

YILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Hata Düz., Mod., veri ile. ve
Bil., Eri., Prog.

Yüksek Lisans Tezi

Ümit Kalkan

1991

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

Kot	: R 152 162
Alındığı Yer	: Fen Bil Ens.
Tarih	: 10.4.95
Fatura	:
Fiyatı	: 329.000.-
Ayniyat No	: 1-6
Kayıt No	: 50991
UDC	:
Ek	:

YILDIZ ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Özet.....	1
Önsöz.....	2

Bölüm 1	3
Veri İletişimi Hakkında Genel Bilgi.....	5

1.1 Dijital Sistem Mimarileri.....	6
1.1.1 Giriş.....	6
1.1.2 Yerel Veri İletişim Ağları.....	10
1.1.3 Yerel Veri İletişim Ağları.....	10
1.1.4 Genel Bilgi.....	14
1.1.5 Ortak Paket Veri İletişim Ağları.....	16
1.1.6 Yerel Veri İletişim Ağları.....	18

1.2 Elektriksel Ara Birimler.....	23
1.2.1 Giriş.....	23
1.2.2 Transmisyon Ortamları.....	24
1.2.2.1 İki Telli Açık Hatlar.....	25
1.2.2.2 Burulmuş Çift İlişkentli Hatlar (Twisted Pair).....	26
1.2.2.3 Koaksiyel Kablo.....	27
1.2.2.4 Optik Kılcalıklar (Optical Fibres).....	28
1.2.2.5 Mikrodalgalar.....	29

YÜKSEK LİSANS TEZİ

Müh. Ümit KALKAN

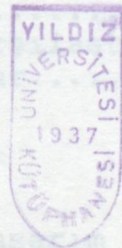
Tez Yöneticisi:
Doç. Dr. Filiz GÜNEŞ

1.3 Hata Tespit Yöntemleri.....	45
1.3.1 Hata Tespiti.....	46
1.3.2 Hata Tespiti Karşılaştırma Metodu.....	47
1.3.3 Polinomial Kodlar.....	47

Bölüm 2	50
UNICOM Yazılımının Temel Özellikleri.....	50

2.1 UNICOM Yazılımının Temel Özellikleri.....	51
2.1.1 Giriş.....	51
2.1.2 Yazılım Modülleri.....	52
2.1.2.1 Çözüm.....	52

İSTANBUL 1991



İÇİNDEKİLER

Özet.....	54
Önsöz.....	55
	57
	58
	59
	59
	59
	60

Bölüm 1

Veri İletişimi Hakkında Genel Bilgi.....	5
1.1 Dağıtılmış Sistem Mimarileri.....	6
1.1.1 Giriş.....	6
1.1.2 İletişim Ağı Sınıflaması.....	7
1.1.3 Tarihi Gelişme.....	10
1.1.4 Özel Bilgisayar İletişim Ağları.....	14
1.1.5 Ortak Paket Anahtarlama Veri Şebekesi.....	16
1.1.6 Yerel Veri İletişim Ağları.....	18
1.2 Elektriksel Ara Birimler.....	23
1.2.1 Giriş.....	23
1.2.2 Transmisyon Ortamları.....	24
1.2.2.1 İki Telli Açık Hatlar.....	25
1.2.2.2 Burulmuş Çift İletkenli Hatlar (Twisted Pair).....	26
1.2.2.3 Koaksiyel Kablo.....	27
1.2.2.4 Optik Kılcallar (Optical Fibres).....	28
1.2.2.5 Mikrodalgalar.....	29
1.2.3 İşaret Tipleri.....	30
1.2.3.1 RS-232C/V.24.....	31
1.2.3.2 20 ma Akım Çevrimi.....	33
1.2.3.3 RS-422/V.11.....	34
1.2.3.4 Modem İşaretleri.....	35
1.2.4 Fiziksel Katman Arabirim Standartları.....	39
1.2.4.1 RS-232C/V.24.....	39
1.2.4.2 RS-449/V.35.....	44
1.3 Hata Sezinleme Yöntemleri.....	45
1.3.1 Parity.....	46
1.3.2 Blok Toplamı Karşılaştırma Metodu.....	47
1.3.3 Polynomial Kodlar.....	47

Bölüm 2

UMICOMM Yazılımının Tasarım Özellikleri.....	50
2.1 UMICOMM Yazılımının Teknik Tanıtımı.....	51
2.1.1 Giriş.....	51
2.1.2 Yazılım Modülleri.....	52
2.1.2.1 CURSES.....	52

2.1.2.2	DISPLYIO.....	54
2.1.2.3	FIELDIO.....	55
2.1.2.4	SCRNIO.....	57
2.1.2.5	DIALOG.....	58
2.1.2.6	DATAMAN.....	58
2.1.2.7	IRQUTIL.....	58
2.1.2.8	COMM.....	59
2.1.2.9	TERMEMUL.....	59
2.1.2.10	XMODEM.....	60
2.1.2.11	UMCOMMON.....	62
2.1.2.12	COMMENU.....	63
2.2	Programların Kullanılışları.....	63
2.2.1	UMICOMM Programının Kullanılışı.....	63
2.2.1.1	ALT-H Yardım.....	64
2.2.1.2	ALT-S Hat Parametrelerini Şartlama.....	64
2.2.1.3	ALT-M Modem Parametrelerini Şartlama.....	64
2.2.1.4	ALT-T Terminal Emulasyonu Şartlama.....	64
2.2.1.5	ALT-D Arama Yapma (Numara Defteri).....	65
2.2.1.6	ALT-K Şartlamaları Saklama.....	65
2.2.1.7	ALT-X Programdan Çıkma.....	66
2.2.1.8	ALT-B BREAK Gönderme.....	66
2.2.1.9	PGUP XMODEM ile Dosya Gönderme.....	66
2.2.1.10	PGDN XMODEM ile Dosya Alma.....	67
2.2.2	UMISRV Programının Kullanılışı.....	67
2.2.2.1	BANNER.FIL Dosyası.....	67
2.2.2.2	PASSWD.FIL Dosyası.....	67
2.2.2.3	MAINMENU.MNU Dosyası.....	68
2.2.2.4	Bilgi Servisinden Faydalanma.....	69

Bölüm 3

Program Kaynak Dökümleri.....	70
DIALOG.C.....	71
COMM.C.....	73
IRQUTIL.C.....	75
UMICOMM.C.....	77
COMMENU.C.....	79
DISPLYIO.C.....	84
CURSES.C.....	88
SCRNIO.C.....	100
FIELDIO.C.....	113
TERMEMUL.C.....	137
XMODEM.C.....	153
DATAMAN.C.....	165
UMCOMMON.C.....	168
COMM.H.....	211
SCRNIO.H.....	213

CURSES.H.....	: 214
FIELDIO.H.....	: 219
NULLPTR.H.....	: 222
DATATYPE.H.....	: 223

EK : HAYES Akıllı Modem Komutları.....	: 224
Kaynaklar.....	: 225
Özgeçmiş.....	: 226

Sayısal veri iletişimi, günümüzde her gün daha büyük sınırlara ulaşarak büyüyen elektronik mühendisliği konusunun geniş bir dalı haline gelmiştir. Artık günümüzde ses, görüntü, doküman gibi verilerin tamamen sayısal olarak iletilmesi mümkündür. Çok yakın bir gelecekte, düşünülebilecek her türlü bilgi sayısal olarak iletilecektir.

Sayısal veri iletişimi konusunun gelişmesinde en büyük rol oynayan araçlar tabii ki bilgisayarlardır. Sayısal veri iletimi konusu, bilgisayarlar ve ilişkili çevre birimleri arasındaki haberleşme ihtiyacı sayesinde gelişmiştir. Bu proje, kişisel bilgisayarların birbirleriyle ve diğer bilgisayar tipleriyle haberleşebilmelerini, kullanıcıların bu sistemler arasında dosya transferi yapabilmelerini sağlamak amacıyla geliştirilmiş bir yazılımın tasarımı ve gerçekleştirilmesi içermekle kalmayıp, küçük bir kişisel bilgisayarda çalıştırılabilecek küçük çaplı bir bilgi servisi programının geliştirilmesini de içine almaktadır. Ayrıca bu dokümanın ilk bölümünde veri iletişimi hakkında temel bilgiler bulunabilir.

Özgeçmiş ve Kaynaklar

Özet

Sayısal veri iletişimi, günümüzde her gün daha büyük sınırlara ulaşarak büyüyen elektronik mühendisliği konusunun geniş bir dalı haline gelmiştir. Artık günümüzde ses, görüntü, döküman gibi verilerin tamamen sayısal olarak iletilmesi mümkündür. Çok yakın bir gelecekte, düşünülebilecek her türlü bilgi sayısal olarak iletilecektir.

Sayısal veri iletişimi konusunun gelişmesinde en büyük rol oynayan araçlar tabii ki bilgisayarlardır. Sayısal veri iletimi konusu, bilgisayarlar ve ilişkili çevre birimleri arasındaki haberleşme ihtiyacı sayesinde gelişmiştir. Bu proje, kişisel bilgisayarların birbirleriyle ve diğer bilgisayar tipleriyle haberleşebilmelerini, kullanıcıların bu sistemler arasında dosya transferi yapabilmelerini sağlamak amacıyla düşünülmüş bir yazılımın tasarımını ve gerçekleşmesini içermekle kalmayıp, küçük bir kişisel bigisayarda çalıştırılabilecek küçük çaplı bir bilgi servisi programının geliştirilmesini de içine almaktadır. Ayrıca bu dökümanın ilk bölümünde veri iletişimi hakkında temel bilgiler bulunabilir.

Abstract

Digital Data Communications has become an important subtitle of Electronics Engineering, which is a rapidly growing context of science. Today, it is possible to transfer informations such as voice, image or any kind of documents using digital transmission methodes. In a close future, we will be able to transfer any type of information using digital transmission methodes.

Computers are most commonly known systems which have played a great role on data communications. This title of electronics engineering has been advenced by the needs of communication requirements between computer systems. The project discussed here does not only include design and realisation of a software product which lets users be able to put two computer systems in to communication and exchange files between them, but also includes development of a small sized information service software running on a personal computer. Some basic information on data communications may be also found in first chapter of this document.

Önsöz

Cok yakın geçmişe kadar veri haberleşmesi konusu, elektronik mühendisliğinin nispeten bir miktar özelleşmiş bir alanıydı. Geçtiğimiz çeyrek yüzyıl içinde bilgisayar teknolojisi meydana gelen büyük gelişmeler, günümüzde kullanılmakta olan ve birbirleriyle haberleşen birçok bilgisayardan kurulu veri işleme sistemlerini ortaya çıkartmıştır. Bu geniş ortam sadece bilgisayarları değil, bunlar için kendiliğinden işlem gücüne sahip veri giriş-çıkış terminalleri, gelişmiş işletim sistemleri, veri kontrol ve işlem birimleri gibi çevre birimlerini de içermektedir. Bu durumda günümüzde tasarlanmakta olan bilgisayar ve akıllı çevre birimlerinde veri haberleşmesinin ne büyük bir öneme sahip olduğunu açıkça görmek mümkündür. Çoğu zaman artık sadece kullanılmakta olan transimisyon birimlerini bilmek yetmemekte, bu amaçla kullanılmakta olan değişik tipe bilgisayar ağlarının gerektirdikleri hakkında da bilgi sahibi olmak gerekmektedir. Veri haberleşmesi ve bilgisayar ağları konuları, günümüz bilgisayar teknolojisinin vazgeçilmez parçaları haline gelmişlerdir.

HATA DÜZELTMELİ MODEM VERİ İLETİŞİM ve BİLGİ ERİŞİM PROGRAMI

Birçok bilgisayar ağı uygulamasında, iki bilgisayarın haberleşebilmesini sağlamak problemin sadece küçük bir bölümünü çözmektedir. Farklı tipte bilgisayarlardan oluşan ve bu bilgisayarlardan birbirlerine bilgi dosyaları transfer ettikleri bir bilgisayar ağında, değişik işletim sistemleri ve dosya düzenleri kullanan bilgisayarlardan özel önlemler alınmaksızın, AÇIK bir şekilde dosya transferi yapabilmelerini sağlamak gereklidir. Bu konu, Uluslararası Standartlar Organizasyonunun (International Standards Organisation - ISO) Açık Sistemler Bağlasını (Open Systems Interconnection - OSI) referans modeline

Önsöz

Çok yakın geçmişe kadar veri haberleşmesi konusu, elektronik mühendisliğinin nispeten bir miktar özelleşmiş bir alanıydı. Geçtiğimiz çeyrek yüzyıl içinde bilgisayar teknolojisinde meydana gelen büyük gelişmeler, günümüzde kullanılmakta olan ve birbiriyle haberleşen birçok bilgisayardan kurulu veri işleme sistemlerini ortaya çıkartmıştır. Bu geniş ortam sadece bilgisayarları değil, bunlar için kendiliğinden işlem gücüne sahip veri giriş-çıkış terminalleri, gelişmiş iş istasyonları, akıllı endüstriyel işlem birimleri gibi çevre birimlerini de içermektedir. Bu durum, günümüzde tasarlanmakta olan bilgisayar ve akıllı çevre birimlerinde veri haberleşmesi konusunun ne büyük bir öneme sahip olduğunu açıkça ortaya koymaktadır. Çoğu zaman artık sadece kullanılmakta olan transmisyon devrelerini bilmek yetmemekte, bu amaçla kullanılmakta olan değişik tipte bilgisayar ağlarının gerektirdikleri hakkında da bilgi sahibi olmak gerekmektedir. Buyüzden, veri haberleşmesi ve bilgisayar ağları konuları, günümüz bilgisayar tasarımı derslerinin vazgeçilmez parçaları haline gelmişlerdir.

Birçok bilgisayar ağı uygulamasında, iki bilgisayarın haberleşebilmesini sağlamak problemin sadece küçük bir bölümünü çözmektedir. Farklı tipte bilgisayarlardan oluşan ve bu bilgisayarların birbirlerine bilgi dosyaları transfer ettikleri bir bilgisayar ağında, değişik işletim sistemleri ve dosya düzenleri kullanan bilgisayarların özel önlemler almaksızın, AÇIK bir şekilde dosya transferi yapabilmelerini sağlamak gereklidir. Bu konu, Uluslararası Standartlar Organizasyonunun (International Standards Organisation - ISO) Açık Sistemler Bağlaşımı (Open Systems Interconnection - OSI) referans modeline

ilişik uygulamaya yönelik protokoller aracılığıyla gerçekleştirilmektedir. Bilgisayar ağlarının uygulaması söz konusu olduğunda, bu protokolleri de anlamak gereklidir.

Veri haberleşmesi konusu geliştikçe ortaya çıkmış önemli bir sonuç da, hata sezme ve düzeltme (Error Checking and Correction) konusudur. Özellikle ses hatları üzerinden modülasyon yöntemleri ile veri iletişimi kurmakta kullanılan MODEM cihazları gelişip yaygınlaştıktan sonra, bunların sağlıklı ve hızlı çalışabilmeleri için transmisyon ortamının band genişliği ve gürültü düzeyi çok önemli iki parametre haline gelmiş ve bu iki parametre de, çok az düzeltilebilmiştir. Bu düzeltme de genellikle veri iletişimi için kiralanan özel hatlar üzerinde olmuştur. Ses hatları üzerinden veri iletişimi kuran insanlar arasında, hattın gürültü düzeyinden gelen veri hataları o denli problemlere sebep olmuştur ki, mühendisler, hata düzeylerini en yüksek gürültülü hatlarda dahi en düşükte tutmak için yeni hata düzeltmesi algoritmaları geliştirmişlerdir.

Veri iletişimi dalındaki önemli bir gelişme de veri giriş çıkış terminallerinde gözlenmiştir. Her terminal yapımcısı firma, kendi terminaline gün geçtikçe yeni özellikler katarak teminalinin daha akıllı olmasını, ana bilgisayarla haberleşme ihtiyacının en alt düzeyde tutulmasını, daha güzel görüntüler elde edebilmek için renkli veya özellikli karakterler gösterebilmesini ve mümkün olduğu kadar çok miktarda tek basışta işe yarayabilecek ve değişik anlamlarda kullanılabilecek tuşların bulunmasını sağlamışlardır. Bu gelişmenin en kötü yanı, bu konuda standartlara pek fazla uyulmamış olmasıdır (Örneğin ANSI gibi standartlar). Bu yüzden zamanla veri giriş çıkış terminalleri birbirlerini taklid edebilme özelliklerine sahip olmuşlardır. Böylece terminali satın alan kişi onu

hangi terminal tipinde kullanacağını kendisi seçebilir olmuştur. Buna ek olarak , kişisel bilgisayar kullanan kişiler de, herhangi bir terminale gereksinim duymaksızın ana bilgisayarlara bağlanabilmek istemişler ve böylece PC ler üzerinde çalışan terminal emülasyon programları ortaya çıkmıştır. Bu programlar zamanla gelişerek sadece terminal emülasyonu yapmakla kalmamışlar, aynı zamanla PC ler arası dosya transferi işlerini de yerine getirir olmuşlardır. Bu tez çalışmasında gerçekleştirilen hata düzeltmeli XMODEM dosya transfer protokolü de böyle ortaya çıkmıştır.

Bu tez çalışması, PC tipi bilgisayarlar üzerinde ve MSDOS işletim sistemi altında çalışmak üzere düşünülmüş hata düzeltmeli veri iletişim ve bilgi erişim programlarının tasarımı ve gerçekleştirilmesini içermektedir. Sonuçta ortaya çıkan veri iletişim programı, terminal emülasyonu ve XMODEM hata düzeltmeli dosya transfer protokolünü desteklemektedir. Bilgi erişim programı ise, küçük bir kişisel veri tabanı olarak çalışmakta ve sistem yöneticisinin belirlediği kişilerin bu bilgi sisteminden kontrollü olarak faydalanabilmesini sağlamaktadır. Bu program da, bu proje için özel olarak geliştirilmiş olan haberleşme alt programlarını kullanmakta ve böylece, uzaktan sisteme ulaşan kişiye XMODEM hata düzeltmeli dosya transfer protokolü ile dosya transferi imkanlarını sağlamaktadır.

Ümit Kalkan

Elektronik ve Haberleşme
Müh.

1.1 Dağıtılmış Sistem Mimarileri

1.1.1 Giriş

Dağıtılmış bilgi işlem sistemleri, fiziksel olarak birbirinden ayrık ve bilgisayar tabanlı sistem grupları arasında bilginin işlenmesi ve transfer edilmesi ile ortaya çıkmaktadır. Genel olarak, bu sistemlerin birçok tipine, Veri Terminali Ekipmanı (Data Terminal Equipment) ya da kısaca DTE denir. Bu sistemler, sadece dağıtılmış bilgisayar sistemlerini değil, akıllı görüntüleme terminalleri, bilgisayar tabanlı iş istasyonları ve bununla ilişkili diğer çevre birimleri, büyük satış yerlerinde kullanılan satış noktası terminalleri (Point Of Sales Terminals), uzaktan takip amacıyla geliştirilmiş mikroişlemci tabanlı akıllı elektriksel ölçüm aletleri gibi birçok yan birimi de içerirler. Bu kadar değişik türde cihazın varlığı, bir çok değişik türde de dağıtılmış bilgisayar sistemleri mimarilerinin bulunduğunu göstermektedir. Örneğin bir sistem, coğrafi olarak geniş bir alana yayılmış birçok görüntüleme terminalinden ve bunların bağlı olduğu tek bir merkezi bilgisayar kompleksinden oluşabilir. Diğer bir sistem de, birkaç ofislik bir bloğa dağıtılmış iş istasyonlarından ve bunların ortak kullandıkları yazıcılar, kopya makineleri ve dosya sistemleri gibi yan birimlerden oluşabilir.

Dağıtılmış sistemlerde yerine getirilen işlevsel fonksiyonların uygulamadan uygulamaya büyük bir çeşitlilik göstermesine rağmen, ihtiyaç duyulan veri iletişimi hizmetlerini karşılayabilecek sınırlı sayıda veri ağı tipi vardır. Ayrıca, iletişim yazılımlarının yapısı ve işlevlerine getirilen uluslararası standartlar da, sistemin uygulamasından bağımsız bir derece ortaklık sağlamıştır. Buyüzden, veri iletişimi sistemleri tasarlanırken şu hususlara dikkat edilmelidir:

(1) Şu anda bulunan çeşitli veri iletişim ağlarının çalışma prensiplerini ve çeşitli alanlardaki tekabül eden çalışma modlarını bilmek gerekir.

(2) Bu ağların kullanılmasına yardımcı olmak amacıyla konmuş uluslararası standartları anlamak gereklidir.

1.1.2 İletişim Ağı Sınıflaması

Dağıtılmış sistemlerin incelemesini derinleştirmeden önce, çeşitli veri iletişim ağlarının bir sınıflandırmasını yapmak yerinde olacaktır. Veri iletişim ağları, haberleşen sistemlerin birbirlerine olan fiziksel uzaklıklarına göre dört sınıfta toplanırlar:

(1) **Minyatür (<5 cm):** Bu ağlar, aynı tümleşik devre üzerinde gerçekleştirilmiş hesaplama ve işleme unitelerinin birbirlerine bağlanmasından oluşurlar.

(2) **Küçük (<50 cm):** Bu tür ağlar ise, aynı rafa ya da kutuya monte edilmiş birden fazla hesaplama ve-veya işleme ünitesinin birbirine bağlanmasıyla oluşurlar.

(3) **Orta (<1 km):** Bu tür ağlar da, üniversite kampüsü gibi yerel alanlara yayılmış çeşitli bilgisayar ve terminal gibi çevre birimlerinin birbirine bağlanmasıyla oluşurlar. Bu yüzden de genellikle Yerel Alan İletişim Ağı (Local Area Network - LAN) olarak bilinirler.

(4) Geniş (>10 km): Bu büyüklükteki iletişim ağları ise, ülke veya dünya çapına yayılmış işlem birimlerinin birbirlerine bağlanmasıyla oluşur ki bunlar da genel olarak Geniş Alan İletişim Ağı (Wide Area Network - WAN) olarak isimlendirilirler.

İlk iki tür iletişim ağında, işlem veya hesaplama birimleri arasındaki mesafe çok küçük olduğundan, birimler arasındaki mesajlar ya da bilgiler, her bir bit için ayrı bir iletkenin kullanılmasıyla oluşan iletken setlerinden kurulu paralel taşıyıcılarla iletilir ve bu sistemler SIKI BAĞLI (closely coupled) sistemler adını alırlar. buna karşın, bundan sonraki iki iletişim ağı türünde, birimler arasındaki mesafenin büyüklüğünden dolayı bütün haberleşme, bir çift iletken üzerinden seri modda gerçekleştirilir. Bu yüzden bu tür iletişim ağlarına da Gevşek Bağlı Sistemler (loosely coupled systems) adı verilir.

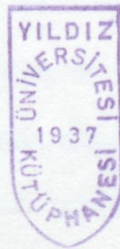
Genellikle sıkı bağlı sistemler, aynı yapıdaki işlem birimlerinin arasında veri alışverişi için kullanılırlar. Bu sistemlerin ana amacı her veri transferi için harcanan zamanı en aza indirmektir. Normal olarak bu tür bağlar, bütün işlem birimleri tarafından erişilebilen bir ortak bellek (shared memory) ile gerçekleştirilir. Bu yolla, birimler arasında mesajların tamamının transfer edilmesi yerine, sadece mesajları işaret eden göstermeçler (pointers) transfer edilir.

Bunun tam aksine gevşek bağlı sistemler, farklı farklı özelliklerdeki bilgisayar sistemleri veya çevre birimlerinin birbirlerine bağlanması ve veri iletmeleri ile ortaya çıkarlar. Bu yapıdaki sistemlerin birbirlerine olan büyük uzaklıklardan dolayı bu sistemlerde mesaj veya verilerin kendileri transfer edilmek zorundadır. Bu yüzden veri transferi esnasında olabilecek transmisyon

1.1 hataları ve bit kayıplarının olasılığı sıkı bağılı sistemlere oranla çok yüksektir. Gevşek bağılı sistemlerdeki başlıca amaçlar şunlardır:

- (1) Güvenilir veri transferi sağlamak. Bu, mesajları taşıyan verinin herhangi bit hataları bulundurmamasını ve gönderildiği bit sırasıyla alınabilmesini sağlamak demektir.
- (2) Transfer edilen mesajların bütün sistemlerde aynı anlama gelmesini sağlamak; yani, bu mesajların bütün sistemlerde aynı şekilde işlenmesini sağlamak.

Bu amaçlara ulaşmak için, gevşek bağılı sistemlerde, uluslararası kabul edilmiş standartların yanında birçok değişik tipte veri iletişim ağı tipi de kullanılmaktadır.

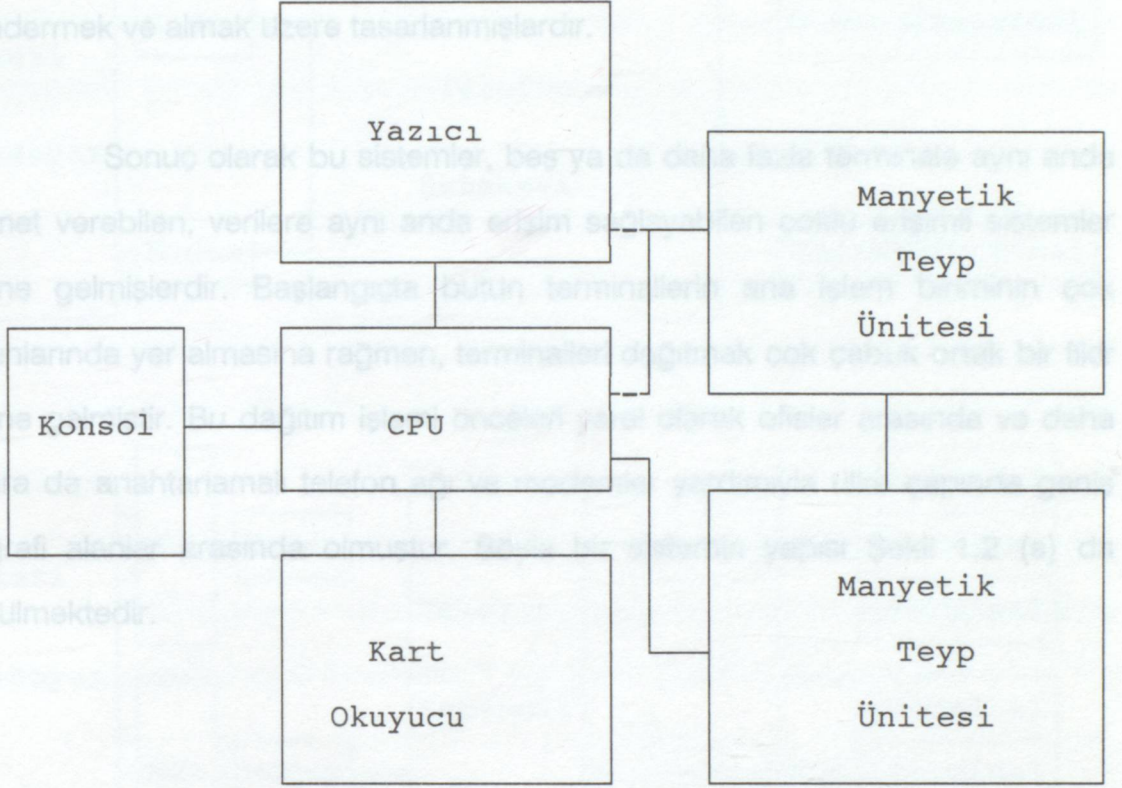


1.1.3 Tarihi Gelişme

Gevşek bağlı sistemlerin tarihsel gelişimini incelemek için herhalde en iyi yol, büyük organizasyonlarda kullanılan işlem birimlerinin gelişmesini gözden geçirmek olacaktır. En eski bilgisayar sistemlerinin karakteristik özelliği, pahalı donanımlara ve çok basit yazılım özelliklerine sahip olmalarıydı. Genellikle firmalar, bu hantal cihazları alırlar ve merkezi bir havalandırmalı odaya yerleştirirlerdi. Bu cihazlar, bir merkezi işlem birimi (Central Processing Unit - CPU), çok az miktarda rastgele erişimli bellek (Random Access Memory - RAM), kısıtlı sayıda manyetik teyp ünitesi, bir yazıcı, bir delikli kart okuyucu ve bir de operator konsolundan oluşurdu. Kullanıcılar, programlarını ve bunlara ilişkin verilerini hiçbir izleme olanağı olmaksızın delikli kartlara kaydederler ve daha sonra operator, delikli kart okuyucu yardımıyla bu programları sırayla çalıştırırdı. Böyle bir sistemin temel yapısı Şekil 1.1 de görülmektedir.

Şekil 1.1: Eski Bir Bilgisayar Sisteminin Yapısı

Bilgisayar teknolojisinin ve işletim sistemleri yazılımlarının gelişmesiyle, hızlı ikinci bellekler (manyetik diskler gibi) ve çoklu programlamak ve kullanımı işletim sistemleri doğmuştur. Böylelikle, merkezi işlem biriminin işlem gücünün, birden fazla aktif kullanıcı veya proses tarafından paylaşılabilmesi sağlanmıştır. Bundan sonra bilgisayar kullanıcıları, kendi programlarını, diğer insanları beklemeden aynı anda ve kendi terminallerinde sonuçlarını görerek ve değişikliklerini yaparak çalıştırmak imkânını bulmuşlardır. O zamanlar kullanılan terminaller, teleks ağlarında kullanılan elektromekanik terminallerin aynısı (TTY)

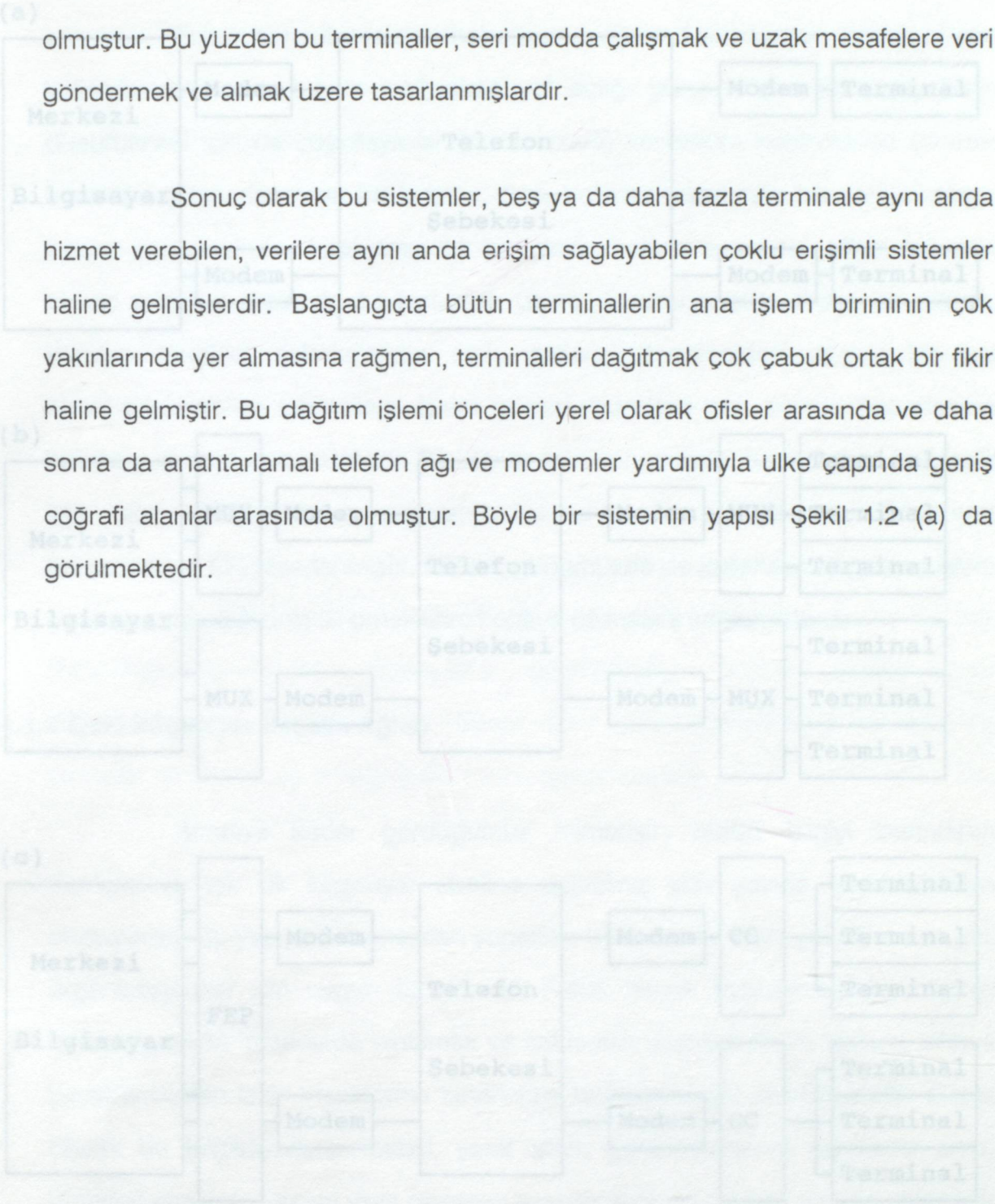


Şekil 1.1: Eski Bir Bilgisayar Sisteminin Yapısı

Bilgisayar teknolojisinin ve işletim sistemleri yazılımlarının gelişmesiyle, hızlı ikincil bellekler (manyetik diskler gibi) ve çoklu programlamalı ve kullanımlı işletim sistemleri doğmuştur. Böylelikle, merkezi işlem biriminin işlem gücünün, birden fazla aktif kullanıcı veya proses tarafından paylaşılabilmesi sağlanmıştır. Bundan sonra bilgisayar kullanıcıları, kendi programlarını, diğer insanları beklemeden aynı anda ve kendi terminallerinde sonuçlarını görerek ve değişikliklerini yaparak çalıştırmak imkanını bulmuşlardır. O zamanlar kullanılan terminaller, teleks ağlarında kullanılan elektromekanik terminallerin aynıları (TTY)

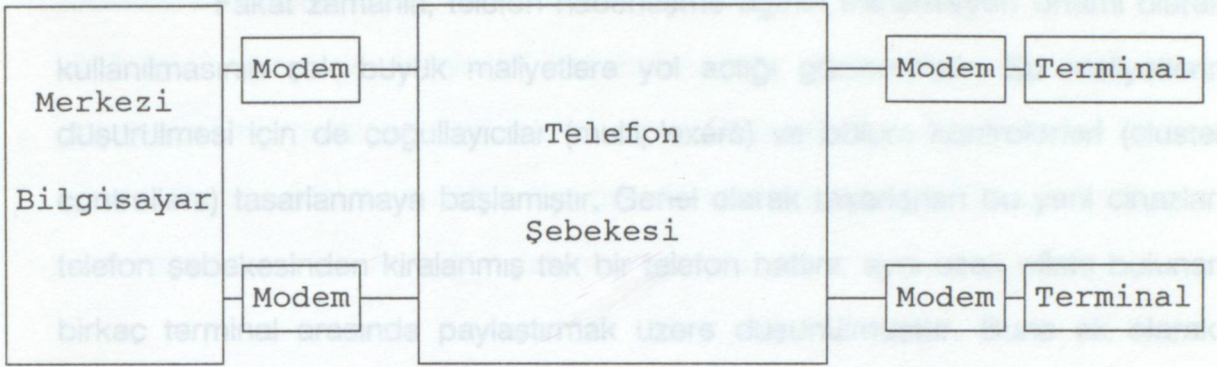
(a) olmuştur. Bu yüzden bu terminaller, seri modda çalışmak ve uzak mesafelere veri göndermek ve almak üzere tasarlanmışlardır.

