"},
                                                                {"TEXT"}};

code t_menu_str  menu_test_sub[3] = {{"RAM TESTi"},
                                                                {"TUS TESTi"},
                                                                {"GOSTERGE
TESTi"}};

/*****************************************************************************/
#pragma         OPTIMIZE(5,SPEED)

/*****************************************************************************/
/*  menu: menu program                                          */
/*****************************************************************************/
void menu()
        {
        menu_state_ptr=p_menu_state_menu_main;
        while(true)
                {
                (*menu_state_ptr)();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_main(void)
        {
        static    byte      xdata     menu_main_choosed_item = e_menu_main_comm;

        choosed_item =  f_choose_menu_item(menu_main_menu,
                                                menu_main_sub,
                                                m_size_of_item(menu_main_sub),
                                                menu_main_choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                menu_main_choosed_item = choosed_item;
                if (choosed_item _EQU_ e_menu_main_comm)
                        {
                        p_call_menu_state(p_menu_state_menu_comm);
                        }
                else if (choosed_item _EQU_ e_menu_main_store)
                        {
                        /* re initialize SCC */
                        p_initialize_8530_with_setup();
                        p_call_menu_state(p_menu_state_menu_store);
                        }
                else if (choosed_item _EQU_ e_menu_main_test)
                        {
                        p_call_menu_state(p_menu_state_menu_test);
```

```
                }
        }
    else if (choosed_item & e_up_pressed_mask)
        {
        }
    else if (choosed_item & e_menu_pressed_mask)
        {
        }
    }

/*****************************************************************************/
void p_menu_state_menu_comm(void)
    {
    choosed_item = system_parameters.comm_type;

    choosed_item =  f_choose_menu_item(&menu_main_sub[e_menu_main_comm],
                                       menu_comm_sub,
                                       m_size_of_item(menu_comm_sub),
                                       choosed_item);
    if (choosed_item & e_item_choosed_mask)
        {
        choosed_item = (choosed_item & e_get_item_mask);
        system_parameters.comm_type = (t_comm_type) choosed_item;
        if (choosed_item _EQU_ e_comm_type_async)
            {
            p_call_menu_state(p_menu_state_menu_async);
            }
        else if (choosed_item _EQU_ e_comm_type_sync)
            {
            system_parameters.data_bits = e_8_data_bits;
            p_call_menu_state(p_menu_state_menu_sync);
            }
        }
    else if (choosed_item & e_up_pressed_mask)
        {
        p_return_from_menu_state();
        }
    else if (choosed_item & e_menu_pressed_mask)
        {
        p_return_to_menu_main_state();
        }
    }

/*****************************************************************************/
void p_menu_state_menu_async(void)
    {
    static    byte    xdata    menu_async_choosed_item = e_menu_async_speed;

    choosed_item =  f_choose_menu_item(&menu_comm_sub[e_comm_type_async],
                                       menu_async_sub,
                                       m_size_of_item(menu_async_sub),
                                       menu_async_choosed_item);
    if (choosed_item & e_item_choosed_mask)
        {
        choosed_item = (choosed_item & e_get_item_mask);
```

```
                menu_async_choosed_item = choosed_item;
                if (choosed_item _EQU_ e_menu_async_speed)
                        {
                        p_call_menu_state(p_menu_state_menu_baud_rate);
                        }
                else if (choosed_item _EQU_ e_menu_async_data_bits)
                        {
                        p_call_menu_state(p_menu_state_menu_data_bits);
                        }
                else if (choosed_item _EQU_ e_menu_async_stop_bits)
                        {
                        p_call_menu_state(p_menu_state_menu_stop_bits);
                        }
                else if (choosed_item _EQU_ e_menu_async_parity)
                        {
                        p_call_menu_state(p_menu_state_menu_parity_bit);
                        }
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_sync(void)
        {
        static  byte    xdata   menu_sync_choosed_item = e_menu_sync_protocol;

        choosed_item =  f_choose_menu_item(&menu_comm_sub[e_comm_type_sync],
                                           menu_sync_sub,
                                           m_size_of_item(menu_sync_sub),
                                           menu_sync_choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                menu_sync_choosed_item = choosed_item;
                if (choosed_item _EQU_ e_menu_sync_protocol)
                        {
                        p_call_menu_state(p_menu_state_menu_sync_protocol);
                        }
                else if (choosed_item _EQU_ e_menu_sync_type)
                        {
                        p_call_menu_state(p_menu_state_menu_sync_type);
                        }
                else if (choosed_item _EQU_ e_menu_sync_speed)
                        {
                        p_call_menu_state(p_menu_state_menu_baud_rate);
                        }
                }
        else if (choosed_item & e_up_pressed_mask)
```

```
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_baud_rate(void)
        {
        choosed_item = system_parameters.baud_rate;

        choosed_item =      f_choose_menu_item(&menu_async_sub[e_menu_async_speed],  /*  or
&menu_sync_sub[e_menu_sync_speed]*/
                                                menu_baud_rate,
                                                m_size_of_item(menu_baud_rate),
                                                choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.baud_rate = (t_baud_rate) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_data_bits(void)
        {
        choosed_item = system_parameters.data_bits;

        choosed_item =  f_choose_menu_item(&menu_async_sub[e_menu_async_data_bits],
                                                menu_data_bits,
                                                m_size_of_item(menu_data_bits),
                                                choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.data_bits = (t_data_bits) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
```

```
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_stop_bits(void)
        {
        choosed_item = system_parameters.comm_setup.async.stop_bits;

        choosed_item =  f_choose_menu_item(&menu_async_sub[e_menu_async_stop_bits],
                                                menu_stop_bits,
                                                m_size_of_item(menu_stop_bits),
                                                choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.comm_setup.async.stop_bits = (t_stop_bits) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_parity_bit(void)
        {
        choosed_item = system_parameters.comm_setup.async.parity;

        choosed_item =  f_choose_menu_item(&menu_async_sub[e_menu_async_parity],
                                                menu_parity_bit,
                                                m_size_of_item(menu_parity_bit),
                                                choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.comm_setup.async.parity = (t_parity) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }
```

```
/*****************************************************************************/
void p_menu_state_menu_sync_protocol(void)
        {
        choosed_item = system_parameters.comm_setup.sync.sync_protocol;

        choosed_item =  f_choose_menu_item(&menu_sync_sub[e_menu_sync_protocol],
                                           menu_sync_protocol,
                                           m_size_of_item(menu_sync_protocol),
                                           choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.comm_setup.sync.sync_protocol      =      (t_sync_protocol)
choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_sync_type(void)
        {
        choosed_item = system_parameters.comm_setup.sync.sync_type;

        choosed_item =  f_choose_menu_item(&menu_sync_sub[e_menu_sync_protocol],
                                           menu_sync_type,
                                           m_size_of_item(menu_sync_type),
                                           choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.comm_setup.sync.sync_type = (t_sync_type) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_store(void)
        {
        static    byte      xdata     menu_store_choosed_item = e_store_trigger_type;
```

```
choosed_item = f_choose_menu_item(&menu_main_sub[e_menu_main_store],
                                  menu_store_sub,
                                  m_size_of_item(menu_store_sub),
                                  menu_store_choosed_item);
if (choosed_item & e_item_choosed_mask)
        {
        choosed_item = (choosed_item & e_get_item_mask);
        menu_store_choosed_item = choosed_item;
        if (choosed_item _EQU_ e_store_trigger_type)
                {
                p_call_menu_state(p_menu_state_menu_trigger_type);
                }
        else if (choosed_item _EQU_ e_store_enter_special_char)
                {
                p_call_menu_state(p_menu_state_menu_special_char);
                }
        else if (choosed_item _EQU_ e_store_trigger_time)
                {
                p_call_menu_state(p_menu_state_menu_trigger_time);
                }
        else if (choosed_item _EQU_ e_store_run)
                {
                system_parameters.data_start_index = 0x0000;
                system_parameters.data_end_index = 0x0000;
                system_parameters.status_start_index = 0x0000;
                system_parameters.status_end_index = 0x0000;
                system_parameters.store_stat = e_store_stat_waiting;        /*  runs  for
trigger */
                p_call_menu_state(p_menu_state_menu_store_runing);
                }
        else if (choosed_item _EQU_ e_store_stop)
                {
                system_parameters.store_stat = e_store_stat_stoped;
                m_clear_display();
                p_line_display(1,(byte*)store_stop[0],strlen((byte*)store_stop[0]));
                p_delay(1000);
                }
        else if (choosed_item _EQU_ e_store_clear)
                {
                system_parameters.data_start_index = 0x0000;
                system_parameters.data_end_index = 0x0000;
                system_parameters.status_start_index = 0x0000;
                system_parameters.status_end_index = 0x0000;
                system_parameters.store_stat = e_store_stat_stoped;
                m_clear_display();
                p_line_display(1,(byte*)store_clear[0],strlen((byte*)store_clear[0]));
                p_delay(1000);
                }
        else if (choosed_item _EQU_ e_store_display_type)
                {
                p_call_menu_state(p_menu_state_menu_display_type);
                }
        else if (choosed_item _EQU_ e_store_display)
                {
```

```
                        p_call_menu_state(p_menu_state_menu_display_buffer);
                        }
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_store_runing(void)
        {
        m_clear_display();
        while(true)
                {
                if (system_parameters.store_stat _EQU_ e_store_stat_waiting)
                        {
                        p_line_display(1,(byte*)store_wait[0],strlen((byte*)store_wait[0]));
                        }
                if (system_parameters.store_stat _EQU_ e_store_stat_runing)
                        {
                        p_line_display(1,(byte*)store_run[0],strlen((byte*)store_run[0]));
                        }
                if (system_parameters.store_stat _EQU_ e_store_stat_stoped)
                        {
                        p_line_display(1,(byte*)store_stop[0],strlen((byte*)store_stop[0]));
                        }
                if (pressed_key _EQU_ e_key_up)
                        {
                        pressed_key = e_key_none;
                        p_return_from_menu_state();
                        break;
                        }
                if (pressed_key _EQU_ e_key_menu)
                        {
                        pressed_key = e_key_none;
                        p_return_to_menu_main_state();
                        break;
                        }
                }
        }

/*****************************************************************************/
void p_menu_state_menu_trigger_type(void)
        {
        choosed_item = system_parameters.trigger_type;

        choosed_item = f_choose_menu_item(&menu_store_sub[e_store_trigger_type],
                                          menu_trigger_type,
                                          m_size_of_item(menu_trigger_type),
                                          choosed_item);
```

```
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.trigger_type = (t_trigger_type) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_special_char(void)
        {

        code    t_menu_str              special_head[1] = {"OZEL KARAKTER"};
        code    t_menu_str              hex_head[1] = {"0x"};
        choosed_item = system_parameters.trigger_special_char;

        p_line_display(2,(byte*)special_head[0],strlen((byte*)special_head[0]));
        p_line_display(2,(byte*)hex_head[0],strlen((byte*)hex_head[0]));
        p_display_at(2,3,p_hex_str(choosed_item),2);

        while(true)
                {
                if (pressed_key _EQU_ e_key_up)
                        {
                        pressed_key = e_key_none;
                        choosed_item = ((choosed_item+1)&0xFF);
                        p_display_at(2,3,p_hex_str(choosed_item),2);
                        }
                else if (pressed_key _EQU_ e_key_down)
                        {
                        pressed_key = e_key_none;
                        choosed_item = ((choosed_item-1)&0xFF);
                        p_display_at(2,3,p_hex_str(choosed_item),2);
                        }
                else if (pressed_key _EQU_ e_key_enter)
                        {
                        pressed_key = e_key_none;
                        system_parameters.trigger_special_char = choosed_item;
                        p_return_from_menu_state();
                        break;
                        }
                else if (pressed_key _EQU_ e_key_menu)
                        {
                        pressed_key = e_key_none;
                        p_return_to_menu_main_state();
                        break;
                        }
```