Sonuç olarak bu sistemler, beş ya da daha fazla terminale aynı anda hizmet verebilen, verilere aynı anda erişim sağlayabilen çoklu erişimli sistemler haline gelmişlerdir. Başlangıçta bütün terminallerin ana işlem biriminin çok yakınlarında yer almasına rağmen, terminalleri dağıtmak çok çabuk ortak bir fikir haline gelmiştir. Bu dağıtım işlemi önceleri yerel olarak ofisler arasında ve daha sonra da anahtarlama telefon ağı ve modemler yardımıyla ülke çapında geniş coğrafi alanlar arasında olmuştur. Böyle bir sistemin yapısı Şekil 1.2 (a) da görülmektedir.

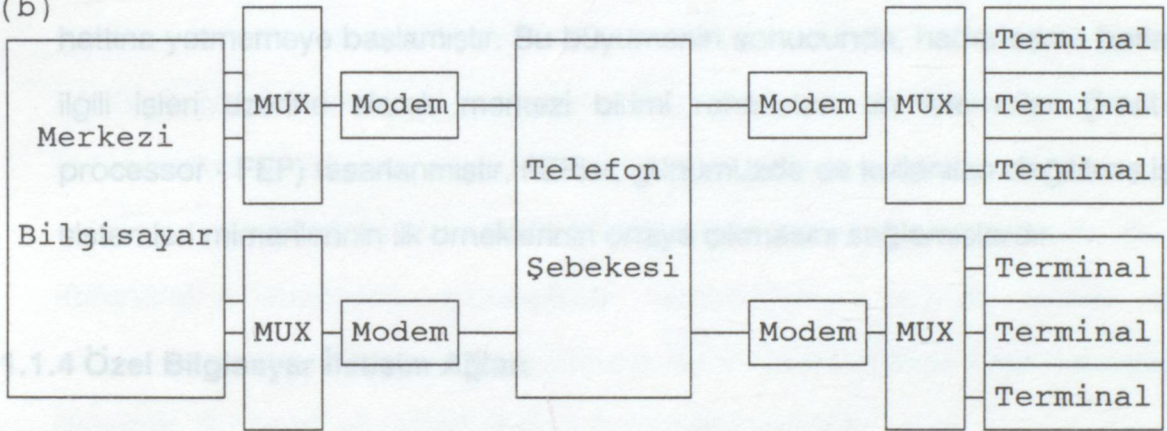


Şekil 1.2: Terminal Tabanlı Veri Ağı Örnekleri

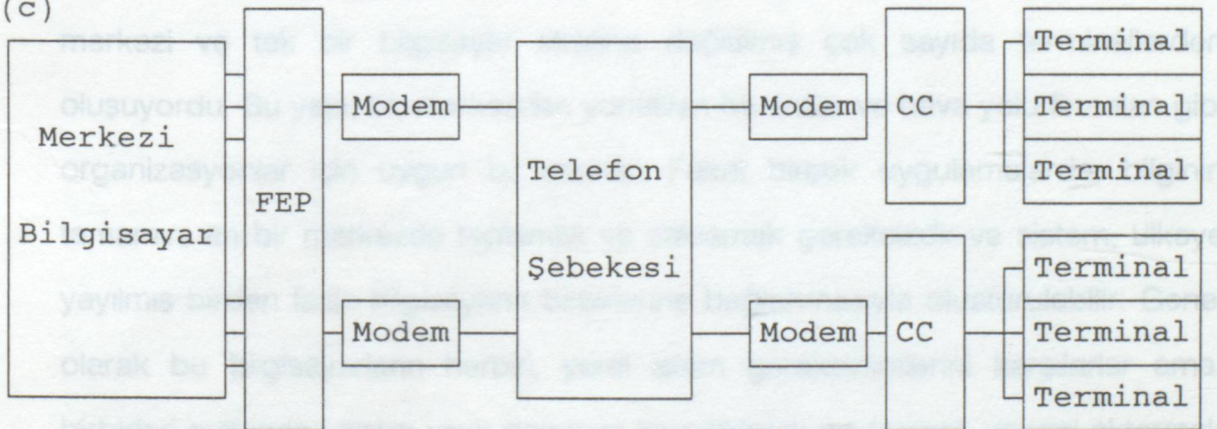
(a)



(b)



(c)



Şekil 1.2: Teminal Tabanlı Veri Ağı Örnekleri

Fakat zamanla, telefon haberleşme ağının transmisyon ortamı olarak kullanılmasının çok büyük maliyetlere yol açtığı gözlenmiştir. Bu maliyetlerin düşürülmesi için de çoğullayıcılar (multiplexers) ve bölüm kontrolörleri (cluster controllers) tasarlanmaya başlamıştır. Genel olarak tasarlanan bu yeni cihazlar, telefon şebekesinden kiralanan tek bir telefon hattını, aynı uzak ofiste bulunan birkaç terminal arasında paylaşdırmak üzere düşünülmüştür. Buna ek olarak, kurulan terminal şebekelerinin artık yüzlerce terminalden oluşur bir hal almasından dolayı işlemcilerin işlem güçleri, bu denli çok sayıda haberleşme hattına yetmemeye başlamıştır. Bu büyümenin sonucunda, haberleşme hatlarıyla ilgili işleri üzerine alarak merkezi birimi rahatlatan ön işlemciler (front-end processor - FEP) tasarlanmıştır. FEPlar, günümüzde de kullanılan dağıtılmış işlem sistemleri mimarilerinin ilk örneklerinin ortaya çıkmasını sağlamışlardır. da, bu ağı kullanarak haberleşen bilgisayarlar birbirlerinden coğrafi olarak uzak

1.1.4 Özel Bilgisayar İletişim Ağları (Geniş Alanlı Şebeke (Wide Area Network) de denebilir. Bu tür bir ağı örneği şekil 1.3 de görülmektedir.

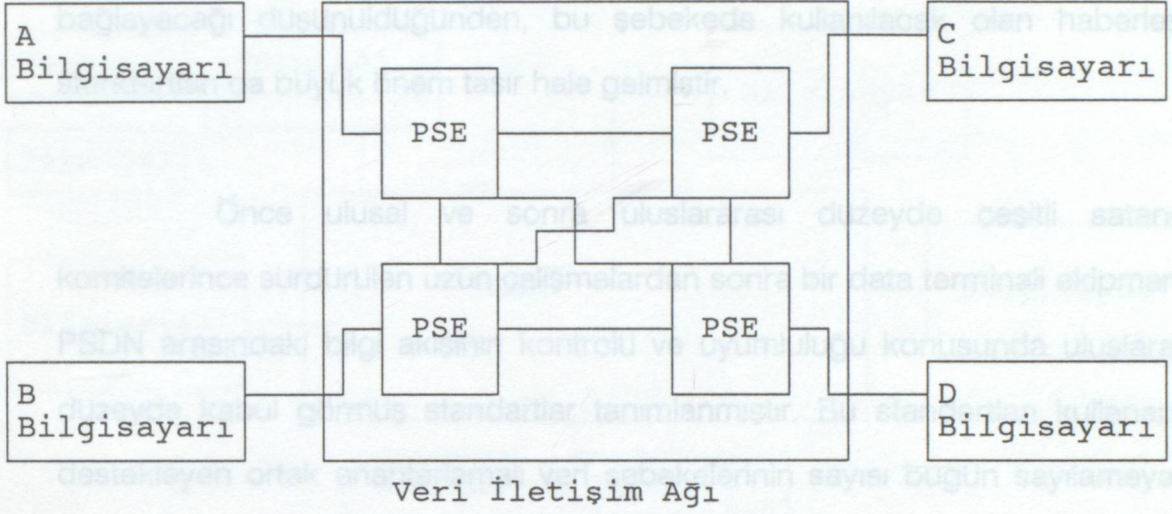
Şimdiye kadar gördüğümüz mimariler, bütün bilgiyi barındıran merkezi ve tek bir bilgisayar etrafına dağıtılmış çok sayıda terminallerden oluşuyordu. Bu yapı, bir merkezden yönetilen bankalar ve hava yolu firmaları gibi organizasyonlar için uygun bir yapıdır. Fakat birçok uygulamalarda, bilginin tamamını an bir merkezde toplamak ve saklamak gereksizdir ve sistem, ülkeye yayılmış birden fazla bilgisayarın birbirlerine bağlanmasıyla oluşturulabilir. Genel olarak bu bilgisayarların herbiri, yerel işlem gereksinimlerini karşılarlar ama, birbirleri arasında yazılım veya donanım kaynaklarını paylaşmak ve veri aktarmak için haberleşme imkanlarına ihtiyaçları vardır. Böyle bir bağlantıyı gerçekleştirmek için, daha esnek ve verimli anahtarlanmış iletişim araçlarına gereksinim doğmuştur. Bu durumda, telefon şebekesinden tamamen ayrı bir veri iletişim

şebekesi kurulmasının daha anlamlı ve kullanışlı olacağına karar verilmiştir. Böyle bir iletişim ağının gereksinimleri hali hazırda var olan ve mesaj saklama ve gönderme modunda çalışan teleks şebekesinin özellikleriyle hemen hemen aynıdır. Mesaj saklama ve dah sonra gönderme modundaki çalışma, bilgisayarlar arası haberleşme için en uygun çalışma modudur.

Teleks şebekesinde, kullanılan birim mesaj uzunluğu büyük olabildiğinden, gönderilen ve buna cevap olarak alınan mesajlar arasındaki gecikme zamanı çok büyük olabilir. Bu gecikmeyi en düşük düzeyde tutabilmek için bilgisayar iletişim ağında paket olarak adlandırılan daha küçük mesaj birimleri kullanılmaktadır. Bu yüzden bu iletişim ağının paket anahtarlama veya paket saklama ve gönderme modunda çalıştığı söylenebilir. Bunun yanında, bu ağı kullanarak haberleşen bilgisayarlar birbirlerinden coğrafi olarak uzak olduklarından, bu şebekeye Uzak (Geniş) Alanlı Şebeke (Wide Area Network) de denebilir. Bu tür bir ağ örneği şekil 1.3 de görülmektedir.

1.1.5 Ortak Paket Anahtarlama Veri Şebekesi

Öncelen firmalar, ortak haberleşme kurumlarından kiraladıkları özel hatları kullanarak kendi özel veri iletişim ağlarını oluşturmışlardır. Bunun hemen ardından bilgisayar üreticisi firmalar, bilgisayarların bu ağlar üzerinden haberleşebilmesini ve veri transferi yapabilmesini sağlayan yazılımlar geliştirmişlerdir. Fakat zamanla, teknolojik ilerlemelerin de getirdiği yeniliklerle, bir firmanın bilgisayarının diğer bir firmanın bilgisayarıyla haberleşmesi ihtiyacı doğmuştur (Örneğin fon bilgilerinin bir banka bilgisayarından diğer bir banka bilgisayarına transfer edilebilmesi gerekmektedir). Bu durumda, ortak haberleşme kuruluşları, ortak anahtarlama telefon şebekesine (PSTN - Public Switched Telephone Network) benzer bir ortak anahtarlama veri şebekesi



PSE: Packet Switching Exchange

Şekil 1.3: Özel Bilgisayar İletişim Ağı

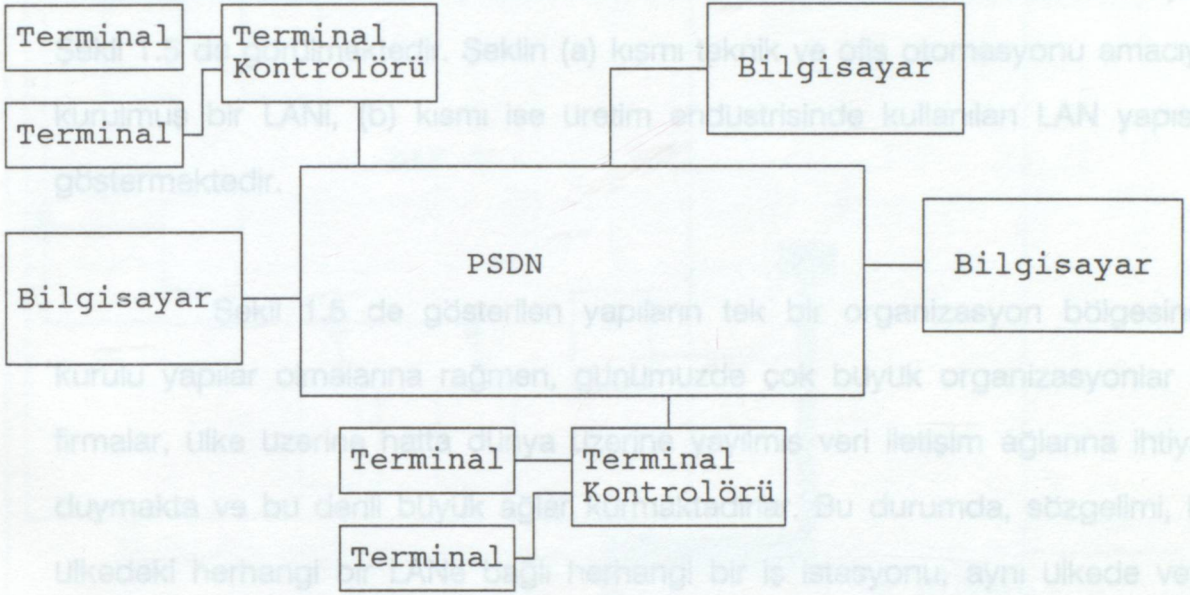
1.1.5 Ortak Paket Anahtarlama Veri Şebekesi

Önceleri firmalar, ortak haberleşme kurumlarından kiraladıkları özel hatları kullanarak kendi özel veri iletişim ağlarını oluşturmuşlardır. Bunun hemen ardından bilgisayar üreticisi firmalar, bilgisayarlarının bu ağlar üzerinden haberleşebilmesini ve veri transferi yapabilmesini sağlayan yazılımlar geliştirmişlerdir. Fakat zamanla, teknolojik ilerlemelerin de getirdiği yeniliklerle, bir firmanın bilgisayarının diğer bir firmanın bilgisayarıyla haberleşmesi ihtiyaçları doğmuştur (Örneğin fon bilgilerinin bir banka bilgisayarından diğer bir banka bilgisayarına transfer edilebilmesi gerekmektedir). Bu durumda, ortak haberleşme kuruluşları, ortak anahtarlama telefon şebekesine (PSTN - Public Switched Telephone Network) benzer bir ortak anahtarlama veri şebekesi

(PSDN - Public Switched Data Network) kurulması fikrinde birleşmişlerdir. Bu şebekenin çok büyük bir alana yayılmış çok çeşitli bilgisayarları birbirine bağlayacağı düşünüldüğünden, bu şebekede kullanılacak olan haberleşme standartları da büyük önem taşır hale gelmiştir.

Önce ulusal ve sonra uluslararası düzeyde çeşitli standart komitelerince sürdürülen uzun çalışmalardan sonra bir data terminali ekipmanı ile PSDN arasındaki bilgi akışının kontrolü ve uyumluluğu konusunda uluslararası düzeyde kabul görmüş standartlar tanımlanmıştır. Bu standartları kullanan ve destekleyen ortak anahtarlamalı veri şebekelerinin sayısı bugün sayılamayacak kadar çoktur. Böyle bir dağıtılmış sistem mimarisinin bir örnek çizimi şekil 1.4 de görülmektedir.

Çok yakın zamanlarda, tümleşik devre teknolojisindeki gelişmeler, birçok ülkenin ortak haberleşme kuruluşlarını (PTT gibi), mevcut telefon şebekesini tamamen sayısal modda çalışır hale getirme fikrine yönlendirmiştir. Bu, şu andaki telefon şebekesiyle veri iletişimi yapabilmek için modemlere ihtiyacımız olmasına rağmen, sayısal uygulamaya geçildiğinde verilerimizi direkt olarak ve şimdikinden çok daha yüksek hızlarda transfer edebileceğimiz anlamına gelmektedir. Ayrıca bu yeni çalışma sisteminde çağrı-bağlantı süreleri de öncesine göre büyük oranda kısalmaktadır. tamamen sayısal modda çalışmaları ve hem veri hem de ses iletişimi için kullanılabileceklerinden dolayı bu yeni tür şebekelere Tümleşik Hizmet Sayısal Şebekeler (Integrated Services Digital Network - ISDN) adı verilmektedir.



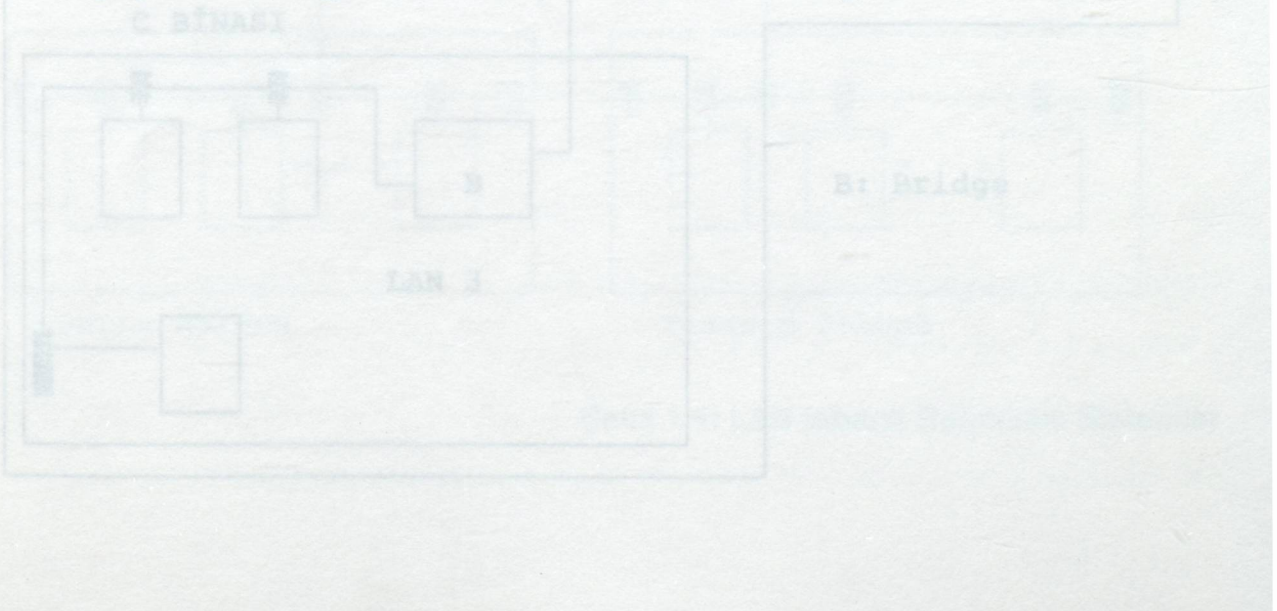
Şekil 1.4: PSDN tabanlı dağıtılmış sistem

1.1.6 Yerel Veri İletişim Ağları

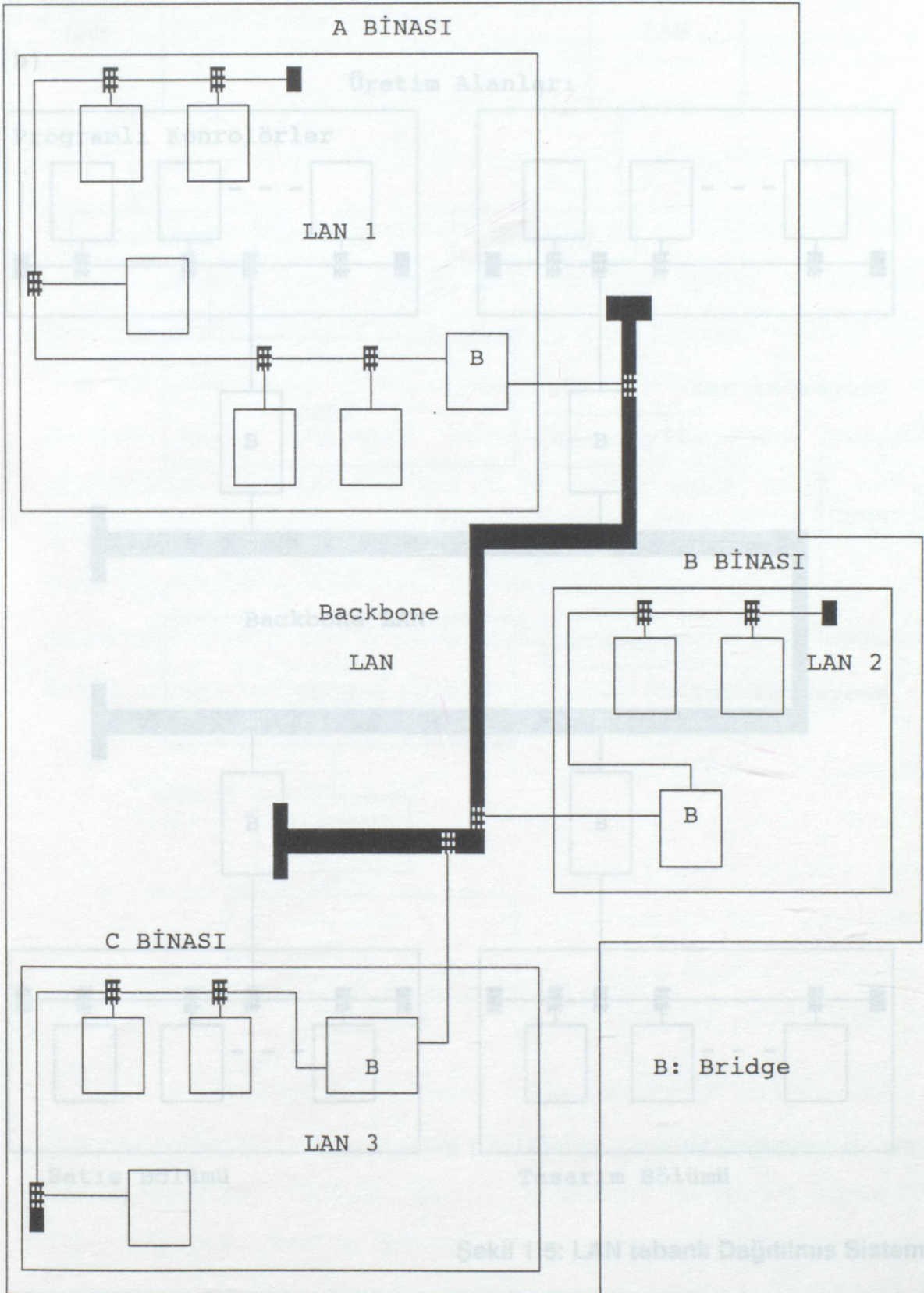
Zamanla, işletmelerin tek bir merkez bilgisayar ile sorunlarını çözememeleri üzerine, aynı binada birden fazla ve değişik türde bilgisayarlar bulundurulmaya başlamıştır. Çoğu zaman bu bireysel bilgisayarların sağladığı işlem gücünün, bireysel işlemler için yeterli kalmasına rağmen, zamanla bu yakın bilgisayarların da, elektronik posta alışverişi gibi nedenlerle birbirlerine bağlanmaları gerekmiştir. Böyle bir bağ ile bağlanmış bilgisayar sistemleri birbirlerine fiziksel olarak çok yakın olduklarından, bunları bağlayan ağa da Yerel Veri İletişim Ağı (Local Area Network - LAN) denmiştir. Çok değişik uygulamalarda kullanılan LAN lerin, çalışma tarzları ve yapıları açısından çeşitli

olmaları da beklenen bir sonuçtur. Örnek olarak teknik amaçlarla veya ofis otomasyonu amacıyla kurulmuş LANler olduğu gibi, üretim endüstrisinde kullanılmakta olan LANler de bulunmaktadır. LAN için verilebilecek iki örnek yapı Şekil 1.5 de görülmektedir. Şeklin (a) kısmı teknik ve ofis otomasyonu amacıyla kurulmuş bir LANi, (b) kısmı ise üretim endüstrisinde kullanılan LAN yapısını göstermektedir.

Şekil 1.5 de gösterilen yapıların tek bir organizasyon bölgesinde kurulu yapılar olmalarına rağmen, günümüzde çok büyük organizasyonlar ve firmalar, ülke üzerine hatta dünya üzerine yayılmış veri iletişim ağlarına ihtiyaç duymakta ve bu denli büyük ağlar kurmaktadır. Bu durumda, sözgelimi, bir ülkedeki herhangi bir LANe bağlı herhangi bir iş istasyonu, aynı ülkede veya değişik bir ülkede kurulu herhangi bir LANe bağlı herhangi bir iş istasyonu ile haberleşebilmektedir. Burada, LANler arasındaki bağ, bir ya da daha fazla PSDN bağı olarak düşünülmekte, hatta bu bağların arasında bir ya da daha çok uydu bağlantısı da yer alabilmektedir. Böyle bir yapı Şekil 1.6 da görülmektedir. Değişik iletişim ağları arasında bağ görevi yapan araçlardan iki LANi bağlayanlara Ağlararası Köprü (Internetwork Bridge), bir LANi bir PSDNe bağlayanlara ise Ağlararası Geçit (Internetwork Gateway) denmektedir.



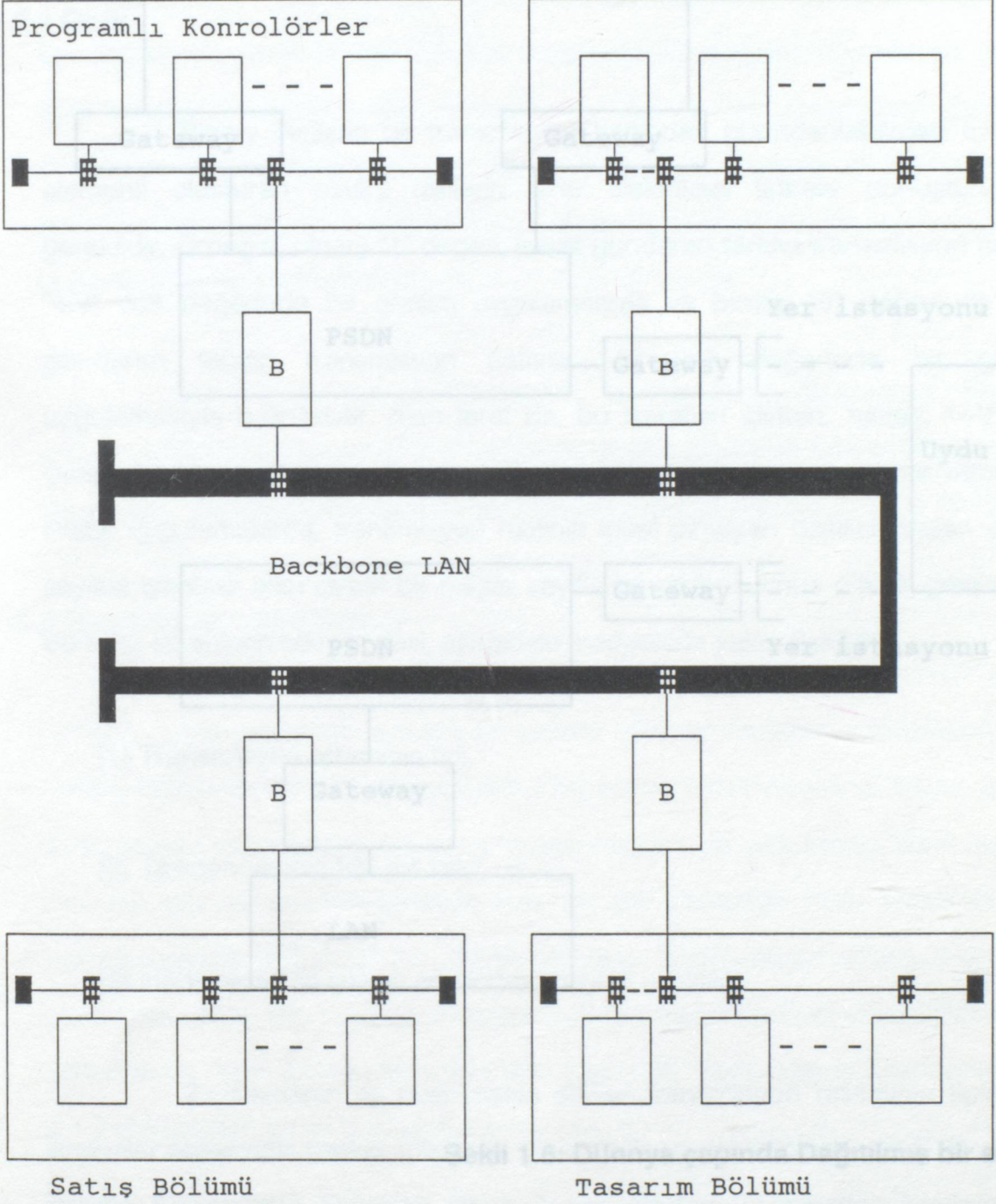
(a)



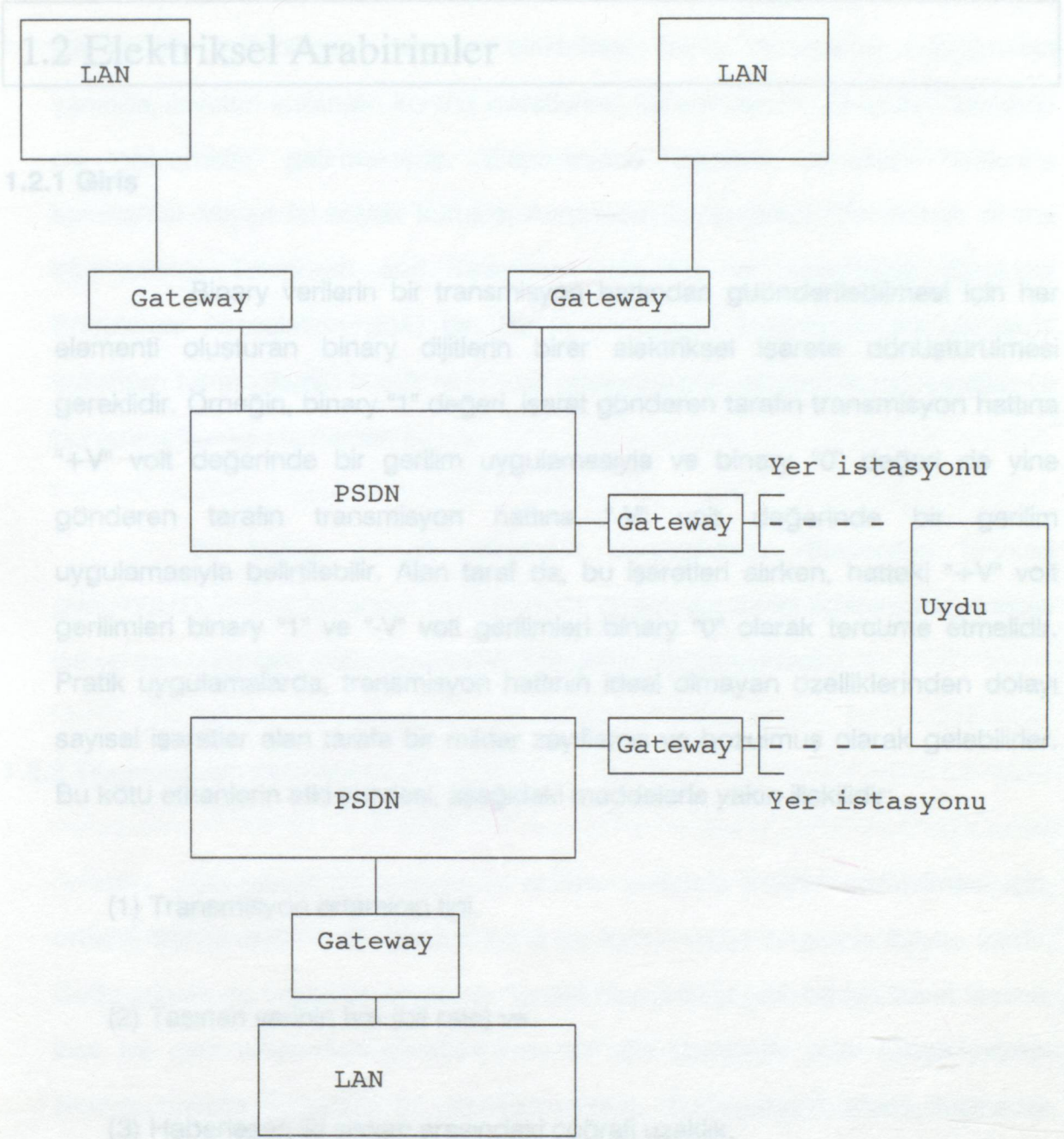
Şekil 1.6: LAN tabanlı Dağıtım Sistemleri

(b)

Üretim Alanları



Şekil 1.5: LAN tabanlı Dağıtılmış Sistemler



Şekil 1.6: Dünyada Dağıtılmış bir sistem

1.2 Elektriksel Arabirimler

1.2.1 Giriş

Binary verilerin bir transmisyon hattından gönderilebilmesi için her elementi oluşturan binary dijitlerin birer elektriksel işarete dönüştürülmesi gereklidir. Örneğin, binary "1" değeri, işaret gönderen tarafın transmisyon hattına "+V" volt değerinde bir gerilim uygulamasıyla ve binary "0" değeri de yine gönderen tarafın transmisyon hattına "-V" volt değerinde bir gerilim uygulamasıyla belirtilebilir. Alan taraf da, bu işaretleri alırken, hattaki "+V" volt gerilimleri binary "1" ve "-V" volt gerilimleri binary "0" olarak tercüme etmelidir. Pratik uygulamalarda, transmisyon hattının ideal olmayan özelliklerinden dolayı sayısal işaretler alan tarafa bir miktar zayıflamış ve bozulmuş olarak gelebilirler.

Bu kötü etkenlerin etki yüzdesi, aşağıdaki maddelerle yakın ilişkilidir:

(1) Transmisyon ortamının tipi.

(2) Taşınan verinin hızı (bit rate) ve

(3) Haberleşen iki sistem arasındaki coğrafi uzaklık.

Zayıflamanın ve bozulmanın etkileri transmisyon ortamının tipine ve sistemler arasındaki uzaklık miktarına göre sınıflandırılabilirdiği için, bir DTE ve Veri İletişimi Sonandırma Ekipmanı (Data Communication Terminating Equipment - DCE) arasındaki fiziksel ve elektriksel bağlantı elemanlarının özellikleri üzerine konmuş ve uluslararası düzeyde kabul görmüş standartlar bulunmaktadır. Bu

1.2 standartlar, kullanılması gereken elektriksel işaret düzeylerini belirlemenin yanında, ilaveten kullanılan kontrol işaretlerinin kullanımları ve anlamları hakkında da açıklamalar getirmektedir. Ekipmanların bağlantı parçaları hakkında standartlar koyan iki büyük kuruluş, Avrupada Consultative Committee of the International Telegraph and Telephone (CCITT) ve Amerikada Electrical Industries Association (EIA) dır. Bu iki kuruluşun tanımladığı standartlarda kullanılan terminolojinin küçük farklılıklar göstermesine rağmen temel işaretler ve bunların anlamları tamamen aynıdır.

İletkenler arasında elektriksel işaret bu iletkenlerden birine, referans gerilimi veya akımı noktası da iletkenlerden diğerine uygulanır.

Bu bölüm iki alt bölümden oluşmaktadır. Bunlardan birincisi günümüzde en çok kullanılan transmisyon ortamı çeşitlerini açıklamakta, ikincisi ise en çok kullanılan elektriksel işaret biçimlerine değinmektedir.

Transmisyon ortamına (örneğin MODEM) bağlamakta kullanılırlar. Daha ilerde görüleceği gibi, bu tür DTE-DCE

1.2.2 Transmisyon Ortamları

İletkenler arasında elektriksel işaret bu iletkenlerden birine, referans gerilimi veya akımı noktası da iletkenlerden diğerine uygulanır.

Elektriksel bir işaretin iki sistem arasında tranfer edilebilmesi için, ortada, transmisyon hattı biçimini almış bir transmisyon ortamına ihtiyaç vardır. Çoğu zaman bu ortam bir çift iletken telden oluşabildiği gibi, bazan işaret taşıma, ince bir cam kılcalından geçirilen ince bir ışık demetiyle veya uzaya yayılan elektromanyetik dalgalarla da yapılabilir. Transmisyon ortamı, taşınacak sayısal işaretlerin saniyedeki bit sayısı (bits per second - bps) olarak belirlenen transmisyon hızını sınırlaması açısından önemlidir. Transmisyon ortamlarının en çok kullanılan birkaç türü aşağıda açıklanmıştır.

İletkenler arasında elektriksel işaret bu iletkenlerden birine, referans gerilimi veya akımı noktası da iletkenlerden diğerine uygulanır. Sızıtı işaretlerinin getirdiği en büyük problem, bu işaretlerin, iletken çiftinden sadece birini etkilemeleridir. Bunun sonucunda iki iletken arasında ilave bir gürültü gerilimi farkı oluşur ve alan taraf da iki iletken arasındaki gerilim farkını

1.2.2.1 İki Telli Açık Hatlar

olarak verilebilir. Bu faktörler, bu tür hatların maksimum uzunluklarını ve kullanılacakları maksimum transmisyon hızlarını sınırlar.

İki telli açık bir hat, transmisyon ortamlarının en basitidir. Bu tür hatlarda, iki iletken birbirlerinden izole edilmişlerdir ve yüzeyleri havayla temas halinde olacak şekilde açıktır. Böyle bir transmisyon ortamı, aralarında küçük mesafeler bulunan (50 m den küçük) ve en fazla 19.2 kbps hızda haberleşen sayısal ekipmanları bağlamak için son derece uygundur. Bir referans noktasına göre gerilim ya da akım farkından oluşan elektriksel işaret bu iletkenlerden birine, referans gerilimi veya akımı noktası da iletkenlerden diğerine uygulanır.

İki telli açık hatlar, iki DTEyi direkt olarak birbirine bağlamakta kullanılabileceği gibi genellikle, bir DTEyi bir DCEnin yerel parçasına (örneğin MODEMe) bağlamakta kullanılırlar. Daha ilerde görüleceği gibi, bu tür DTE-DCE bağlantıları birden fazla bağlantı içerirler ve bu bağlantıda her bir işaret için izole edilmiş ayrı bir iletken ve ortak bir referans iletkeni kullanılır. sonuçta ortaya çıkan iletkenler grubu ya tek bir muhafazanın içine alınarak multicore kablo haline, ya da herbir iletken ayrı korumaya alınarak flat ribbon kablo haline getirilirler.

Böyle bir bağlantıda, aynı kablonun içinde yan yana duran tellerde taşınan elektriksel işaretlerin birbirlerine sızmamalarına özen göstermek gerekir (bu sızıntı kapasitif kuplaj vasıtasıyla olmaktadır). Bunun yanında bu tür bağlantıların açık yapısı, taşınan elektriksel işaretlerin, çevredeki elektriksel gürültü kaynaklarından elektromanyetik radyasyon yoluyla etkilenmeleri de söz konusudur. Sızıntı işaretlerinin getirdiği en büyük problem, bu işaretlerin, iletken çiftinden sadece birini etkilemeleridir. Bunun sonucunda iki iletken arasında ilave bir gürültü gerilimi farkı oluşur ve alan taraf da iki iletken arasındaki gerilim farkını

1.2 izlediğinden, hatalı kararlar verebilir. Bu faktörler, bu tür hatların maksimum uzunluklarını ve kullanılabilecekleri maksimum transmisyon hızlarını sınırlarlar.

Burulmuş çift kabloların en büyük kısıtlaması deri etkisi denen olay ile

1.2.2.2 Burulmuş Çift İletkenli Hatlar (Twisted Pair)

frekansı yükseldikçe, tellerin

içerisinden akan akım iletkenlerin sadece kesit çaplarına yakın küçük bir

katmandan Gürültü kaynaklarının etkisi, birbiri üzerinde burulmuş iletken çiftleri

kullanılarak azaltılabilir ve böylece twisted pair olarak adlandırılan hatlar ortaya

çıkar. Buradaki mantık, elektriksel gürültü kaynaklarının, burulmuş çiftteki her iki

iletkeni de ortak etkilemesi olasılığını fazlaştıracak gürültünün, sonuçta kullanılan

fark işareti üzerindeki etkisini azaltmaktır. Bunun gibi, aynı kabloda bulunan

twisted pair iletken çiftleri de birbirlerini mümkün olduğu kadar az etkilemeleri için

birbirleri üzerine tekrar burulurlar. Bu her iki bozucu etkiyi de en aza indiren

transmisyon ortamı tiplerinden biri de koaksiyel (coaxial - Ortak eksenli)

kablodur. Twisted pair hatlar, uygun hat sürücüleri ve alıcıları kullanılarak, kısa

mesafelerde (100m den küçük) fakat yüksek transmisyon hızlarında (1 Mbps den

düşük) kullanılabildikleri gibi, uzun mesafelerde fakat düşük transmisyon

hızlarında da kullanılabiliirler. Bazı twisted pair kablolarda, dış gürültülerden

etkilenmeyi daha da azaltmak için kablonun en dışında bir ekran katmanı

kullanılır ve bu tür kablolar Ekranlanmış Burulmuş Çift Kablolar (Shielded

Twisted Pair Cable) adı verilir. Bu mümkün olmadığından bu bölüme kab yapı

veya balpeteği yapıli dielektrik maddeler kullanılır.

Bu geometriyi getirdiği sonuç sayesinde ortadaki iletken ile

gürültülerden büyük ölçüde korunmuştur ve radyasyon ve deri olayı etkilerinden

doğan kayıplar da en aza düşürülmüştür. Koaksiyel kablo, eğer iyi yapılmış

seçilmiş malzemelerden yapılmış ve uygun şekilde korunmuş ise, twisted pair kablolarla

1.2.2.3 Koaksiyel Kablo

Mbps hızlara kadar ve birçok yüz metre mesafelere kadar kullanılabilir.

Burulmuş çift kabloların en büyük kısıtlaması deri etkisi denen olay ile gelmektedir. İletilen işaretin hızı ve dolayısıyla frekansı yükseldikçe, tellerin içerisinde akan akım iletkenlerin sadece kesit çeperlerine yakın küçük bir katmandan akmaya başlamaktadır. Bu durumda mümkün olan iletim yüzeyinin sadece küçük bir bölümü kullanıldığından, yüksek frekanslarda ve transmisyon hızlarında kablonun empedansı gitgide büyümekte ve işaret zayıflamış olarak iletilmektedir. Bunun yanında, yüksek frekanslarda radyasyon etkisi yüzünden işaret gücünün büyük bir bölümü kayıp olmaktadır. Bu yüzden transmisyon hızının 1 Mbps dan daha yüksek olması gereken yerlerde başka bir transmisyon ortamının kullanılması gerekmektedir. Bu her iki bozucu etkiyi de en aza indiren transmisyon ortamı tiplerinden biri de koaksiyel (coaxial - Ortak eksenli) kablodur.

Bunun dışında, işaretin ışık olarak taşınması, fibre optik kablolarda elektromanyetik gürültülerin ve komşu iletken sızıntılarının (crosstalk)

Koaksiyel bir kabloda, işaret ve referans taşıyıcı iletkenler, sabit merkezli olarak silindirik biçimli bir ekran iletkeninin içinde uzanan bir iletken biçimini alırlar. Ortadaki iletken işaret taşıyıcı, dış ekran ise referans taşıyıcı olarak kullanılır. İdeal olarak, iki iletkeni birbirinden ayıran izolasyon ortamının hava olması gerekirken, pratikte bu mümkün olmadığından bu bölümde katı yapılı veya balpeteği yapılı dielektrik maddeler kullanılır.

Bu geometriyi getirdiği sonuç sayesinde ortadaki iletken dış gürültülerden büyük ölçüde korunmuştur ve radyasyon ve deri olayı etkilerinden doğan kayıplar da en aza düşürülmüştür. Koaksiyel kablo, eğer iyi yapılmış

işaretine çeviren bir optik veri tarafından üretilir. Benzer şekilde, hatırı diğer

olursa, 10 veya 20 Mbps hızlara kadar ve birkoç yüz metre mesafelere kadar kullanılabilir.

Yayan Diyot (Light Emitting Diode-LED) ve alıcı da ışığa duyarlı

diyot veya transistör (photo diode - photo transistor) elemanlarını kullanır.

1.2.2.4 Optik Kılcallar (Optical Fibres)

Her nekadar koaksiyel kablonun geometrisi çeşitli sınırlamaları en aza

indiriyorsa da, katı bir iletken ile ulaşılabilcek maksimum işaret frekansı ve

dolayısıyla transmisyon hızı yüksek olmasına rağmen sınırlı ve yetersizdir. Bu

durum, twisted pair kabloda da aynıdır. Optik kılcal kablo, taşınan bilgiyi

elektriksel bir işaret olarak değil, bir cam kılcalından geçen kırpışan bir ışık ışını

olarak taşımasıyla diğer transmisyon ortamlarından tamamen ayrılır. Işık dalgaları,

elektrik dalgalarından çok daha büyük bir band genişliğine sahip olduklarından,

1.2 fibre optik hatlar, yüzlerce Mbps gibi çok yüksek transmisyon hızlarında

kullanılabilirler. Bunun ötesinde, işaretin ışık olarak taşınması, fibre optik

kablolarda elektromanyetik gürültülerin ve komşu iletken sızıntılarının (crosstalk)

bulunmasını imkansız kılar. Bu yüzden bu kablolar, örneğin çelik işleme

fabrikaları gibi birçok yüksek gerilim ve akım anahtarlama ekipmanlarının

bulunduğu yerlerde, düşük hızlı veri iletişiminin hatasızlığını sağlamak için de

kullanılabilirler. Ayrıca, fibre optik bir kabloya ara bağlantı yapmanın zor olması

nedeniyle, yüksek düzeyde güvenlik istenen yerlerde de sıkça fibre optik kablolar

kullanılmaktadır.

genişliğine sahiptir ve çoğullama tekniği ile yüksek hızı yüzlerce

alt kanalı bulundursabilir. Bu durumda, kablının toplam kapasitesi, bu alt kanallara

paylaştırılır. Bir fibre optik kablo, her işaret için ayrılmış, koruma ve dış ışık

kaynaklarından izolasyon işine yarayan bir muhafazaya konmuş tek bir cam

kılcalından oluşur. Işık işareti, DTE tarafından kullanılan elektriksel işareti ışık

işaretine çeviren bir optik verici tarafından üretilir. Benzer şekilde, hattın diğer

ucundaki bir alıcı ters dönüştürme işini yapar. Genellikle burada bahsedilen verici bir Işık Yayan Diyot (Light Emmitting Diode-LED) ve alıcı da ışığa duyarlı diyot veya transistör (photo diode - photo transistor) elemanlarını kullanırlar. Kılcalı çevreleyen dış koruma maddesinin içe bakan tarafı ışığı yansıtan özellikte bir maddeyle kaplı olduğundan, LED tarafından üretilen ışık tamamen kılcalın içinde kalır ve buyüzden zayıflatma etkisi çok düşüktür. Genel olarak fibre optik kablolar koaksiyel kablolardan daha pahalıdır ve kırılğan yapılarından dolayı işlenmeleri ve yerleştirilmeleri zordur. Bu kablolar bölündüklerinde ve eklendiklerinde ortaya çıkan bağlantı kayıpları çok büyük olduğundan, sadece çok yüksek transmisyon hızlarına veya çok büyük derecede gürültü izolasyonuna gerek duyulan uygulamalarda kullanılırlar.

kurulmasının mümkün olmadığı veya çok pahalıya mal olduğu durumlarda da, veri iletişimi dalında

1.2.2.5 Mikrodalgalar

Örneğin, transmisyon hatının bir nehrü veya çok yoğun trafikli bir çevre yolunu geçmesi gerekebilir. Bu tür bağlara yerden yere mikrodalga bağı adı verilir. Şu ana kadar ele alınan transmisyon ortamları, bilgiyi bulunduran elektriksel işareti taşıyan fiziksel hatlar kullanılmaktadırlar. Ancak veriler, serbest uzayda elektromanyetik dalgalar aracılığı ile de taşınabilirler. Böyle bir ortamın bir örneği uydulardır. Bilginin üzerine module edildiği yönlendirilmiş bir mikrodalga ışını, yerden uyduya yayınlanır ve bu uydu tarafından, yerdeki önceden belirlenmiş yeni bir hedefe aktarılır. Tipik bir uydu haberleşme kanalı, çok çok yüksek bir band genişliğine sahiptir ve çoğullama tekniği ile yüksek hızlı yüzlerce alt kanalı bulundurabilir. Bu durumda, kanalın toplam kapasitesi, bu alt kanallara paylaştırılmaktadır.

Haberleşme amacıyla kullanılan uydular normal olarak durağandırlar. Yani, dünyanın çevresini her yirmidört saatte bir dolaşacak bir hızla yörüngede

hareket ederler ve böylece yerden bakıldığında hep aynı yerde duruyor gözükürler. Böylece uydu, görüş alanına giren alma ve gönderme istasyonlarına hizmet verebilir. Uydu tarafından aktarılan yeni mikrodalga ışınının açısı geniş olup, uydunun görüş alanındaki tüm istasyonlar tarafından alınabileceği gibi, bu açı daraltılarak ve çapı küçük verici antenleri kullanılarak sadece belirli bir istasyon veya istasyon grubunca alınabilmesi de sağlanabilmektedir. Uydular, dünya üzerinde bulunan çeşitli veri iletişim ağlarını birbirine bağlamak için ve aynı ülkede uzak mesafelerde bulunan iletişim ağlarının yüksek hızda birbirlerine bağlanabilmeleri için kullanılmaktadırlar.

1.2.3.1 RS-232C Mikrodalga bağları ayrıca, fiziksel hatların kurulmasının mümkün olmadığı veya çok pahalıya mal olduğu durumlarda da, veri iletişimi dalında kullanılmaktadırlar. Örneğin, transmisyon hattının bir nehri veya çok yoğun trafikli bir çevre yolunu geçmesi gerekebilir. Bu tür bağlara yerden yere mikrodalga bağı adı verilir ve bu bağlarda kullanılan mikrodalgalar dünya atmosferi içinde kaldığı için insan yapısı bozucu etkenlerden ve hava şartlarından etkilenebilirler. Buna rağmen, 50 km ye kadar olan mesafelerde kullanılabilirler.

1.2.3 İşaret Tipleri

Eğer birbirleriyle haberleşmeleri gereken iki DTE birbirlerine çok yakın bulunuyorlar ve haberleşme hızı çok önemli değilse bu durumda, ortam olarak sadece iki iletken ve DTE lerden gelen elektriksel işaretleri kullanılan kabloya uygun elektriksel işaret düzeylerine dönüştüren birkaç arabirim devre kullanmak yeterli olacaktır. Buna karşın, DTE ler arası uzaklık ve gerekli transmisyon hızı yükseldikçe, daha gelişmiş devreler ve teknikler kullanılmalıdır. Bunun yanında,

eğer iki DTE ülkenin uzak köşelerinde yer alıyorsa ve ortak kullanılabilecek, ulusal PTT tarafından sağlanmış bir veri iletişimi ağı yoksa bu durumda en ekonomik yol, PTT nin ses haberleşmesi için verdiği hatlardan faydalanmaktır. Bu durumda da, DTE tarafından üretilen işaretleri, ses işaretleri özelliğinde yeni işaretlere dönüştürmek gerekmektedir. Alıcı taraftaki DTE için de, alınan ve ses işareti özelliği taşıyan işaretleri DTEnin kullandığı elektriksel işaretlere çevirmek gerekmektedir. Bu iki fonksiyonu da yerine getiren cihazlara MODEM denir. Modemler ve diğer bazı transmisyon hatları tarafından kullanılan farklı işaret biçimlerinin bazıları aşağıdaki bölümlerde ele alınmaktadır.

1.2.3.1 RS-232C/V.24

EIA tarafından tanımlanan RS-232C arabirim standardı ve CCITT tarafından tanımlanan V.24 arabirim standardı, herhangi bir DTEyi, PTT tarafından verilmiş veya onaylanmış bir modeme bağlamak ve böylece değişik ekipman üreticilerinin anahtarlamalı telefon şebekesinin imkanlarından faydalanabilmelerini sağlamak amacıyla tasarlanmışlardır. Bu yüzden, DTE ve modem arasındaki maksimum mesafe düşük ve gereken maksimum transmisyon hızı 9600 bps olarak düşünülmüştür. Bu alanda getirilen ve geçerlilik gören ilk standartlar olmaları dolayısıyla zamanla bu arabirimler, herhangi bir karakter tabanlı çevre birimini (VDU,printer gibi) ana bilgisayara bağlamakta kullanılan arabirim standardı olarak genelleşmişlerdir. Böylelikle de, değişik üreticilerin bilgisayarlarıyla değişik üreticilerin çevre birimleri sorunsuz olarak birbirlerine bağlanabilir olmuşlardır.

DTEnin içindeki alt uniteler arasındaki uzaklık çok çok küçük olduğundan, bu birimler arasındaki bağlarda kullanılan lojik seviyelere tekabül eden elektriksel seviyeler genellikle çok küçüktür. Örneğin, sayısal ekipmanlarda kullanılan lojik tümleşik devre ailelerinden biri olan Transistor Transistor Logic (TTL) ailesinde, 2.0V ve 5.0V arasındaki gerilim değerleri lojik "1" ve 0.2V ve 0.8V arasındaki gerilim değerleri de lojik "0" olarak anlaşılmaktadır. Bu iki sınır arasındaki gerilim değerleri belirsiz bir konuma götürebilir. En kötü olasılıkla, eğer gerilim değeri sınırlardan birine çok yakınsa, zayıflatmanın ve gürültünün etkileri çok büyük olacaktır. Bu yüzden iki ekipmanın birbirlerine bağlanmasında kullanılan gerilim değerleri, ekipmanların içlerindeki alt uniteleri birbirine bağlarken kullanılan gerilim değerlerinden yüksektir.

RS-232C (V.24) arabirim standardı için tanımlanan gerilim düzeyleri şöyledir: transmisyon hattında lojik sıfır için en az +3V, lojik bir için ise en az -3V bulunmalıdır. Pratikte, bu seviyelerin aktüel değerleri, arabirim devrelerine uygulanan besleme gerilimi ile belirlenir ki bu da +-12V dan +-15V a kadar olabilir. DTEnin içine yerleştirilmiş gönderme ara birim devreleri, DTE içinde kullanılan düşük gerilim değerlerinden, RS-232C de kullanılan yüksek gerilim değerlerine dönüşüm yaparlar. Benzer şekilde yine DTE içine yerleştirilen arabirim alma devreleri de, RS-232C de kullanılan yüksek gerilim değerlerinden, DTE içinde kullanılan alçak gerilim düzeylerine ters dönüşüm sağlarlar. Hat Sürücü ve Hat Alıcı (line driver - line receiver) adını alan bu arabirim devreleri aynı zamanda gerekli gerilim evirme işlemlerini de üstlenirler.

Bu arabirim standardında yüksek gerilim değerlerinin kullanılması, gürültü ve zayıflatmanın etkilerini, TTL ve diğer normal lojik seviyelere göre büyük

ölçüde azaltmaktadır. RS-232C (V.24) arabirimi, ekipmanları birbirine bağlamak için ya flat ribbon kablo ya da tek referans iletkenli multicore kablo kullandığından, tek bir iletken üzerinde oluşan gürültü, problematik olabilir. Komşu kanal sızıntılarını önlemek için, verici devrenin çıkışına bir kapasitör bağlamak alışılmış bir çözümdür. Bunun, gönderilen sayısal işaretin köşelerini yuvarlatmak gibi bir yan etkisi olduğu gibi, problem olabilecek yüksek frekanslı bileşenlerin bastırılmasını sağlamaktadır. Uzaklıkların artması ve hızın yükselmesiyle zayıflatmanın etkisi, hat üzerindeki gerilimleri çok küçük gürültü kaynaklarından dahi etkilenebilecek kadar düşük düzeylere çekmektedir.

transmisyon hızlarında kullanılmak zorundadır. Bunun sebebi de, kullanılan anahtarların RS-232C (V.24) standartları, maksimum mesafe olarak 15m ve maksimum transmisyon hızı olarak 9.6 kbps değerlerini tanımlamakta ise de, pratikte bu değerlerin üstünde bağlı çevre birimleri ile karşılaşılabilir.

arabirimlerden istediğini kullanmakta serbest bırakılmaktadır.

1.2.3.2 20 ma Akım Çevrimi (20 ma Current Loop)

1.2.3.3 RS-422/V.11

RS-232C de tanımlanan elektriksel arabirim standardına alternatif olarak kullanılan diğer bir standart da 20 ma akım çevrimidir. Bu standartta, adından da belli olduğu gibi, gerilim düzeyleri değil akım düzeyleri kullanılmaktadır ve böylelikle transmisyon hızı sınırında bir iyileştirme olmamasına rağmen bağlanan aletler arasındaki fiziksel mesafe sınırı büyütülmektedir.

Genel olarak, bir anahtarın (röle veya benzeri diğer bir anahtarlama birimi) konumu, gönderilen bit katarına göre kontrol edilir. Lojik "1" değeri için anahtar kapalıdır ve karşı tarafa 20 ma lik bir akım gönderilir, lojik "0" için ise anahtar açıktır ve karşıya akım gönderilmez. Alıcı tarafta, akımın akışı akıma

duyarlı bir devre tarafından sezinlenir ve gönderilen bit katarı tekrar elde edilmiş olur.

Bu standartta akım değerlerinin kullanılması, gerilim değerlerinin kullanılmasından daha iyidir, çünkü akım çevresinin oluşturulabilmesi için bir çift iletken kablo kullanılmakta ve böylece dış gürültü kaynakları her iki iletkeni de etkileyerek ortak modda bir gürültü işareti oluşturmakta ve bu da akım farklarını sezinleyen alıcı devre üzerinde etkili olmamaktadır. Buna göre 20 ma akım çevrimi 1 km ye kadar olan uzun mesafelerde kullanılabilir ancak alçak transmisyon hızlarında kullanılmak zorundadır. Bunun sebebi de, kullanılan anahtarlama elemanları ve akıma duyarlı devrelerin çalışma hızlarının kısıtlılığıdır. Bu yüzden, günümüzde ekipman üreticileri, RS-232C ve 20 ma akım çevrimi çıkışlarının her ikisini de ekipmanlarına koymakta ve böylece kullanıcı bu arabirimlerden istediğini kullanmakta serbest bırakılmaktadır.

1.2.3.3 RS-422/V.11

Eğer ekipmanlar arası mesafe ve transmisyon hızı beraberce büyük olmak zorunda ise, RS-422/V.11 işaret tanımlamaları kullanılmalıdır. Bu işaret standartları, bir twisted pair kablo ile farksal modda çalışan sürücü ve alıcılara dayanmaktadır.

Farksal modda çalışan bir hat sürücü, her lojik seviyesi için eşit genlikte fakat ters işaretli (ters polariteli) bir çift işaret üretir. Alıcı uçtaki farksal modda çalışan alıcı sadece uçları arasındaki potansiyel farkına duyarlı olduğundan, kablo çiftinin iki elemanına birden sızan gürültülerin hiçbir etkisi

olmayacaktır. Bu yüzden farksal alıcıların iyi bir ortak mod bastırma oranına (Common Mode Rejection Ratio-CMMR) sahip oldukları söylenir. RS-422 nin bir türevi olan RS-423, RS-232 tarafından üretilen tek yönlü işaretlerin farksal bir alıcı tarafından alınabilmesine olanak vermektedir. RS-422, 100m ye kadar olan mesafelerde ve 1 Mbps hızda, twisted pair kablo ile kullanıma son derece elverişlidir. Daha uzak mesafelerde daha düşük hızlarda kullanılabilir.

Birinci dönüşümün tersi yapılarak sayısal işaret tekrar elde edilmelidir. Birinci dönüşüm Herhangi bir transmisyon hattının önemli bir parametresi, o hattın karakteristik empedansıdır (Z_0). Bu empedansın tanımı şöyledir: Bir alıcıya maksimum işaret gücü transferi, ancak alıcının bağlı olduğu transmisyon hattı, yine kendi karakteristik empedansı olan Z_0 ile sonlandırılmışsa gerçekleşir. Eğer böyle değilse, alınan işaretin bozulmasına sebep olan işaret yansımaları meydana gelir. Bu yüzden bu tür hatlar, karakteristik empedansları olan 100 ila 200 ohmluk dirençlerle sonlandırılırlar.

1.2.3.4 Modem İşaretleri

Eğer veri iletişimi, ortak anahtarlama telefon şebekesi (PSTN) üzerinden yapılıyorsa herşeyden önce, DTE tarafından üretilen işaretlerin, PSTN in kabul edebileceği bir forma çevrilmesi gerekmektedir. Bunun gereği tabi ki, tamamen ses iletişimi için tasarlanmış olan PSTN hatları ve anahtarlama devrelerinin 400 Hz den 3400 Hz e kadar uzanabilen dar band genişliğidir.

Bir devrenin bozmadan geçirebildiği işaretlerin frekanslarının aralığı band genişliği olarak bilinir. Yani, PSTNin 400 Hzden 3400 Hz e kadar uzanan 3000 Hz lik bir band genişliği vardır. Bunun anlamı, telefon hattının, kendine

gelen çok alçak frekanslı işaretleri geçirmeyeceğidir. Böyle bir durum, örneğin, bitlerinin hepsi "1" veya "0" olan uzun bit katarlarının bu hattın peşisıra geçirilmeye çalışılmasında ortaya çıkabilir. Buyüzden, iki gerilim düzeyinden oluşan sayısal işaretlerin direkt olarak telefon hattına uygulanarak taşınması mümkün değildir. Bunun yerine sayısal işaretler, frekans domeninde ses bölgesine düşen işaretlere dönüştürülerek gönderilmeli ve alıcı tarafta da bu dönüşümün tersi yapılarak sayısal işaret tekrar elde edilmelidir. Birinci dönüşümü yerine getiren devrelere modülatör, ters dönüşümü gerçekleştiren devrelere demodülatör denmektedir. Veri iletişim hattının her iki tarafının da gönderme ve alma işlemlerinin ikisini de yapabilmesi gerektiğinden, bu iki tür devre bir araya getirilerek MODEM cihazları oluşturulmuştur.

Modemler kullanılarak, gerek önceden numara çevirerek fiziksel bağlantı kurulduktan sonra PSTN üzerinden, gerekse PTT den kiralanan özel kiralık hatlar üzerinden veri iletişimi sağlanabilmektedir. PSTN in kullanılmasından dolayı normal bir telefon görüşmesi gibi ücretlendirilen bilgisayar görüşmeleri de, pahalıya mal olmaktadır. Bu yüzden ancak ortak anahtarlama ihtiyacı varsa bu yola gidilmektedir. Kiralık hatların bir iyi tarafı da, normal telefon hatlarına rağmen daha belirli olduklarından, daha yüksek band genişliklerine ve dolayısıyla daha yüksek transmisyon hızlarına sahip olmalarıdır.

Sayısal bir işareti PSTNe uygun bir hale getirmek üzere kullanılabilecek üç çeşit modülasyon türü vardır. Bunlar Genlik Modülasyonu, Frekans Modülasyonu ve Faz Modülasyonudur.

Genlik Modülasyonunda, tek frekanslı bir taşıyıcı işaretinin genliği, gönderilmek istenen bit katarına göre değiştirilir. Bu tür modülasyon, çok basit olmasına rağmen, PSTN hatlarının büyük zayıflatma oranlarından çok etkilenir ve bu yüzden fazla kullanılmaz. İlk zamanlarda, kısa mesafeli kiralık hatlarda kullanılmıştır. Bu yüzden bu çeşit modülasyona Farksal PSK (differential PSK) adı verilir.

Frekans Modülasyonunda, sabit genlikli bir taşıyıcı işaretinin frekansı, gönderilmek istenen bit katarına göre iki değer arasında anahtarlanır. Taşıyıcı belli bir anda sadece iki frekans değerinden birinde bulunabildiğinden, bu tür modülasyona Sayısal Frekans Modülasyonu veya Frekans Anahtarlama (Frequency Shift-Keying FSK) da denir. Bu modülasyon türü, PSTN üzerinden kurulmak istenen alçak hızlı bağlantıları kuran modemlerde sıkça kullanılmaktadır ve demodüle etmek için gerekli devrenin yapısının basitliği ve gerekli band genişliğinin düşüklüğü açısından kullanışlıdır.

Faz Modülasyonunda, taşıyıcının genliği ve frekansı sabit tutulmakta, bunlar yerine taşıyıcının fazı, gönderilmek istenen bit katarına göre anahtarlanmaktadır. Faz modülasyonunun bir çeşidi, lojik "1" ve "0" değerleri için birbirleri arasında 180 derece faz farkı olan iki taşıyıcı işaret kullanılmaktadır ve bu yüzden Faz Anahtarlama (Phase Shift Keying - PSK) adını almıştır. Böyle bir faz modülasyonu türünde, alıcı tarafta, gelen işaretle faz açısından karşılaştırılmak üzere bir referans taşıyıcı işaret örneği bulundurulması gerektiğinden, buna ayrıca Faz Sağlamalı PSK (phase-coherent PSK) de denir. Pratikte, böyle bir düzen, karmaşık demodülasyon devreleri gerektirir ve taşıyıcı fazındaki beklenmedik değişikliklere karşı son derece duyarlıdır. Genellikle başka bir faz modülasyonu türünün kullanılmasıdaki sebep de budur. Bu yeni türde, gönderilmek istenen

bir bit için yapılması gereken faz ötelemesi, bir önce gönderilmiş olan bitin neden olduğu faza bağlı olarak yapılır. Yani, 90 derecelik bir faz sıçraması lojik "0" ve 270 derecelik bir faz sıçraması lojik "1" anlamına gelmektedir. Böylelikle alıcı tarafın, taşıyıcının fazının tam değerini anlaması yerine, sadece faz sıçraması miktarını sezinlemesi yeterli olacaktır. Bu yüzden bu çeşit modülasyona Farksal PSK (differential PSK) adı verilir.

Bütün bunların hepsi, fiziksel katman arabirim standartlarını oluştururlar. Bu standartların hepsini detaylı olarak burada incelemek mümkün olmamakla beraber, en çok kullanılan birkaçından bahsedilecektir.

1.2.4.1 RS-232C/V.24

Daha önce bahsedildiği gibi RS232-C/V.24, bir DTE yi, PTT tarafından sağlanan veya onaylanan bir modeme, daha da geneli bir DCEye bağlamak üzere geliştirilmiştir. Bu arabirim standardında kullanılan ilave kontrol işaretleri Şekil 1.17 de görülmektedir.

TxD (transmit data) ve RxD (receive data) sırasıyla veri gönderen ve alan hatlardır. Diğer hatların hepsi, PSTN üzerinden anahtarlama bir bağlantının kurulması ve koparılması için gerekli zamanlama ve kontrol fonksiyonlarını yerine getiren kontrol işaretleridir.