```
                }
        }

/*****************************************************************************/
void p_menu_state_menu_trigger_time(void)
        {
        choosed_item = system_parameters.trigger_time;

        choosed_item =  f_choose_menu_item(&menu_store_sub[e_store_trigger_time],
                                           menu_trigger_time,
                                           m_size_of_item(menu_trigger_time),
                                           choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.trigger_time = (t_trigger_time) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_display_type(void)
        {
        static    byte      xdata     menu_display_type_choosed_item = e_display_status_source;

        choosed_item =  f_choose_menu_item(&menu_store_sub[e_store_display_type],
                                           menu_display_sub,
                                           m_size_of_item(menu_display_sub),
                                           menu_display_type_choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                menu_display_type_choosed_item = choosed_item;
                if (choosed_item _EQU_ e_display_status_source)
                        {
                        p_call_menu_state(p_menu_state_menu_status_source);
                        }
                else if (choosed_item _EQU_ e_display_data_source)
                        {
                        p_call_menu_state(p_menu_state_menu_data_source);
                        }
                else if (choosed_item _EQU_ e_display_data_char_type)
                        {
                        p_call_menu_state(p_menu_state_menu_data_char_type);
                        }
                }
        else if (choosed_item & e_up_pressed_mask)
```

```
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_status_source(void)
        {
        choosed_item = system_parameters.display_status_source;

        choosed_item =  f_choose_menu_item(&menu_display_sub[e_display_status_source],
                                           menu_display_status,
                                           m_size_of_item(menu_display_status),
                                           choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.display_status_source = (t_display_status_source)choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_data_source(void)
        {
        choosed_item = system_parameters.display_data_source;

        choosed_item =  f_choose_menu_item(&menu_display_sub[e_display_data_source],
                                           menu_display_data,
                                           m_size_of_item(menu_display_data),
                                           choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.display_data_source = (t_display_data_source) choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
```

```
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
void p_menu_state_menu_data_char_type(void)
        {
        choosed_item = system_parameters.display_data_char_type;

        choosed_item =  f_choose_menu_item(&menu_display_sub[e_display_data_char_type],
                                           menu_display_data_char_type,
                                           m_size_of_item(menu_display_data_char_type),
                                           choosed_item);
        if (choosed_item & e_item_choosed_mask)
                {
                choosed_item = (choosed_item & e_get_item_mask);
                system_parameters.display_data_char_type        =       (t_display_data_char_type)
choosed_item;
                p_return_from_menu_state();
                }
        else if (choosed_item & e_up_pressed_mask)
                {
                p_return_from_menu_state();
                }
        else if (choosed_item & e_menu_pressed_mask)
                {
                p_return_to_menu_main_state();
                }
        }

/*****************************************************************************/
word    xdata   status_ram_index;
byte    idata   status_byte;

word    xdata   data_ram_index;
byte    idata   data_byte;
code    byte    display_status_char[2][2] = {{0x00,0x00},  /* signal low */

        {0x01,0x01}};   /* signal high */
code    t_status display_status_mask[5] = {
                                e_stat_rts_mask,
                                e_stat_cts_mask,
                                e_stat_dtr_mask,
                                e_stat_dsr_mask,
                                e_stat_dcd_mask};
code    t_status display_data_mask[2] = {
                        e_stat_CHA_rx_char_mask,
                        e_stat_CHB_rx_char_mask};

void p_menu_state_menu_display_buffer(void)
        {
        code    t_menu_str      display_not_ready[2] = {{"BELLEK BOS"},

        {"ONCE CALISTIRINIZ"}};
```

```
status_ram_index = system_parameters.status_start_index;
data_ram_index = system_parameters.data_start_index;

if ((system_parameters.store_stat _NOT_EQU_ e_store_stat_stoped) _OR_
        (system_parameters.status_start_index _EQU_ system_parameters.status_end_index))
        {
        p_line_display(1,(byte*)display_not_ready[0],strlen((byte*)display_not_ready[0]));
        p_line_display(2,(byte*)display_not_ready[1],strlen((byte*)display_not_ready[1]));
        p_delay(1000);
        p_return_from_menu_state();
        return;
        }

p_display_ram_data();

while(true)
/* wait key loop */
        {
        if (pressed_key)
                {
                if (pressed_key _EQU_ e_key_down)
                        {
                        pressed_key = e_key_none;

                        p_dec_indexes();
                        p_dec_indexes();
                        p_dec_indexes();
                        p_dec_indexes();
                        p_dec_indexes();
                        p_dec_indexes();
                        }
                else if (pressed_key _EQU_ e_key_up)
                        {
                        pressed_key = e_key_none;
                        }
                else if (pressed_key _EQU_ e_key_enter)
                        {
                        pressed_key = e_key_none;
                        p_return_from_menu_state();
                        break;
                        }
                else if (pressed_key _EQU_ e_key_menu)
                        {
                        pressed_key = e_key_none;
                        p_return_to_menu_main_state();
                        break;
                        }
                /* then display new */
                p_display_ram_data();
                }
        }
}

/*****************************************************************************/
void p_inc_indexes(void)
```

```
        {
        m_read_status_ram(status_ram_index,status_byte);
        if (status_byte & e_stat_CHA_rx_char_mask)
                {
                m_inc_circular_buffer_index(data_ram_index,DATA_RAM_LENGTH);
                if (data_ram_index _EQU_ system_parameters.data_end_index)
                /* last byte empty */
                        {
                        data_ram_index = system_parameters.data_start_index;
                        }
                }
        if (status_byte & e_stat_CHB_rx_char_mask)
                {
                m_inc_circular_buffer_index(data_ram_index,DATA_RAM_LENGTH);
                if (data_ram_index _EQU_ system_parameters.data_end_index)
                /* last byte empty */
                        {
                        data_ram_index = system_parameters.data_start_index;
                        }
                }
        m_inc_circular_buffer_index(status_ram_index,STATUS_RAM_LENGTH);
        if (status_ram_index _EQU_ system_parameters.status_end_index)
        /* last byte empty */
                {
                status_ram_index = system_parameters.status_start_index;
                }
        }

/*****************************************************************************/
void p_dec_indexes(void)
        {
        /* last byte empty */
        if (status_ram_index _EQU_ system_parameters.status_start_index)
                {
                status_ram_index = system_parameters.status_end_index;
                }
        m_dec_circular_buffer_index(status_ram_index,STATUS_RAM_LENGTH);
        m_read_status_ram(status_ram_index,status_byte);
        if (status_byte & e_stat_CHA_rx_char_mask)
                {
                if (data_ram_index _EQU_ system_parameters.data_start_index)
                /* last byte empty */
                        {
                        data_ram_index = system_parameters.data_end_index;
                        }
                m_dec_circular_buffer_index(data_ram_index,DATA_RAM_LENGTH);
                }
        if (status_byte & e_stat_CHB_rx_char_mask)
                {
                if (data_ram_index _EQU_ system_parameters.data_start_index)
                /* last byte empty */
                        {
                        data_ram_index = system_parameters.data_end_index;
                        }
                m_dec_circular_buffer_index(data_ram_index,DATA_RAM_LENGTH);
```

```
                }
        }

/*****************************************************************************/
void p_display_ram_data(void)
        {
        p_line_display(1,(byte*)menu_display_status[system_parameters.display_status_source],strlen(
(byte*)menu_display_status[system_parameters.display_status_source]));
        p_line_display(2,(byte*)menu_display_data[system_parameters.display_data_source],strlen((by
te*)menu_display_data[system_parameters.display_data_source]));

        p_display_at(1,5,p_hex_str((status_ram_index&0xFF00)>>8),2);
        p_display_at(1,7,p_hex_str((status_ram_index&0xFF)),2);

        p_display_at(2,5,p_hex_str((data_ram_index&0xFF00)>>8),2);
        p_display_at(2,7,p_hex_str((data_ram_index&0xFF)),2);

        /* First byte */
        m_read_status_ram(status_ram_index,status_byte);
        if (status_byte & display_status_mask[system_parameters.display_status_source])
                {
                p_display_at(1,10,(byte*)&display_status_char[(byte)e_signal_on][0],2);
                }
        else
                {
                p_display_at(1,10,(byte*)&display_status_char[(byte)e_signal_off][0],2);
                }

        if              (display_data_mask[system_parameters.display_data_source]               _EQU_
e_stat_CHA_rx_char_mask)
                {
                if (status_byte & e_stat_CHA_rx_char_mask)
                        {
                        m_read_data_ram(data_ram_index,data_byte);
                        if (system_parameters.display_data_char_type _EQU_ e_display_data_hex)
                                {
                                p_display_at(2,10,p_hex_str(data_byte),2);
                                }
                        else
                                {
                                p_display_at(2,10,p_text_str(data_byte),1);
                                }
                        }
                }
        else    if      (display_data_mask[system_parameters.display_data_source]       _EQU_
e_stat_CHB_rx_char_mask)
                {
                if (status_byte & e_stat_CHB_rx_char_mask)
                        {
                        if (status_byte & e_stat_CHA_rx_char_mask)
                                {
                                m_read_data_ram(data_ram_index+1,data_byte);
                                }
                        else
                                {
```

```
                        m_read_data_ram(data_ram_index,data_byte);
                        }
                    if (system_parameters.display_data_char_type _EQU_ e_display_data_hex)
                        {
                        p_display_at(2,10,p_hex_str(data_byte),2);
                        }
                    else
                        {
                        p_display_at(2,10,p_text_str(data_byte),1);
                        }
                    }
                }
        p_inc_indexes();

        /* Second byte */
        m_read_status_ram(status_ram_index,status_byte);
        if (status_byte & display_status_mask[system_parameters.display_status_source])
                {
                p_display_at(1,12,(byte*)&display_status_char[(byte)e_signal_on][0],2);
                }
        else
                {
                p_display_at(1,12,(byte*)&display_status_char[(byte)e_signal_off][0],2);
                }

        if          (display_data_mask[system_parameters.display_data_source]          _EQU_
e_stat_CHA_rx_char_mask)
                {
                if (status_byte & e_stat_CHA_rx_char_mask)
                        {
                        m_read_data_ram(data_ram_index,data_byte);
                        if (system_parameters.display_data_char_type _EQU_ e_display_data_hex)
                                {
                                p_display_at(2,12,p_hex_str(data_byte),2);
                                }
                        else
                                {
                                p_display_at(2,12,p_text_str(data_byte),1);
                                }
                        }
                }
        else    if      (display_data_mask[system_parameters.display_data_source]      _EQU_
e_stat_CHB_rx_char_mask)
                {
                if (status_byte & e_stat_CHB_rx_char_mask)
                        {
                        if (status_byte & e_stat_CHA_rx_char_mask)
                                {
                                m_read_data_ram(data_ram_index+1,data_byte);
                                }
                        else
                                {
                                m_read_data_ram(data_ram_index,data_byte);
                                }
                        if (system_parameters.display_data_char_type _EQU_ e_display_data_hex)
```

```
                        {
                        p_display_at(2,12,p_hex_str(data_byte),2);
                        }
                else
                        {
                        p_display_at(2,12,p_text_str(data_byte),1);
                        }
                }
        }
p_inc_indexes();

/* Third byte */
m_read_status_ram(status_ram_index,status_byte);
if (status_byte & display_status_mask[system_parameters.display_status_source])
        {
        p_display_at(1,14,(byte*)&display_status_char[(byte)e_signal_on][0],2);
        }
else
        {
        p_display_at(1,14,(byte*)&display_status_char[(byte)e_signal_off][0],2);
        }

if (display_data_mask[system_parameters.display_data_source] _EQU_
e_stat_CHA_rx_char_mask)
        {
        if (status_byte & e_stat_CHA_rx_char_mask)
                {
                m_read_data_ram(data_ram_index,data_byte);
                if (system_parameters.display_data_char_type _EQU_ e_display_data_hex)
                        {
                        p_display_at(2,14,p_hex_str(data_byte),2);
                        }
                else
                        {
                        p_display_at(2,14,p_text_str(data_byte),1);
                        }
                }
        }
else if (display_data_mask[system_parameters.display_data_source] _EQU_
e_stat_CHB_rx_char_mask)
        {
        if (status_byte & e_stat_CHB_rx_char_mask)
                {
                if (status_byte & e_stat_CHA_rx_char_mask)
                        {
                        m_read_data_ram(data_ram_index+1,data_byte);
                        }
                else
                        {
                        m_read_data_ram(data_ram_index,data_byte);
                        }
                if (system_parameters.display_data_char_type _EQU_ e_display_data_hex)
                        {
                        p_display_at(2,14,p_hex_str(data_byte),2);
                        }
```