Bağlantı, uzaktaki terminali kullanan kullanıcının, bilgisayar tarafındaki modemini bağlı olduğu telefon hatının telefonunu çevirmesiyle kurulur. Burada, otomatik arama fonksiyonu olan modemler de kullanılabilir. Eğer hat boşsa ve bilgisayar da hazırda, çalma sesi duyulacak ve kullanıcı tek tonlu bir ses duyacaktır.

1.2.4 Fiziksel Katman Arabirim Standartları

Bundan Önceki bölümler, çeşitli transmisyon ortamları ve bunlarda kullanılabilecek işaret türlerine ilişkindi. Bununla beraber, standartlar, kullanılması gereken işaret türlerini belirlemenin yanında, bu arabirimlerde kullanılan ve veri transferinin zamanlaması ve sıralaması için gerekli olan ilave işaretleri de tanımlamaktadırlar. Bütün bunların hepsi, fiziksel katman arabirim standartlarını oluştururlar. Bu standartların hepsini detaylı olarak burada incelemek mümkün olmamakla beraber, en çok kullanılan birkaçından bahsedilecektir.

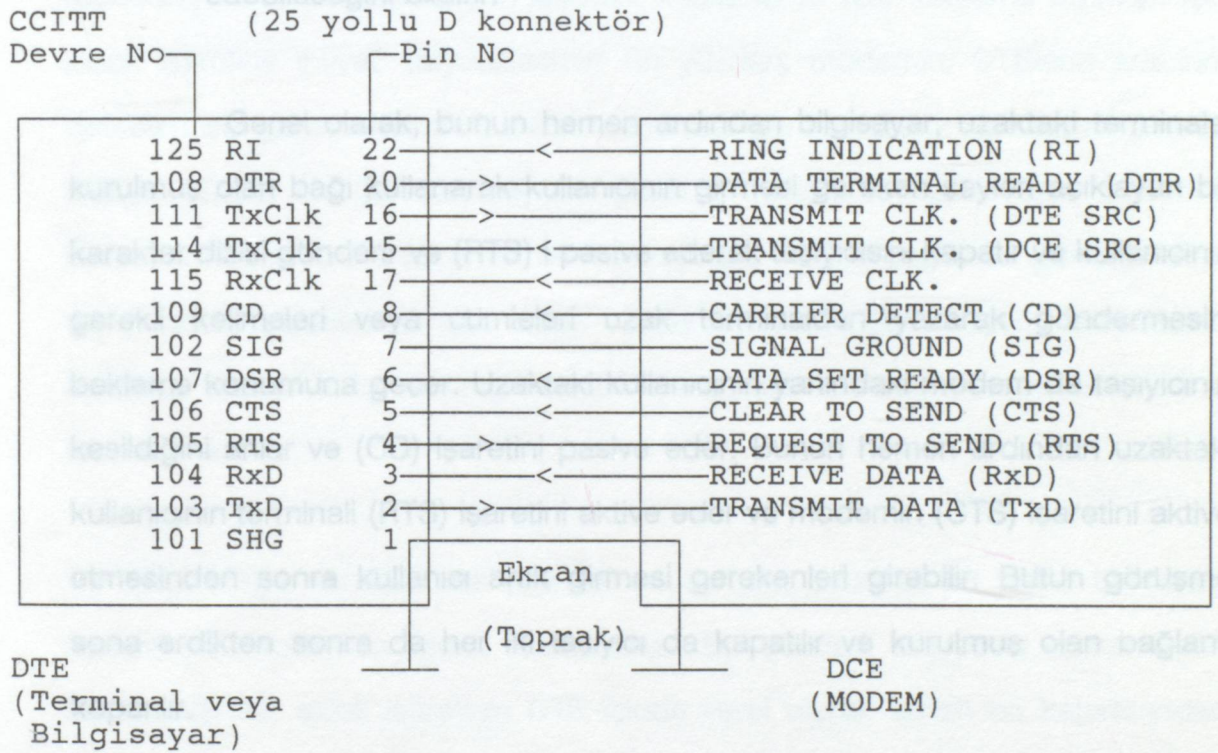
1.2.4.1 RS-232C/V.24

Daha önce bahsedildiği gibi RS232-C/V.24, bir DTE yi, PTT tarafından sağlanan veya onaylanan bir modeme, daha da geneli bir DCEye bağlamak üzere geliştirilmiştir. Bu arabirim standardında kullanılan ilave kontrol işaretleri Şekil 1.17 de görülmektedir.

TxD (transmit data) ve RxD (receive data) sırasıyla veri gönderen ve alan hatlardır. Diğer hatların hepsi, PSTN üzerinden anahtarlamalı bir bağlantının kurulması ve koparılması için gerekli zamanlama ve kontrol fonksiyonlarını yerine getiren kontrol işaretleridir.

Bağlantı, uzaktaki terminali kullanan kullanıcının, bilgisayar tarafındaki modemle bağlı olduğu telefon hattının telefonunu çevirmesiyle kurulur. Burada, otomatik arama fonksiyonu olan modemler de kullanılabilir. Eğer hat boşsa ve bilgisayar da hazırsa, çalma sesi duracak ve kullanıcı tek tonlu bir ses duyacaktır.

Bunun üzerine kullanıcı, veri iletişimi bağıını kurmak için modem üzerindeki özel bir tuşa basar. Bu, yerel terminali, kurulmuş olan telefon bağlantısına yönlendirir ve yerel modem bu terminale DSR (data set ready) işaretini aktive ederek cevap verir. Bu sırada, yerel modemin üzerinde bu işarete tekabül eden bir ışık yanarak veri iletişimi hattının kurulduğunu belirtir.



Şekil 1.7: RS-232C İşaret Tanımlamaları

Numara çevrildiğinde ve bilgisayar tarafındaki modemin telefonu çaldığında, bu modem çağrı geldiğini belirtmek için (RI) işaretini aktive eder. Bilgisayar çağrıyı almaya hazır konumda ise (DTR) bilgisayar tarafından zaten aktive edilmiştir. Bu durumda bilgisayar (RTS) işaretini aktive ederek cevap verir. Bunun iki sonucu olur:

yapı (1) Modem, karşı tarafa, çağrının bilgisayar tarafından kabul edildiğini iki modem belirtmek üzere taşıyıcı işaretini gönderir; ve her iki modem de sürekli taşıyıcı işareti göndermektedirler.

(2) Uzaktaki kullanıcının modeminin gönderilecek veriyi hazırlaması için beklenen küçük bir süreden sonra bilgisayar tarafındaki modem için bir (CTS) işaretini aktive ederek bağlı olduğu bilgisayara, veri transfer modulası edebileceğini bildirir.

Genel olarak, bunun hemen ardından bilgisayar, uzaktaki terminale, kurulmuş olan bağı kullanarak kullanıcının girmesi gereken şeyleri açıklayan bir karakter dizisi gönderir ve (RTS) i pasive ederek taşıyıcısını kapatır ve kullanıcının gerekli kelimeleri veya cümleleri uzak terminalden yazarak göndermesini bekleme konumuna geçer. Uzaktaki kullanıcının yanındaki modem de taşıyıcının kesildiğini anlar ve (CD) işaretini pasive eder. Bunun hemen ardından uzaktaki kullanıcının terminali (RTS) işaretini aktive eder ve modemin (CTS) işaretini aktive etmesinden sonra kullanıcı artık girmesi gerekenleri girebilir. Bütün görüşme sona erdikten sonra da her iki taşıyıcı da kapatılır ve kurulmuş olan bağlantı koparılır.

Burada, kontrol işaretlerinin tamamının görevlerini anlatabilmek için örnek olarak özellikle half duplex bağlantı seçilmiştir. Unutmamak gerekir ki, half duplex bağlantılarda gönderme modundan alma moduna geçmek için gerekli olan ve konum değiştirme zamanı (turn-around time) olarak adlandırılan süre hiç de kısa değildir. Bu yüzden, sadece half duplex modda çalışmak gerekse bile, full duplex bir devreyle çalışmak tercih sebebidir. Full duplex bir devre kullanıldığında tabi ki gönderme ve alma fonksiyonları aynı anda

yapılabilmektedir ve bu durumda her iki DTE nin RTS işaretleri sürekli aktif, her iki modemde de CTS işaretleri sürekli aktif kalmakta ve her iki modem de sürekli taşıyıcı işareti göndermektedirler.

(1) Terminal modem rolü oynar ve bağlantı türü değişmemiş olur.

Senkron haberleşme kullanıldığında, bit senkronizasyonunu sağlamak için bir clock işareti kaynağı gereklidir. Eğer senkron bir modem kullanılıyorsa, modülasyon ve demodülasyon işlevleri, kodlama ve ters kodlama işlemleri için clock işaretine ihtiyaç duyacaklardır. Bu yüzden, modemde, DTElerin arabirim devreleri için gereken clock işaretini sağlaması gerekir. Clock işareti modemden DTEye (TxClk - DCE Source) ve (RxCLK) hatları ile iletilir. Eğer modem, işaretleşme elementi başına bir bit kullanıyorsa, DTEye sağlanan gönderme ve alma clock işaretleri kanalın baud hızıyla aynıdır. Eğer modem işaretleşme elementi başına iki bit kullanıyorsa, DTEye gönderilen clock işareti kanal baud hızının iki katıdır.

Diğer taraftan asenkron bir modemde, baud ve bit hızları tamamen aynıdır ve modemde clock kullanılmaz. Bu yüzden asenkron transmisyon kullanıldığında, clock işaretinin DTE içinde yerel olarak alınan bit katarlarından elde edilmesi gerekir ve bu amaçla da genellikle sayısal faz kilitlemeli çevrim (Digital Phase Lock Loop) kullanılır. Ayrıca, bir terminali bir modeme bağlamak için gereken ek devreleri azaltmak amacıyla arabirim kontrol devrelerinin (UARTlar ve USRTler) RTS-CTS ve DTR-DSR işaretlerini otomatik çalıştıran birer modem kontrol bölümleri vardır.

Şekil 1.7 deki bağlantı türünün birebir bağlantı olduğu görülmektedir. Bu sadece DTE ve DCE birbirlerine bağlanırken geçerlidir. Eğer iki DTE

1.2 aralarında 9 modem olmaksızın direkt olarak birbirlerine RS-232C ile bağlanacaklarsa, Bu durumda şu üç seçenekten birinin yerine getirilmesi gerekir:

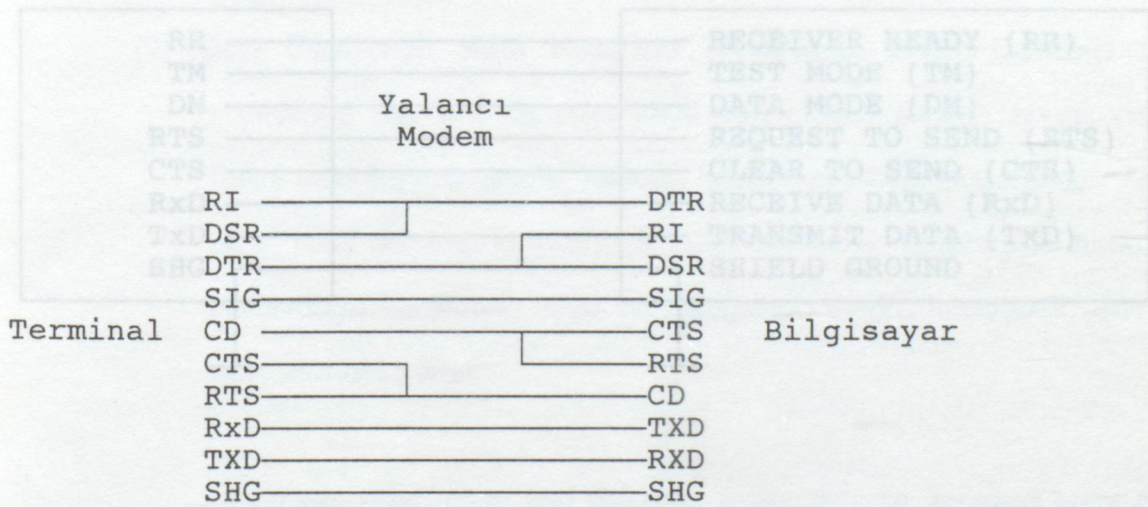
RS-422 elektriksel işaretleri kullandığında ortaya çıkan arabirim star (1) Terminal modem rolü oynar ve bağlantı türü değişmemiş olur. Şekil 1.19 da

görülmektedir. Burada kullanılan RS-422 standardından dolayı her kontrol işareti için (2) Bilgisayar modem rolü oynar ve bağlantı türü değişmemiş olur. çoğu RS-

232C standardındaki kontrol işaretleriyle aynıdır. Ayrıca Data Mode ve Receiver Re (3) Bilgisayar ve terminal değiştirilmez, aradaki bağlantı şekli değiştirilir. ktedir.

Test Mode işareti, RS-449'a özgüdür ve haberleşme ekipmanlarının bir bölümüdür. Bu seçeneklerden en uygununun üçüncü seçenek olduğu açıktır. Bu

çapraz bağlantıyı sağlayan aktif ya da pasif cihazlara yalancı modem (null modem) denir. Aktif yalancı modemler, bağlantı çaprazlamasının yanında, her iki DTEnin de clock işaretlerini sağlayamamaları durumunda ve senkron transmisyon halinde clock işaretini de sağlarlar. ve genel olarak modem yerine geçen devreler (modem eliminator) adını alırlar. Pasif bir yalancı modemin iç bağlantısı Şekil 1.18 de görülmektedir.

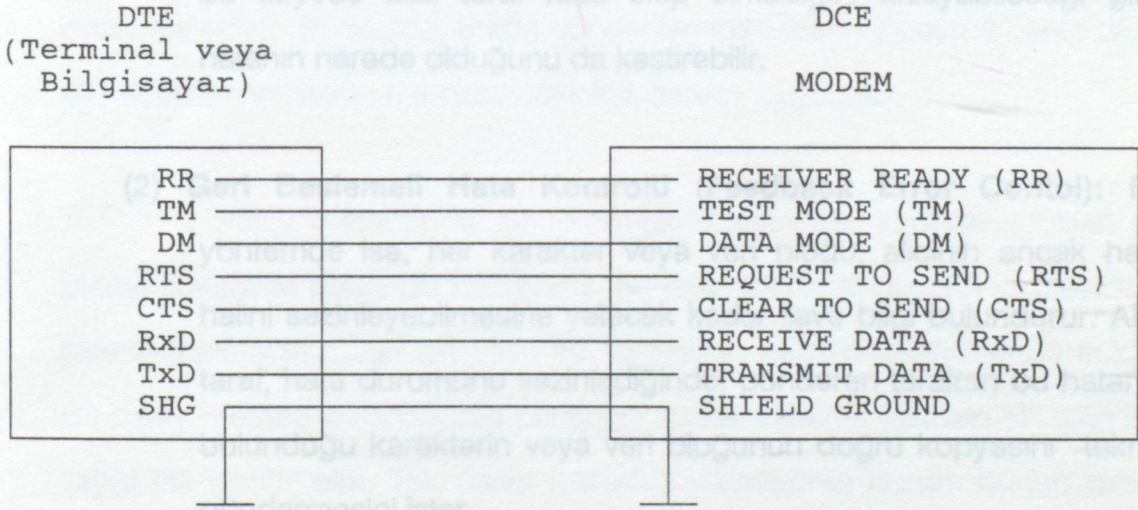


Şekil 1.19: RS-449/V.35 İşaret Tanımları

Şekil 1.18: Yalancı Modem Bağlantısı

1.2.4.2 RS-449/V.35

RS-422 elektriksel işaretleri kullanıldığında ortaya çıkan arabirim standardı RS-449 dur. Bu standartta kullanılan kontrol işaretleri Şekil 1.19 da görülmektedir. Burada kullanılan RS-422 standardından dolayı her kontrol işareti için bir çift iletken mevcuttur. Görüldüğü gibi kontrol işaretlerinin birçoğu RS-232C standardındaki kontrol işaretleriyle aynıdır. Ayrıca Data Mode ve Receiver Ready işaretleri de RS-232C deki DSR ve DTR işaretlerine tekabül etmektedir. Test Mode işareti, RS-449 a özgüdür ve haberleşme ekipmanlarının bir bölümünün test edilebilmesini sağlamak düşüncesiyle eklenmiştir. Genel olarak bu işaret, DTEnin çıkışını DCE üzerinden tekrar DTEnin girişine bağlamak için kullanılır. Böylelikle DTE, haberleşme ekipmanlarından bazılarının doğru çalışıp çalışmadığını test edebilmektedir.



Şekil 1.19: RS-449/V.35 İşaret Tanımları

1.3 Hata Sezinleme Yöntemleri

İki DTE arasında veri iletişimi yapılırken çoğunlukla çeşitli hataların meydana geldiği görülür. Burada hatadan kasıt, aslında "1" olarak gönderilen bir bitin "0" olarak algılanması veya tersidir. Bu hata hallerine özellikle kullanılan transmisyon ortamı, PSTN gibi gürültü açısından zengin bir ortamsa rastlanır. Bir alıcı tarafından alınan bilginin gönderilen orijinal bilgiyle aynı olması olasılığını fazlalaştırmak için, alıcı tarafta hata durumlarının sezinlenebilmesi gereklidir. Bunun da ötesinde, hatalı olduğu anlaşılan bilginin doğru halinin de tekrar elde edilebilmesi gereklidir. Bunu başarabilmek için günümüzde iki yaklaşım vardır:

(1) İleri Hata Kontrolü (Forward Error Control): Bu yöntemde, gönderilen her karakter veya veri bloğu fazlalık ek bir bilgi içerir ve bu sayede alıcı taraf hata olup olmadığını anlayabileceği gibi, hatanın nerede olduğunu da kestirebilir.

(2) Geri Beslemeli Hata Kontrolü (Feedback Error Control): Bu yöntemde ise, her karakter veya veri bloğu, alıcının ancak hata halini sezinleyebilmesine yetecek kadar ilave bilgi bulundurur. Alıcı taraf, hata durumunu sezinlediğinde, gönderen taraftan bu hatanın bulunduğu karakterin veya veri bloğunun doğru kopyasını tekrar göndermesini ister.

Pratikte, gönderilen veri bitlerinin sayısı arttıkça, forward error control metodu için gereken fazlalık bitlerin sayısı da hızla artmaktadır. Bu yüzden

1.3 günümüzde en çok kullanılan hata kontrol metodu feedback error control metodudur.

Anlaşılabileceği gibi feedback error control metodu iki kısma ayrılabilir. Bunlardan birincisi güvenilir bir biçimde hata durumlarının sezinlenmesini sağlayan teknikler ve diğeri de mümkün olabilecek yeniden gönderme metodlarını belirleyen kontrol algoritmalarıdır. Bu bölümde en çok kullanılan hata sezinleme yöntemlerine değinilecektir.

1.3.1 Parity

Her defasında gönderilen bitlerin sayısının az olduğu ve hata olasılığının çok düşük olduğu uygulamalarda en uygun hata sezinleme yolu parity metodudur. Bu metodda, gönderen taraf, gönderdiği her karaktere bir parity biti ekler. Bu metod, bütün gönderilen kelimelerin 7 veya 8 bitten oluştuğu asenkron transmisyon metodları için son derece uygundur.

Parity metodunda, gönderilmek istenen her karakterin bitleri ön bir işleme sokularak parity biti hesaplanır. Bu hesaplama, karakterin içindeki "1" olan bitlerin toplam sayısının çift veya tek olmasının belirlenmesinden ibarettir. Eğer çift parity metodu kullanılıyorsa parity biti ancak karakter içindeki "1" olan bitlerin sayısı çift ise "1" olur. Tek parity metodu kullanılıyorsa durum bunun tamamen tersidir. Bu hesaplanan parity biti karakterin sonuna eklenerek karşı tarafa gönderilir. Alıcı taraf da, karakter üzerinde aynı hesaplamayı yaparak elde ettiği parity bitini gönderen taraftan aldığı parity bitiyle karşılaştırır ve hatayı sezinleme şansı bulur.

1.3.2 Blok Toplamı Karşılaştırma Metodu

Transmisyonun bloklar halinde yapıldığı bir uygulamada hata sezinleme olasılığı, her karakterden sonra gönderilen her bir parity bitine ilave olarak, bloktaki tüm karakterlerin tüm bitlerinden hesaplanmış ilave parity bitlerinin kullanılmasıyla artırılabilir. Böyle bir yöntemde bir taraftan USART tarafından yine her karakter için birer parity biti hesaplanırken, kontrolör arabirim tarafından bütün bloktaki karakterler üzerinden her bit pozisyonu için birer parity biti hesaplanır. Yani önce her karakterin birinci bitleri parity işlemine konur ve elde edilen sonuç yeni bir karakterin birinci bitine atanır, daha sonra her karakterin ikinci bitleri parity işlemine sokulur ve elde edilen bit sonuç karakterinin ikinci bitine atanır ve bütün bitler için bu işlem yapılır. Bu işlemin sonucunda ek bir karakter elde edilir ki buna blok kontrol toplamı denir. Bu karakter bloğun sonuna fazlalık olarak eklenerek karşı tarafa gönderilir. Karşı taraf da, aynı işlemleri tekrar yaparak blok kontrol toplamı değeriyle karşılaştırır ve hata olup olmadığını sezinler.

Bu yöntemin sağlamlığı, aynı bit pozisyonunda aynı anda iki bit hatasının olma olasılığının düşük olduğu uygulamalarda yüksektir. Daha yüksek gürültülü hatlarda aşağıda anlatılan polynomial hata sezinleme metodu kullanılır.

1.3.3 Polynomial Kodlar

PSTN üzerinden yapılan veri iletişimi esnasında meydana gelen hataların bir çoğu, santrallerdeki anahtarlama elemanlarının oluşturduğu gürültü darbeleri gibi elektriksel patlamaların yol açtığı hatalardır. Bu olaylar, veri katarının

içerisinde ardışıl birkaç bitin bozulmasına yol açarlar ve bu yüzden bu tür hatalara hata patlaması (Error Burst) denir. Bir error burst, hatalı bir bitle başlar ve hatalı bir bitle son bulur, aradaki bitler bozulmuş veya bozulmamış olabilir. Yani, error burst, baştaki ve sondaki hatalı bitler de dahil olmak üzere iki hatalı bit arasında kalan tüm bit katarı olarak tanımlanabilir. O halde, bir error burstün en son bitiyle diğer bir error burstün ilk biti arasında B bit kadar bozulmamış bit bulunmalıdır. Burada B error burst uzunluğunu vermektedir.

Parity ve onun türevi olan blok kontrol toplamı yöntemleri, error burst türü hatalara karşı güvenilir değildirler. Bu durumda güvenli olan polynomial kodlara dayalı yöntemler kullanılır. Polynomial kodlar, blok modunda transmisyonun kullanıldığı uygulamalarda kullanılır ve her bir blok için belli hane sayılı yeni bir sayı hesaplanarak karşı tarafa gönderilir. Alıcı taraf da, alınan blok üzerinde aynı hesaplamayı tekrar yapar ve gönderilmiş olan değeri bulmaya çalışır. Bulamıyor ise bu blokta hatalı bir bilgi var demektir.

İlave olarak gönderilecek olan bilginin bit hane adedi, karşılaşılabilecek en uzun error burste göre belirlenir. En çok kullanılan değerler 16 ve 32 bittir. Bu bitler Çatı Kontrol Sırası (Frame Check Sequence - FCS) veya Aralıklı Fazlalık Kontrolü (Cyclic Redundancy Check - CRC) olarak isimlendirilirler. Yanı sıra, blokun başına eklenerek karşı tarafa gönderilir. Alıcı taraf, FCS değeri de dahil olmak üzere bütün bloğu aynı işlemi yapar ve karşı tarafın gönderdiği değere karşılaştırır. Eğer karşı tarafın gönderdiği değere eşit değilse, blokta hata var demektir.

Polynomial kodların kullanılmasındaki mantık ikilik düzen aritmetiği kullanılarak şöyle açıklanabilir:

M = k bitlik bir sayı,
R = n bitlik bir sayı (k>n)
G = (n+1) bitlik bir sayı

olmak üzere, eğer

$$R = \frac{M \times 2^n}{G} \quad \text{ise,} \quad \frac{M \times 2^n + R}{G} = 0 \text{ dır.}$$

Bunun doğruluğunu görmek için ikinci ifadede R yi yerine koyarsak,

$$\frac{M \times 2^n + M \times 2^n}{G}$$

ifadesini elde ederiz ki bu sığıra eşittir. Çünkü herhangi bir sayı ikilik düzen aritmetiği ile kendiyile toplanırsa sıfır elde edilir.

Yani, bloğun tamamı olan Mnin sonuna FCS dijitlerinin sayısı kadar sıfır eklenir ve Üretici Plynomiali denen (Generator Polynomial) ve FCS dijitlerinden bir fazla dijit bulunduran ikinci bir binary sayıya (G) ikilik düzende bölünür. Bölme işlemi, paralel olarak bloktaki tüm bitler için bit bit EXOR işlemiyle yapılır. sonuçta kalan olarak elde edilen R, FCS değeridir. Bu FCS, bloğun sonuna eklenerek karşı tarafa gönderilir. Alıcı taraf, FCS bitleri de dahil olmak üzere bütün bloğu aynı üretici polynomial değerine böler. Burada elde edilen kalanın (R) sıfır olması gerekir. Eğer değilse alınmış olan bit katarında hata var demektir.

UMICOMM programı, büyük bir kullanıcı potansiyeline sahip olan IBM PC ve bununla uyumlu kişisel bilgisayarlarda kullanılmak üzere tasarlanmış bir veri iletişim programıdır. Bu programın karşısında yardımcı olarak bir bilgi servisi programı olan UMISIR programı tasarlanmıştır.

BÖLÜM 2

UMICOMM

YAZILIMININ

TASARIM

ÖZELLİKLERİ

UMICOMM YAZILIMININ TASARIM ÖZELLİKLERİ

2.1 UMICOMM Yazılımının Teknik Tanıtımı

2.1.1 Giriş

UMICOMM programı, büyük bir kullanıcı potansiyeline sahip olan IBM PC ve bununla uyumlu kişisel bilgisayarlarda kullanılmak üzere tasarlanmış bir veri iletişim programıdır. Bu programın karşısında yardımcı olarak bir bilgi servisi programı olan UMISRV programı tasarlanmıştır.

UMICOMM programı, PC kullanıcısının, sistemine bağlı bir modem aracılığı ile uzaktaki diğer bir sisteme bağlanmasını ve buradaki servislerden faydalanabilmesini sağlar. Bunu yaparken de, uzaktaki sistemin, kendine bağlı olmasını beklediği terminal tipini taklit eder. Gerek haberleşme hattına ait parametreler, gerekse terminal emülasyonuna ve modem tipine ait parametreler kullanıcı tarafından kolaylıkla değiştirilebilecek şekilde tasarlanmıştır. Ayrıca bu programı kullanan kişi, XMODEM hata düzeltmeli dosya transfer protokolünü kullanarak uzaktaki sistemle dosya alışverişi de yapabilir.

UMISRV programı ise, kendilerine ait ve dışarıdaki kullanıcıların faydalanabileceği bir bilgi servisi oluşturmak isteyen kullanıcıların ihtiyaçlarına servis vermek üzere tasarlanmış bir programdır. Bu programda da, haberleşme hattına ve modeme ait parametreler kolaylıkla değiştirilebilir. Ayrıca, sisteme uzaktan bağlanacak olan kişilere gösterilecek olan bilgiler yine bu sistemin yöneticisi olan kişi tarafından istenen biçimde düzenlenebilir.

Bu bölümde, UMICOMM ve UMISRV programlarının yazılım yapısına değinilerek, bu yazılımları oluşturan modüller kısaca açıklanacaktır. Bu proje için geliştirilen tüm yazılımda "C" programlama dili kullanılmıştır.

Şimdi bu modülün içerdği bazı alt programlara göz atalım:

2.1.2 Yazılım Modülleri

İnitsr(): CURSES modülünü kullanan tüm programların en başında

Bu proje için "C" diliyle yazılarak geliştirilmiş olan yazılım modülleri CURSES, COMM, DATAMAN, DIALOG, TERMEMUL, XMODEM, IRQUTIL, DISPLYIO, SCRNIIO, FIELDIO, COMMENU ve UMCOMMONdur. Bu bölümde, bu modüllerin ne işler yaptıklarına kısaca birer göz atacağız.

endwin(): Bu alt program da, CURSES kullanan programların en son

2.1.2.1 CURSES

raww(): Bu modül, tamamen UNIX işletim sisteminde kullanılan Curses

modülünün DOS işletim sistemine aktarılmış halidir. Curses modülü, "C" dilinde programlama yapan bir programcının, ekran düzenlemelerini istediği biçimde yapabilmesini kolaylaştıran alt programları içeren bir modüldür. Bu yüzden, bu projedeki ekran düzenlerinin bu modül ile yapılmasına karar verilmiş ve MS-DOS işletim sistemi altında çalışan bir PC üzerinde kullanılabilecek bir CURSES modülü yazılmıştır. Bu modül sayesinde ekranda çeşitli pencereler açılabilmekte, bu pencerelere ayrı ayrı erişilebilmekte ve içlerine istenen renkte ve özellikte karakterler yazılabilmekte, bu pencerelerin ekran üzerindeki yerleri değiştirilebilmekte ve pencereler üst üste çakıştırılabilmektedir. Bu modülün en önemli özelliklerinden birisi de, renkli ekranda çalıştırılacağı düşünülen yazılımların tek renkli ekran kartına sahip makinelerde çalıştırılması durumunda,

renklide elde edilecek görüntüye en yakın görüntüyü elde etmeye çalışan bir yazılım olmasıdır. `underilen pencereye yazar.`

`mvwa` Şimdi bu modülün içerdiği bazı alt programlara göz atalım:

initscr(): CURSES modülünü kullanan tüm programların en başında mutlaka bir kez çağırılmalıdır. Bu alt program, CURSES modülündeki diğer bir takım alt programların kullandığı ortak parametreleri başlangıç değerine getirir ve ekranı temizler.

endwin(): Bu alt program da, CURSES kullanan programların en son çağırması gereken alt programdır.

newwin(): Yeni bir pencere açar. Bu pencerenin ekran üzerinde nerede olacağı ve ne genişlikte açılacağı, bu alt programa parametre olarak verilmelidir. Yeni açılan pencere, `wrefresh()` altprogramı kullanılarak çağırılana kadar ekranda gösterilmeyecektir.

2.1.2.2 waddch(): İstenen pencereye istenen karakterin yazılmasını sağlar.

Pencere ve karakter değerleri bu alt programa parametre olarak gönderilir. Karakter, bu pencerenin imlecinin bulunduğu yere yazılır.

CURSES Bu imlecin yeri `wmove()` alt programı ile değiştirilebilir. Eklenen karakterler `wrefresh()` çağırılana kadar gösterilmeyecektir.

wmove(): İmlecin pencere içerisindeki yerini değiştirir.

waddstr(): Parametre olarak gönderilen karakter katarını yine parametre olarak gönderilen pencereye yazar.

mvwaddch(),mvwaddstr(): Bu altprogramlar waddch() ve waddstr() ile aynı işi görür. Tek fark, bu alt programların, karakteri veya katarı yazmadan önce, imleci istenen yere taşımalarıdır.

wrefresh(): Bu alt program, parametre olarak gönderilen pencerenin en son halinin ekranda gösterilmesini sağlar.

wattrset(): Bu alt program yardımıyla, pencerelere istenen renkte yazı yazdırılabilir. Bu altprograma pencere ve renk bilgisi parametre olarak gönderilir. Artık bundan sonra bu pencereye yazılan karakterler bu renkle yazılacaktır.

CURSES modülü daha birçok alt program içermesine rağmen en çok kullanılan alt programlar yukarıda açıklanmıştır.

2.1.2.2 DISPLYO : Bir önceki alt program gibidir, tek farkı karakterleri sağa yazmasıdır.

Bu modül, CURSES modülünün üzerine bir katman olarak yazılmıştır.

2.1 CURSES alt programlarını kullanan fakat daha genelleştirilmiş ekrana yazma ve ekrandan okuma alt programlarından oluşur. Bu alt programlardan bazıları aşağıda kısaca açıklanmıştır.

olarak belirlenmiş veri tipleri için ekrandan kullanıcı tarafından veri girişini sağlamak amacıyla yazılmıştır. Bu veri tipleri Alfanümerik, Nümerik, Tarih ve Saat olarak belirlenmiştir. Zamanla yeni veri tipleri de

wtowin(): İstenen pencereye istenen karakter dizisini yazar. Ancak, eğer karakter dizisinin uzunluğu ekranda yazılmak istenen alandan büyükse, karakter dizisinin sadece sığan kısmı yazılır. Tersine, eğer karakter dizisinin uzunluğu alandan daha kısa ise, alanın boş kalan kısımları o anki renkte boşluk karakterleriyle doldurulur. Karakter dizisi, pencere ve alan uzunluğu ve yeri, bu alt programa parametre olarak gönderilir.

rfromwin(): Bu alt program ise, belirtilen pencerenin belirtilen bölümündeki karakterleri parametre olarak gönderilen karakter dizisinin içine yerleştirir. Yani ekrandan geri okuma işi yapar.

set_color(): İstenen pencerenin istenen bölgesinin rengini değiştirir. Bu bölgedeki karakterlerde herhangi bir değişiklik yapmaz.

wljustify(): Bu alt program, istenen bir pencerenin istenen bölümündeki karakterleri sol tarafa doğru yaslar.

wrjustify(): Bir önceki alt program gibidir, tek farkı karakterleri sağa yaslamasıdır.