```
                else
                        {
                        p_display_at(2,14,p_text_str(data_byte),1);
                        }
                }
        }
    p_inc_indexes();

    }

/*****************************************************************************/
void p_menu_state_menu_test(void)
    {
    static  byte    xdata   menu_test_choosed_item = e_test_ram;

    choosed_item =  f_choose_menu_item(&menu_main_sub[e_menu_main_test],
                                       menu_test_sub,
                                       m_size_of_item(menu_test_sub),
                                       menu_test_choosed_item);
    if (choosed_item & e_item_choosed_mask)
        {
        choosed_item = (choosed_item & e_get_item_mask);
        menu_test_choosed_item = choosed_item;
        if (choosed_item _EQU_ e_test_ram)
            {
            p_call_menu_state(p_menu_state_menu_test_ram);
            }
        else if (choosed_item _EQU_ e_test_key)
            {
            p_call_menu_state(p_menu_state_menu_test_key);
            }
        else if (choosed_item _EQU_ e_test_display)
            {
            p_call_menu_state(p_menu_state_menu_test_display);
            }
        }
    else if (choosed_item & e_up_pressed_mask)
        {
        p_return_from_menu_state();
        }
    else if (choosed_item & e_menu_pressed_mask)
        {
        p_return_to_menu_main_state();
        }
    }

/*****************************************************************************/
#pragma         OPTIMIZE(3,SPEED)

/*****************************************************************************/
void p_menu_state_menu_test_ram(void)
    {
    code    word    ram_lengths[3] = {      SYSTEM_RAM_LENGTH,

    DATA_RAM_LENGTH,
```

```
        STATUS_RAM_LENGTH};

        code t_menu_str  ram_mess[3] = {{"SiSTEM BELLEGi"},
                                                                {"VERi BELLEGi"},
                                                                {"DURUM
BELLEGi"}};

        code t_menu_str  error_mess[1] = {"HATA :"};
        code t_menu_str  ram_test_end[1] = {"RAM TESTi BiTTi"};

        for (choosed_item = 0;choosed_item < 3; choosed_item++)
                {
                m_clear_display();

        p_line_display(1,(byte*)ram_mess[choosed_item],strlen((byte*)ram_mess[choosed_item]));
                for (index = 0x0000;index < ram_lengths[choosed_item]; index++)
                        {
                        if (choosed_item _EQU_ 0)
                                {
                                m_read_system_ram(index,ram_actual_byte);
                                m_write_system_ram(index,0xAA);
                                m_read_system_ram(index,test_val);
                                m_write_system_ram(index,ram_actual_byte);
                                }
                        else if (choosed_item _EQU_ 1)
                                {
                                m_read_data_ram(index,ram_actual_byte);
                                m_write_data_ram(index,0xAA);
                                m_read_data_ram(index,test_val);
                                m_write_data_ram(index,ram_actual_byte);
                                }
                        else if (choosed_item _EQU_ 2)
                                {
                                m_read_status_ram(index,ram_actual_byte);
                                m_write_status_ram(index,0xAA);
                                m_read_status_ram(index,test_val);
                                m_write_status_ram(index,ram_actual_byte);
                                }

                        if (test_val _EQU_ 0xAA)
                                {
                                if ((index & 0xFF) _EQU_ 0xFF)
                                        {
                                        p_display_at(2,1,p_hex_str((index&0xFF00)>>8),2);
                                        p_display_at(2,3,p_hex_str((index&0xFF)),2);
                                        }
                                }
                        else
                                {
                                p_line_display(2,(byte*)error_mess[0],strlen((byte*)error_mess[0]));
                                p_display_at(2,10,p_hex_str((index&0xFF00)>>8),2);
                                p_display_at(2,12,p_hex_str((index&0xFF)),2);
                                p_display_at(2,15,p_hex_str(test_val),2);
                                }
```

```
                }
            }
        m_clear_display();
        p_line_display(1,(byte*)ram_test_end[0],strlen((byte*)ram_test_end[0]));
        p_delay(1000);
        p_return_from_menu_state();
        }

/*****************************************************************************/
#pragma         OPTIMIZE(5,SPEED)

/*****************************************************************************/
void p_menu_state_menu_test_key(void)
        {
        code t_menu_str  key_test_start[1] = {"BiR TUSA BASINIZ"};
        code t_menu_str  key_test_end[1] = {"TUS TESTi BiTTi"};
        code t_menu_str  key_str[5] = {{"BASILI DEGiL"},
                                                        {"YUKARI BASILDI"},
                                                        {"ASAGI BASILDI"},
                                                        {"ENTER BASILDI"},
                                                        {"MENU BASILDI"}};

        p_line_display(1,(byte*)key_test_start[0],strlen((byte*)key_test_start[0]));
        p_line_display(2,(byte*)key_str[pressed_key],strlen((byte*)key_str[pressed_key]));
        while(true)
                {
                if (pressed_key)
                        {

        p_line_display(2,(byte*)key_str[pressed_key],strlen((byte*)key_str[pressed_key]));
                        if (pressed_key _EQU_ e_key_menu)
                                {
                                pressed_key = e_key_none;
                                break;
                                }
                        else
                                {
                                pressed_key = e_key_none;
                                }
                        }
                }
        p_line_display(2,(byte*)key_test_end[0],strlen((byte*)key_test_end[0]));
        p_delay(1000);
        p_return_from_menu_state();
        }

/*****************************************************************************/
void p_menu_state_menu_test_display(void)
        {
        code    byte    test_char[1] = {0xFF};
        code    byte    space_char[1] = {0x20};

        for (choosed_item = 1;choosed_item < (LCD_COLUMN_NO+1); choosed_item++)
                {
                p_display_at(1,choosed_item,test_char,1);
```

```
            p_delay(20);
            }
        for (choosed_item = 1;choosed_item < (LCD_COLUMN_NO+1); choosed_item++)
            {
            p_display_at(2,choosed_item,test_char,1);
            p_delay(20);
            }
        m_clear_display();
        for (choosed_item = 1;choosed_item < (LCD_COLUMN_NO+1); choosed_item++)
            {
            p_display_at(1,choosed_item,test_char,1);
            p_delay(20);
            p_display_at(1,choosed_item,space_char,1);
            }
        for (choosed_item = 1;choosed_item < (LCD_COLUMN_NO+1); choosed_item++)
            {
            p_display_at(2,choosed_item,test_char,1);
            p_delay(20);
            p_display_at(2,choosed_item,space_char,1);
            }
        p_return_from_menu_state();
        }

/*****************************************************************************/
byte f_choose_menu_item(t_menu_str      *head_str,
                        t_menu_str      *item_array,
                        byte            total_item,
                        byte            active_item)
        {
        p_line_display(1,(byte*)head_str[0],strlen((byte*)head_str[0]));
        p_line_display(2,(byte*)item_array[active_item],strlen((byte*)item_array[active_item]));
        p_delay(100);

        while(true)
            {
            if (pressed_key)
                {
                if (pressed_key _EQU_ e_key_up)
                    {
                    pressed_key = e_key_none;
                    return ((byte)e_up_pressed_mask);
                    }
                else if (pressed_key _EQU_ e_key_down)
                    {
                    pressed_key = e_key_none;
                    active_item++;
                    if (active_item >= total_item)
                        {
                        active_item=0;
                        }

        p_line_display(2,(byte*)item_array[active_item],strlen((byte*)item_array[active_item]));
                    }
                else if (pressed_key _EQU_ e_key_enter)
                    {
```

```
                pressed_key = e_key_none;
                return (active_item|(byte)e_item_choosed_mask);
                }
            else if (pressed_key _EQU_ e_key_menu)
                {
                pressed_key = e_key_none;
                return ((byte)e_menu_pressed_mask);
                }
            }
        }
    }

/*****************************************************************************/
void p_call_menu_state(t_void_arg_proc_ptr          new_state_ptr)
    {
    menu_stack.stack[menu_stack.stack_pointer] = menu_state_ptr;
    menu_stack.stack_pointer++;
    menu_state_ptr = new_state_ptr;
    }

/*****************************************************************************/
void p_return_from_menu_state(void)
    {
    if (menu_stack.stack_pointer > 0)
        {
        menu_stack.stack_pointer--;
        menu_state_ptr = menu_stack.stack[menu_stack.stack_pointer];
        }
    }

/*****************************************************************************/
void p_return_to_menu_main_state(void)
    {
    while(menu_stack.stack_pointer > 0)
        {
        menu_stack.stack_pointer--;
        }
    menu_state_ptr = menu_stack.stack[menu_stack.stack_pointer];
    }

/*****************************************************************************/
#pragma     OPTIMIZING_LEVEL

/*****************************************************************************/
/*                                                   */
/* SCCUTIL.C                                             */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*****************************************************************************/
#define _SCCUTIL_C
/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
```

```
#include <c:\project\tez\software\mcs51\include\sccutil.h>

/*****************************************************************************/
static    byte                          idata          t;        /* for macro activation*/

/*****************************************************************************/
t_AM8530_int_copy                       xdata    AM8530_int_copy;
t_union_of_word_with_bytes      code     time_constant[s_max_baud]={
                                m_time_constant(300),
                                m_time_constant(600),
                                m_time_constant(1200),
                                m_time_constant(2400),
                                m_time_constant(4800),
                                m_time_constant(7200),
                                m_time_constant(9600),
                                m_time_constant(12000),
                                m_time_constant(14400),
                                m_time_constant(19200)
                                };
byte code rx_data_bits[4]={
                /*e_5_data_bits*/       0x00,
                /*e_6_data_bits*/       0x80,
                /*e_7_data_bits*/       0x40,
                /*e_8_data_bits*/       0xC0
                };

byte code char_mask[4]={
                /*e_5_data_bits*/       0x1F,
                /*e_6_data_bits*/       0x3F,
                /*e_7_data_bits*/       0x7F,
                /*e_8_data_bits*/       0xFF
                };

byte code rx_stop_bits[3]={
                /*e_1_stop_bits*/0x04,
                /*e_1_5_stop_bits*/     0x08,
                /*e_2_stop_bits*/0x0C
                };

byte code rx_parity_bits[3]={
                /*e_none_parity*/       0x00,
                /*e_odd_parity*/ 0x01,
                /*e_even_parity*/       0x03
                };


/*****************************************************************************/
#define  WR1_init_val\
        (e_ext_int_enable|e_parity_is_special_condition|e_int_on_all_rx_chars_or_special_condition)
/*****************************************************************************/
#define  WR9_init_val\
        (e_nv|e_interrupt_masking_without_intack)
/*****************************************************************************/
#define  WR9_second_val\
        (e_nv|e_mie|e_interrupt_masking_without_intack)
```

```
/*****************************************************************************/
#define  WR11_init_val\
        (e_trxc_output|e_trxc_out_src_tx_clk|e_tx_clk_src_br_generator_output|e_rx_clk_src_br_gener
ator_output)
/*****************************************************************************/
#define  WR14_init_val\
        (e_br_generator_enable|e_br_generator_src_is_PCLK_input)
/*****************************************************************************/
#define  WR15_init_val\
        (e_break_abort_ie)

/*****************************************************************************/
void p_initialize_8530_with_setup()
        {
        byte    xdata           temp;

        m_disable_ints();

        m_rd_reg_e_RR0(CHA,temp);
        m_rd_reg_e_RR0(CHB,temp);

        m_reset_scc();

        if (system_parameters.comm_type _EQU_ e_comm_type_async)
                {
                temp =  m_clock_mode
                                | rx_stop_bits[system_parameters.comm_setup.async.stop_bits]
                                | rx_parity_bits[system_parameters.comm_setup.async.parity];
                m_wr_reg(CHA,e_WR4,temp);
                m_wr_reg(CHB,e_WR4,temp);
                }
        else
                {
                if (system_parameters.comm_setup.sync.sync_protocol _EQU_ e_sync_sdlc)
                        {
                        m_wr_reg(CHA,e_WR4,0x20);
                        m_wr_reg(CHB,e_WR4,0x20);
                        }
                else
                        {
                        if (system_parameters.comm_setup.sync.sync_type _EQU_ e_sync_mono)
                                {
                                m_wr_reg(CHA,e_WR4,0x00);
                                m_wr_reg(CHB,e_WR4,0x00);
                                }
                        else
                                {
                                m_wr_reg(CHA,e_WR4,0x10);
                                m_wr_reg(CHB,e_WR4,0x10);
                                }
                        }
                }

        m_wr_reg(CHA,e_WR2,0x00);
        m_wr_reg(CHB,e_WR2,0x00);
```

```
m_wr_reg(CHA,e_WR3,rx_data_bits[system_parameters.data_bits]);
m_wr_reg(CHB,e_WR3,rx_data_bits[system_parameters.data_bits]);

m_wr_reg(CHA,e_WR5,0x00);
m_wr_reg(CHB,e_WR5,0x00);

m_wr_reg(CHA,e_WR6,0x7E);
m_wr_reg(CHB,e_WR6,0x7E);

m_wr_reg(CHA,e_WR7,0x7E);
m_wr_reg(CHB,e_WR7,0x7E);

m_wr_reg(CHA,e_WR9,WR9_init_val);
m_wr_reg(CHB,e_WR9,WR9_init_val);

if (system_parameters.comm_type _EQU_ e_comm_type_async)
        {
        m_wr_reg(CHA,e_WR10,0x00);
        m_wr_reg(CHB,e_WR10,0x00);
        }
else
        {
        m_wr_reg(CHA,e_WR10,0x01);
        m_wr_reg(CHB,e_WR10,0x01);
        }

if (system_parameters.comm_type _EQU_ e_comm_type_async)
        {
        m_wr_reg(CHA,e_WR11,WR11_init_val);
        m_wr_reg(CHB,e_WR11,WR11_init_val);
        }
else
        {
        m_wr_reg(CHA,e_WR11,0x01);
        m_wr_reg(CHB,e_WR11,0x01);
        }

temp = time_constant[system_parameters.baud_rate].word_in_two_bytes.low_byte;
m_wr_reg(CHA,e_WR12,temp);
m_wr_reg(CHB,e_WR12,temp);
temp = time_constant[system_parameters.baud_rate].word_in_two_bytes.high_byte;
m_wr_reg(CHA,e_WR13,temp);
m_wr_reg(CHB,e_WR13,temp);

if (system_parameters.comm_type _EQU_ e_comm_type_async)
        {
        m_wr_reg(CHA,e_WR14,WR14_init_val);
        m_wr_reg(CHB,e_WR14,WR14_init_val);
        }
else
        {
        m_wr_reg(CHA,e_WR14,0x00);
        m_wr_reg(CHB,e_WR14,0x00);
        }
```

```
        m_wr_reg(CHA,e_WR15,WR15_init_val);
        m_wr_reg(CHB,e_WR15,WR15_init_val);

        temp =  rx_data_bits[system_parameters.data_bits]
                        | e_rx_enable;
        m_wr_reg(CHA,e_WR3,temp);
        m_wr_reg(CHB,e_WR3,temp);

        m_wr_reg(CHA,e_WR1,WR1_init_val);
        m_wr_reg(CHB,e_WR1,WR1_init_val);

        m_wr_reg(CHA,e_WR9,WR9_second_val);
        m_wr_reg(CHB,e_WR9,WR9_second_val);

        m_enable_ints();
        }

/*****************************************************************************/
/*                                                                           */
/* MAINPRG.C                                                                 */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
#define _MAINPRG_C

/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
#include <c:\project\tez\software\mcs51\include\display.h>
#include <c:\project\tez\software\mcs51\include\sccutil.h>
#include <c:\project\tez\software\mcs51\include\time0int.h>
#include <c:\project\tez\software\mcs51\include\ext0int.h>
#include <c:\project\tez\software\mcs51\include\menu.h>

/*****************************************************************************/
static  byte                    idata           t;      /* for macro activation*/

/*****************************************************************************/
bdata   byte    system_flag_area;
sbit    CHA_rx_char_ready       = system_flag_area ^ 0;
sbit    CHB_rx_char_ready       = system_flag_area ^ 1;
sbit    busy_state              = system_flag_area ^ 7;

/*****************************************************************************/
/*  main: main program                                                       */
/*****************************************************************************/
void main()
        {
        code t_menu_str  aydin_parin[1] = {"AYDIN   PARIN"};
        code t_menu_str  ytu[1] =               {"YTU - iSTANBUL"};

        m_disable_ints();                       /* disable all interrupts  */
```

```
        p_initialize_display();
        p_initialize_8530_with_setup();
        p_timer0_init();
        p_int0_init();

        /*PX0 = 1;*/                                    /* int0 priority highest */
        PT0 = 1;                                        /* T0int priority highest */
        m_enable_ints();                        /* enable all interrupts  */

        p_line_display(1,(byte*)aydin_parin[0],strlen((byte*)aydin_parin[0]));
        p_line_display(2,(byte*)ytu[0],strlen((byte*)ytu[0]));
        p_delay(1000);

        menu();
        }

/*****************************************************************************/
/*                                                                           */
/* GENUTIL.C                                                                 */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
#define _GENUTIL_C

/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>

/*****************************************************************************/
byte    idata   *p_hex_str(byte  byte_data)
        {
        byte    idata   hex_str[2];
        hex_str[0]=m_convert_asc(m_calc_hi_nibble(byte_data));
        hex_str[1]=m_convert_asc(m_calc_lo_nibble(byte_data));
        return  (hex_str);
        }

/*****************************************************************************/
/*                                                                           */
/* EXT0INT.C                                                                 */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/

#define _EXT0INT_C

/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
#include <c:\project\tez\software\mcs51\include\sccutil.h>
```

```
#include <c:\project\tez\software\mcs51\include\ext0int.h>

/*****************************************************************************/
static  byte                            idata           t;      /* for macro activation*/

/*****************************************************************************/
extern          bit                                     CHA_rx_char_ready;
extern          bit                                     CHB_rx_char_ready;
extern          byte    code            char_mask[4];

/*****************************************************************************/
byte                    idata   CHA_last_received_byte;
byte                    idata   CHB_last_received_byte;

/*****************************************************************************/
#ifdef CPU_8051
        extint0() interrupt 0    /* using 2 use registerbank 2 for interrupt     */
#endif
        {
        register byte   idata   interrupt_source;
        register byte   idata   temp;

        m_rd_reg(CHA,e_RR3,interrupt_source);

        if(interrupt_source & e_channel_A_rx_ip)
                {
                m_rd_reg(CHA,e_RR1,temp);
                if(temp & (e_rx_overrun_error | e_parity_error |e_crc_framing_error))
                        {
                        m_wr_reg_e_WR0(CHA,e_error_reset);
                        }
                else
                        {
                        m_scc_inportd(CHA,CHA_last_received_byte);
                        CHA_last_received_byte          =       (CHA_last_received_byte          &
char_mask[system_parameters.data_bits]);
                        if (system_parameters.store_stat _NOT_EQU_ e_store_stat_stoped)
                                {
                                CHA_rx_char_ready = true;
                                }
                        }
                }
        if(interrupt_source & e_channel_A_tx_ip)
                {
                /*m_scc_outportd(CHA,0x55);*/
                m_wr_reg_e_WR0(CHA,e_reset_tx_int_pending);
                }
        if(interrupt_source & e_channel_A_ext_or_stat_ip)
                {
                m_rd_reg_e_RR0(CHA,temp);
                if (temp & e_break_abort)
                        {
                        m_scc_inportd(CHA,temp);
                        }
                m_wr_reg_e_WR0(CHA,e_reset_ext_status_interrupts);
```

```
        }
    /*m_wr_reg_e_WR0(CHA,e_reset_highest_ius);*/

    if(interrupt_source & e_channel_B_rx_ip)
        {
        m_rd_reg(CHB,e_RR1,temp);
        if(temp & (e_rx_overrun_error | e_parity_error |e_crc_framing_error))
            {
            m_wr_reg_e_WR0(CHB,e_error_reset);
            }
        else
            {
            m_scc_inportd(CHB,CHB_last_received_byte);
            CHB_last_received_byte = (CHB_last_received_byte &
char_mask[system_parameters.data_bits]);
            if (system_parameters.store_stat _NOT_EQU_ e_store_stat_stoped)
                {
                CHB_rx_char_ready = true;
                }
            }
        }
    if(interrupt_source & e_channel_B_tx_ip)
        {
        /*m_scc_outportd(CHB,0xAA);*/
        m_wr_reg_e_WR0(CHB,e_reset_tx_int_pending);
        }
    if(interrupt_source & e_channel_B_ext_or_stat_ip)
        {
        m_rd_reg_e_RR0(CHB,temp);
        if (temp & e_break_abort)
            {
            m_scc_inportd(CHB,temp);
            }
        m_wr_reg_e_WR0(CHB,e_reset_ext_status_interrupts);
        }
    /*m_wr_reg_e_WR0(CHB,e_reset_highest_ius);*/

    }

/*****************************************************************************/
void p_int0_init()
    {
    IT0 = 0;                                    /* EXT0 interrupts falling edge */
    EX0 = 1;                                    /* enable EXT0 interrupts  */
    }

/*****************************************************************************/
/*                                           */
/* DISPLAY.C                                     */
/*                                           */
/*                                           */
/*                                           */
/*                                           */
/*                                           */
/*****************************************************************************/
```

```
#define _DISPLAY_C

/*****************************************************************************/
/*#define        DISPLAY_STATUS*/

/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\display.h>
#include <c:\project\tez\software\mcs51\include\menu.h>

/*****************************************************************************/
static  byte                    idata           t;      /* for macro activation*/

/*****************************************************************************/
extern  bit                                     busy_state;

/*****************************************************************************/
sbit    busy_state_port_bit     = m_display_port ^ 7;
static  byte    idata   count;

/*****************************************************************************/
static  byte    code    special_fonts[]={
        0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x00,
        0x1F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
        0x0E,0x11,0x10,0x10,0x10,0x15,0x0E,0x00,
        0x0E,0x1F,0x10,0x16,0x11,0x11,0x0E,0x00,
        0x04,0x1F,0x04,0x04,0x04,0x04,0x1F,0x00,
        0x0A,0x0E,0x11,0x11,0x11,0x11,0x0E,0x00,
        0x0E,0x11,0x10,0x0E,0x01,0x05,0x0E,0x00,
        0x0A,0x11,0x11,0x11,0x11,0x11,0x0E,0x00
        };

/*****************************************************************************/
#pragma         OPTIMIZE(5,SPEED)

/*****************************************************************************/
void p_delay(word milisec)
        {
        register byte   idata   delay_count;

        while(milisec)
                {
                for(delay_count=0x70;delay_count>0;delay_count--)
                        {
                        }
                milisec--;
                }
        }

/*****************************************************************************/
void p_wait_busy_flag(void)
        {
        RW(1);
        RS(0);
        do
```

```
                {
                m_wr_display_port_FF();
                m_inhibit_interrupts();
                EN(1);
                m_wait_1_t_clk_on_rd();
                busy_state = busy_state_port_bit;
                EN(0);
                m_uninhibit_interrupts();
                }while(busy_state);
        }