2.1.2.3 FIELDIO

Bu modül, genel olarak belirlenmiş veri tipleri için ekrandan kullanıcı tarafından veri girişini sağlamak amacıyla yazılmıştır. Bu veri tipleri Alfanümerik, Nümerik, Tarih ve Saat olarak belirlenmiştir. Zamanla yeni veri tipleri de

2.1 eklenebilir. Bu herbir veri tipi için özel birer giriş alt programı yazılmıştır. Genel olarak bu altprogramlarda kullanıcı karakterleri girebilmekte, silebilmekte, imleç hareket tuşlarını kullanarak verinin üzerinde dolaşabilmektedir.

bilgi gösterme modülüdür. Bu modülü kullanan bir programcının, çeşitli alanlardan

Veri tipleri bazındaki bu alt programların yanında, eğer girilmesi beklenen verinin bir listenin içinden seçilmesi mümkünse, bu seçme işini yerine getirecek altprogramlar da yazılmıştır ve kullanılabilir. Bu imkanın olup olmadığı, bu modülün alt programlarına parametre olarak gönderilen veri yapılarında programcı tarafından belirtilmektedir. Bu veri yapılarında programcı ayrıca bazı alanları sadece gösterime açarak buralara veri girilmesini de engelleyebilmektedir.

screen_input() alt programına girildikten sonra artık kullanıcı veri alanları için

Giriş alanının penceresini, pencere içindeki yerini ve uzunluğunu, ayrıca bu alanın veri tipi gibi özelliklerini içeren bir veri yapısı bu modülün genel giriş alt programı olan **field_input()** altprogramına gönderilmekte ve bu altprogram, verinin tipine tekabül eden ilgili alt programı gerekli parametrelerle çağırmaktadır. Bununla aynı mantıkta çalışan fakat alanların ekranda gösterilmesini sağlayan **field_display()** isimli genel bir alan gösterme ve onun kullandığı her veri tipi için ayrı birer alan gösterme alt programı da bu modül kapsamındadırlar. kalınan yerden devam edilir.

Bu projede de görüldüğü gibi bu modül, SCRNIIO modülü tarafından kullanılır ve genel olarak direkt kullanım için tasarlanmamıştır.

2.1.2.4 SCRNIO

SCRNIO modülü artık en genelleşmiş, ekrandan bilgi alma ve ekranda bilgi gösterme modülüdür. Bu modülü kullanan bir programcının, çeşitli alanlardan oluşan bir bilginin ekrandan girilmesini sağlamak için yazacağı program, sadece alanların özelliklerini ve yerlerini, sayılarını belirten bir veri yapısını doldurmak ve bu modülün genel giriş alt programı olan **screen_input()** alt programını çağırmaktan ibaret olacaktır. Aynı şekilde, bu bilgi topluluklarının ekranda gösterilmesini sağlayan bir de **screen_display()** isimli gösterme alt programı bulunmaktadır.

2.1.2.6 DATAMAN

screen_input() alt programına girildikten sonra artık kullanıcı veri alanları içinde istediği gibi dolaşabilir ve değiştirmesine izin verilen alanlarda değişiklik yapabilir. Bu alt programdan, programcı tarafından belirlenen başka bir alt programa geçilmesi mümkündür. Bunun için kullanıcının, bu alt programa programcı tarafından parametre olarak gönderilen özel tuşlardan birine basması gerekmektedir. Bu durumda çağırılan alt program (callback procedure) da, **screen_input()** alt programına gönderilen parametrelerden biridir. Bu yeni alt programdan geri döndürülen değere göre ya **screen_input()** alt programından çıkılır ya da işleme kalınan yerden devam edilir.

2.1.2.5 DIALOG

Kullanıcıyla sürekli konuşma halinde olan bir program geliştirirker karşılaşılan ihtiyaçlardan birisi de, kullanıcıya hata mesajlarının gösterilmesinde, kısa sorular sorulmasında ve bunların cevaplarının alınmasında ortak bir yöntemin belirlenmesidir. Mesala her iki durumda da ekrana yazılan mesajlar çerçeveye alınabilir ve hata mesajlarıyla soru cümlelerinin çerçeve ve kutu renkleri birbirinden farklı olabilir. Bu projede bu yöntem kullanılmıştır ve bu mesaj işlemleri için DIALOG modülü yazılmıştır.

2.1.2.6 DATAMAN

Bu modül, projenin genel veri işleme (Data Manipulation) modülüdür. Alfaniümerik verilerin sonlarındaki boşluk karakterlerinin ayıklanması (**rmtrailblanks()**), nümerik verilerin ondalık noktası ve sonrasındaki hanelerin düzenlenmesi (**arrange_num()**), tarih ve saat tipi verilerin doğruluğunun kontrol edilmesi (**check_date()**, **check_time()**) işlerine yarayan alt programlardan oluşmaktadır.

2.1.2.7 IRQUTIL

UMICOMM programı Interrupt driven çalışan bir programdır. Bunun anlamı, RS-232C arabirimine bir karakter geldiğinde, bunun mikroişlemciye üretilen bir interrupt ile bildirilmesi ve bunun üzerine interrupt alt programının çağırılmasıdır. Bu durumda, zaman zaman, UART'ın interrupt üretmesini engellemek veya tekrar sağlamak için sistemdeki 8259 programlanabilir interrupt

2.1. kontrolörünü programlamak gerekmiştir. İstenen interrupt hattının interrupt üretmesini sağlamak için **enable_irq()** ve engellemek için de **disable_irq()** altprogramları yazılarak, daha sonra 8259 ile ilgili yeni alt programların da yazılabileceği düşünülerek IRQUTIL modülü oluşturulmuştur.

2.1.2.8 COMM

Bu modül, PC lerin kendi firmware ROMlarında bulunan hazır alt programları kullanacak şekilde tasarlanmış bir haberleşme modülüdür. İçeriği, haberleşme portlarının başlangıç değerlerinin şartlanmasına, bu portlardan bilgi gönderilmesi ve alınmasına yarayan alt programlardan oluşmaktadır. Ayrıca, bir haberleşme portunda okunmaya hazır bilgi olup olmadığını sezinlemeye yarayan bir alt programı da içermektedir. PCnin kendi firmwareindeki alt programların kullanılmasında bazı problemler olduğundan, UMICOMM paketi bu alt programların yeniden yazılmış yeni kopyalarını kullanmaktadır ve bu alt programlar UMCOMMON modülündedir.

2.1.2.9 TERMEMUL

Bu modül, terminal emülasyonu modülü olarak tasarlanmıştır. Digital VT-52 ve VT-100 ile ADDS Viewpoint terminalleri için gerekli terminal emülasyonu tuştakımı ve kontrol karakterleri emülasyonu alt programlarını içerir. Bu modülde yapı, daha sonra yeni emülasyon modlarının da eklenebilmesine olanak verecek biçimde geliştirilmiştir. UMICOMM programı, bir tuşa basıldığında veya porttan bir karakter alındığında, bunları, hangi emülasyon modu kullanılıyor ise ona ilişkin klavye veya ekran alt programına gönderir.

2.1.2.10 XMODEM

Bu modül, XMODEM protokolü ile dosya transferi modülüdür. Bu modüldeki alt programların işlevleri XMODEM protokolünün açıklamasıyla beraber aşağıda anlatılmıştır.

XMODEM, dosyaların içlerindeki bilgilerin bloklar halinde transfer edildiği hata düzeltmeli bir protokoldür. Her veri paketinin uzunluğu 133 karakterdir. İlk karakter, veri paketinin cinsini belirtmek için veya komut karakteri olarak kullanılır.

İlk byte 01 ise, bu bir veri paketidir anlamına gelmektedir. Bunu izleyen iki bytedan birincisi blok numarası, ikincisi ise blok numarasının ikilik tümleyeni olmalıdır. Böylece alan tarafta blok numarasının doğru olup olmadığı kontrol edilebilir. Bunu 128 adet veri bytei izler. Bunların hemen ardından da iki byte'lık FCS gelir.

Alan taraf, 128 adet veri bytei üzerinden FCS in doğru olup olmadığını kontrol eder. FCS doğru ise, gönderen tarafa ACK (acknowledgement) karakteri ile cevap verir ve gönderen taraf yeni bir veri bloğu gönderir. FCS yanlış ise, alan taraf gönderen tarafa NAK (negative acknowledgement) karakteri gönderir ve gönderen taraf bu veri bloğunu tekrar gönderir. Bu tekrar göndermelerin sayısı sınırlanmıştır. Bir veri paketinin formatı Şekil 2.1 de görülmektedir.

Dosyanın sonuna gelindiğinde gönderen taraf alan tarafa EOT (end of transmission) karakteri gönderir ve alan tarafın da buna ACK ile cevap

vermesiyle dosya transferi modundan çıkılır. Ayrıca, gönderen ya da alan taraflardan herhangi biri, bloklar arasında CAN (Cancel) karakteri göndererek transferi kesebilir.

Transferin ilk başında, gönderen taraf, veri bloğunu göndermeye başlamadan önce alan taraftan "C" veya NAK karakterinin gelmesini beklemelidir.

XMODEM modülünde hatalı olayların düzeltilmesi için belirlenmiş tekrar deneme adedi 10 dur. Karşı taraftan bir cevap gelmesini bekleme süresi ise 3 sn olarak belirlenmiştir. Bu her iki limitte aşıldığında transfer modundan çıkılır.

Xmodem protokolüne göre veri bloğu gönderme işini **XMsenddata()**, alma işini **XMreceivedata()** alt programları yerine getirmektedir. Baştan sona bir dosyayı gönderen alt program **XMxmitfile()** ve alan alt program da **XMreceivefile()** alt programıdır.

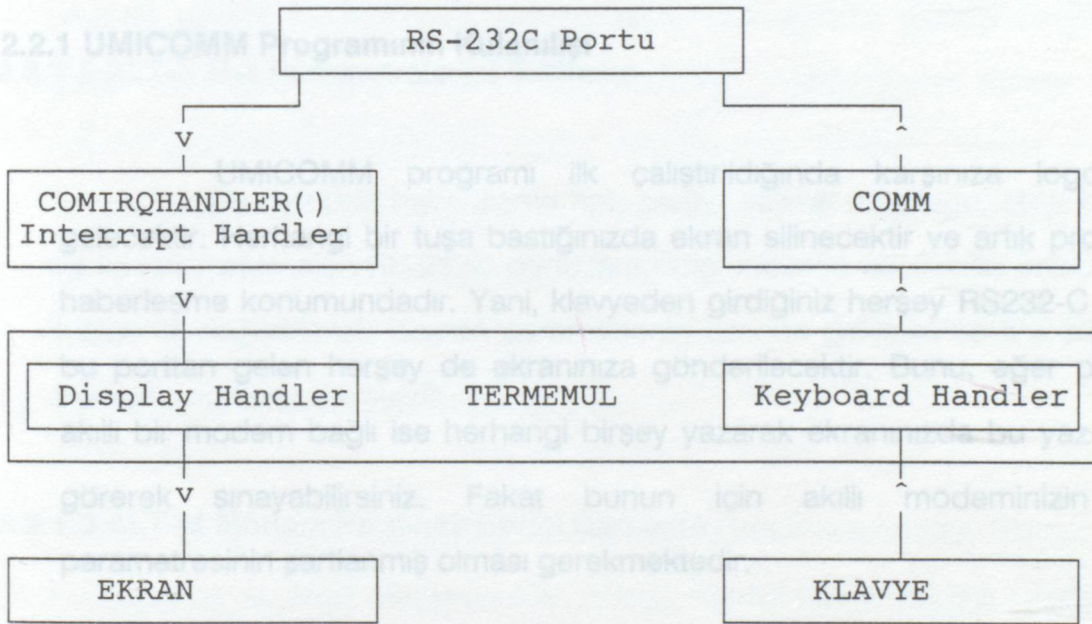
ACK
NAK
EOT
CAN

01	Blok No	Blok No Tümleyen	128 byte Veri	2 byte FCS
----	---------	------------------	---------------	------------

Şekil 2.1: XMODEM Kontrol Karakterleri ve Veri Paketi Formatı

2.1.2.11 UMCOMMON

Bu modül hem UMICOMM hem de UMISRV programları tarafından ortak kullanılan bir modüldür ve genel haberleşme alt programlarıyla, interrupt ve keyboard handler alt programlarını da içermektedir. Bunun yanında parametre şartlama ekranlarının veri yapılarından, yardım alt programına ve telefon numarası arama defterine kadar ortak olan herşey buradadır. Şekil 2.2, UMICOMM programındaki genel veri akışını sembolik olarak göstermektedir.



Şekil 2.2: UMICOMM içinde Veri Akışı

2.1.2.12 COMMENU

Bu modül sadece UMISRV programı tarafından kullanılır ve uzak kullanıcının RS-232C portu üzerinden, ardışıl menülerle yakın sistemdeki verilere erişebilmesini sağlar. Bu modülün kullandığı veri dosyalarının adları ve formatları

Bölüm 2.2.2 UMISRV Programının Kullanılışı başlığında bulunabilir.

2.2 Programların Kullanılışları

2.2.1 UMICOMM Programının Kullanılışı

UMICOMM programı ilk çalıştırıldığında karşınıza logo ekranı gelecektir. Herhangi bir tuşa bastığınızda ekran silinecektir ve artık programınız haberleşme konumundadır. Yani, klavyeden girdiğiniz herşey RS232-C portuna, bu porttan gelen herşey de ekranınıza gönderilecektir. Bunu, eğer portunuza akıllı bir modem bağlı ise herhangi birşey yazarak ekranınızda bu yazdıklarınızı görerek sınavabilirsiniz. Fakat bunun için akıllı modeminizin ECHO parametresinin şartlanmış olması gerekmektedir.

Bu durumda, eğer UMICOMMu modeminize uygun bir haberleşme hızına şartlanmış iseniz, modemin numara çevirme komutlarını girerek istediğiniz yerle bağlantı kurabilirsiniz. Ancak, ileride anlatılacağı gibi, bu işi UMICOMM zaten sizin için yapabilmektedir.

UMICOMM çalışırken herhangi bir zaman modem, terminal veya hat parametrelerini şartlama menülerine gidebilir ve ESC tuşu ile bu menülerden geri

dönebilirsiniz. Bunun için ALT tuşuyla beraber bu menülerin aktive etme tuşlarına basmalısınız. Eğer bu tuşları unutmuş veya bilmiyor iseniz, ALT-H tuşlarına bastığınızda UMICOMM size bütün tuş olasılıklarını ve ne işe yaradıklarını gösterecektir. Bunları burada kısaca görelim.

2.2.1.1 ALT-H Yardım

Yardım ekranı getirilir ve kullanıcıya hangi tuşların ne işe yaradığı gösterilir.

2.2.1.2 ALT-S Hat Parametrelerini Şartlama

Bu menüde haberleşme hızı, parity, kelime uzunluğu, stop biti sayısı ve kullanılmakta olan RS-232C portu istenildiği biçimde seçilebilir. İmleç hareket tuşları ile değiştirilmek istenen parametrenin üzerine gidilmeli ve ara çubuğu ile istenen değere getirilmelidir.

2.2.1.3 ALT-M Modem Parametrelerini Şartlama

Bu menüde, programın ilk başında modemi şartlamak için modeme gönderilen karakter dizisi, modeme numara çevirtmek için gereken komut dizisi, modemin bağlantı kurmaya çalışırken karşılaştığı meşgul, çevir sesi yok ve taşıyıcı yok gibi durumlarda bilgisayara gönderdiği karakter dizileri ile bağlantının kurulması anında gönderdiği karakter dizileri şartlanır. Bu şartlamalar doğru yapıldığında artık akıllı modem ile UMICOMM programı anlaşabilir hale

geleceklerdir. Akıllı modemlerde genellikle HAYES komut seti kullanılır. Bu setin komutları hakkında kısa açıklamaları eklerde bulabilirsiniz.

2.2.1.4 ALT-T Terminal Emulasyonu Şartlama

Bu menüde UMICOMMun hangi terminal tipini taklit edeceğini, CR karakterlerini aldığında bunlara line feed (LF) ekleyip eklemeyeceğini, gönderirken CR lere LF ekleyip eklemeyeceğini, silme (BS=BACKSPACE) karakteri geldiğinde sadece imleci mi kaydıracağını yoksa hem imleci kaydırıp hem de soldaki karakteri silip silmeyeceğini, satır sonuna geldiğinde yeni satıra geçip geçmeyeceğini, ekranın son satırına geldiğinde ekranı kaydırıp kaydırmayacağını, ekrandaki satır sayısını ve düdük sesinin kullanılıp kullanılmayacağını seçebilirsiniz. Ayrıca bu menüden el sıkışma yönteminin XON-XOFF mu DTR mı olduğunu da şartlayabilirsiniz.

2.2.1.5 ALT-D Arama Yapma (Numara Defteri)

Bu bölüm, tamamen bir telefon defteri bölümüdür. Sıkça bağlantı kurduğunuz yerlerin isimlerini ve telefon numaralarını buraya yazabilirsiniz. Bağlantı kurmak istediğiniz isimin veya numaranın üstüne gidip tekrar ALT-D tuşlarına bastığınızda, arama süreci UMICOMM tarafından otomatik olarak yerine getirilecek ve bağlantı kurulacaktır.

2.2.1.6 ALT-K Şartlamaları Saklama

2.2.1.9 PGDN Aslında her şartlama menüsünden çıkarken yaptığınız değişikliklerin saklanmasını isteyip istemediğiniz sorulmaktadır. Ancak burada sadece girdiğiniz menüdeki parametreler saklanmaktadır. Bütün parametre şartlamalarını saklamak isterseniz ALT-K tuşlarına basmalısınız.

2.2.1.7 ALT-X Programdan Çıkma

2.2.2 UMISHV Programdan çıkmak istediğiniz herhangi bir anda ALT-X tuşlarına basınız. Program emin olup olmadığını soracak ve eğer olumlu yanıt vererseniz programdan çıkılacaktır.

2.2.1.8 ALT-B BREAK Gönderme

BREAK göndermek demek, RS-232C arabiriminin gönderme çıkışı, bir karakter için gereken süreden daha uzun bir süre aktif tutmak demektir. Bu sayede alan taraf, gönderen taraftan bir kesme isteği geldiğini anlar. Bu işaret UNIX işletim sistemiyle çalışan bilgisayarlarda kullanılmaktadır. UMICOMMun BREAK göndermesi için ALT-B tuşlarına basmalısınız.

2.2.1.9 PGUP XMODEM ile Dosya Gönderme

UMICOMMu kullanırken XMODEM protokolü ile dosya göndermek için PageUp tuşuna basmalısınız. Bu durumda gönderilecek olan dosyanın adı sorulacaktır ve bu verildikten sonra transfer başlayacaktır.

2.2.1.9 PGDN XMODEM ile Dosya Alma

şifresiz erişim yapabilsin isteniyorsa "." karakterinden sonra herhangi birşey yazılmaz. Eğer açıklayıcı satırlar konmak isteniyorsa UMICOMMu kullanırken XMODEM protokolü ile dosya almak için PageDown tuşuna basmalısınız. Bu durumda uzaktan alınacak olan dosyanın yerel PCde yerleştirileceği dosya adı sorulacaktır ve bu verildikten sonra transfer başlayacaktır.

2.2.2 UMISRV Programının Kullanılışı

sahip her kullanıcının önüne gelecek olan ilk menüyü tanımlar. Kullanılan ortak menü dosyası formatı şöyledir:

Bu programın kullanılışındaki şartlamalar tamamen UMICOMM ile aynı yöntemlerle yapılır. Zaten kendi başına bırakıldığında herhangi bir uzak kullanıcının sisteme bağlanmasını bekleyecek ve bunu sezinlediğinde, güvenlik kontrolü işleminden sonra bu kullanıcıya hizmet vermeye başlayacaktır. Doldurulması gereken dosyalar ve formatları aşağıda açıklanmıştır.

2.2.2.1 BANNER.FIL Dosyası

adi, yine "." ve seçeneğin açıklamasını oluşturan karakter dizisi olmalıdır. Eğer işlem türü karakteri "S" ise, verilen dosya adı yine bu ortak Bu dosya, bağlantı kuran herkese ilk başta gösterilen bir dosyadır. Bu yüzden, bilgi servisinin adını ve bu servise nasıl abone olunacağını belirten satırlara sahip olması mantıklıdır. Sistemin yöneticisi bu dosyayı herhangi bir editör ile istediği gibi değiştirebilir.

2.2.2.2 PASSWD.FIL Dosyası

Bu dosya sistemdeki bilgilere erişim hakkı olan herkesin kullanıcı adını ve varsa şifrelerini içerir. Her satır, birbirinden ":" ile ayrılmış kullanıcı adı ve

şifreden oluşmaktadır. Eğer kullanıcı şifresiz erişim yapabilsin isteniyorsa ":" karakterinden sonra herhangi birşey yazılmaz. Eğer açıklayıcı satırlar konmak isteniyorsa bu satırlar "#" karakteri ile başlamalıdır. Bu durumda bu satırlar şifre aranırken işleme katılmayacaklardır.

2.2.2.3 MAINMENU.MNU Dosyası

Bu dosya, erişim hakkına sahip her kullanıcının önüne gelecek olan ilk menüyü tanımlar. Kullanılan ortak menü dosyası formatı şöyledir.

Dosyanın içinde ":" ile başlamayan her satır ("#" ile başlayan açıklama satırları dışında) kullanıcının ekranına gönderilir. ":" ile başlayan satırlar seçenek satırlarıdır. Her seçenek satırında ":" dan sonra bu seçeneği seçmek için kullanıcının girmesi gereken karakter dizisi, yine ":", alt menü seçenekleri için "S" veya dosya transferi seçenekleri için "X", yine ":" ve alt menü dosyasının veya transfer edilecek dosyanın adı, yine ":" ve seçeneğin açıklamasını oluşturan karakter dizisi olmalıdır. Eğer işlem türü karakteri "S" ise, verilen dosya adı yine bu ortak menü dosyası formatında yeni bir alt menü dosyasının adı olmalıdır. Kullanıcıya sadece bu seçenek dizisi, işlem türü karakteri ("S" veya "X") ve açıklama gösterilir.

2.2.2.4 Bilgi Servisinden Faydalanma

Bilgi servisinden faydalanmak isteyen kişiye önce kullanıcı adı sorulur. Eğer kullanıcı adı kayıtlı ise ve şifresi yoksa, kullanıcının servisten faydalanma hakkı var demektir ve ana menüye geçilir. Eğer şifresi varsa şifre sorulur ve doğru şifreyi girmesi beklenir. Kullanıcı en fazla beş kez yanlış kullanıcı adı veya şifre girme hakkına sahiptir. Bu limit aşıldığında bağlantı koparılmaktadır.

Ana menüye geçildiğinde kullanıcı ekranda gördüğü seçeneklerden herhangi birini seçebilir. Eğer seçeneğin yanında "S" harfi varsa bu bir alt menüdür. Alt menülerden seçenek olarak "b" girilerek bir önceki menüye, "m" girilerek ana menüye geri dönülebilir. Eğer seçenek olarak "I" girilirse, kullanıcının sistemden ayrılmak istediği anlaşılır ve bağlantı koparılır.

Eğer seçeneğin yanında "X" harfi varsa bu bir dosya transferi seçeneğidir ve kullanıcı, XMODEM dosya transferi moduna geçmesi için uyarılır. Kullanıcının bunu yerine getirmesiyle dosya transfer edilir ve menüye geri dönülür.

Herhangi bir kullanıcının sisteme bağlandıktan sonra sorulan soruları cevapsız bırakması ve bu yüzden bağlanmak isteyen diğer kullanıcıları bloke etmesinin önüne geçilmesi için, kullanıcı belli bir süre (2 dakika) cevap vermezse bağlantı kesilmektedir.


```

**                                     DIALOG.C                                     **
**                                     Dialog Facilities Module                       **
**                                                                                   **
**      Creation : 10/3/1991 by Unit KALEM                                         **
**                                                                                   **
**-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<iomanip.h>

int promptoryesno(question,answer,&answerinfo)
char          *question,*answer,*answerinfo;
COLOR_INFO    *colorinfo;
{
    int         size,posx,posy;
    WINDOW      *dialogwin;
    int         opus_n,opus_y,cevre_x,cevre_y;
                scr_buf[25*80];

    size=strlen(question)+2;
    posx=(CMSG-size)/2;
    posy=(LINES-1 -1)/2;

    keypadon   (3,size,posy,posx,scr_buf);
    getchpos   (&opus_y,&opus_x);
    get_cursor_size (&cevre_x,&cevre_y);
    set_cursor_size (-1,-1);

    dialogwin = newwin(1,size,posy,posx);
    wclear     (dialogwin," " [colorinfo->text_color]);
    wattent    (dialogwin,colorinfo->text_color);
    box        (dialogwin,SWR_XXX,SWR_XXX,SWR_XXX,SWR_XXX,SWR_XXX,SWR_XXX,SWR_XXX);
    wattent    (dialogwin,colorinfo->text_color);
    mvwprintw  (dialogwin,1,1,question);
    wrefresh   (dialogwin);
}

```



```

/*****
*****
**                                     **
** keyrefresh(dialogwin);          DIALOG.C          **
** 15 refresh(answer0,key1->w11_ptr) (key->break);    **
** 15 refresh(answer1, Dialog Facilities Module);    **
** }                                     **
**      Creation : 15/3/1991 by Umit KALKAN          **
** save_scrn (3,size,posy,posx,scr_buff);           **
*****
*****/

dialogwin = newwin(3,size,posy,posx);

#include <stdio.h>
#include < curses.h>
#include < string.h>
#include < nullptr.h>
#include < fieldio.h>

int promptforyesno(question,answer0,answer1,colorinfo)
char      *question,*answer0,*answer1;
COLOR_INFO *colorinfo;
{
    int      size,posx,posy;
    WINDOW   *dialogwin;
    int      cpos_x,cpos_y,csz_b,csz_e,key,
            scr_buff[25*80];

    size=strlen(question)+2;
    posx=(COLS-size)/2;
    posy=(LINES-3 )/2;

    savescrn (3,size,posy,posx,scr_buff);
    getcurpos (&cpos_y,&cpos_x);
    get_cursor_size (&csz_b,&csz_e);
    set_cursor_size (-1,-1);

    dialogwin = newwin(3,size,posy,posx);
    fillwin (dialogwin,' '|colorinfo->text_color);
    wattrset (dialogwin,colorinfo->box_color);
    box (dialogwin,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);
    wattrset (dialogwin,colorinfo->text_color);
    mvwaddstr (dialogwin,1,1,question);
    wrefresh (dialogwin);

```



```

comsendok(port,chr);
/*****
****
**  (comsendok(port)) (1);
**  bioscom(COM_WRITE,chr,port,COMM.C
**
**
**      Communication Module
**
**
**  Creation : 15/3/1991 by Umit KALKAN
**
****
****
*****/
return (retc);
#include <stdio.h>
#include <bios.h>
#include <comm.h>

char  comhit(port)
int   port;
{
    char  retc;
    if ( bioscom(COM_GETSTAT,'Z',port) & COM_DATAREADY ) return (1);
    return (0);
}

char  comsendok(port)
int   port;
{
    char  retc;
    if ( (bioscom(COM_GETSTAT,'Z',port) &
          (COM_CD|COM_DSR)) == (COM_CD|COM_DSR) )
    {
        retc=1;
    }
    else
    {
        retc=0;
    }
    return (retc);
}

```



```

comsendchr(port,chr)
{
    char chr,port;
    while (!comsendok(port)) {}
    bioscom(COM_SNDCHAR,chr,port);
}

comreceivechr(port)
{
    char port;
    char retc=(bioscom(COM_RCVCHAR,' ',port)&0x00FF);
    return (retc);
}

#define INT_CTRLS

comsendstr(port,buff)
{
    char port,*buff;
    {
        char chr,*ptr=buff;
        while ((chr=*ptr++)!=NULL) comsendchr(port,chr);
    }
}

mask_irq(irqnum)
{
    char irqnum;
    {
        disable();
        outportb(INT_CTRLS+1,0x0000);
        enable();
    }
}

```



```

/*****
*****
**
**                               **
**          IRQUTIL.C          **
**                               **
**      Interrupt Request Parameters Manipulation Modeule      **
**                               **
**      Creation : 15/3/1991 by Umit KALKAN                      **
**                               **
*****
*****/

```

```
#include <stdio.h>
```

```
#include <dos.h>
```

```
#define          INT_CTRLR      0x020
```

```

unmask_irq(irqnum)
    char  irqnum;
    {
        disable();
        outportb(INT_CTRLR+1, ((inportb(INT_CTRLR+1)) & ~(1<<irqnum)));
        enable();
    }

```

```

mask_irq(irqnum)
    char  irqnum;
    {
        disable();
        outportb(INT_CTRLR+1, ((inportb(INT_CTRLR+1)) | (1<<irqnum)));
        enable();
    }

```



```

/*****
*****
** chr = COMPOSITECHR(); **
** DISPLAYNAME IN (chr); **
**
** Umicomm Main Program **
**
** Creation : 15/3/1991 by Umit KALKAN **
**
*****/

```

```

#include<stdio.h>
#include<conio.h>
#include<bios.h>
#include<dos.h>
#include<fcntl.h>
#include<stat.h>
#include<io.h>
#include<string.h>
#include<nullptr.h>
#include<alloc.h>
#include<curses.h>
#include<comm.h>
#include<scrnio.h>

```

```

#define DATAFILE "umicomm.dat"

```

```

/*****
*****
***** COMMON DEFINITIONS & ROUTINES *****
*****
*****/

```

```

#include<umcommon.c>

```

```

/*****
*.....MAIN.....*
*****/

```

```

main ()
{
    int chr=0;
    initscr ();
    prog_init ();
    prog_reset();
}

```



```

while (1)
{
  chr = COMRECEIVECHR();
  DISPLAYHANDLER(chr);
}

}

/*
 * Creation : 12/2/1991 by Galt KALSHN
 */

#define MAX_MENU_FILESIZE 51024
#define MAX_MENU_LEVEL 100

#define SEL_KEY 1
#define STOP_KEY 2
#define FNAME_KEY 4
#define COMMAND_KEY 16
#define SUBMENU_INDICATOR 'S'
#define FILENAME_INDICATOR 'F'

#define MENU_TIMEOUT_INTERVAL 30

char *MENU_EXIT_MESSAGE = "b";
char *MENU_MAIN_MESSAGE = "M";
char *MENU_EXIT_MESSAGE = "l";

char *MENU_PROMPT_STR = "\n\nEnter Your Selection\n\n ('b'-previous menu 'M'-main menu 'l'-logout) : ";
char *MENU_FILESIZE_ERR_STR = "File Transfer Aborted as an Error Occured.\n\n";
char *MENU_FILESIZE_OK_STR = "File Transfer Succeeded.\n\n";
char *MENU_INVALID_SEL_ERR = "Invalid Selection.\n\n";
char *MENU_PREPARE_FILESTR = "Get prepared for XMODEM DOWNLOAD.\n\n";
char *MENU_RESPONSE_TIMEOUT_ERR = "XMODEM timed out to Respond.\n\n";
char *MENU_CONTINUE_STR = "\n\nPress ANYKEY to Continue..";

int MENU_LEVEL=0;

```



```

/*****
*****
** Remote Menu System Modeule
**
** Creation : 15/3/1991 by Umit KALKAN
**
*****/

while (file=fopen(FILENAME, "r") != NULL)
{
#define MAX_MENU_FILESIZE      5*1024
#define MAX_NOOF_LINES (MAX_MENU_FILESIZE/sizeof(char))

#define SEL_BEG                1
#define TYPE_BEG               4
#define FNAME_BEG              6
#define COMMENT_BEG           19
#define SUBMENU_INDICATOR      'S'
#define FILEXFER_INDICATOR     'X'

#define MENU_TIMEOUT_INTERVAL   60

char *MENU_BACK_ANSWER        ="b";
char *MENU_MAIN_ANSWER        ="m";
char *MENU_EXIT_ANSWER        ="l";

char *MENU_PROMPT_STR          ="\r\nEnter Your Selection\r\n ('b'-previous menu  'm'-main menu  'l'-
logout) : ";
char *MENU_FILEXFER_BAD_STR    ="File Transfer Aborted or an Error Occured!.\r\n";
char *MENU_FILEXFER_OK_STR     ="File Transfer Succeeded!.\r\n";
char *MENU_INVALID_SEL_STR     ="Invalid Selection!.\r\n";
char *MENU_PREPARE_FILEXFER_STR="Get prepared for XMODEM DOWNLOAD!.\r\n";
char *MENU_RESPONSE_TIMEOUT_STR="\r\nUSER Timed out to Respond!.\r\n";
char *MENU_CONTINUE_STR       =" \r\nPress NEWLINE to Continue..";

int MENU_LEVEL=0;

```



```

/*****
*.....READ FILE.....*
*****/

int readfile(filename, buff, maxlen)
char *filename, *buff;
long maxlen;
{
    int handle, filesize=0, amount;
    if ((handle=open(filename, O_RDONLY|O_TEXT))!=-1)
    {
        while (filesize<MAX_MENU_FILESIZE && filesize<maxlen)
        {
            if ((amount=read(handle, buff, 512))>0)
            {
                buff+=amount;
                filesize+=amount;
            }
            else
            {
                break;
            }
        }
        *buff = (char)NULL;
        close (handle);
        return (filesize);
    }
    return(-1);
}

/*****
*.....CLEAR SCREEN.....*
*****/

comclrscrn()
{
    int cntr=0;
    while (cntr<30)
    {
        _comsendstr(PORT, NEW_LN_STR);
        cntr++;
    }
}

```



```

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

returns:
    0: Go to previous
    1: Go to main
    2: Logout
   -1: Malloc or file error
   -2: Response timeout
*****/

int commenu(menufilename)
char *menufilename;
{
    char *buffer,*ptr,nooflines=0,lncntr,
        selfilename[LINE_BUFF_LEN],menuanswer[LINE_BUFF_LEN],
        *lines[MAX_NOOF_LINES];
    int rc,found;
    if ((buffer=malloc(MAX_MENU_FILESIZE+1))!=NULL_PTR)
    {
        if (readfile(menufilename,buffer,MAX_MENU_FILESIZE)>0)
        {
            ptr=buffer;
            while (nooflines<MAX_NOOF_LINES)
            {
                lines[nooflines]=ptr;
                while (*ptr!=(char)NULL && *ptr!=CR && *ptr!=LF) ptr++;
                if (*ptr==(char)NULL) {nooflines++;break;}
                *ptr = (char)NULL;
                ptr++;
                nooflines++;
            }

            while (1)
            {
                found=0;
                comclrscrn();
                lncntr=0;
                while(lncntr<nooflines)
                {
                    if (lines[lncntr][0]!=COMMENT_INDICATOR)
                    {
                        if (lines[lncntr][0]!=FIELD_SEPARATOR)
                        {
                            _comsendstr(PORT,lines[lncntr]);
                            _comsendstr(PORT,NEW_LN_STR);
                        }
                        else
                        {
                            _comsendstr(PORT," ");
                            _comsendchr(PORT,lines[lncntr][SEL_BEG]);
                            _comsendchr(PORT,lines[lncntr][SEL_BEG+1]);
                            _comsendstr(PORT," ");
                        }
                    }
                }
            }
        }
    }
}

```



```

        _comsendchr(PORT, lines[lncntr][TYPE_BEG]);
        _comsendstr(PORT, " ");
        _comsendstr(PORT, &(lines[lncntr][COMMENT_BEG]));
        _comsendstr(PORT, NEW_LN_STR);
    }
}
lncntr++;
}
_comsendstr(PORT, MENU_PROMPT_STR);
if (waitevent(MENU_TIMEOUT_INTERVAL, COMHIT) != 1)
{
    _comsendstr(PORT, MENU_RESPONSE_TIMEOUT_STR);
    free(buffer);
    return(-2);
}
_comreadln (PORT, menuanswer, 0);
_comsendstr(PORT, NEW_LN_STR);
if (strcmp(menuanswer, MENU_BACK_ANSWER) == 0)
{
    if (MENU_LEVEL != 0)
    {
        free(buffer);
        return(0);
    }
    found = 1;
}
if (strcmp(menuanswer, MENU_MAIN_ANSWER) == 0)
{
    if (MENU_LEVEL != 0)
    {
        free(buffer);
        return(1);
    }
    found = 1;
}
if (strcmp(menuanswer, MENU_EXIT_ANSWER) == 0)
{
    free(buffer);
    return(2);
}

lncntr = 0;
while (lncntr < nooflines)
{
    if (lines[lncntr][0] == FIELD_SEPARATOR)
    {
        if (strncmp(&(lines[lncntr][SEL_BEG]), menuanswer, 2) == 0)
        {
            selfilename[0] = (char) NULL;
            strncpy(selfilename, &(lines[lncntr][FNAME_BEG]), 12);
            selfilename[12] = (char) NULL;
            sscanf(selfilename, "%s", selfilename);
            switch(lines[lncntr][TYPE_BEG])

```



```

{
case SUBMENU_INDICATOR:
    found=1;
    MENU_LEVEL++;
    rc=commenu(selffilename);
    MENU_LEVEL--;
    if (rc!=0)
    {
        if (rc!=1 || MENU_LEVEL!=0)
        {
            free(buffer);
            return(rc);
        }
    }
    break;
case FILEXFER_INDICATOR:
    found=1;
    _comsendstr(PORT,MENU_PREPARE_FILEXFER_STR);
    if (XMxmitfile(selffilename)==0)
    {
        _comsendstr(PORT,MENU_FILEXFER_OK_STR);
    }
    else
    {
        _comsendstr(PORT,MENU_FILEXFER_BAD_STR);
    }
    _comsendstr(PORT,MENU_CONTINUE_STR);
    if (waitevent(MENU_TIMEOUT_INTERVAL,COMHIT)!=1)
    {
        _comsendstr(PORT,MENU_RESPONSE_TIMEOUT_STR);
        free(buffer);
        return(-2);
    }
    _comreadln(PORT,menuanswer,0);
}
}
lnctr++;
}
if (!found) _comsendstr(PORT,MENU_INVALID_SEL_STR);
}
}
free(buffer);
}
return(-1);
}

```



```

/*****
*****
**      pos_y,pos_x,len;                               **
**      DISPLAYIO.C                                   **
**                                                     **
**      Display Input Output Module                   **
**      (win=NULL_PTR && pos_y==0 && pos_x==0 && len==0 && **
**      pos_y Creation : 15/3/1991 by Umit KALKAN :=NULL_PTR) **
**                                                     **
*****
*****/

#include<stdio.h>
#include<curses.h>
#include<nullptr.h>

wtowin (win,pos_y,pos_x,len,str)
WINDOW *win;
int pos_y,pos_x,len;
char *str;
{
    int lin,beg,len,color;
    char *ptr=str,past_end=0,cntr;
    if (win!=NULL_PTR && pos_y>=0 && pos_x>=0 && len>=0 &&
        pos_y<win->_maxy && pos_x<win->_maxx)
    {
        if (str!=NULL_PTR)
        {
            for (cntr=pos_x;cntr<win->_maxx&&cntr<pos_x+len;cntr++,ptr++)
            if (past_end)
                {mvwaddch(win,pos_y,cntr,' ');}
            else
            {
                if(*ptr==NULL)
                    {past_end=1;cntr--;}
                else
                    {mvwaddch(win,pos_y,cntr,*ptr);}
            }
        }
        else
        {
            for (cntr=0;cntr<len;cntr++) mvwaddch(win,pos_y,pos_x+cntr,' ');
        }
    }
}

```



```

rfromwin (win,pos_y,pos_x,len,str)
WINDOW *win;
int      pos_y,pos_x,len;
char     *str;
{
    char *ptr=str,cntr;
    if (win!=NULL_PTR && pos_y>=0 && pos_x>=0 && len>=0 &&
        pos_y<win->_maxy && pos_x<win->_maxx && str!=NULL_PTR)
    {
        for (cntr=pos_x;cntr<win->_maxx&&cntr<pos_x+len;cntr++,ptr++)
        {
            *ptr=(mvwinch(win,pos_y,cntr)&A_CHARTEXT);
        }
        str[len] = (char) NULL;
    }
}

```

```

set_color(win,lin,beg,len,color)
WINDOW *win;
int      lin,beg,len,color;
{
    int cntr;
    for (cntr=0;cntr<len;cntr++)
    {
        mvwaddch(win,lin,beg+cntr,
            ((mvwinch(win,lin,beg+cntr)&A_CHARTEXT)|color));
    }
}

```

```

wsptozero(win,lin,beg,len)
WINDOW *win;
int      lin,beg,len;
{
    int cntr,chr;
    for (cntr=0;cntr<len;cntr++)
    {
        chr=mvwinch(win,lin,beg+cntr);
        if ( (chr&A_CHARTEXT)==' ' ) waddch(win,'0'|(chr&A_ATTRIBUTE));
    }
}

```



```

wljustify(win,y,x,len)
WINDOW *win;
int      y,x,len;
{
int      scnt,dcnt,chr,found=0;
scnt=dcnt=0;
while (scnt<len)
{
chr=mvwinch(win,y,x+scnt);
if (!found)
{
if ( (chr&A_CHARTEXT)!=' ' )
{
found=1;
continue;
}
}
else
{
mvwaddch(win,y,x+dcnt,chr);
dcnt++;
}
scnt++;
}
for (;dcnt<len;dcnt++)
{
chr=mvwinch(win,y,x+dcnt)&A_ATTRIBUTE;
waddch (win,chr|' ');
}
}

```

```

wrjustify(win,y,x,len)
WINDOW *win;
int      y,x,len;
{
int      scnt,dcnt,chr,found=0;
scnt=dcnt=len-1;
while (scnt>=0)
{
chr=mvwinch(win,y,x+scnt);
if (!found)
{
if ( (chr&A_CHARTEXT)!=' ' )
{
found=1;
continue;
}
}
}
}

```



```

else
{
    mvwaddch(win,y,x+dentr,chr);
    dentr--;
}
sentr--;
}
for (;dentr>=0;dentr--)
{
    chr=mvwinch(win,y,x+dentr)&A_ATTRIBUTE;
    waddch(win,chr|' ');
}
}
#include<stdio.h>
#include<dos.h>
#include<time.h>
#include<unistd.h>
#include<conio.h>

#define VIRTUAL_SCREEN(LINES)(COLS)
int
    count_refresh_called=0;
int
    count_beg_x=COLS-1;
int
    count_end_x=0;
int
    count_beg_y=LINES-1;
int
    count_end_y=0;
int
    REAL_SCREEN_BASE;
int
    color_set=0;
int
    initial_cur_beg,initial_cur_end;

#define NO_OF_REVERSES 40
char
    reverse_table[NO_OF_REVERSES]=
    {36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,
    53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,
    103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125};

```



```

*****
*****
**                                     **
**      *stat_ptr = MK_FP(0x40000,0x0300);**
**      [stat_ptr = 0x0300]          CURSES.C**
**                                     **
**                                     **
**      Optimized Screen Input Output Module with Windows**
**                                     **
**      *NOD**                                     **
**      Creation : 15/3/1991 by Umit KALKAN**
**      REAL_SCREEN_SIZE = (int *)MK_FP(0x40000,0x0300);**
*****
*****

```

```
#include <stdio.h>
#include <dos.h>
#include <bios.h>
#include <alloc.h>
#include <conio.h>

#define NO_OF_REVERSES 49

char reverse_table[NO_OF_REVERSES]=
{16,17,32,33,34,36,37,38,39,40,41,44,45,46,48,49,50,51,52,
53,54,56,57,60,61,64,65,68,70,80,84,85,96,97,100,101,102,
112,113,114,115,116,117,118,119,120,121,124,125};
```



```

initscr() {
    {
        int *stat_ptr = MK_FP(0x4000U, 0063U);
        if (*stat_ptr == 0x03b4)
        {
            /* win_ptr=malloc(sizeof(WINDOW)); */
            /*****
             * Mono
             */
            REAL_SCREEN_BASE = ((int *)MK_FP(0xb000U, 0000U));
        }
    }
    else
    {
        /* ptr->begy=0; */
        /*****
         * Color
         */
        REAL_SCREEN_BASE = ((int *)MK_FP(0xb800U, 0000U));
        color_set_begy=0 = 1;
    }
    ptr->maxx=win_ptr->maxx;
    clrscrn();
    get_cursor_size(&initial_cur_beg, &initial_cur_end);
    gotoxy(0, 0);
}

endwin() {
    {
        clrscrn();
        set_cursor_size(initial_cur_beg, initial_cur_end);
        gotoxy(0, 0);
    }
}

clrscrn()
{
    int *ptr, cnt;
    ptr=REAL_SCREEN_BASE;
    for (cnt=0; cnt<LINES*COLS; cnt++, ptr++)
    {
        *(ptr)=' '|A_NORMAL;
    }
}

```



```

WINDOW *newwin(y_size,x_size,y_pos,x_pos)
int y_size,x_size,y_pos,x_pos;
{
WINDOW *_win_ptr;
if ((_win_ptr=malloc(sizeof(WINDOW)))==NULL)
{
printf ("\n\nrnewwin: Out of memory!\n\nr");
exit (1);
}
_win_ptr->_begx=x_pos;
_win_ptr->_begy=y_pos;
_win_ptr->_curx=0;
_win_ptr->_cury=0;
_win_ptr->_maxx=x_size==0?COLS:x_size;
_win_ptr->_maxy=y_size==0?LINES:y_size;
_win_ptr->_ubegx=0;
_win_ptr->_ubegy=0;
_win_ptr->_umaxx=_win_ptr->_maxx;
_win_ptr->_umaxy=_win_ptr->_maxy;
_win_ptr->_attr=A_NORMAL;
if ((_win_ptr->_buffer=malloc(_win_ptr->_maxx*_win_ptr->_maxy*sizeof(int)))
==NULL)
{
printf ("\n\nrnewwin: Out of memory!\n\nr");
exit (1);
}
clrwin (_win_ptr);
return (_win_ptr);
}

waddch (_win,_chr)
WINDOW *_win;
int _chr;
{
if ( (_chr&A_ATTRIBUTE)!=0 )
{
*(_win->_buffer+_win->_cury*_win->_maxx+_win->_curx)= _chr;
}
else
{
*(_win->_buffer+_win->_cury*_win->_maxx+_win->_curx)= (_win->_attr|_chr);
}
_win->_ubegx = (_win->_ubegx>_win->_curx?_win->_curx:_win->_ubegx);
_win->_ubegy = (_win->_ubegy>_win->_cury?_win->_cury:_win->_ubegy);
_win->_umaxx = (_win->_umaxx<_win->_curx+1?_win->_curx+1:_win->_umaxx);
_win->_umaxy = (_win->_umaxy<_win->_cury+1?_win->_cury+1:_win->_umaxy);
}

```



```

int winch(_win)
wdelch(_win)
{
    WINDOW *_win;
    { turn( (*_win->buffer*_win->_cury*_win->_maxx*_win->_maxx) );
    int cntx,currx;
    currx=_win->_curx;
    for (cntx=_win->_curx;cntx<_win->_maxx-1;cntx++)
    {
        mvwaddch(_win,_win->_cury,cntx,mvwinch(_win,_win->_cury,cntx+1));
    }
    mvwaddch (_win,_win->_cury,_win->_maxx-1,' ');
    wmove (_win,_win->_cury,currx);
    }
    return (winch(_win));
}

```

```

mvwdelch (_win,_lin,_col)
{
    WINDOW *_win;
    int _lin,_col;
    {
        wmove (_win,_lin,_col);
        wdelch (_win);
    }
}

```

```

winstch(_win,_chr)
{
    WINDOW *_win;
    int _chr;
    {
        int cntx,currx;
        currx=_win->_curx;
        for (cntx=_win->_maxx-1;cntx>currx;cntx--)
        {
            mvwaddch(_win,_win->_cury,cntx,mvwinch(_win,_win->_cury,cntx-1));
        }
        mvwaddch (_win,_win->_cury,currx,_chr);
    }
}

```

```

mvwinstch(_win,_lin,_col,_chr)
{
    WINDOW *_win;
    int _lin,_col,_chr;
    {
        wmove (_win,_lin,_col);
        winstch (_win,_chr);
    }
}

```



```
int winch(_win)
WINDOW *_win;
{
    return( *(_win->_buffer+_win->_cury*_win->_maxx+_win->_curx) );
}

```

```
int mvwinch(_win, _lin, _col)
WINDOW *_win;
int _lin, _col;
{
    wmove (_win, _lin, _col);
    return (winch(_win));
}

```

```
void waddch ( _win, _ch )
WINDOW *_win;

```

```

/*****
 *      _KBHIT()
 * This routine works similar to standard
 * kbhit() routine of conio module but only
 * difference is that _kbhit() does not
 * the user to break program by typing
 * CTRL-C as standard kbhit() do!
 *****/

```

```

void wrefresh ( _win, _flag )
WINDOW *_win;
int _flag;
{
    return (bioskey(1));
}

```

```
int wgetch()
{
    unsigned int key;
    if (((key=bioskey(0))&0xFF)==0)
    {
        key=(key>>8)|0x0100;
    }
    else
    {
        key&=0xFF;
    }
    return(key);
}

```



```
mvwaddstr (_win, _lin, _col, _str)
```

```
WINDOW * _win;
```

```
int _lin, _col;
```

```
char * _str;
```

```
wmove (_win, _lin, _col)
```

```
WINDOW * _win;
```

```
int _lin, _col;
```

```
{
```

```
if (_lin >= 0 && _col >= 0 && _lin < _win->_maxy && _col < _win->_maxx)
```

```
{
```

```
_win->_cury = _lin;
```

```
_win->_curx = _col;
```

```
}
```

```
}
```

```
mvwaddch (_win, _
```

```
mvwaddch (_win, _lin, _col, _chr)
```

```
WINDOW * _win;
```

```
int _lin, _col, _chr;
```

```
{
```

```
wmove (_win, _lin, _col);
```

```
waddch (_win, _chr);
```

```
}
```

```
waddstr (_win, _str)
```

```
WINDOW * _win;
```

```
char * _str;
```

```
{
```

```
int length, cnt;
```

```
length = strlen(_str);
```

```
for (cnt = 0; cnt < length; cnt++)
```

```
{
```

```
waddch (_win, _str[cnt]);
```

```
if ((++_win->_curx) >= _win->_maxx)
```

```
{
```

```
_win->_curx = _win->_maxx - 1;
```

```
break;
```

```
}
```

```
}
```



```
mvwaddstr (_win, _lin, _col, _str)
```

```
WINDOW *_win;
```

```
int _lin, _col;
```

```
char *_str;
```

```
{
    wmove (_win, _lin, _col);
```

```
    waddstr (_win, _str);
```

```
}
```

```
clrwin (_win)
```

```
WINDOW *_win;
```

```
{
    fillwin(_win, ' ');
}
```

```
fillwin (_win, _chr)
```

```
WINDOW *_win;
```

```
int _chr;
```

```
{
    int lim, cnt, attr;
```

```
    attr = _chr & A_ATTRIBUTE;
```

```
    attr = attr != 0 ? attr : A_NORMAL;
```

```
    lim = _win->_maxy * _win->_maxx;
```

```
    for (cnt = 0; cnt < lim; cnt++)
```

```
    {
        _win->_buffer[cnt] = _chr | attr;
    }
```

```
    touchwin(_win);
```

```
    wmove (_win, 0, 0);
```

```
}
```

```
box (_win, _hor, _ver, _upl, _upr, _lol, _lor)
```

```
WINDOW *_win;
```

```
int _hor, _ver, _upl, _upr, _lol, _lor;
```

```
{
    int g_cnt, _curry, _currx;
```

```
    _curry = _win->_cury;
```

```
    _currx = _win->_curx;
```

```
    for (g_cnt = 0; g_cnt < _win->_maxy; g_cnt++)
```

```
    {
        mvwaddch (_win, g_cnt, 0, _ver);
        mvwaddch (_win, g_cnt, _win->_maxx-1, _ver);
    }
```

```
    for (g_cnt = 0; g_cnt < _win->_maxx; g_cnt++)
```



```

{
    mvwaddch (_win, 0, g_cntr, _hor);
    mvwaddch (_win, _win->_maxy-1, g_cntr, _hor);
}

mvwaddch (_win, 0, 0, _upl);
mvwaddch (_win, 0, _win->_maxx-1, _upr);
mvwaddch (_win, _win->_maxy-1, 0, _lol);
mvwaddch (_win, _win->_maxy-1, _win->_maxx-1, _lor);
wmove (_win, _curry, _currx);
}

wattrset (_win, _attribute)
WINDOW *_win;
int _attribute;
{
    _win->_attr = _attribute;
}

wrefresh (_win)
WINDOW *_win;
{
    int _lin, _col, *_source_ptr, *_dest_ptr;
    if (noout_beg_y < _win->_begy && _win->_ubegy < _win->_ubegy)
    if (noout_beg_x < _win->_begx && _win->_ubegx < _win->_ubegx)
    if (noout_end_y < _win->_begy && _win->_ubegy < _win->_ubegy-1)
    if (noout_end_x < _win->_begx && _win->_ubegx < _win->_ubegx-1)
wrefresh (_win)
WINDOW *_win;
{
    int _lin, _col, *_source_ptr, *_dest_ptr;
    if (noout_refresh_called==1)
    {
        wnoutrefresh(_win);
        for (_lin=noout_beg_y; _lin<=noout_end_y; _lin++)
        {
            for (_col=noout_beg_x; _col<=noout_end_x; _col++)
            {
                *(REAL_SCREEN_BASE+_lin*COLS+_col) = VIRTUAL_SCREEN[_lin][_col];
            }
        }
        noout_end_x=noout_end_y=0;
        noout_beg_x=COLS-1; noout_beg_y=LINES-1;
        noout_refresh_called=0;
    }
    for (_lin=_win->_begy+_win->_ubegy;
        _lin<_win->_umaxy+_win->_begy&& _lin<LINES;
        _lin++)
    {
        _dest_ptr = REAL_SCREEN_BASE+_lin*COLS+_win->_begx+_win->_ubegx;
        _source_ptr = _win->_buffer+(_lin-_win->_begy)*_win->_maxx+_win->_ubegx;
        int _attr = _win->_attribute;
        int _ty = _win->_currenty;
        if (_ty < _lin)
        {
            _ty = _lin;
        }
    }
}

```



```

    for (_col=_win->_begx+_win->_ubegx;
        _col<_win->_umaxx+_win->_begx&&_col<COLS;
        _col++)
    {
        VIRTUAL_SCREEN[_lin][_col] = *(_source_ptr);
        *(_dest_ptr++) = color_translate(*(_source_ptr++));
    }
}

gotoyx(_win->_begy+_win->_cury, _win->_begx+_win->_curx);
_win->_ubegx=_win->_maxx;
_win->_ubegy=_win->_maxy;
_win->_umaxx=0;
_win->_umaxy=0;
}

wnoutrefresh (_win)
WINDOW *_win;
{
    int _lin, _col, cntr, *_source_ptr, *_dest_ptr;
    if (noout_beg_y>_win->_begy+_win->_ubegy) noout_beg_y=_win->_begy+_win->_ubegy;
    if (noout_beg_x>_win->_begx+_win->_ubegx) noout_beg_x=_win->_begx+_win->_ubegx;
    if (noout_end_y<_win->_begy+_win->_umaxy-1) noout_end_y=_win->_begy+_win->_umaxy-1;
    if (noout_end_x<_win->_begx+_win->_umaxx-1) noout_end_x=_win->_begx+_win->_umaxx-1;
    for (_lin=_win->_begy+_win->_ubegy;
        _lin<_win->_umaxy+_win->_begy&&_lin<LINES;
        _lin++)
    {
        _source_ptr = _win->_buffer+(_lin-_win->_begy)*_win->_maxx+_win->_ubegx;
        for (_col=_win->_begx+_win->_ubegx;
            _col<_win->_umaxx+_win->_begx&&_col<COLS;
            _col++)
        {
            VIRTUAL_SCREEN[_lin][_col] = color_translate(*(_source_ptr++));
        }
    }
    _win->_ubegx=_win->_maxx;
    _win->_ubegy=_win->_maxy;
    _win->_umaxx=0;
    _win->_umaxy=0;
    noout_refresh_called=1;
}

int color_translate(_chr)
int _chr;
{
    int attr = _chr&A_ATTRIBUTE;
    int rc = _chr&A_CHARTEXT;
    if ( !color_set )
    {
        unsigned char byte_attr, cntr;

```



```

byte_attr = attr>>8;
byte_attr &= ((A_BGCOLOR|A_FGCOLOR|A_CALITE)>>8);
attr &= ~A_BGCOLOR;
attr |= A_FGCOLOR;
for (cntr=0;cntr<NO_OF_REVERSES;cntr++)
{
    if (byte_attr==reverse_table[cntr])
    {
        attr &= ~A_FGCOLOR;
        attr |= A_BGCOLOR;
        break;
    }
}
return (rc|attr);
}

```

```

touchwin (_win)
WINDOW *_win;
{
    _win->ubegx=0;
    _win->ubegy=0;
    _win->umaxx=_win->maxx;
    _win->umaxy=_win->maxy;
}

```

```

gotoyx (_y,_x)
int _y,_x;
{
    union REGS regs;
    regs.h.ah = 2;
    regs.h.dh = _y;
    regs.h.dl = _x;
    regs.h.bh = 0;
    int86(0x10,&regs,&regs);
}

```

```

getcurpos (_y,_x)
int *_y,*_x;
{
    union REGS regs;
    regs.h.ah = 3;
    regs.h.bh = 0;
    int86(0x10,&regs,&regs);
}

```



```

*_y = regs.h.dh;
*_x = regs.h.dl;
}
int *buff_ptr, *scr_ptr, y, x;
buff_ptr = buffer;
scr_ptr = REAL_SCREEN_SIZE - beg_y * COLS + beg_x;

```

```

set_cursor_size(_beg, _end)

```

```

int _beg, _end;
{
    for (i=0; i<size_x; i++) buff_ptr++;
    union REGS regs;
    regs.h.ah = 1;
    regs.h.ch = _beg;
    regs.h.cl = _end;
    int86(0x10, &regs, &regs);
}

```

```

get_cursor_size(_beg, _end)

```

```

int *_beg, *_end;
{
    union REGS regs;
    regs.h.ah = 3;
    regs.h.bh = 0;
    int86(0x10, &regs, &regs);
    *_beg = regs.h.ch;
    *_end = regs.h.cl;
}

```

```

delwin (_win)

```

```

WINDOW *_win;
{
    free (_win->_buffer);
    free (_win);
}

```

```

beep()

```

```

{
    putchar(007);
}

```



```
savescrn(size_y,size_x,beg_y,beg_x,buffer)
```

```
int beg_y,beg_x,size_y,size_x,*buffer;
```

```
{
```

```
int *buff_ptr,*scr_ptr,y,x;
```

```
buff_ptr = buffer;
```

```
scr_ptr = REAL_SCREEN_BASE+beg_y*COLS+beg_x;
```

```
for (y=0;y<size_y;y++,scr_ptr+=COLS)
```

```
{
```

```
for (x=0;x<size_x;x++,buff_ptr++)
```

```
{
```

```
*buff_ptr = *(scr_ptr+x);
```

```
}
```

```
}
```

```
}
```

```
restorescrn(size_y,size_x,beg_y,beg_x,buffer)
```

```
int beg_y,beg_x,size_y,size_x,*buffer;
```

```
{
```

```
int *buff_ptr,*scr_ptr,y,x;
```

```
buff_ptr = buffer;
```

```
scr_ptr = REAL_SCREEN_BASE+beg_y*COLS+beg_x;
```

```
for (y=0;y<size_y;y++,scr_ptr+=COLS)
```

```
{
```

```
for (x=0;x<size_x;x++,buff_ptr++)
```

```
{
```

```
*(scr_ptr+x) = *buff_ptr;
```

```
}
```

```
}
```

```
}
```




```

*****
**                               **
**                               SCRNI0.H                               **
**                               **
**                               Screened data Input Output Module      **
**                               **
**                               Creation : 15/3/1991 by Umit KALKAN    **
**                               **
*****
*****
#include <stdio.h>
#include <nullptr.h>
#include <curses.h>
#include <fieldio.h>
#include <scrnio.h>

/*****
*.....INPUT ROUTINE.....*
*****/

#define FINFOS          sinp_info->field_infos
#define NO_OF_FIELDS    sinp_info->no_of_fields
#define CURR_FIELD      sinp_info->current_field

int screen_input(win,sinp_info,key_info,color_info,user_proc)
WINDOW *win;
SCR_INP_INFO *sinp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
int (*user_proc)();
{
    int retcode= -1,curfld;
    if (sinp_info!=NULL_PTR&&NO_OF_FIELDS>0&&CURR_FIELD>=0)
    {
        wattreset      (win,color_info->text_color);
        screen_display (win,sinp_info);
        wrefresh       (win);

        if (CURR_FIELD>=NO_OF_FIELDS)
            CURR_FIELD = 0;

        while (1)
        {
            if (FINFOS[CURR_FIELD]->inp_style!=DISP_ONLY)
            {
                int userirc;
                CALL_FLAG callf=FINFOS[CURR_FIELD]->call_flag;

                if (callf==JUST_CALL||callf==CALL_AND_DO||callf==CALL_DO_CALL)

```



```

if ((userrc=user_proc(win,
                        sinp_info,
                        key_info,
                        color_info,
                        retcode,
                        BEFORE_DOING))!=0)
    if (userrc==DONT_MOVE) continue; else return (userrc);

if (callf==JUST_DO||callf==DO_AND_CALL)
    retcode=field_input(win,FINFOS[Curr_FIELD],key_info,color_info);

if (callf==DO_AND_CALL||callf==CALL_DO_CALL)
    if ((userrc=user_proc(win,
                        sinp_info,
                        key_info,
                        color_info,
                        retcode,
                        AFTER_DOING))!=0)
        if (userrc==DONT_MOVE) continue; else return (userrc);

    next_field (sinp_info,retcode);
}
else
{
    int    currf = Curr_FIELD;
    for (Curr_FIELD=currf+1;
        Curr_FIELD!=currf&&FINFOS[Curr_FIELD]->inp_style==DISP_ONLY;
        Curr_FIELD++) {if (Curr_FIELD>=NO_OF_FIELDS) Curr_FIELD=0;}
    if (Curr_FIELD==currf) return (0);
}
}
}
return (-1);
}

```

```

/*****
*.....MOTION ROUTINES.....*
*****/

```

```

int    next_field_right(sinp_info)
SCR_INP_INFO  *sinp_info;
{
    int    cntr,nxtfld,found=0;
    for (cntr=Curr_FIELD+1<NO_OF_FIELDS?Curr_FIELD+1:0;
        cntr!=Curr_FIELD;cntr = ++cntr<NO_OF_FIELDS?cntr:0)
    {
        if (FINFOS[cntr]->position.y==FINFOS[Curr_FIELD]->position.y)
        {

```



```

int if (!found)
{
    if (FINFOS[ctr]->position.x>=FINFOS[Curr_FIELD]->position.x)
    {
        nxtdfld=ctr;
        found=1;
    }
}
else
{
    if (FINFOS[ctr]->position.x>=FINFOS[Curr_FIELD]->position.x &&
        FINFOS[ctr]->position.x<=FINFOS[nxtdfld]->position.x)
        nxtdfld=ctr;
}
}
if (found) return(nxtdfld); else return(-1);
}

int next_field_left(sinp_info)
SCR_INP_INFO *sinp_info;
{
    int ctr,nxtdfld,found=0;
    for (ctr=Curr_FIELD+1<NO_OF_FIELDS?Curr_FIELD+1:0; ctr<NO_OF_FIELDS; ctr++)
    {
        if (FINFOS[ctr]->position.y==FINFOS[Curr_FIELD]->position.y)
        {
            if (!found)
            {
                if (FINFOS[ctr]->position.x<=FINFOS[Curr_FIELD]->position.x)
                {
                    nxtdfld=ctr;
                    found=1;
                }
            }
            else
            {
                if (FINFOS[ctr]->position.x<=FINFOS[Curr_FIELD]->position.x &&
                    FINFOS[ctr]->position.x>=FINFOS[nxtdfld]->position.x)
                    nxtdfld=ctr;
            }
        }
    }
    if (found) return(nxtdfld); else return(-1);
}

```



```

int     next_field_down(sinp_info)
SCR_INP_INFO *sinp_info;
{
    int     cntnr,nxtfld,found=0;
    for (cntnr=CURR_FIELD+1<NO_OF_FIELDS?CURR_FIELD+1:0;
        cntnr!=CURR_FIELD;cntnr = ++cntnr<NO_OF_FIELDS?cntnr:0)
    {
        if (
            (FINFOS[cntnr]->position.x>=FINFOS[CURR_FIELD]->position.x&&
             FINFOS[cntnr]->position.x<=FINFOS[CURR_FIELD]->position.x+
             FINFOS[CURR_FIELD]->size.x-1) ||
            (FINFOS[CURR_FIELD]->position.x>=FINFOS[cntnr]->position.x&&
             FINFOS[CURR_FIELD]->position.x<=FINFOS[cntnr]->position.x+
             FINFOS[cntnr]->size.x-1)
        )
        {
            if (!found)
            {
                if (FINFOS[cntnr]->position.y>=FINFOS[CURR_FIELD]->position.y)
                {
                    nxtfld=cntnr;
                    found=1;
                }
            }
            else
            {
                if (FINFOS[cntnr]->position.y>=FINFOS[CURR_FIELD]->position.y &&
                    FINFOS[cntnr]->position.y<=FINFOS[nxtfld]->position.y)
                {
                    if (FINFOS[cntnr]->position.y!=FINFOS[nxtfld]->position.y||
                        FINFOS[cntnr]->position.x< FINFOS[nxtfld]->position.x)
                        nxtfld=cntnr;
                }
            }
        }
    }
    if (found) return(nxtfld); else return(-1);
}

```

```

int     next_field_up(sinp_info)
SCR_INP_INFO *sinp_info;
{
    int     cntnr,nxtfld,found=0;
    for (cntnr=CURR_FIELD+1<NO_OF_FIELDS?CURR_FIELD+1:0;
        cntnr!=CURR_FIELD;cntnr = ++cntnr<NO_OF_FIELDS?cntnr:0)
    {
        if (
            (FINFOS[cntnr]->position.x>=FINFOS[CURR_FIELD]->position.x&&
             FINFOS[cntnr]->position.x<=FINFOS[CURR_FIELD]->position.x+
             FINFOS[CURR_FIELD]->size.x-1) ||

```



```

(FINFOS[Curr_Field]->position.x>=FINFOS[ctr]->position.x&&
FINFOS[Curr_Field]->position.x<=FINFOS[ctr]->position.x+
FINFOS[ctr]->size.x-1)
)
{
if (!found)
{
if (FINFOS[ctr]->position.y<=FINFOS[Curr_Field]->position.y)
{
nxtfld=ctr;
found=1;
}
}
else
{
if (FINFOS[ctr]->position.y<=FINFOS[Curr_Field]->position.y &&
FINFOS[ctr]->position.y>=FINFOS[nxtfld]->position.y)
{
if (FINFOS[ctr]->position.y!=FINFOS[nxtfld]->position.y||
FINFOS[ctr]->position.x< FINFOS[nxtfld]->position.x)
nxtfld=ctr;
}
}
}
}
if (found) return(nxtfld); else return(-1);
}

```

```

/*****
*.....GENERAL MOTION ROUTINE.....*
*****/

```

```

int    next_field(sinp_info,key)
SCR_INP_INFO  *sinp_info;
int          key;
{
int  curfld;
switch (key)
{
case 0:
CURR_FIELD = ++CURR_FIELD<NO_OF_FIELDS?CURR_FIELD:0;
break;

case KEY_RIGHT:
if ( (curfld=next_field_right(sinp_info))!= -1 )
CURR_FIELD=curfld;
break;

```



```

case KEY_DOWN:
    if ( (curfld=next_field_down(sinp_info))!= -1 )
        CURR_FIELD=curfld;
    break;

case KEY_LEFT:
    if ( (curfld=next_field_left(sinp_info))!= -1 )
        CURR_FIELD=curfld;
    break;

case KEY_UP:
    if ( (curfld=next_field_up(sinp_info))!= -1 )
        CURR_FIELD=curfld;
}
}
#include <conio.h>
#include <dos.h>
#include <fcntl.h>

/*****
*.....DISPLAY ROUTINE.....*
*****/
#include <fcntl.h>
#include <ioctl.h>
#include <poll.h>

int screen_display(win,sinp_info)
    WINDOW *win;
    SCR_INP_INFO *sinp_info;
{
    int cnt;
    for (cnt=0;cnt<sinp_info->no_of_fields;cnt++)
    {
        field_display(win,sinp_info->field_infos[cnt]);
    }
}

***** COMMON DEFINITIONS & ROUTINES *****
*****
*****

#include <conio.h>

#define LINE_SIZE 256
#define MAX_LOGIN_LENGTH 20
#define LOGIN_WINDOW_HEIGHT 10 /* window */
#define MAX_FILE_LEN 100
#define FIELD_SEPARATOR '\t'
#define COMMENT_SEPARATOR '#'
char *BANNER_FILE = "banner.fil";
char *PASSWORD_FILE = "passwd.fil";
char *MAINMENU_FILE = "mainmenu.menu";
char *LOGIN_PROMPT = "User Name : ";
char *PASSWORD_PROMPT = "Password : ";
char *NEW_ID_STR = "\n";
char *UNKNOWN_USER_STR = "Unknown User Name!\n";