/*****************************************************************************/
void p_display_cmd_BF(byte cmd)
        {
        p_wait_busy_flag();
        RW(0);
        RS(0);
        m_inhibit_interrupts();
        EN(1);
        m_display_port = cmd;
        m_wait_1_t_clk_on_wr();
        EN(0);
        m_uninhibit_interrupts();
        }

/*****************************************************************************/
void p_display_cmd_NBF(byte cmd)
        {
        RW(0);
        RS(0);
        m_inhibit_interrupts();
        EN(1);
        m_display_port = cmd;
        m_wait_1_t_clk_on_wr();
        EN(0);
        m_uninhibit_interrupts();
        }

/*****************************************************************************/
void p_set_DDRA(byte row,byte chrix)
        {
        p_wait_busy_flag();
        RW(0);
        RS(0);
        m_inhibit_interrupts();
        EN(1);
        if (row _EQU_ 1)
                {
                m_display_port = (chrix | 0x80);
                }
        else
                {
                m_display_port = (chrix | 0xC0);
                }
        m_wait_1_t_clk_on_wr();
```

```
        EN(0);
        m_uninhibit_interrupts();
        }

/*****************************************************************************/
void p_WDDRA(byte c)
        {
        p_wait_busy_flag();
        RW(0);
        RS(1);
        m_inhibit_interrupts();
        EN(1);
        m_display_port = c;
        m_wait_1_t_clk_on_wr();
        EN(0);
        m_uninhibit_interrupts();
        }

/*****************************************************************************/
byte p_RDDRA(void)
        {
        register byte     idata     ch;

        p_wait_busy_flag();
        RW(1);
        RS(1);
        m_wr_display_port_FF();
        m_inhibit_interrupts();
        EN(1);
        m_wait_1_t_clk_on_rd();
        ch = m_display_port;
        EN(0);
        m_uninhibit_interrupts();
        return (ch);
        }

/*****************************************************************************/
void p_set_CGRA(byte CGRA)
        {
        CGRA = (CGRA | 0x40);
        p_wait_busy_flag();
        RW(0);
        RS(0);
        m_inhibit_interrupts();
        EN(1);
        m_display_port = CGRA;
        m_wait_1_t_clk_on_wr();
        EN(0);
        m_uninhibit_interrupts();
        }

/*****************************************************************************/
void p_reset_display(void)
        {
        EN(0);
```

```
        RW(0);
        RS(0);
        p_delay(20);
        for(count=0;count<3;count++)
                {
                RW(0);
                RS(0);
                m_inhibit_interrupts();
                EN(1);
                m_display_port = 0x3F;
                m_wait_1_t_clk_on_wr();
                EN(0);
                m_uninhibit_interrupts();
                p_delay(5);
                }
        }

/*****************************************************************************/
code    byte    display_store_stat[3] = {(byte)'<',
                                         (byte)'=',
                                         (byte)'>'};
void p_line_display(byte row, byte *mess_ptr, byte mess_len)
        {
        register byte    idata    char_index;

        for(char_index=0;char_index<mess_len;char_index++)
                {
                p_set_DDRA(row,char_index);
                p_WDDRA(mess_ptr[char_index]);
                }
        if (mess_len < LCD_COLUMN_NO)
                {
                while(char_index < LCD_COLUMN_NO)
                        {
                        p_set_DDRA(row,char_index);
                        p_WDDRA(' ');
                        char_index++;
                        }
                }
#ifdef DISPLAY_STATUS
        p_set_DDRA(1,(LCD_COLUMN_NO-1));
        p_WDDRA((byte)display_store_stat[(byte)system_parameters.store_stat]);
#endif
        }

/*****************************************************************************/
void p_display_at(byte row, byte column, byte *mess_ptr, byte mess_len)
        {
        register byte    idata    char_index;

        column--;
        for(char_index=0; char_index<mess_len; char_index++)
                {
                p_set_DDRA(row, char_index+column);
                p_WDDRA(mess_ptr[char_index]);
```

```
                }
#ifdef DISPLAY_STATUS
        p_set_DDRA(1,(LCD_COLUMN_NO-1));
        p_WDDRA((byte*)&display_store_stat[system_parameters.store_stat]);
#endif
        }

/*****************************************************************************/
void p_write_special_fonts_to_CGRAM()
        {
        for(count=0;count<sizeof(special_fonts);count++)
                {
                p_set_CGRA(count);
                p_WDDRA(special_fonts[count]);
                }
        }

/*****************************************************************************/
void p_initialize_display()
        {
        byte    code    INSTR[]={0x38,0x08,0x01,0x06,0x0C,0x14};

        p_reset_display();
        for(count=0;count<sizeof(INSTR);count++)
                {
                p_display_cmd_BF(INSTR[count]);;
                }
        p_delay(5);
        p_write_special_fonts_to_CGRAM();
        p_delay(5);
        m_clear_display();
        }

/*****************************************************************************/
#pragma         OPTIMIZING_LEVEL

/*****************************************************************************/
/*                                                   */
/* TIME0INT.H                                              */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*****************************************************************************/

#ifndef _TIME0INT_H
#define _TIME0INT_H
/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
#include <c:\project\tez\software\mcs51\include\statport.h>

/*****************************************************************************/
#define m_T0_int_period         1                       /* for 1 msec int */
#define T0_initial_value   (0xFFFF - (1000*m_T0_int_period))
```

```
#define TH0_initial_value m_calc_hi_byte(T0_initial_value)
#define TL0_initial_value m_calc_lo_byte(T0_initial_value)
#define T0_mode                         0x01            /* 16 bit timer */

/*****************************************************************************/
extern  t_key   idata   pressed_key;

/*****************************************************************************/
extern void p_timer0_init();

#endif

/*****************************************************************************/
/*                                                   */
/* TARGET.H                                                */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*****************************************************************************/
#ifndef  _TARGET_H
#define  _TARGET_H

#define CPU_8051

#define DEBUG_MODE                  DEBUG
                                                    /* DEBUG NODEBUG */
#define MEMORY_MODEL                LARGE
                                                    /* SMALL COMPACT LARGE */
#define ROM_SIZE                    ROM(LARGE)
                                                    /* SMALL COMPACT LARGE */
#define VAR_TYPE_INFO               OBJECTEXTEND
                                                    /*                 OBJECTEXTEND
NOOBJECTEXTEND */
#define ASM_CODE_REQ                CODE
                                                    /* CODE NOCODE */
#define OPTIMIZING_LEVEL    OPTIMIZE(3,SPEED)
                                                    /* ( 0..5 , SIZE SPEED ) */

#define REGISTER_USE            AREGS
                                                    /* ( NOAREGS, AREGS ) */
/* Target code model definition */
#pragma DEBUG_MODE MEMORY_MODEL ROM_SIZE VAR_TYPE_INFO ASM_CODE_REQ
OPTIMIZING_LEVEL REGISTER_USE

/*****************************************************************************/
#ifdef CPU_8051
        #include <c:\project\tez\software\mcs51\include\reg51.h>
        #include <c:\embed\mcs51\c51p\inc\intrins.h>
        #include <c:\embed\mcs51\c51p\inc\string.h>
#else
        #error CPU_TYPE tanimlanmamis
#endif
```

```
#include <c:\project\tez\software\mcs51\include\glotyp.h>
#include <c:\project\tez\software\mcs51\include\genutil.h>

/*****************************************************************************/
#define  CPU_CLOCK_FRQ                                    12000000
#define  A16(bit_val)                                     TXD=bit_val
#define  A17(bit_val)                                     RXD=bit_val

/*****************************************************************************/
#define  ABSOLUTE_ADDRESS_OF_SYSTEM_RAM       0x30000L        /*    0x30000    length
0x8000 */
#define  ADDRESS_OF_SYSTEM_RAM                0x0000
#define  SYSTEM_RAM_LENGTH                          0x1000
#define  m_enable_system_ram()
         A17(((ABSOLUTE_ADDRESS_OF_SYSTEM_RAM&0x20000)>>17));        \

         A16(((ABSOLUTE_ADDRESS_OF_SYSTEM_RAM&0x10000)>>16))
#define  SYSTEM_RAM                                       (*(byte        volatile        xdata
*)ADDRESS_OF_SYSTEM_RAM)
#define m_read_system_ram(address,var)   var = ((byte*)&SYSTEM_RAM)[address]
#define m_write_system_ram(address,var)  ((byte*)&SYSTEM_RAM)[address] = var

/*****************************************************************************/
#define  ABSOLUTE_ADDRESS_OF_DATA_RAM
         (ABSOLUTE_ADDRESS_OF_SYSTEM_RAM+SYSTEM_RAM_LENGTH)
#define  ADDRESS_OF_DATA_RAM                        0x1000
#define  DATA_RAM_LENGTH                            0x3800
#define  m_enable_data_ram()
         A17(((ABSOLUTE_ADDRESS_OF_DATA_RAM&0x20000)>>17));  \

         A16(((ABSOLUTE_ADDRESS_OF_DATA_RAM&0x10000)>>16))
#define  DATA_RAM                                         (*(byte        volatile        xdata
*)ADDRESS_OF_DATA_RAM)
#define m_read_data_ram(address,var)\
         ti=address;\
         m_enable_data_ram();\
         t=((byte*)&DATA_RAM)[ti];\
         m_enable_system_ram();\
         var=t
#define m_write_data_ram(address,var)\
         ti=address;\
         t=var;\
         m_enable_data_ram();\
         ((byte*)&DATA_RAM)[ti]=t;\
         m_enable_system_ram()

/*****************************************************************************/
#define  ABSOLUTE_ADDRESS_OF_STATUS_RAM
         (ABSOLUTE_ADDRESS_OF_DATA_RAM+DATA_RAM_LENGTH)
#define  ADDRESS_OF_STATUS_RAM                0x4800
#define  STATUS_RAM_LENGTH                          0x3800
#define  m_enable_status_ram()
         A17(((ABSOLUTE_ADDRESS_OF_STATUS_RAM&0x20000)>>17));        \

         A16(((ABSOLUTE_ADDRESS_OF_STATUS_RAM&0x10000)>>16))
```

```
#define  STATUS_RAM                                    (*(byte                    volatile
xdata*)ADDRESS_OF_STATUS_RAM)
#define m_read_status_ram(address,var)\
        ti=address;\
        m_enable_status_ram();\
        t=((byte*)&STATUS_RAM)[ti];\
        m_enable_system_ram();\
        var=t
#define m_write_status_ram(address,var)\
        ti=address;\
        t=var;\
        m_enable_status_ram();\
        ((byte*)&STATUS_RAM)[ti]=t;\
        m_enable_system_ram()

/*****************************************************************************/
#define  ADDRESS_OF_AM8530                          0x20000
#define  ADDRESS_OF_STATUS_PORT                     0x28000

#endif

/*****************************************************************************/
/*                                                */
/* STATPORT.H                                            */
/*                                                */
/*                                                */
/*                                                */
/*                                                */
/*                                                */
/*****************************************************************************/

#ifndef _STATPORT_H
#define _STATPORT_H
/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>

/*****************************************************************************/
#define  STATUS_PORT                                   (*(byte
xdata*)((word)ADDRESS_OF_STATUS_PORT & 0xFFFF))
#define  m_enable_status_port()
        A17(((ADDRESS_OF_STATUS_PORT&0x20000)>>17));    \

        A16(((ADDRESS_OF_STATUS_PORT&0x10000)>>16))
#define m_read_status_port(var)                 m_enable_status_port();   \
                                                                          var          =
STATUS_PORT; \

        m_enable_system_ram()

/*****************************************************************************/
#define m_key_int_period(100*m_T0_int_period)   /* 10*T0_int_priod */

typedef enum    {
                e_key_none,
                e_key_up,
```

```
                e_key_down,
                e_key_enter,
                e_key_menu
                } t_key;
typedef enum    {
                e_key_none_code,
                e_key_up_code=0x01,
                e_key_down_code=0x02,
                e_key_enter_code=0x04,
                e_key_menu_code=0x08
                } t_key_code;

typedef enum    {
                e_stat_all_off=0xF0,
                e_stat_CHA_rx_char_mask=0x01,
                e_stat_CHB_rx_char_mask=0x02,
                e_stat_dcd_mask=0x08,
                e_stat_rts_mask=0x10,
                e_stat_dtr_mask=0x20,
                e_stat_cts_mask=0x40,
                e_stat_dsr_mask=0x80
                } t_status;         /* definition on DTE */

#endif

/*****************************************************************************/
/*                                                                           */
/* SCCUTIL.H                                                                 */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/

#ifndef _SCCUTIL_H
#define _SCCUTIL_H
/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
#include <c:\project\tez\software\mcs51\include\am8530.h>
#include <c:\project\tez\software\mcs51\include\menu.h>

/*****************************************************************************/
/*
#define  m_handle_wait_PCLK_cycles
*/

#ifdef    m_handle_wait_PCLK_cycles
          #define  m_wait_6_PCLK()              _nop_()
          #define  m_wait_11_PCLK()       _nop_();_nop_();_nop_()
#else
          #define  m_wait_6_PCLK()              do{}while(0)
          #define  m_wait_11_PCLK()       do{}while(0)
#endif
```

```
/*****************************************************************************/
#define  m_read_wr_copy(ch_type,reg)      \
         (AM8530_int_copy.ch_type.write_regs[reg##_int_copy] & reg##_mask)

/*****************************************************************************/
#define  m_rd_reg_e_RR0(ch_type,var)\
         do{\
         m_enable_AM8530();t=AM8530.ch_type.control_port;\
         m_enable_system_ram();m_wait_6_PCLK();\
         var=t;}while(0)

/*****************************************************************************/
#define  m_rd_reg(ch_type,reg,var)\
do{\
         t=reg;\
         m_enable_AM8530();AM8530.ch_type.control_port=t;\
         m_wait_6_PCLK();t=AM8530.ch_type.control_port;\
         m_enable_system_ram();m_wait_6_PCLK();var=t;}while(0)

/*****************************************************************************/
#define m_scc_inportd(ch_type,var)\
         do{\
         m_enable_AM8530();\
         t=AM8530.ch_type.data_port;\
         m_enable_system_ram();var=t;}while(0)

/*****************************************************************************/
#define  m_wr_reg_e_WR0(ch_type,val)\
         do{\
         t=val;AM8530_int_copy.ch_type.write_regs[e_WR0_int_copy]=t;\
         m_enable_AM8530();AM8530.ch_type.control_port=t;\
         m_enable_system_ram();m_wait_6_PCLK();}while(0)

/*****************************************************************************/
#define  m_wr_reg(ch_type,reg,val)\
         do{\
         t=val;AM8530_int_copy.ch_type.write_regs[reg##_int_copy]=t;\
         m_enable_AM8530();AM8530.ch_type.control_port=reg;\
         m_wait_6_PCLK();AM8530.ch_type.control_port=t;\
         m_enable_system_ram();m_wait_6_PCLK();}while(0)

/*****************************************************************************/
#define m_scc_outportd(ch_type,val)\
         do{\
         t=val;m_enable_AM8530();\
         AM8530.ch_type.data_port=t;\
         m_enable_system_ram();}while(0)

/*****************************************************************************/
#define m_reset_scc()\
         m_wr_reg(CHA,e_WR9,e_force_hardware_reset);    \
         m_wait_11_PCLK()
/*****************************************************************************/
#define m_ch_reset(ch_type)\
```

```
        m_wr_reg(ch_type,e_WR9,m_read_wr_copy(ch_type,e_WR9)|e_##ch_type##_channel_reset);
        \
        m_wait_11_PCLK()
/*****************************************************************************/
#define m_tx_reset(ch_type)\
        AM8530.ch_type.control_port = e_reset_ext_status_interrupts;        \
        m_wr_reg(ch_type,e_WR5,m_read_wr_copy(ch_type,e_WR5)&(_INVERSE_ e_tx_enable))
/*****************************************************************************/
#define m_rx_reset(ch_type)\
        AM8530.ch_type.control_port = e_reset_ext_status_interrupts;        \
        m_wr_reg(ch_type,e_WR3,m_read_wr_copy(ch_type,e_WR3)&(_INVERSE_ e_rx_enable))
/*****************************************************************************/
#define m_tx_enable(ch_type)\
        m_wr_reg(ch_type,e_WR5,m_read_wr_copy(ch_type,e_WR5)|e_tx_enable)
/*****************************************************************************/
#define m_tx_disable(ch_type)\
        m_wr_reg(ch_type,e_WR5,m_read_wr_copy(ch_type,e_WR5)&(_INVERSE_ e_tx_enable))
/*****************************************************************************/
#define m_tx_int_enable(ch_type)\
        m_wr_reg(ch_type,e_WR1,m_read_wr_copy(ch_type,e_WR1)|e_tx_int_enable)
/*****************************************************************************/
#define m_tx_int_disable(ch_type)\
        m_wr_reg(ch_type,e_WR1,m_read_wr_copy(ch_type,e_WR1)&(_INVERSE_
e_tx_int_enable))
/*****************************************************************************/
#define m_check_tx_enable(ch_type)\
        (AM8530_int_copy.ch_type.write_regs[e_WR5_int_copy]& e_tx_enable        ?
        e_signal_on     :       e_signal_off)
/*****************************************************************************/
#define m_rx_enable(ch_type)\
        m_wr_reg(ch_type,e_WR3,m_read_wr_copy(ch_type,e_WR3)|e_rx_enable)
/*****************************************************************************/
#define m_rx_disable(ch_type)\
        m_wr_reg(ch_type,e_WR3,m_read_wr_copy(ch_type,e_WR3)&(_INVERSE_ e_rx_enable))



/*****************************************************************************/
extern  t_AM8530_int_copy                       xdata   AM8530_int_copy;
extern  t_union_of_word_with_bytes      code    time_constant[s_max_baud];

/*****************************************************************************/
extern void p_initialize_8530_with_setup();

#endif


/*  BYTE Register  */
sfr P0  = 0x80;
sfr P1  = 0x90;
sfr P2  = 0xA0;
sfr P3  = 0xB0;
sfr PSW = 0xD0;
sfr ACC = 0xE0;
sfr B   = 0xF0;
```

```
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;


/*  BIT Register  */
/*  PSW   */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;

/*  TCON  */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
sbit IE0  = 0x89;
sbit IT0  = 0x88;

/*  IE   */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
sbit EX1  = 0xAA;
sbit ET0  = 0xA9;
sbit EX0  = 0xA8;

/*  IP   */
sbit PS   = 0xBC;
sbit PT1  = 0xBB;
sbit PX1  = 0xBA;
sbit PT0  = 0xB9;
sbit PX0  = 0xB8;

/*  P1  */
sbit P1_0  = P1 ^ 0;
sbit P1_1  = P1 ^ 1;
sbit P1_2  = P1 ^ 2;
```

```
sbit P1_3  = P1 ^ 3;
sbit P1_4  = P1 ^ 4;
sbit P1_5  = P1 ^ 5;
sbit P1_6  = P1 ^ 6;
sbit P1_7  = P1 ^ 7;

/*  P3  */
sbit RD  = 0xB7;
sbit WR  = 0xB6;
sbit T1  = 0xB5;
sbit T0  = 0xB4;
sbit INT1 = 0xB3;
sbit INT0 = 0xB2;
sbit TXD  = 0xB1;
sbit RXD  = 0xB0;

/*  SCON  */
sbit SM0  = 0x9F;
sbit SM1  = 0x9E;
sbit SM2  = 0x9D;
sbit REN  = 0x9C;
sbit TB8  = 0x9B;
sbit RB8  = 0x9A;
sbit TI  = 0x99;
sbit RI  = 0x98;

/*****************************************************************************/
/*                                                                           */
/* MENU.H                                                                    */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/

#ifndef _MENU_H
#define _MENU_H
/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
#include <c:\project\tez\software\mcs51\include\statport.h>
#include <c:\project\tez\software\mcs51\include\time0int.h>
#include <c:\project\tez\software\mcs51\include\display.h>

/*****************************************************************************/
typedef enum  {
              e_menu_main_comm,
              e_menu_main_store,
              e_menu_main_test
              } t_menu_main;

typedef enum  {
              e_comm_type_async,
              e_comm_type_sync
              } t_comm_type;
```

```
typedef enum  {
              e_menu_async_speed,
              e_menu_async_data_bits,
              e_menu_async_stop_bits,
              e_menu_async_parity
              } t_menu_async;

typedef enum  {
              e_menu_sync_protocol,
              e_menu_sync_type,
              e_menu_sync_speed
              } t_menu_sync;

typedef enum  {
              e_sync_sdlc,
              e_sync_hdlc
              } t_sync_protocol;

typedef enum  {
              e_sync_mono,
              e_sync_bi
              } t_sync_type;

typedef enum  {
              e_300_baud,
              e_600_baud,
              e_1200_baud,
              e_2400_baud,
              e_4800_baud,
              e_7200_baud,
              e_9600_baud,
              e_12000_baud,
              e_14400_baud,
              e_19200_baud
              } t_baud_rate;
#define       s_max_baud      10

typedef enum  {
              e_5_data_bits,
              e_6_data_bits,
              e_7_data_bits,
              e_8_data_bits
              } t_data_bits;

typedef enum  {
              e_1_stop_bit,
              e_1_5_stop_bit,
              e_2_stop_bit
              } t_stop_bits;

typedef enum  {
              e_none_parity,
              e_odd_parity,
              e_even_parity
```

```
                } t_parity;

typedef struct  {
                /*t_sync_protocol*/     byte    sync_protocol;
                /*t_sync_type*/         byte    sync_type;
                }t_sync_setup_parameters;

typedef struct  {
                /*t_stop_bits*/ byte            stop_bits;
                /*t_parity*/    byte            parity;
                }t_async_setup_parameters;

typedef union   {
                t_sync_setup_parameters         sync;
                t_async_setup_parameters async;
                }t_setup_parameters;

typedef enum    {
                e_store_trigger_type,
                e_store_enter_special_char,
                e_store_trigger_time,
                e_store_run,
                e_store_stop,
                e_store_clear,
                e_store_display_type,
                e_store_display
                } t_menu_store;

typedef enum    {
                e_trigger_dte_first_char,
                e_trigger_dce_first_char,
                e_trigger_dte_special_char,
                e_trigger_dce_special_char,
                e_trigger_enter_key,
                e_trigger_rts,
                e_trigger_cts,
                e_trigger_dtr,
                e_trigger_dsr,
                e_trigger_dcd
                } t_trigger_type;

typedef enum    {
                e_time_start_of_buffer,
                e_time_center_of_buffer,
                e_time_end_of_buffer
                } t_trigger_time;

typedef enum    {
                e_display_status_source,
                e_display_data_source,
                e_display_data_char_type
                } t_menu_display_type;

typedef enum    {
                e_display_rts,
```

```
                e_display_cts,
                e_display_dtr,
                e_display_dsr,
                e_display_dcd
                } t_display_status_source;

typedef enum    {
                e_display_DTE_data,
                e_display_DCE_data
                } t_display_data_source;

typedef enum    {
                e_display_data_hex,
                e_display_data_text
                } t_display_data_char_type;

typedef enum    {
                e_test_ram,
                e_test_key,
                e_test_display
                } t_menu_test;

typedef enum    {
                e_store_stat_waiting,
                e_store_stat_runing,
                e_store_stat_stoped
                } t_store_stat;

typedef  struct     {
                /*t_comm_type*/         byte                    comm_type;
                /*t_baud_rate*/         byte                    baud_rate;
                /*t_data_bits*/         byte                    data_bits;
                t_setup_parameters                                    comm_setup;
                /*t_trigger_type*/      byte                    trigger_type;
                /*t_trigger_time*/      byte                    trigger_time;
                /*t_store_stat*/   byte                  store_stat;
                /*t_display_status_source*/       byte    display_status_source;
                /*t_display_data_source*/ byte    display_data_source;
                /*t_display_data_char_type*/ byte display_data_char_type;
                byte                                                  trigger_special_char;
                word                                                  data_start_index;
                word                                                  data_end_index;
                word                                                  status_start_index;
                word                                                  status_end_index;
                }t_system_parameters;