```



```

/*****
*****
**  "LOGIN_TIMEOUT_STR"      =      "Login Timeout Set:\r\n";
**  "NEW_LN_STR"            =      "UMISRV.C  are logged out!... Bye Bye!...\r\n";
**
**
**      Umisrv Main Program
**
**
**      Creation : 15/3/1991 by Umit KALKAN
**
*****
*****/

#include<stdio.h>
#include<conio.h>
#include<bios.h>
#include<dos.h>
#include<fcntl.h>
#include<stat.h>
#include<io.h>
#include<string.h>
#include<nullptr.h>
#include<alloc.h>
#include<curses.h>
#include<comm.h>
#include<scrnio.h>

#define DATAFILE "umisrv.dat"

/*****
*****
*****  COMMON DEFINITIONS & ROUTINES  *****/
*****
*****/

while (rcbuff[0]!='(char) NULL) _com_sendin(PORT,rcbuff,1);

#include<umcommon.c>

#define LINE_BUFF_LEN      513
#define MAX_LOGIN_RETRIES  5
#define LOGIN_TIMEOUT_INTERVAL  30      /* seconds */
#define MAX_FILE_LINE_LEN  80
#define FIELD_SEPARATOR    ':'
#define COMMENT_INDICATOR  '#'
char  *BANNER_FILE        =      "banner.fil";
char  *PASSWD_FILE        =      "passwd.fil";
char  *MAINMENU_FILE      =      "mainmenu.mnu";
char  *LOGIN_PROMPT       =      "User Name : ";
char  *PASSWD_PROMPT      =      "Password : ";
char  *NEW_LN_STR         =      "\r\n";
char  *UNKNOWN_USER_STR   =      "Unknown User Name!\r\n";

```



```

char *LOGIN_INCORRECT_STR = "Incorrect Password!\r\n";
char *NO_ACCOUNT_STR = "You Are Not Allowwd to LOGIN Here!\r\n";
char *LOGIN_TIMEOUT_STR = "\r\nLogin Timed Out!\r\n";
char *BYE_STR = "\r\nYou are Logged Out!...Bye Bye!...\r\n";

```

```

modifi(file_name);

```

```

char *file_name;

```

```

/*****
*****
***** MENU DEFINITIONS & ROUTINES *****/
*****
*****/

```

```

#include <commenu.c>

```

```

_comreadln(PORT,com_buff);

```

```

close(handle);

```

```

return (0);

```

```

/*****
*..... SERVER SIDE PROCEDURES.....*
*****/

```

```

/*****
*.....WAIT CONNECTION.....*
*****/

```

```

wait_for_connection()

```

```

{
char *buff;

```

```

while (1)

```

```

{

```

```

FLUSHCOMBUFF();

```

```

resbuff[0]=0;

```

```

while ( !(COMHIT()) ) {handle_kbd();};

```

```

while (resbuff[0]==(char) NULL) _comreadln(PORT,resbuff,1);

```

```

if (strcmp(resbuff,CONNECT1)==0||

```

```

    strcmp(resbuff,CONNECT2)==0||

```

```

    strcmp(resbuff,CONNECT3)==0 ) {sleep(2);return(0);}

```

```

}

```

```

}

```

```

buff[0]='\0';

```

```

return;

```

```

case 0:

```

```

if (ctrl==0) ctrl--;

```

```

if ( !strcmp(ctrl,"_comreadln(PORT,ctrl);

```

```

break;

```

```

default:

```

```

buff[ctrl]='\0';

```

```

if ( !strcmp(ctrl,"_comreadln(PORT,ctrl);

```

```

ctrl++;

```

```

/*****
*.....CAT A FILE ON COMM PORT.....*
*****/

buff[cntr]=chr; NULL;

comcatfile(file_name)
char *file_name;
{
    int handle;
    char chr_buff;
    if ( (handle=open(file_name,O_RDONLY)) != -1 )
    {
        while ( read(handle,&chr_buff,1) != 0 )
        {
            if (chr_buff==LF) _comsendchr(PORT,CR);
            _comsendchr(PORT,chr_buff);
        }
        close(handle);
        return (0);
    }
    return(-1);
}

/*****
*.....READ A LINE FROM COMM PORT.....*
*****/

_comreadln(port,buff,supress_echo)
char *buff;
char port,supress_echo;
{
    int cnter=0;
    char chr;
    while (cnter<LINE_BUFF_LEN-1)
    {
        chr=COMRECEIVECHR();
        switch (chr)
        {
            case CR:
                buff[cnter]=(char) NULL;
                return;
            case LF:
                buff[cnter]=(char) NULL;
                return;
            case BS:
                if (cnter>0) cnter--;
                if ( !supress_echo ) _comsendchr(port,chr);
                break;
            default:
                buff[cnter]=chr;
                if ( !supress_echo ) _comsendchr(port,chr);
                cnter++;
        }
    }
}

```



```

        break;
    }
}

buff[cnter]=(char) NULL;

}

/*****
*.....FIND USER'S PASSWORD.....*
*****/

int findpasswd(username,password)
char *username,*password;
{
    int handle,rc,cntr,currline=0;
    char ch,tmpbuff[MAX_FILE_LINE_LEN+2];
    if ( (handle=open(PASSWD_FILE,O_RDONLY|O_TEXT)) != -1)
    {
        cntr=0;
        while (cntr<MAX_FILE_LINE_LEN)
        {
            rc=read(handle,&ch,1);
            if (ch!=LF && ch!=CR && rc!=0)
            {
                tmpbuff[cntr]=ch;
                cntr++;
            }
            else
            {
                char *sep_loc;
                currline++;
                tmpbuff[cntr]=(char) NULL;
                cntr=0;
                if (tmpbuff[0]!=COMMENT_INDICATOR)
                {
                    if ( (sep_loc=strchr(tmpbuff,FIELD_SEPARATOR))!=NULL_PTR)
                    {
                        *sep_loc=(char) NULL;
                        sep_loc++;
                        if (strcmp(tmpbuff,username)==0)
                        {
                            sprintf(password,"%s",sep_loc);
                            close (handle);
                            if (password[0]!=(char)NULL) return (0); else return(1);
                        }
                    }
                    else
                    {
                        printf ("Invalid User definition line in file %s line %d!\n",
                                PASSWD_FILE,currline);
                    }
                }
                if (rc==0) {close(handle);return(-1);}
            }
        }
    }
}

```

```

    close (handle);
    return (-1);
}

printf ("Cannot find password file %s.\n",PASSWD_FILE);
return (-1);
}

/*
 *.....LOGIN PROCEDURE.....*
 */

int comlogin()
{
    char password[LINE_BUFF_LEN];
    int rc,ctr=0;
    while (ctr<MAX_LOGIN_RETRIES)
    {
        ctr++;
        FLUSHCOMBUFF();
        password[0] =(char) NULL;
        resbuff[0] =(char) NULL;
        cmdbuff[0] =(char) NULL;
        _comsendstr(PORT,LOGIN_PROMPT);
        while (cmdbuff[0]==(char)NULL)
        {
            if (waitevent(LOGIN_TIMEOUT_INTERVAL,COMHIT)!=1)
            {
                _comsendstr (PORT,LOGIN_TIMEOUT_STR);
                printf ("Login Timed Out!\n");
                return(-1);
            }
            _comreadln (PORT,cmdbuff,0);
        }
        _comsendstr(PORT,NEW_LN_STR);
        if ( (rc=findpasswd(cmdbuff,password)) == 0 )
        {
            _comsendstr(PORT,PASSWD_PROMPT);
            if (waitevent(LOGIN_TIMEOUT_INTERVAL,COMHIT)!=1)
            {
                _comsendstr (PORT,LOGIN_TIMEOUT_STR);
                printf ("Login Timed Out!\n");
                return(-1);
            }
            _comreadln (PORT,resbuff,1);
            _comsendstr(PORT,NEW_LN_STR);
            if (strcmp(password,resbuff)==0)
            {
                printf ("User \"%s\" logged in!\n",cmdbuff);
                return (0);
            }
        }
    }
}

```



```

else
{
    _comsendstr(PORT, LOGIN_INCORRECT_STR);
}
}
else
{
    if (rc==1)
    {
        printf ("User \"%s\" logged in!\n", cmdbuff);
        return (0);
    }
    else
    {
        _comsendstr(PORT, UNKNOWN_USER_STR);
    }
}
}
_comsendstr(PORT, NO_ACCOUNT_STR);
printf ("Unauthorized login attempt! Login refused!\n");
return (-1);
}

```

```

/*****
*.....MAIN.....*
*****/

```

```

main ()
{
    initscr ();
    prog_init ();
    prog_reset();

    while (1)
    {
        drop_connection();
        wait_for_connection();
        printf("\nGuest Connected!\n");
        comcatfile(BANNER_FILE);
        if (comlogin()==0)
        {
            _comsendstr(PORT, "Successfull LOGIN...\r\n");
            switch(commenu(MAINMENU_FILE))
            {
                case 2:
                    _comsendstr(PORT, BYE_STR);
                    printf ("User Logged out!\n");
                    break;
            }
        }
    }
}

```

```

case -1:
printf ("An Error Occured while menu operation!.\n");
break;
case -2:
printf ("User response timed out!.\n");
}
}
else
{
printf ("Connection is being closed due to login error!.\n");
}
}
}

```

```

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <conio.h>
#include <fcntl.h>

```

```

.....PROCEDURE TABLE.....
.....

```

```

.....

```

```

let (*inp_procedure_of_data_type) = (simple_input,
                                     data_input,
                                     time_input);

```

```

.....

```

```

let (*sel_procedure_of_data_type) = (simple_selection,
                                     data_selection);

```

```

.....

```

```

let (*disp_procedure_of_data_type) = (simple_display,
                                     data_display,
                                     time_display);

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```

```

.....

```



```

/*****
**
**                               FIELDIO.C
**
**      Single selection window (info,key_info,color_info)
**      Field Based Screen Input Output Module
**
**      Creation : 15/3/1991 by Umit KALKAN
**
*****/

#include <stdio.h>
#include <string.h>
#include <nullptr.h>
#include <curses.h>
#include <fieldio.h>

/*.....PROCEDURE TABLES.....*/

int (*inp_procs[NO_OF_DATA_TYPES])() = {alpha_input,
                                         num_input,
                                         date_input,
                                         time_input};

int (*sel_procs[NO_OF_SEL_STYLES])() = {simple_selection,
                                         boxed_selection};

(*disp_procs[NO_OF_DATA_TYPES])()= {alpha_display,
                                     num_display,
                                     date_display,
                                     time_display};

```

```

/*****
*.....SELECTION ROUTINES.....*
*****/

int simple_selection(win, finp_info, key_info, color_info)
WINDOW *win;
FIELD_INP_INFO *finp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
{
WINDOW *sel_win;
int cpos_x, cpos_y, csize_b, csize_e, cntr;
int selection, key, no_of_sel,
returning=0, usrkey=0, keyrc = -1;
char **sel_list;

getcurpos (&cpos_y, &cpos_x);
get_cursor_size (&csize_b, &csize_e);
set_cursor_size (-1, -1);

no_of_sel = finp_info->sel_info.no_of_sel;
sel_list = finp_info->sel_info.sel_list;
selection = *(finp_info->selection);
selection = selection<0?0:selection;
selection = selection>no_of_sel?no_of_sel-1:selection;

sel_win = newwin(1, finp_info->size.x,
finp_info->position.y+win->begy,
finp_info->position.x+win->begx);

while (1)
{
wattrset (sel_win, color_info->sel_color);
wtowin (sel_win,
0, 0,
finp_info->size.x,
sel_list[selection]);
wrefresh (sel_win);

key = GETKEY (win);
for (cntr=0; cntr<key_info->no_of_keys; cntr++)
{
if (key==key_info->key_list[cntr])
{
keyrc = key;
usrkey = 1;
break;
}
}
}
}

```



```

if (usrkey) break;
switch(key)
{
case KEY_PGUP:
    selection = --selection<0?no_of_sel-1:selection;
    break;
case KEY_SPACE:
case KEY_PGDN:
    selection = ++selection>no_of_sel?0:selection;
    break;
case KEY_UP:
case KEY_DOWN:
case KEY_LEFT:
case KEY_RIGHT:
    keyrc = key;
    returning = 1;
    break;
case KEY_RETURN:
    keyrc = 0;
    returning = 1;
    break;
case KEY_ESC:
    keyrc = -1;
    returning = 1;
    break;
}
if (returning) break;
}

if (keyrc != -1)
{
    if (finp_info->out_buff!=NULL_PTR)
        strcpy(finp_info->out_buff,sel_list[selection]);
    *(finp_info->selection)=selection;
}

if (finp_info->inp_style==SEL_ONLY)
{
    wattrset (win,color_info->text_color);
    wtowin (win,
        finp_info->position.y,
        finp_info->position.x,
        finp_info->size.x,
        finp_info->out_buff);
}

touchwin (win);
wrefresh (win);
delwin (sel_win);
gotoyx (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
return (keyrc);
}

```

```

if (finp_info->key_style==KEY_STYLE)
{
    wattrset (win,color_info->sel_color);
    wdown (win);
    finp_info->position.y;
    finp_info->position.x;
}

int boxed_selection(win,finp_info,key_info,color_info)
WINDOW *win;
FIELD_INP_INFO *finp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
{
    WINDOW *sel_win;
    int cpos_x,cpos_y,csz_b,csz_e,cntr,
        scr_buff[LINES*COLS];
    int selection,no_of_sel,offset=0,key,siz_y,siz_x,pos_y,pos_x,
        scrolled=1,returning=0,usrkey=0,keyrc = -1;
    char **sel_list;

    wattrset (win,color_info->sel_color);

    siz_y = finp_info->size.y;
    siz_x = finp_info->size.x+2;
    pos_y = finp_info->position.y+1;
    pos_x = finp_info->position.x-1;

    if (siz_y<=2 || siz_x<=2 || pos_y<=0 || siz_x>COLS)
        return (-1);

    if (pos_y+siz_y>LINES) siz_y=LINES-pos_y;
    if (siz_y<=2) {siz_y = finp_info->size.y;
        pos_y-= (siz_y+1);}
    if (pos_y<0) {pos_y = 0;
        siz_y = finp_info->position.y;}
    if (pos_x+siz_x>COLS) pos_x--;
    if (pos_x<0) pos_x=0;

    no_of_sel = finp_info->sel_info.no_of_sel;
    sel_list = finp_info->sel_info.sel_list;

    savescrn (siz_y,siz_x,win->_begy+pos_y,win->_begx+pos_x,scr_buff);
    getcurpos (&cpos_y,&cpos_x);
    get_cursor_size (&csz_b,&csz_e);
    set_cursor_size (-1,-1);

    sel_win = newwin(siz_y,siz_x,win->_begy+pos_y,win->_begx+pos_x);

    selection = *(finp_info->selection);
    selection = selection<0?0:selection;
    selection = selection>=no_of_sel?no_of_sel-1:selection;

    wattrset (sel_win,color_info->box_color);
    box (sel_win,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);

```



```

if (finp_info->inp_style==SEL_ONLY)
{
    wattrset      (win,color_info->sel_color);
    wtowin        (win,
        finp_info->position.y,
        finp_info->position.x,
        finp_info->size.x,
        finp_info->out_buff);
}
touchwin        (win);
wrefresh        (win);

while (1)
{
    if (selection<offset)      {offset=selection;scrolled=1;}
    if (selection>offset+siz_y-3) {offset=selection-siz_y+3;scrolled=1;}
    if (scrolled)
    {
        wattrset (sel_win,color_info->text_color);
        for (cntr=offset;cntr<offset+siz_y-2;cntr++)
        {
            if (cntr>=no_of_sel)
            {
                wattrset (sel_win,color_info->box_color);
                for (cntr--=(offset-1);cntr<siz_y-1;cntr++)
                    wtowin (sel_win,cntr,1,siz_x-2,"");
                break;
            }
            if (cntr==selection)
            {
                wattrset (sel_win,color_info->sel_color);
                wtowin (sel_win,cntr-offset+1,1,siz_x-2,sel_list[cntr]);
                wattrset (sel_win,color_info->text_color);
            }
            else
            {
                wtowin (sel_win,cntr-offset+1,1,siz_x-2,sel_list[cntr]);
            }
        }
    }
    wrefresh      (sel_win);
    key = GETKEY (sel_win);
    for (cntr=0;cntr<key_info->no_of_keys;cntr++)
    {
        if (key==key_info->key_list[cntr])
        {
            keyrc = key;
            usrkey = 1;
            break;
        }
    }
    if (usrkey) break;
}

```

```

switch(key)
{
case      KEY_PGUP:
    selection = --selection<0?0:selection;
    break;

case      KEY_PGDN:
    selection = ++selection>no_of_sel?no_of_sel-1:selection;
    break;

case      KEY_UP:
case      KEY_DOWN:
case      KEY_LEFT:
case      KEY_RIGHT:
    keyrc    = key;
    returning = 1;
    break;

case      KEY_RETURN:
    keyrc    = 0;
    returning = 1;
    break;

case      KEY_ESC:
    keyrc    = -1;
    returning = 1;
    break;
}
if (returning) break;
}
if (keyrc != -1)
{
    if (finp_info->out_buff!=NULL_PTR)
        strcpy(finp_info->out_buff,sel_list[selection]);
    *(finp_info->selection)=selection;
}
if (finp_info->inp_style==SEL_ONLY)
{
    wattrset      (win,color_info->text_color);
    wtowin        (win,
        finp_info->position.y,
        finp_info->position.x,
        finp_info->size.x,
        finp_info->out_buff);
}
touchwin        (win);
wrefresh        (win);
restorescrn     (siz_y,siz_x,win->begy+pos_y,win->begx+pos_x,scr_buff);
gotoxy          (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
delwin          (sel_win);
return          (keyrc);
}

```



```

switch (key)
{
    case KEY_RETURN:
        keyrc = 1;
        return * 1;
        break;
}

/*****
*.....INPUT ROUTINES.....*
*****/

if (keyrc == 1)
{
    key_info = keyrc;
    color_info = 1;
}

int alpha_input(win, finp_info, key_info, color_info)
WINDOW = *win;
FIELD_INP_INFO *finp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
{
    int cpos_x, cpos_y, csize_b, csize_e, cntr;
    int key, siz_x, pos_y, pos_x,
    insert_mode=0, returning=0, usrkey=0, keyrc = -1;

    siz_x = finp_info->size.x;
    pos_y = finp_info->position.y;
    pos_x = finp_info->position.x;

    getcurpos (&cpos_y, &cpos_x);
    get_cursor_size (&csize_b, &csize_e);
    set_cursor_size (initial_cur_beg, initial_cur_end);

    if (finp_info != NULL_PTR && finp_info->out_buff != NULL_PTR)
    {
        int x=pos_x;
        wattrset (win, color_info->sel_color);
        wtpwin (win, pos_y, pos_x, siz_x, finp_info->out_buff);

        while (1)
        {
            wmove (win, pos_y, x);
            wrefresh (win);
            key = GETKEY (win);
            for (cntr=0; cntr<key_info->no_of_keys; cntr++)
            {
                if (key==key_info->key_list[cntr])
                {
                    keyrc = key;
                    usrkey = 1;
                    break;
                }
            }
        }
        if (usrkey) break;
    }
}

```

```

switch (key)
{
    case KEY_RETURN:
        keyrc = 0;
        returning = 1;
        break;

    case SEL_REQ_KEY:
        if (finp_info->inp_style==ENTR_AND_SEL)
        {
            if ( (keyrc=sel_procs[finp_info->sel_style](win,
                finp_info,
                key_info,
                color_info)) != -1)
            {
                if (keyrc!=0)
                {returning =1;}
            }
            else
            {
                wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
                keyrc = -1;
                x = pos_x;
            }
        }
        break;

    case KEY_ESC:
        keyrc = -1;
        returning = 1;
        break;

    case KEY_UP:
    case KEY_DOWN:
    case KEY_LEFT:
    case KEY_RIGHT:
    case KEY_PGUP:
    case KEY_PGDN:
        keyrc = key;
        returning = 1;
        break;

    case KEY_cLEFT:
        if (x==pos_x)
            {keyrc = KEY_LEFT;returning = 1;}
        else
            {x = --x<pos_x?pos_x:x;}
        break;

    case KEY_cRIGHT:
        if (x==pos_x+siz_x-1)
            {keyrc = KEY_RIGHT;returning =1;}
        else

```



```

        {x = ++x>=pos_x+siz_x?pos_x+siz_x-1:x;}
        break;

        for (cntr=pos_x+1; cntr<pos_x+siz_x; cntr++)

case KEY_BACKSPACE:
    if (x>pos_x)
    {
        --x;
        for (cntr=x+1; cntr<pos_x+siz_x; cntr++)
        {
            mvwaddch (win,pos_y,cntr-1,mvwinch(win,pos_y,cntr));
        }
        mvwaddch (win,pos_y,cntr-1, ' ');
    }
    else
    {
        beep();
    }
    break;

case KEY_DEL:
    for (cntr=x+1; cntr<pos_x+siz_x; cntr++)
    {
        mvwaddch (win,pos_y,cntr-1,mvwinch(win,pos_y,cntr));
    }
    mvwaddch (win,pos_y,cntr-1, ' ');
    break;

case KEY_INS:
    insert_mode = !insert_mode;
    if (insert_mode)
    {set_cursor_size(initial_cur_end-5,initial_cur_end);}
    else
    {set_cursor_size(initial_cur_beg,initial_cur_end);}
    break;

case KEY_HOME:
    x = pos_x;
    break;

case KEY_END:
    x = pos_x+siz_x-1;
    break;

default:
    if ( (key>=32&&key<=126) || (key>=128&&key<=254) )
    {
        if (!insert_mode)
        {
            waddch (win,key);
            x = ++x>=pos_x+siz_x?pos_x+siz_x-1:x;
        }
        else
        {
            //

```

```

    if ((mvwinch(win,pos_y,pos_x+siz_x-1)&A_CHARTEXT)==' ')
    {
        for (cntr=pos_x+siz_x-1;cntr>x;cntr--)
        {
            mvwaddch (win,pos_y,cntr,mvwinch(win,pos_y,cntr-1));
        }
        mvwaddch (win,pos_y,cntr,key);
        x = ++x>pos_x+siz_x?pos_x+siz_x-1:x;
    }
    else
    {
        beep();
    }
}

if (returning) break;
}

if (keyrc != -1)
{
    rfromwin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
}

wattrset      (win,color_info->text_color);
wtowin        (win,pos_y,pos_x,siz_x,finp_info->out_buff);
wrefresh      (win);
gotoyx        (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
return        (keyrc);
}

return (-1);
}

int num_input(win,finp_info,key_info,color_info)
WINDOW *win;
FIELD_INP_INFO *finp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
{
    int cpos_x,cpos_y,csize_b,csize_e,cntr;
    int key,siz_x,siz_m,pos_y,pos_x,
        insert_mode=0,returning=0,usrkey=0,keyrc = -1;

    siz_x = finp_info->size.x;
    siz_m = finp_info->size.mant_len;
    pos_y = finp_info->position.y;
    pos_x = finp_info->position.x;

    getcurpos (&cpos_y,&cpos_x);
    get_cursor_size (&csize_b,&csize_e);

```



```

set_cursor_size (initial_cur_beg,initial_cur_end);

if (finp_info != NULL_PTR && finp_info->out_buff != NULL_PTR)
{
    int x=pos_x;
    int beg,siz,retpos=pos_x;
    char chr,fraction_state=0,state_changed=1,first_time=1;

    if (siz_m<0 || siz_x<=0 || pos_y<0 || pos_x<0)
        return (-1);
    if (siz_m>siz_x)
        siz_m=siz_x;

    beg=0;
    siz=siz_m;

    arrange_num (finp_info->out_buff,siz_x,siz_m);
    wattrset (win,color_info->text_color);
    wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
    wattrset (win,color_info->sel_color);
    wljustify (win,pos_y,pos_x,siz_m);

    if (siz_m==0) fraction_state = 1;

    while (1)
    {
        if (!fraction_state & !state_changed)
        {
            if ( (mvwinch(win,pos_y,pos_x)&A_CHARTEXT) == ' ' )
                mvwaddch(win,pos_y,pos_x,
                    (mvwinch(win,pos_y,pos_x)&A_ATTRIBUTE) | '0');
        }
        if (state_changed)
        {
            state_changed = 0;
            set_color(win,pos_y,pos_x+beg,siz,color_info->text_color);
            if (fraction_state)
            {
                wrjustify(win,pos_y,pos_x+beg,siz);
                siz = siz_x-siz_m-1;
                beg = siz_m+1;
                x = pos_x+siz_m+1;
            }
            else
            {
                wljustify(win,pos_y,pos_x+beg,siz);
                if (!first_time)
                    wsptozero(win,pos_y,pos_x+beg,siz);
                siz = siz_m;
                beg = 0;
                x = retpos;
            }
            set_color(win,pos_y,pos_x+beg,siz,color_info->sel_color);

```

```

    wljustify(win,pos_y,pos_x+beg,siz);
    wattrset (win,color_info->sel_color);
}
returning = 1;
wmove (win,pos_y,x);
wrefresh (win);

key = GETKEY (win);
for (cntr=0;cntr<key_info->no_of_keys;cntr++)
{
    if (key==key_info->key_list[cntr])
    {
        keyrc = key;
        usrkey = 1;
        break;
    }
}
if (usrkey) break;

switch (key)
{
    case KEY_RETURN:
        keyrc = 0;
        returning = 1;
        break;

    case SEL_REQ_KEY:
        if (finp_info->inp_style==ENTR_AND_SEL)
        {
            if ( (keyrc=sel_procs[finp_info->sel_style](win,
                finp_info,
                key_info,
                color_info)) != -1)
            {
                if (keyrc!=0)
                {returning =1;}
            }
            else
            {
                arrange_num (finp_info->out_buff,siz_x,siz_m);
                wattrset (win,color_info->text_color);
                wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
                wattrset (win,color_info->sel_color);
                wljustify (win,pos_y,pos_x,siz_m);

                fraction_state=0;state_changed=first_time=1;keyrc= -1;
                if (siz_m==0) fraction_state = 1;
            }
        }
    }
}
break;

```



```

case KEY_ESC:
    keyrc = -1;
    returning = 1;
    break;

case KEY_UP:
case KEY_DOWN:
case KEY_LEFT:
case KEY_RIGHT:
case KEY_PGUP:
case KEY_PGDN:
    keyrc = key;
    returning = 1;
    break;

case KEY_BACKSPACE:
case KEY_CLEFT:
    if (x==pos_x)
        {keyrc = KEY_LEFT;returning = 1;}
    else
    {
        if (siz_m>0&&siz_m<siz_x-1)
        {
            if (fraction_state&&x==pos_x+beg)
            {
                fraction_state = !fraction_state;
                state_changed = 1;
                break;
            }
        }
        x = --x<pos_x+beg?pos_x+beg:x;
    }
    break;

case KEY_CRIGHT:
    if (x==pos_x+siz_x-1)
        {keyrc = KEY_RIGHT;returning = 1;}
    else
    {
        if (siz_m>0&&siz_m<siz_x-1)
        {
            if (!fraction_state&&x==pos_x+beg+siz-1)
            {
                fraction_state = !fraction_state;
                state_changed = 1;
                retpos = x;
                break;
            }
        }
        if ( (mvwinch(win,pos_y,x)&A_CHARTEXT)==' ' )
        {
            if (fraction_state)
            {
                keyrc = key;
                returning = 1;
            }
        }
    }

```

```

    }
    else
    {
        fraction_state = 1;
        state_changed = 1;
        retpos = x;
    }
    break;
}
x = ++x > pos_x + beg + siz ? pos_x + beg + siz - 1 : x;
}
break;

case KEY_BACKSPACE:
    if (x > pos_x + beg)
    {
        --x;
        for (cntr = x + 1; cntr < pos_x + beg + siz; cntr++)
        {
            mvwaddch (win, pos_y, cntr - 1, mvwinch(win, pos_y, cntr));
        }
        mvwaddch (win, pos_y, cntr - 1, ' ');
    }
    else
    {
        beep();
    }
    break;

case KEY_DEL:
    for (cntr = x + 1; cntr < pos_x + beg + siz; cntr++)
    {
        mvwaddch (win, pos_y, cntr - 1, mvwinch(win, pos_y, cntr));
    }
    mvwaddch (win, pos_y, cntr - 1, ' ');
    break;

case KEY_INS:
    insert_mode = !insert_mode;
    if (insert_mode)
        {set_cursor_size(initial_cur_end - 5, initial_cur_end);}
    else
        {set_cursor_size(initial_cur_beg, initial_cur_end);}
    break;

case KEY_HOME:
    x = pos_x + beg;
    break;

case KEY_END:
    while ((mvwinch(win, pos_y, x) & A_CHARTEXT) != ' ' && x < pos_x + beg + siz)
        x++;

```



```

    x=x<pos_x+beg+siz?x:pos_x+beg+siz-1;
    return break;

case KEY_DOT:
    if (siz_m>0&&siz_m<siz_x-1)
    {
        fraction_state = !fraction_state;
        retpos = pos_x;
        state_changed = 1;
    }
    break;

default:
    if (key>='0'&&key<='9')
    {
        if (!insert_mode)
        {
            waddch (win,key);
            x = ++x>=pos_x+beg+siz?pos_x+beg+siz-1:x;
        }
        else
        {
            if ((mvwinch(win,pos_y,pos_x+beg+siz-1)&A_CHARTEXT)==' ')
            {
                for (cntr=pos_x+beg+siz-1;cntr>x;cntr--)
                {
                    mvwaddch (win,pos_y,cntr,mvwinch(win,pos_y,cntr-1));
                }
                mvwaddch (win,pos_y,cntr,key);
                x = ++x>=pos_x+beg+siz?pos_x+beg+siz-1:x;
            }
            else
            {
                beep();
            }
        }
    }
    first_time = 0;
    if (returning) break;
}

if (keyrc != -1)
{
    wrjustify(win,pos_y,pos_x,siz_m);
    wljustify(win,pos_y,pos_x+siz_m+1,siz_x-siz_m-1);
    wsptozero(win,pos_y,pos_x+siz_m+1,siz_x-siz_m-1);
    rfromwin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
}
wattrset (win,color_info->text_color);
wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
wrefresh (win);
gotoyx (cpos_y,cpos_x);

```

```

    set_cursor_size (csize_b,csize_e);
    return          (keyrc);
} if (usrkey: break;
return (-1);
} switch (key)
{
    case KEY_RIGHT:
        keyrc = 0;
        returning = 1;
int date_input(win,finp_info,key_info,color_info)
WINDOW *win;
FIELD_INP_INFO *finp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
{
    if (keyrc: gross(finp_info->sel_color)(win,
int cpos_x,cpos_y,csize_b,csize_e,ctr;
int key,siz_x,pos_y,pos_x,
    returning=0,usrkey=0,keyrc = -1;

siz_x = DATE_LEN;
pos_y = finp_info->position.y;
pos_x = finp_info->position.x;

getcurpos (&cpos_y,&cpos_x);
get_cursor_size (&csize_b,&csize_e);
set_cursor_size (initial_cur_beg,initial_cur_end);

if (finp_info != NULL_PTR && finp_info->out_buff != NULL_PTR)
{
    int x=pos_x;

    if (pos_y<0 || pos_x<0)
        return (-1);

    wattrset (win,color_info->sel_color);
    wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
    mvwaddch (win,pos_y,pos_x+2,'/');
    mvwaddch (win,pos_y,pos_x+5,'/');
    wspotzero (win,pos_y,pos_x,siz_x);

while (1)
{
    wmove (win,pos_y,x);
    wrefresh (win);

    key = GETKEY (win);
    for (ctr=0;ctr<key_info->no_of_keys;ctr++)
    {
        if (key==key_info->key_list[ctr])
        {
            keyrc = key;
            usrkey = 1;
            break;

```



```

    }
    }
    if (usrkey) break;

    switch (key)
    {
    case KEY_RETURN:
        keyrc = 0;
        returning = 1;
        break;

    case SEL_REQ_KEY:
        if (finp_info->inp_style==ENTR_AND_SEL)
        {
            if ( (keyrc=sel_procs[finp_info->sel_style](win,
                                                         finp_info,
                                                         key_info,
                                                         color_info)) != -1)
            {
                if (keyrc!=0)
                {returning =1;}
                else
                {
                    wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
                    keyrc = -1;
                    x = pos_x;
                }
            }
        }
        break;

    case KEY_ESC:
        keyrc = -1;
        returning = 1;
        break;

    case KEY_UP:
    case KEY_DOWN:
    case KEY_LEFT:
    case KEY_RIGHT:
    case KEY_PGUP:
    case KEY_PGDN:
        keyrc = key;
        returning = 1;
        break;

```

```

case KEY_cLEFT:
case KEY_BACKSPACE:
    if (x==pos_x)
        {keyrc = (key==KEY_cLEFT?KEY_LEFT:key);returning = 1;}
    else
    {
        x = --x<pos_x?pos_x:x;
        if (x==pos_x+2||x==pos_x+5) x--;
    }
    break;

case KEY_cRIGHT:
    if (x==pos_x+siz_x-1)
        {keyrc = KEY_RIGHT;returning =1;}
    else
    {
        x = ++x>pos_x+siz_x?pos_x+siz_x-1:x;
        if (x==pos_x+2||x==pos_x+5) x++;
    }
    break;

case KEY_HOME:
    x = pos_x;
    break;

case KEY_END:
    x = pos_x+siz_x-1;
    break;

default:
    if (key>='0'&&key<='9')
    {
        waddch (win,key);
        x = ++x>pos_x+siz_x?pos_x+siz_x-1:x;
        if (x==pos_x+2) x++;
        if (x==pos_x+5) x=pos_x+8;
    }
}

if (returning)
{
    if (keyrc != -1)
    {
        rfromwin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
        if ( check_date(finp_info->out_buff) )
        {
            returning=usrkey=0;keyrc=-1;
            continue;
        }
    }
}
break;
}
}

```