/*****************************************************************************/
typedef struct      {
                byte                                  stack_pointer;
                t_void_arg_proc_ptr             stack[20];
                } t_menu_stack;

/*****************************************************************************/
typedef enum        {
                e_item_choosed_mask=0x80,
```

```
                e_up_pressed_mask=0x40,
                e_menu_pressed_mask=0x20,
                e_get_item_mask=0x1F
                } t_choose_return_code_mask;

/*****************************************************************************/
typedef  byte (*code t_menu_str)[LCD_COLUMN_NO+1];
        /* max_display_char+1 (+ 1 for 0x00 string terminator char) */

/*****************************************************************************/
extern   t_system_parameters                xdata    system_parameters;

/*****************************************************************************/
extern   void menu();

#endif

/*****************************************************************************/
/*                                                   */
/* GLOTYP.H                                                  */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*****************************************************************************/

#ifndef  _GLOTYP_H
#define  _GLOTYP_H

/************** global  type  definition ************************************/
typedef unsigned char     byte;
typedef unsigned int      word;
typedef unsigned long     dword;
typedef enum      {
                  b_false,
                  b_true
                  } t_boolean;

typedef enum      {
                  e_off,
                  e_on
                  } t_bit_operations;

typedef enum      {
                  e_signal_off,
                  e_signal_on
                   } t_signal_values;

/*****************************************************************************/
typedef void (*t_void_arg_proc_ptr) (void);
typedef void (*t_byte_arg_proc_ptr) (byte );
        /* t_byte_rtn_void_arg_funct_ptr is synonym for "pointer to   function returning byte" */
typedef byte (*t_byte_rtn_void_arg_funct_ptr) (void);
        /*t_byte_rtn_void_arg_funct_ptr is synonym for "pointer to   function returning byte"  */
```

```
/*****************************************************************************/
#ifdef CPU_8051
        typedef struct      {
                                unsigned int      high_byte       :8;
                                unsigned int      low_byte        :8;
                                } t_word_in_bit_fields;

        typedef struct      {
                                byte      high_byte;
                                byte      low_byte;
                                } t_word_in_two_bytes;
#else
        typedef struct      {
                                unsigned int      low_byte        :8;
                                unsigned int      high_byte       :8;
                                } t_word_in_bit_fields;

        typedef struct      {
                                byte      low_byte;
                                byte      high_byte;
                                } t_word_in_two_bytes;
#endif

typedef union     {
                        word                                body;
                        t_word_in_bit_fields        word_in_bit_fields;
                        t_word_in_two_bytes                 word_in_two_bytes;
                } t_union_of_word_with_bytes;

/*****************************************************************************/
#define  _AND_                          &&
#define  _OR_                           ||
#define  _EQU_                          ==
#define  _NOT_EQU_                      !=
#define  _GE_                           >=
#define  _GT_                           >
#define  _LE_                           <=
#define  _LT_                           <
#define  _NOT_                          !
#define  _ONES_COMPLEMENT_              ~
#define  _INVERSE_                      _ONES_COMPLEMENT_

/*****************************************************************************/
#define  binary(b7,b6,b5,b4,b3,b2,b1,b0) \
        ((b7<<7)|(b6<<6)|(b5<<5)|(b4<<4)|(b3<<3)|(b2<<2)|(b1<<1)|b0)

/*****************************************************************************/
#define bit0_on         0x01
#define bit1_on         0x02
#define bit2_on         0x04
#define bit3_on         0x08
#define bit4_on         0x10
#define bit5_on         0x20
```

```
#define bit6_on         0x40
#define bit7_on         0x80
#define all_bits_on             0xFF
#define bit0_off        0xFE
#define bit1_off        0xFD
#define bit2_off        0xFB
#define bit3_off        0xF7
#define bit4_off        0xEF
#define bit5_off        0xDF
#define bit6_off        0xBF
#define bit7_off        0x7F
#define all_bits_off    0x00

/*****************************************************************************/
#define null    0
#define true    1
#define false   0
#define first   0

/*****************************************************************************/
#define  m_size_of_item(array)    (sizeof(array)/sizeof(array[0]))

/*****************************************************************************/
#define m_calc_circular_buffer_ptrs_diff(ptr1, ptr2,circular_buffer_length) \
        (ptr1>=ptr2 ? (ptr1-ptr2) : (ptr1+circular_buffer_length-ptr2))

/*****************************************************************************/
#define m_inc_circular_buffer_index(index, circular_buffer_length) \
        do{if (index _EQU_ (circular_buffer_length-1))      {index=0;}      else{index++;}}while(0)

/*****************************************************************************/
#define m_dec_circular_buffer_index(index, circular_buffer_length) \
        do{if (index _EQU_ 0)    {index=(circular_buffer_length-1);}      else{index--;}}while(0)

#endif

/*****************************************************************************/
/*                                                                           */
/* GENUTIL.H                                                                 */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
#ifndef  _GENUTIL_H
#define  _GENUTIL_H

/*****************************************************************************/
#define m_convert_asc(hex_digit) \
        (((byte)hex_digit<0x0a) ? (byte)(hex_digit+0x30) : (byte)(hex_digit+0x37))
#define m_calc_hi_nibble(hex_digit) \
        ((byte)(hex_digit&0xF0) >> 4)
#define m_calc_lo_nibble(hex_digit) \
        ((byte)hex_digit&0x0F)
```

```
#define m_calc_hi_byte(word_var) \
        ((byte)(((word)word_var&0xFF00) >> 8))
#define m_calc_lo_byte(word_var) \
        ((byte)((word)word_var&0x00FF))
#define  m_disable_ints()                    EA = 0
#define  m_enable_ints()                             EA = 1

/*****************************************************************************/
extern byte        idata     *p_hex_str(byte   byte_data);

#endif

/*****************************************************************************/
/*                                                   */
/* EXT0INT.H                                              */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*****************************************************************************/

#ifndef _EXT0INT_H
#define _EXT0INT_H
/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>
#include <c:\project\tez\software\mcs51\include\sccutil.h>

/*****************************************************************************/
extern   byte     idata     CHA_last_received_byte;
extern   byte     idata     CHB_last_received_byte;


/*****************************************************************************/
extern void p_int0_init();

#endif

/*****************************************************************************/
/*                                                   */
/* DISPLAY.H                                              */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*****************************************************************************/
#ifndef  _DISPLAY_H
#define  _DISPLAY_H

/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>

/*****************************************************************************/
#define  m_display_port_write_FF
```

```
/*
#define  m_display_uninterruptable
#define  m_display_port_read_slow
#define  m_display_port_write_slow
#define  m_display_port_write_FF
*/

#ifdef   m_display_port_write_FF
         #define  m_wr_display_port_FF()          m_display_port=0xFF
#else
         #define  m_wr_display_port_FF()          do{}while(0)
#endif

#ifdef   m_display_uninterruptable
         #define  m_inhibit_interrupts()          m_disable_ints()
         #define  m_uninhibit_interrupts()  m_enable_ints()
#else
         #define  m_inhibit_interrupts()          do{}while(0)
         #define  m_uninhibit_interrupts()  do{}while(0)
#endif

#ifdef   m_display_port_write_slow
         #define  m_wait_1_t_clk_on_wr()                    _nop_();_nop_()
#else
         #define  m_wait_1_t_clk_on_wr()                    do{}while(0)
#endif

#ifdef   m_display_port_read_slow
         #define  m_wait_1_t_clk_on_rd()                    _nop_()
#else
         #define  m_wait_1_t_clk_on_rd()                    do{}while(0)
#endif

/*****************************************************************************/
#define LCD_COLUMN_NO                      16
#define EN(bit_val)                        INT1=bit_val
#define RW(bit_val)                        T0=bit_val
#define RS(bit_val)                        T1=bit_val
#define  m_display_port                    P1
#define  m_display_busy_flag               0x80
#define  m_clear_display()                 p_display_cmd_BF(0x01)
#define  m_display_cursor_on()     p_display_cmd_BF(0x0D)
#define  m_display_cursor_off()    p_display_cmd_BF(0x0F)

/*****************************************************************************/
extern   void p_delay(word milisec);
extern   void p_wait_busy_flag(void);
extern   void p_display_cmd_BF(byte cmd);
extern   void p_display_cmd_NBF(byte cmd);
extern   void p_set_DDRA(byte row,byte char_index);
extern   void p_WDDRA(byte c);
extern   byte p_RDDRA(void);
extern   void p_set_CGRA(byte address);
extern   void p_reset_display(void);
```

```
extern   void p_line_display(byte row, byte *mess_ptr, byte mess_len);
extern   void p_display_at(byte row, byte column, byte *mess_ptr, byte mess_len);
extern   void p_write_special_fonts_to_CGRAM();
extern   void p_initialize_display();

#endif

/*****************************************************************************/
/*                                                                           */
/* AM8530.H                                                                  */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
#ifndef _AM8530_H
#define _AM8530_H
/*****************************************************************************/
#include <c:\project\tez\software\mcs51\include\target.h>

/*****************************************************************************/
typedef  enum    {
                 e_WR0_int_copy,
                 e_WR1_int_copy,
                 e_WR2_int_copy,
                 e_WR3_int_copy,
                 e_WR4_int_copy,
                 e_WR5_int_copy,
                 e_WR6_int_copy,
                 e_WR7_int_copy,
                 e_WR7_prime_int_copy,
                 e_WR8_int_copy,
                 e_WR9_int_copy,
                 e_WR10_int_copy,
                 e_WR11_int_copy,
                 e_WR12_int_copy,
                 e_WR13_int_copy,
                 e_WR14_int_copy,
                 e_WR15_int_copy,
                 s_write_regs_number
                 }  t_int_copy_write_registers;

typedef  enum    {
                 e_WR0_mask=0x0F,
                 e_WR1_mask=0xFF,
                 e_WR2_mask=0xFF,
                 e_WR3_mask=0xFF,
                 e_WR4_mask=0xFF,
                 e_WR5_mask=0xFF,
                 e_WR6_mask=0xFF,
                 e_WR7_mask=0xFF,
                 e_WR8_mask=0xFF,
                 e_WR9_mask=0x3F,
                 e_WR10_mask=0xFF,
```

```
                e_WR11_mask=0xFF,
                e_WR12_mask=0xFF,
                e_WR13_mask=0xFF,
                e_WR14_mask=0x1F,
                e_WR15_mask=0xFF
                }  t_int_copy_mask;

typedef struct  {
                byte    write_regs[s_write_regs_number];
                }       t_AM8530_int_copy_channel;

typedef struct  {
                t_AM8530_int_copy_channel       CHB;
                t_AM8530_int_copy_channel       CHA;
                }       t_AM8530_int_copy;

/*****************************************************************************/
typedef enum    {
                e_WR0,
                e_WR1,
                e_WR2,
                e_WR3,
                e_WR4,
                e_WR5,
                e_WR6,
                e_WR7,
                e_WR8,
                e_WR9,
                e_WR10,
                e_WR11,
                e_WR12,
                e_WR13,
                e_WR14,
                e_WR15
                }  t_write_registers;

typedef enum    {
                e_RR0,
                e_RR1,
                e_RR2,
                e_RR3,
                e_RR6=6,
                e_RR7=7,
                e_RR8=8,
                e_RR10=10,
                e_RR12=12,
                e_RR13=13,
                e_RR15=15
                }  t_read_registers;

typedef enum    {
                e_point_high                                    =binary(0,0,0,0,1,0,0,0),
                e_reset_ext_status_interrupts   =binary(0,0,0,1,0,0,0,0),
                e_send_abort                                    =binary(0,0,0,1,1,0,0,0),
                e_enable_int_on_next_rx_char    =binary(0,0,1,0,0,0,0,0),
```

```
                e_reset_tx_int_pending                  =binary(0,0,1,0,1,0,0,0),
                e_error_reset                           =binary(0,0,1,1,0,0,0,0),
                e_reset_highest_ius                     =binary(0,0,1,1,1,0,0,0),
                e_reset_rx_crc_checker                  =binary(0,1,0,0,0,0,0,0),
                e_reset_tx_crc_checker                  =binary(1,0,0,0,0,0,0,0),
                e_reset_tx_underrun_eom_latch   =binary(1,1,0,0,0,0,0,0)
                }       t_WR0;

typedef enum    {
                e_ext_int_enable                        =(bit0_on),
                e_tx_int_enable                         =(bit1_on),
                e_parity_is_special_condition   =(bit2_on),
                e_rx_int_disable
        =all_bits_off,
                e_rx_int_on_first_char_or_special_condition      =binary(0,0,0,0,1,0,0,0),
                e_int_on_all_rx_chars_or_special_condition      =binary(0,0,0,1,0,0,0,0),
                e_rx_int_on_special_condition_only
        =binary(0,0,0,1,1,0,0,0),
                e_wait_dma_request_on_rx_tx
        =(bit5_on),
                e_wait_dma_request_function
        =(bit6_on),
                e_wait_dma_request_enable
        =(bit7_on)
                }       t_WR1;

typedef enum    {
                e_rx_enable
                =(bit0_on),
                e_sync_char_load_inhibit                                =(bit1_on),
                e_address_search_mode
        =(bit2_on),
                e_rx_crc_enable
        =(bit3_on),
                e_enter_hunt_enable
        =(bit4_on),
                e_auto_enables
        =(bit5_on),
                e_rx_5_bits_char
        =all_bits_off,
                e_rx_7_bits_char
        =binary(0,1,0,0,0,0,0,0),
                e_rx_6_bits_char
        =binary(1,0,0,0,0,0,0,0),
                e_rx_8_bits_char
        =binary(1,1,0,0,0,0,0,0)
                }       t_WR3;

typedef enum    {
                e_parity_disable        =(bit0_off),
                e_parity_enable                 =(bit0_on),
                e_parity_odd                    =(bit1_off),
                e_parity_even                   =(bit1_on),
                e_sync_modes_enable             =all_bits_off,
                e_1_stop_bit_char               =binary(0,0,0,0,0,1,0,0),
```

```
                e_1_5_stop_bits_char     =binary(0,0,0,0,1,0,0,0),
                e_2_stop_bits_char              =binary(0,0,0,0,1,1,0,0),
                e_8_bits_sync_char              =all_bits_off,
                e_16_bits_sync_char             =binary(0,0,0,1,0,0,0,0),
                e_sdlc_mode                             =binary(0,0,1,0,0,0,0,0),
                e_external_sync_mode    =binary(0,0,1,1,0,0,0,0),
                e_x1_clk_mode                   =all_bits_off,
                e_x16_clk_mode                  =binary(0,1,0,0,0,0,0,0),
                e_x32_clk_mode                  =binary(1,0,0,0,0,0,0,0),
                e_x64_clk_mode                  =binary(1,1,0,0,0,0,0,0)
                }       t_WR4;

typedef enum    {
                e_tx_crc_enable
        =(bit0_on),
                e_rts
                =(bit1_on),
                e_sdlc_crc_16
        =(bit2_on),
                e_tx_enable
                =(bit3_on),
                e_send_break
        =(bit4_on),
                e_tx_5_bits_char
        =all_bits_off,
                e_tx_7_bits_char
        =binary(0,0,1,0,0,0,0,0),
                e_tx_6_bits_char
        =binary(0,1,0,0,0,0,0,0),
                e_tx_8_bits_char
        =binary(0,1,1,0,0,0,0,0),
                e_dtr
                =(bit7_on)
                }       t_WR5;

typedef enum    {
                e_auto_tx_flag
        =(bit0_on),
                e_auto_eom_reset
        =(bit1_on),
                e_auto_rts_deactivation
        =(bit2_on),
                e_txd_forced_high_in_sdlc_nrzi_mode                             =(bit3_on),
                e_dtr_req_timing_mode
        =(bit4_on),
                e_receive_crc
        =(bit5_on),
                e_extended_read_enable
        =(bit6_on)
                }       t_WR7_prime;

typedef enum    {
                e_vis
        =(bit0_on),
```

```
        e_nv
    =(bit1_on),
        e_dlc
    =(bit2_on),
        e_mie
    =(bit3_on),
        e_status_high_status_low                =(bit4_on),
        e_interrupt_masking_without_intack      =(bit5_on),
        e_channel_reset_B                                   =(bit6_on),
        e_CHB_channel_reset
    =e_channel_reset_B,
        e_channel_reset_A                                   =(bit7_on),
        e_CHA_channel_reset
    =e_channel_reset_A,
        e_force_hardware_reset
    =(e_channel_reset_A|e_channel_reset_B)
        }       t_WR9;

typedef enum    {
        e_6_bit_8_bit_sync              =(bit0_on),
        e_loop_mode                             =(bit1_on),
        e_abort_flag_on_underrun        =(bit2_on),
        e_mark_flag_idle                =(bit3_on),
        e_go_active_on_poll             =(bit4_on),
        e_nrz                                   =all_bits_off,
        e_nrzi                                  =binary(0,0,1,0,0,0,0,0),
        e_fm1                                   =binary(0,1,0,0,0,0,0,0),
        e_fm0                                   =binary(0,1,1,0,0,0,0,0),
        e_crc_preset_1_or_0             =(bit7_on)
        }       t_WR10;

typedef enum    {
        e_trxc_out_src_xtal_output                      =all_bits_off,
        e_trxc_out_src_tx_clk                           =binary(0,0,0,0,0,0,0,1),
        e_trxc_out_src_br_generator_output      =binary(0,0,0,0,0,0,1,0),
        e_trxc_out_src_dpll_output                      =binary(0,0,0,0,0,0,1,1),
        e_trxc_output                                           =(bit2_on),
        e_tx_clk_src_rtxc_pin                           =all_bits_off,
        e_tx_clk_src_trxc_pin                           =binary(0,0,0,0,1,0,0,0),
        e_tx_clk_src_br_generator_output =binary(0,0,0,1,0,0,0,0),
        e_tx_clk_src_dpll_output                =binary(0,0,0,1,1,0,0,0),
        e_rx_clk_src_rtxc_pin                           =all_bits_off,
        e_rx_clk_src_trxc_pin                           =binary(0,0,1,0,0,0,0,0),
        e_rx_clk_src_br_generator_output =binary(0,1,0,0,0,0,0,0),
        e_rx_clk_src_dpll_output                =binary(0,1,1,0,0,0,0,0),
        e_rtxc_xtal                                                             =(bit7_on)
        }       t_WR11;

typedef enum    {
        e_br_generator_enable   =(bit0_on),
        e_br_generator_src_is_PCLK_input        =(bit1_on),
        e_dtr_req_function              =(bit2_on),
        e_auto_echo                             =(bit3_on),
        e_local_loopback        =(bit4_on),
        e_enter_search_mode             =binary(0,0,1,0,0,0,0,0),
```

```
                e_reset_missing_clk             =binary(0,1,0,0,0,0,0,0),
                e_disable_dpll                  =binary(0,1,1,0,0,0,0,0),
                e_set_src_br_generator  =binary(1,0,0,0,0,0,0,0),
                e_set_src_rtxc                  =binary(1,0,1,0,0,0,0,0),
                e_set_fm_mode                   =binary(1,1,0,0,0,0,0,0),
                e_set_nrzi_mode                 =binary(1,1,1,0,0,0,0,0)
                }       t_WR14;

typedef enum    {
                e_sdlc_hdlc_enhancement_enable                          =(bit0_on),
                e_zero_count_ie
        =(bit1_on),
                e_10_x_19_bit_frame_status_fifo_enable          =(bit2_on),
                e_dcd_ie
                =(bit3_on),
                e_sync_hunt_ie
        =(bit4_on),
                e_cts_ie
        =(bit5_on),
                e_tx_underrun_eom_ie
        =(bit6_on),
                e_break_abort_ie
        =(bit7_on)
                }       t_WR15;

typedef enum    {
                e_rx_char_available             =(bit0_on),
                e_zero_count                    =(bit1_on),
                e_tx_buffer_empty               =(bit2_on),
                e_dcd                                   =(bit3_on),
                e_sync_hunt                             =(bit4_on),
                e_cts                                   =(bit5_on),
                e_tx_underrun_eom               =(bit6_on),
                e_break_abort                   =(bit7_on)
                }       t_RR0;

typedef enum    {
                e_all_sent                              =(bit0_on),
                e_residue_code_2                        =(bit1_on),
                e_residue_code_1                        =(bit2_on),
                e_residue_code_0                        =(bit3_on),
                e_parity_error                  =(bit4_on),
                e_rx_overrun_error              =(bit5_on),
                e_crc_framing_error             =(bit6_on),
                e_end_of_frame                  =(bit7_on)
                }       t_RR1;

typedef enum    {
                e_channel_B_ext_or_stat_ip      =(bit0_on),
                e_channel_B_tx_ip                               =(bit1_on),
                e_channel_B_rx_ip                               =(bit2_on),
                e_channel_A_ext_or_stat_ip      =(bit3_on),
                e_channel_A_tx_ip                               =(bit4_on),
                e_channel_A_rx_ip                               =(bit5_on)
                }       t_RR3;
```

```
typedef enum    {
                e_msb_byte_count                                =binary(0,0,1,1,1,1,1,1),
                e_fifo_data_available_status    =(bit6_on),
                e_fifo_overflow_status                  =(bit7_on)
                }       t_RR7;

typedef enum    {
                e_on_loop                               =(bit1_on),
                e_loop_sending                  =(bit4_on),
                e_two_clks_missing              =(bit6_on),
                e_one_clk_missing               =(bit7_on)
                }       t_RR10;

typedef t_WR15 t_RR15;

/*************************************************************************/
typedef struct  {
                byte    control_port;
                byte    data_port;
                }       t_AM8530_channel;

typedef struct  {
                t_AM8530_channel        CHB;
                t_AM8530_channel        CHA;
                }       t_AM8530;

/*************************************************************************/
#define AM_8530_PCLK_FRQ                                (6144000)
#define m_enable_AM8530()
        A17(((ADDRESS_OF_AM8530&0x20000)>>17));  \

        A16(((ADDRESS_OF_AM8530&0x10000)>>16))
#define AM8530                                                          (*(t_AM8530
xdata*)((word)ADDRESS_OF_AM8530 & 0xFFFF))
#define m_clock_mode                                    e_x16_clk_mode
#define clock_mode                                              16

/*************************************************************************/
#define m_time_constant(baud_rate)      \
        (word)( \
        ((dword)(AM_8530_PCLK_FRQ)/ \
        (dword)2/       \
        (dword)clock_mode/      \
        (dword)baud_rate)-(dword)2)

#endif
```

ÖZGEÇMİŞ

Doğum Tarihi : 30.09.1967

Doğum Yeri : Gönence

İlkokul Mezuniyeti : 1978 Bursa Cumhuriyet İlkokulu

Ortaokul Mezuniyeti : 1981 Bursa Duaçınarı Ortaokulu

Lise Mezuniyeti : 1984 Bursa Demirtaşpaşa Endüstri Meslek Lisesi Elektronik Bölümü

Üniversite Mezuniyeti : 1990 Yıldız Üniversitesi, Mühendislik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümü

1991 yılında Yıldız Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Yüksek Lisans Programı'na kayıt oldum.