```

wattrset (win,color_info->text_color);
wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
wrefresh (win);
gotoyx (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
return (keyrc);
}
return (-1);
}

int time_input(win,finp_info,key_info,color_info)
WINDOW *win;
FIELD_INP_INFO *finp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
{
    int cpos_x,cpos_y,csize_b,csize_e,cntr;
    int key,siz_x,pos_y,pos_x;
    returning=0,usrkey=0,keyrc = -1;

    siz_x = TIME_LEN;
    pos_y = finp_info->position.y;
    pos_x = finp_info->position.x;

    getcurpos (&cpos_y,&cpos_x);
    get_cursor_size (&csize_b,&csize_e);
    set_cursor_size (initial_cur_beg,initial_cur_end);

    if (finp_info != NULL_PTR && finp_info->out_buff != NULL_PTR)
    {
        int x=pos_x;

        if (pos_y<0 || pos_x<0)
            return (-1);

        wattrset (win,color_info->sel_color);
        wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
        mvwaddch (win,pos_y,pos_x+2,':');
        wsptozero (win,pos_y,pos_x,siz_x);

        while (1)
        {
            wmove (win,pos_y,x);
            wrefresh (win);

            key = GETKEY (win);
            for (cntr=0;cntr<key_info->no_of_keys;cntr++)
            {
                if (key==key_info->key_list[cntr])
                {

```

```

    keyrc = key;
    usrkey = 1;
    break;
}
}
if (usrkey) break;

switch (key)
{
case KEY_RETURN:
    keyrc = 0;
    returning = 1;
    break;

case SEL_REQ_KEY:
    if (finp_info->inp_style==ENTR_AND_SEL)
    {
        if ( (keyrc=scl_procs[finp_info->scl_style](win,
                                                    finp_info,
                                                    key_info,
                                                    color_info)) != -1)
        {
            if (keyrc!=0)
            {returning =1;}
            else
            {
                wtowin (win,pos_y,pos_x,siz_x,finp_info->out_buff);
                keyrc = -1;
                x = pos_x;
            }
        }
        break;
    }

case KEY_ESC:
    keyrc = -1;
    returning = 1;
    break;

case KEY_UP:
case KEY_DOWN:
case KEY_LEFT:
case KEY_RIGHT:
case KEY_PGUP:
case KEY_PGDN:
    keyrc = key;
    returning = 1;
    break;

case KEY_cLEFT:
case KEY_BACKSPACE:
    if (x==pos_x)
    {keyrc = (key==KEY_cLEFT?KEY_LEFT:key);returning = 1;}

```



```

else
{
    x = --x < pos_x ? pos_x : x;
    if (x == pos_x + 2) x--;
}
break;

return (-1);

case KEY_CRIGHT:
    if (x == pos_x + siz_x - 1)
        {keyrc = KEY_RIGHT; returning = 1;}
    else
    {
        x = ++x > pos_x + siz_x ? pos_x + siz_x - 1 : x;
        if (x == pos_x + 2) x++;
    }
    break;

case KEY_HOME:
    x = pos_x;
    break;

case KEY_END:
    x = pos_x + siz_x - 1;
    break;

default:
    if (key >= '0' && key <= '9')
    {
        waddch (win, key);
        x = ++x > pos_x + siz_x ? pos_x + siz_x - 1 : x;
        if (x == pos_x + 2) x++;
    }
    if (returning)
    {
        if (keyrc != -1)
        {
            rfromwin (win, pos_y, pos_x, siz_x, finp_info->out_buff);
            if (check_time(finp_info->out_buff))
            {
                returning = usrkey = 0; keyrc = -1;
                continue;
            }
        }
    }
    break;
}

if (keyrc != -1)
{
    rfromwin (win, pos_y, pos_x, siz_x, finp_info->out_buff);
}
wattrset (win, color_info->text_color);

```



```

/*****
*.....DISPLAY ROUTINES.....*
*****/

FIELD_INP_INFO *finp_info;

{
    wtowin      (win,
                 finp_info->position.y,
                 finp_info->position.x,
                 finp_info->size.x,
                 finp_info->out_buff);
}

alpha_display(win,finp_info)
WINDOW      *win;
FIELD_INP_INFO *finp_info;
{
    wtowin      (win,
                 finp_info->position.y,
                 finp_info->position.x,
                 finp_info->size.x,
                 finp_info->out_buff);
}

/*****
*.....GENERAL DISPLAY.....*
*****/

num_display(win,finp_info)
WINDOW      *win;
FIELD_INP_INFO *finp_info;
{
    field_display(win,finp_info);
    arrange_num (finp_info->out_buff,
                 finp_info->size.x,
                 finp_info->size.mant_len);
    wtowin      (win,
                 finp_info->position.y,
                 finp_info->position.x,
                 finp_info->size.x,
                 finp_info->out_buff);
}

date_display(win,finp_info)
WINDOW      *win;
FIELD_INP_INFO *finp_info;
{
    wtowin      (win,
                 finp_info->position.y,
                 finp_info->position.x,
                 DATE_LEN,
                 finp_info->out_buff);
}

```

```

time_display(win,finp_info)
WINDOW      *win;
FIELD_INP_INFO *finp_info;
{
    wtowin    (win,
               finp_info->position.y,
               finp_info->position.x,
               TIME_LEN,
               finp_info->out_buff);
}

```

```

/*****
*.....GENERAL DISPLAY ROUTINE.....*
*****/

```

```

int field_display(win,finp_info)
WINDOW      *win;
FIELD_INP_INFO *finp_info;
{
    if (finp_info->data_type<NO_OF_DATA_TYPES)
    {
        if (finp_info->inp_style==SEL_ONLY&&finp_info->out_buff!=NULL_PTR)
        {
            if (finp_info->selection!=NULL_PTR)
            {
                sprintf(finp_info->out_buff,"%s",
                        finp_info->sel_info.sel_list[(finp_info->selection)]);
            }
            else
            {
                sprintf(finp_info->out_buff,"%s",
                        finp_info->sel_info.sel_list[0]);
            }
        }
        return (disp_procs[finp_info->data_type](win,finp_info));
    }
    return (-1);
}

```

```

switch(chc)
{

```

```

case 'C': /* Cancel current selection */
case 'D': /* Delete current selection */
break;

```


[illegible]

```

break;

case 'A': /* Move up one line */
    decrease_line();
    break;

case 'B': /* Move Down one line */
    increase_line();
    break;

case 'C': /* Move right one column */
    increase_col();
    break;

case 'D': /* Move left one column */
    decrease_col();
    break;

case 'H': /* Move to home position */
    curx=cury=0;
    gotoyx(cury,curx);
    break;

case 'I': /* Move up with scroll */
    decrease_line();
    break;

case 'Y': /* Cursor addressing */
    if ((tmp1=COMRECEIVECHR())==CAN || tmp1==SUB) break;
    if ((tmp2=COMRECEIVECHR())==CAN || tmp2==SUB) break;
    cury=tmp1-' ';
    curx=tmp2-' ';
    cury=cury>=0?cury:0;
    cury=cury<TERMLINES?cury:0;
    curx=curx>=0?curx:0;
    curx=curx<COLS?curx:0;
    gotoyx(cury,curx);
    break;

case 'P': /* Select graphics char set */
    break;

case 'G': /* Select std ASCII character set */
    break;

case 'J': /* Erase from cursor to end of display */
    etoeod();
    break;

case 'X': /* Erase from cursor to end of line */
    etoeol();
    break;

case 'V': /* Print cursor line */

```



```

        break;

case 'J': /* Print screen */
    break;

case 'W': /* Transparent print mode on */
    break;

case 'X': /* Transparent print mode off */
    break;

case '^': /* Copy print mode on */
    break;

case '_': /* Copy print mode off */
    break;

case '=': /* Keypad application mode on */
    break;

case '>': /* Keypad application mode off */
    break;

case '<': /* Enter VT100 mode */
    break;

case 'Z': /* Identify terminal */
    _comsendstr(PORT, "52\r");
    break;
    }
}
else
{
    if (chr!=NUL && chr!=DEL) displaychar(chr);
}
}

/*****
*.....VT-52 KEYBOARD HANDLER.....*
*****/

vt52_keyboard_handler(chr)
int chr;
{
    switch(chr)
    {
        case KEY_DEL:
            _comsendchr(PORT, DEL);
            break;

```

```

break;

case KEY_UP:
    _comsendstr(PORT, "\033A");
    break;

case KEY_DOWN:
    _comsendstr(PORT, "\033B");
    break;

case KEY_LEFT:
    _comsendstr(PORT, "\033D");
    break;

case KEY_RIGHT:
    _comsendstr(PORT, "\033C");
    break;

case KEY_F1:
    _comsendstr(PORT, "\033OP");
    break;

case KEY_F2:
    _comsendstr(PORT, "\033OQ");
    break;

case KEY_F3:
    _comsendstr(PORT, "\033OR");
    break;

case KEY_F4:
    _comsendstr(PORT, "\033OS");
    break;

default:
    if (chr < 256) _comsendchr(PORT, chr);
}
}

/*****
*.....VT-100 DISPLAY HANDLER.....*
*****/

vt100_display_handler(chr)
char chr;
{
char termchr;
switch (chr)
{
case BEL: /*...Sound Bell.....*/ Enable top half of S-H, P-W line...*/
    if (vt100bellon) beep();

```



```

break;

case ESC: /*...Escape Sequence.....*/
    chr=COMRECEIVECHR();
    switch(chr)
    {
        case CAN: /*...Cancel escape sequence.....*/
        case SUB:
            break;

        case '=': /*...Keypad application mode on..*/
            break;

        case '>': /*...Keypad application mode off..*/
            break;

        case '(': /*...Change G0.....*/
            switch(COMRECEIVECHR())
            {
                case 'O': /*..ANSI Graphics..*/
                case 'A': /*..UK ANSI.....*/
                case 'B': /*..Standard ANSI..*/
                    break;
            }
            break;

        case ')': /*...Change G1.....*/
            switch(COMRECEIVECHR())
            {
                case 'O': /*..ANSI Graphics..*/
                case 'A': /*..UK ANSI.....*/
                case 'B': /*..Standard ANSI..*/
                    break;
            }
            break;

        case '#': /*...Select Bell volume or
                    display screen alignment pattern or
                    double hight and width printing..*/
            switch(COMRECEIVECHR())
            {
                case '?': /*...Loud.....*/
                case '>': /*...Medium.....*/
                case '=': /*...Low.....*/
                    vt100bellon=1;
                    break;

                case '<': /*...Off.....*/
                    vt100bellon=0;
                    break;

                case '3': /* Enable top half of D-H, D-W line....*/
                case '4': /* Enable bottom half of D-H, D-W line..*/
            }
    }

```

```

case '5': /* Enable S-H, S-W line.....*/
if (COM1) case '6': /* Enable S-H, D-W line.....*/
case ':': /* Enable top half of D-H, S-W line....*/
/* case ' '; /* Enable bottom half of D-H, S-W line.*/
case '8': /*...Display screen alignment pattern..*/
break;
}
case break; /*...Cursor up one line;scroll...*/
decrease_line();
case ',': /*...Delay processing for 250 ms.*/
break;
case 'D': /*...Delete down one line;scroll..*/
case 'P': /*...Request Product type.....*/
_comsendstr(PORT,"120\r");
break;
case 'X': /*...Scroll to start of next line;scroll..*/
case 'c': /*...Reset terminal to initial state..*/
vt100savedcurx=vt100savedcury=curx=cury=0;
vt100savedattr=curattr=TEXTATTR;
vt100bellon =1;
gotoyx(cury,curx); /*...at cursor position.....*/
etoeod();
break;
case 'A': /*...Request primary attributes..*/
_comsendstr(PORT,"(0)(1)(2)");
break;
case 'B': /*...Restore cursor pos. char set and attr..*/
curx = vt100savedcurx;
cury = vt100savedcury;
curattr = vt100savedattr;
gotoyx(cury,curx);
break;
case '7': /*...Save cursor pos. char set and attr....*/
vt100savedcurx = curx;
vt100savedcury = cury;
vt100savedattr = curattr;
break;
case 'Z': /*...Exit.....*/
_comsendstr(PORT,"(0)(1)(2)");
break;
case 'S': /* CSI with no parameters */
switch(ch)
{
case 'I': /* CSI Sequence 1 */
_comsendstr(PORT,"(0)(1)(2)");
break;
case 'L': /* CSI Sequence with one parameter */
switch(ch)

```



```

case 'I':
    if (COMRECEIVECHR()=='p')
    {
        /* Reset all terminal modes */
    }
    break;

case 'M': /*...Cursor up one line;scroll...*/
    decrease_line();
    break;

case 'D': /*...Cursor down one line;scroll...*/
    increase_line();
    break;

case 'E': /*...Cursor to start of next line;scroll...*/
    curx=0;
    increase_line();
    break;

case 'H': /*...Set tab stop at cursor position.....*/
    break;

case 'Z': /*...Request primary attributes...*/
    _comsendstr(PORT,"033[?1;2c");
    break;

case '8': /*...Restore cursor pos. char set and attr */
    curx = vt100savedcurx;
    cury = vt100savedcury;
    curattr = vt100savedattr;
    gotoyx(cury,curx);
    break;

case '7': /*...Save cursor pos. char set and attr....*/
    vt100savedcurx = curx;
    vt100savedcury = cury;
    vt100savedattr = curattr;
    break;

case '[': /*...CSI.....*/
    termchr=vt100_get_decimals();
    switch(VT100NOOFDECIMALS)
    {
        case 0: /* CSI with no parameters */
            switch(termchr)
            {
                case '?': /* CSI? Sequence */
                    termchr=vt100_get_decimals();
                    switch(VT100NOOFDECIMALS)
                    {
                        case 1: /* CSI? sequence with one parameter */
                            switch(termchr)

```

```

{
    case 'i': /* Media copy part two */
        switch(vt100decimals[1])
        {
            case 1: /* Copy cursor line to printer port */
            case 3: /* Copy cursor line to data port */
            case 4: /* Turn off auxiliary print mode */
            case 5: /* Turn on auxiliary print mode */
                break;
        }
        break;

    case 'h': /* Set modes */
    case 'l': /* Reset modes */
        break;
    }
    break;

    default: /* CSI? Sequence with several parameters */
        switch(termchr)
        {
            case 'h': /* Set modes */
                break;

            case 'l': /* Reset modes */
                break;
        }
        break;

    case 'u': /*...Restore cursor pos. char set and attr */
        curx = vt100savedcurx;
        cury = vt100savedcury;
        curattr = vt100savedattr;
        gotoyx(cury, curx);
        break;

    case 's': /*...Save cursor pos. char set and attr...*/
        vt100savedcurx = curx;
        vt100savedcury = cury;
        vt100savedattr = curattr;
        break;

    case 'C': /*...Cursor right one column....*/
        increase_col();
        break;
    }
    break;

    case 1: /* CSI with one parameter */
        switch(termchr)
        {
            case 'C':

```



```

if (vt100decimals[1]==0)
{
    /* Request primary attributes */
    _comsendstr(PORT, "\033[?1;2c");
}
break;

case 'C': /* Cursor right n columns */
    curx+=vt100decimals[1];
    if (curx>=COLS)
    {
        curx=COLS-1;
        increase_col();
    }
    gotoyx(cury, curx);
    break;

case 'D': /* Cursor left n columns */
    curx-=vt100decimals[1];
    if (curx<0)
    {
        curx=0;
        decrease_col();
    }
    gotoyx(cury, curx);
    break;

case 'F': /* Cursor up n lines and to column 1 */
    curx=0;
case 'A': /* Cursor up n lines */
    cury-=vt100decimals[1];
    if (cury<0)
    {
        cury=0;
        decrease_line();
    }
    gotoyx(cury, curx);
    break;

case 'E': /* Cursor down n lines and to column 1 */
    curx=0;
case 'B': /* Cursor down n lines */
    cury+=vt100decimals[1];
    if (cury>=TERMLINES)
    {
        cury=TERMLINES-1;
        increase_line();
    }
    gotoyx(cury, curx);
    break;

case 'G': /* Cursor to n.th column */
    curx=vt100decimals[1]-1;

```

```

if (curx<0) curx=0;
if (curx>=COLS) curx=COLS-1;
gotoyx(cury,curx);
break;

case 'I': /* Move cursor forward n tab stops */
    for (;vt100decimals[1]>0;vt100decimals[1]--)
        movetotabforward();
    break;

case 'Z': /* Move cursor backward n tab stops */
    for (;vt100decimals[1]>0;vt100decimals[1]--)
        movetotabbackward();
break;

case '@': /* Insert n NULL characters */
    instchr(vt100decimals[1]);
    break;

case 'L': /* Insert n NULL lines */
    instline(vt100decimals[1]);
    break;

case 'P': /* Delete n characters */
    delchr(vt100decimals[1]);
    break;

case 'M': /* Delete n lines */
    Umdelline(vt100decimals[1]);
    break;

case 'J': /* Erase in display */
    switch(vt100decimals[1])
    {
        case 0: /* From cursor to end of screen */
            etoeod();
            break;

        case 1: /* From beginning through cursor */
            etobod();
            break;

        case 2: /* Entire screen */
            escr();
            break;
    }
    break;

case 'K': /* Erase in line */
    switch(vt100decimals[1])
    {
        case 0: /* From cursor to end of line */
            etoeol();

```



```

        break;

case 1: /* From start of line through cursor */
    etobol();
    break;

case 2: /* Entire line */
    eline();
    break;
}
break;

case 'X': /* Erase n characters beginning at cursor column */
    echars(vt100decimals[1]);
    break;

case 'i': /* Media copy part one */
    switch(vt100decimals[1])
    {
        case 0: /* Copy entire screen to printer port */
        case 2: /* Copy entire screen to data port */
        case 4: /* Turn off transparent print mode */
        case 5: /* Turn on transparent print mode */
            break;
    }
    break;

case 'g': /* Clear tab stops */
    switch(vt100decimals[1])
    {
        case 0: /* Clear tab stop at cursor position */
        case 3: /* Clear all tab stops */
            break;
    }

case 'n': /* Request status report */
    switch(vt100decimals[1])
    {
        case 5: /* Terminal status */
            _comsendstr(PORT, "\033[0n");
            break;

        case 6: /* Cursor position */
            sprintf(vt100buff,
                "\033[%d;%dR",
                cury+1, curx+1);
            _comsendstr(PORT, vt100buff);
            break;

        case 15: /* Printer status */
            _comsendstr(PORT, "\033[?11n"); /* Busy */
            break;
    }
}

```

```

break;

case 'U': /*...Display Next page...*/
break;

case 'V': /*...Display Previous page...*/
break;

default:
break;

if (curx==0 || curx==COLS) displaychar(0);

case 'h': /* Set modes */
case 'l': /* Reset modes */
break;
}
break;

case 2: /* CSI with two parameters.... */
switch(termchr)
{
case 'H':
case 'f': /*...Cursor to line; col.....*/
cury=vt100decimals[1]-1;
curx=vt100decimals[2]-1;
curx=curx<0?0:curx;
cury=cury<0?0:cury;
curx=curx>=COLS?COLS-1:curx;
cury=cury>=TERMLINES?TERMLINES-1:cury;
gotoyx(cury,curx);
break;

case 'r': /*...Define scrolling region...*/
break;
}
break;

default: /* CSI with several parameters */
switch(termchr)
{
case 'h': /* Set modes */
break;

case 'l': /* Reset modes */
break;

case 'm': /* Define video attribute */
break;

case 'q': /* Control simulated LEDs */
break;
}
}
break;
}
break;
}
break;

case SI: /* Select G0 Character Set */

```



```

    case break;
    _comsendstr(PORT, "\033[08");

case SO: /* Select G1 Character Set */
    break;

case KEY_F5:
    _comsendstr(PORT, "\033[15");

default:
    if (chr!=NUL && chr!=DEL) displaychar(chr);
}
} case KEY_F6:
    _comsendstr(PORT, "\033[17");
    break;

case KEY_F7:
    _comsendstr(PORT, "\033[18");

/*****
*.....VT-100 KEYBOARD HANDLER.....*
*****/
_comsendstr(PORT, "\033[18");

vt100_keyboard_handler(chr)
int chr;
{
switch(chr)
{
case KEY_UP:
    _comsendstr(PORT, "\033[A");
    break;

case KEY_DOWN:
    _comsendstr(PORT, "\033[B");
    break;

case KEY_LEFT:
    _comsendstr(PORT, "\033[D");
    break;

case KEY_RIGHT:
    _comsendstr(PORT, "\033[C");
    break;

case KEY_F1:
    _comsendstr(PORT, "\033OP");
    break;

case KEY_F2:
    _comsendstr(PORT, "\033OQ");
    break;

case KEY_F3:
    _comsendstr(PORT, "\033OR");
    break;

```

```
case KEY_F4:
    _comsendstr(PORT, "\\033OS");
    break;
```

```
case KEY_F5:
    _comsendstr(PORT, "\\033[16~");
    break;
```

```
case KEY_F6:
    _comsendstr(PORT, "\\033[17~");
    break;
```

```
case KEY_F7:
    _comsendstr(PORT, "\\033[18~");
    break;
```

```
case KEY_F8:
    _comsendstr(PORT, "\\033[19~");
    break;
```

```
case KEY_F9:
    _comsendstr(PORT, "\\033[20~");
    break;
```

```
case KEY_F10: /* F10 Key */
    _comsendstr(PORT, "\\033[21~");
    break;
```

```
case KEY_F11: /* F11 Key */
    _comsendstr(PORT, "\\033[22~");
    break;
```

```
case KEY_F12:
    _comsendstr(PORT, "\\033[23~");
    break;
```

```
case KEY_F13:
    _comsendstr(PORT, "\\033[24~");
    break;
```

```
case KEY_F14:
    _comsendstr(PORT, "\\033[25~");
    break;
```

```
case KEY_F15:
    _comsendstr(PORT, "\\033[26~");
    break;
```

```
case KEY_F16: /* F16 Key */
    _comsendstr(PORT, "\\033[27~");
    break;
```



```
case KEY_F9:
    _comsendstr(PORT, "\033[20~");
    break;
```

```
case KEY_F10:
    _comsendstr(PORT, "\033[21~");
    break;
```

```
case KEY_sF1:
    _comsendstr(PORT, "\033[23~");
    break;
```

```
case KEY_sF2:
    _comsendstr(PORT, "\033[24~");
    break;
```

```
case KEY_sF3:
    _comsendstr(PORT, "\033[25~");
    break;
```

```
case KEY_sF4:
    _comsendstr(PORT, "\033[26~");
    break;
```

```
case KEY_sF5: /* HELP Key */
    _comsendstr(PORT, "\033[28~");
    break;
```

```
case KEY_sF6: /* DO Key */
    _comsendstr(PORT, "\033[29~");
    break;
```

```
case KEY_sF7:
    _comsendstr(PORT, "\033[31~");
    break;
```

```
case KEY_sF8:
    _comsendstr(PORT, "\033[32~");
    break;
```

```
case KEY_sF9:
    _comsendstr(PORT, "\033[33~");
    break;
```

```
case KEY_sF10:
    _comsendstr(PORT, "\033[34~");
    break;
```

```
case KEY_HOME: /* FIND Key */
    _comsendstr(PORT, "\033[1~");
    break;
```

```

caseKEY_INS: /* INSERT HERE Key */
    _comsendstr(PORT, "\033[2~");
    break;

caseKEY_DEL: /* REMOVE Key */
    _comsendstr(PORT, "\033[3~");
    break;

caseKEY_END: /* SELECT Key */
    _comsendstr(PORT, "\033[4~");
    break;

caseKEY_PGUP: /* PREVSCREEN Key */
    _comsendstr(PORT, "\033[5~");
    break;

caseKEY_PGDN: /* NEXTSCREEN Key */
    _comsendstr(PORT, "\033[6~");
    break;

default:
    if (chr<256) _comsendchr(PORT, chr);
}

}

#define KM_DATA 1
#define KM_EOF 254
#define KM_CANCEL 255
#define KM_EOF 254
#define KM_BACK 255
#define KM_TIMEOUT 255
#define KM_BREAK 254

vt100_get_decimals()
{
    int cnt=0;
    char *ptr=vt100buff, terminator='';
    vt100buff[0] = (char) NULL;
    vt100decimals[0] = 0;
    while (terminator=='&&cnt<16)
    {
        while ( (*ptr=COMRECEIVECHR())>='0' && (*ptr)<='9' ) ptr++;
        terminator = *ptr;
        *ptr = (char) NULL;
        ptr = vt100buff;
        if (vt100buff[0] != (char) NULL)
        {
            vt100decimals[++cnt]=atoi(vt100buff);
            vt100decimals[0]=cnt;
        }
    }
    return(terminator);
}

```



```

*****
**
**                                XMODEM.C                                **
**
**                                XMODEM Error Free File Transfer Protocol Module        **
**
**                                Creation : 15/3/1991 by Umit KALKAN                **
**
*****
*****
*****
#define GENERATOR_POLINOMIAL      0x8810
#define XMODEM_DATA_LEN          128

#define XM_MAX_RETRIES            10
#define XM_TIMEOUT_INTERVAL      3 /* seconds */

#define XM_OK                     0
#define XM_DATA                   1
#define XM_EOT                    EOT
#define XM_CANCEL                 CAN
#define XM_ACK                    ACK
#define XM_NACK                   NAK
#define XM_TIMEOUT                255
#define XM_BREAK 254
#define XM_CRC                    253
#define XM_ERR                    252

unsigned char  xmdatabuff[XMODEM_DATA_LEN+2];
unsigned char  XM_RC;

```

```

/*****
*.....WAIT FOR COMM PORT TO HIT.....*
*****/

```

```

unsigned int swap_bytes(value)

```

```

int waitcomm(duration)

```

```

    int    duration;

```

```

    { return ((value&0xFF) << 8) | ((value>>8) && 0xFF); }

```

```

    long etime,ctime;

```

```

    int rc;

```

```

    etime = time(&ctime)+duration;

```

```

    while (time(&ctime)<etime)
    {

```

```

        if (COMHIT()!=0) return(1);

```

```

    } .....CALCULATE CRC VALUE OF A BLOCK.....

```

```

    return (0);

```

```

}

```

```

unsigned int calc_crc16(unsigned char *buff, unsigned int len)

```

```

unsigned char *buff;

```

```

unsigned int len;

```

```

{
    unsigned int crc_value=0,

```

```

    crc_word;

```

```

    char_ptr;

```

```

    bit_ptr;

```

```

/*****

```

```

*.....RECEIVE PROCEDURE FOR XMODEM MODULE.....*

```

```

*****/

```

```

for (bit_ptr=0; bit_ptr<8; bit_ptr++)

```

```

unsigned char X_COMRECEIVECHR()

```

```

{
    unsigned char chr;

```

```

    while (comindest==cominsrc) {;;};

```

```

    chr = cominbuff[cominsrc];

```

```

    cominsrc= rotate(cominsrc);

```

```

    return (chr);

```

```

}
    crc_value = crc_value << 1;

```

```

}

```

```

}

```

```

}

```

```

return (crc_value);

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```

```

}

```



```

/*****
*.....SWAP BYTES OF AN INTEGER.....*
*****/

mov     blocknumber,0

unsigned int swap_bytes(value)
unsigned int value;
{
    mov     cx,0
    return ((value&0xFF) << 8) | ((value&0xFF00) >> 8);
}

mov     di,0
mov     al,al
mov     al,ah
mov     ah,al

/*****
*.....CALCULATE CRC VALUE OF A BLOCK.....*
*****/

mov     cx,0

unsigned int calc_crc(buff,byte_count)
unsigned char *buff;
unsigned int byte_count;
{
    mov     ax,0
    unsigned int crc_value=0,
                curr_word,
                char_cntr,
                bit_cntr;
    for (char_cntr=0;char_cntr<byte_count;char_cntr+=2)
    {
        loopb
        curr_word = (buff[char_cntr]<<8) | buff[char_cntr+1];
        for (bit_cntr=0;bit_cntr<16;bit_cntr++)
        {
            rol     cx,1
            if ( (((curr_word=curr_word<<1) ^ crc_value) & 0x8000) )
            {
                mov     crc_value = (((crc_value ^ GENERATOR_POLINOMIAL) << 1) | 1);
            }
            else
            {
                mov     crc_value = crc_value << 1;
            }
        }
    }
    return (crc_value);
}

```

```

/*****
PROC .....WAIT FOR ACK OR NACK.....
    mov     di,crcadr
    mov     blocknumber,0
    mov     bcc,0
    mov     bx,0
    mov     cx,0
loopcx: mov     ax,word ptr es:[di]
    inc     di
    inc     di
    mov     al_,al
    mov     al,ah
    mov     ah,al_
    mov     save,ax
    mov     dx,0
loop1:
    mov     ax,save
    xor     ax,bcc
    shl     save,1
    and     ax,10000000000000000b
    cmp     ax,0
    jnz     bir
    shl     bcc,1
    inc     dx
    cmp     dx,16
    jne     loop1
    jmp     loop2
bir:
    xor     bcc,10001000000010000b
    rol     bcc,1
    or      bcc,1
    inc     dx
    cmp     dx,16
    jne     loop1
loop2:
    inc     cx
    inc     bx
    inc     bx
    cmp     cx,64
    jne     loopcx
    ret
crc     ENDP

```

```

*****/

```



```

/*****
*.....WAIT FOR ACK OR NACK.....*
*****/

```

```

unsigned char XMwaitacknack()
{
    unsigned char *buff;
    int rc;
    unsigned char chr;
    if (waitcomm(XM_TIMEOUT_INTERVAL)==1)
    {
        switch ( (chr=X_COMRECEIVECHR()) )
        {
            case XM_ACK:
            case XM_CANCEL:
                return(chr);
            default:
                return(XM_NACK);
        }
    }
    return(XM_TIMEOUT);
}

```

```

/*****
*.....SEND A DATA BLOCK TO REMOTE.....*
*****/

```

```

unsigned char XMsenddata(buff,blkno)
unsigned char *buff,blkno;
{
    unsigned int cntr=0,crc;
    crc=calc_crc(buff,XMODEM_DATA_LEN);
    _comsendchr(PORT,XM_DATA);
    _comsendchr(PORT,blkno);
    _comsendchr(PORT,"blkno");
    while (cntr<XMODEM_DATA_LEN) {_comsendchr(PORT,buff[cntr]);cntr++;}
    _comsendchr( PORT, (char) ((crc&0xFF00)>>8) );
    _comsendchr( PORT, (char) (crc&0xFF) );
    return (XMwaitacknack());
}

```

```

/*****
*.....RECEIVE A DATA BLOCK FROM REMOTE.....*
*****/

unsigned char XMreceivedata(buff,blkno)
unsigned char *buff;
unsigned char *blkno;
{
    switch (waitcomm(XM_TIMEOUT_INTERVAL))
    {
        unsigned int cntr=0,crc=0,badblkno=0;
        while (cntr<XMODEM_DATA_LEN+4)
        {
            if ( (char)_COMRECEIVECHR() != 0 ) { cntr=XM_DATA_LEN; }
            switch (waitcomm(XM_TIMEOUT_INTERVAL))
            {
                default:
                case XM_DATA_LEN+1: /* Comm port hit */
                    switch (cntr)
                    {
                        {
                            case XMODEM_DATA_LEN+2:
                                crc = (X_COMRECEIVECHR()<<8);
                                break;
                            case XMODEM_DATA_LEN+3:
                                crc |= X_COMRECEIVECHR();
                                break;
                            case 0:
                                *blkno = X_COMRECEIVECHR();
                                break;
                            case 1:
                                if (X_COMRECEIVECHR() != ~(*blkno)) badblkno=1;
                                break;
                            default:
                                buff[cntr-2]=X_COMRECEIVECHR();
                                break;
                            default:
                                return (XM_TIMEOUT);
                        }
                    }
                    cntr++;
                }
            if (crc!=calc_crc(buff,XMODEM_DATA_LEN) || badblkno==1)
            {
                _comsendchr(PORT,XM_NACK);
                return (XM_CRC);
            }
            else
            {
                _comsendchr(PORT,XM_ACK);
                return(XM_OK);
            }
        }
        if (waitcomm(XM_TIMEOUT_INTERVAL)!=-1) return(X_COMRECEIVECHR());
        return(XM_TIMEOUT);
    }
}

```



```

/*****
*.....ESTABLISH PROTOCOL CONNECTION FOR XMIT.....*
*****/
.....CLOSE RECEIVE PROTOCOL CONNECTION.....

unsigned char XMestablishxmit()
{
    unsigned char chr;
    switch (waitcomm(XM_TIMEOUT_INTERVAL))
    {
        case 1: /* Comm port hit */
            if ( (chr=X_COMRECEIVECHR())=='C' || chr==XM_NACK)
                return(XM_OK); else return(XM_ERR);
            default:
                return(XM_TIMEOUT);
    }
}

/*****
*.....ESTABLISH PROTOCOL CONNECTION FOR RECEIVE.....*
*****/
char chr=0;

unsigned char XMestablishreceive()
{
    _comsendchr(PORT, 'C');
    if (waitcomm(XM_TIMEOUT_INTERVAL)==1) return(XM_OK);
    _comsendchr(PORT, XM_NACK);
    switch (waitcomm(XM_TIMEOUT_INTERVAL))
    {
        case 1: /* Comm port hit */
            return(XM_OK);
            default:
                return(XM_TIMEOUT);
    }
}

/*****
*.....CLOSE XMIT PROTOCOL CONNECTION.....*
*****/
.....SETUP DATA TERMINAL PORT TO XMIT.....

Xmitfile:file:

unsigned char XMclosexmit()
{
    _comsendchr(PORT, XM_EOT);
    if (waitcomm(XM_TIMEOUT_INTERVAL)==1) return(X_COMRECEIVECHR());
    return(XM_TIMEOUT);
}
.....SETUP DATA TERMINAL PORT TO RECEIVE.....

```

```

/*****
*.....CLOSE RECEIVE PROTOCOL CONNECTION.....*
*****/

```

```

unsigned char XMclosereceive()

```

```

{
    _comsendchr(PORT,XM_ACK);
    return(XM_OK);
}

```

```

/*****
*.....CANCEL XMODEM PROTOCOL TRANSMISSION.....*
*****/

```

```

XMcancel()

```

```

{
    char cntr=0;
    while (cntr<10)
    {
        _comsendchr(PORT,XM_CANCEL);
        cntr++;
    }
    return(XM_OK);
}

```

```

/*****
*.....XMIT A FILE USING XMODEM PROTOCOL.....*
*****/

```

```

XMxmitfile(filename)

```

```

unsigned char *filename;

```

```

{
    int handle,ecntr,dcntr,hs;
    hs=setup_data.terminal_settings.hshake;
    setup_data.terminal_settings.hshake=0;
    if ((handle=open(filename,O_RDONLY|O_BINARY)) != -1)
    {
        unsigned char blkno=0;

        hs=setup_data.terminal_settings.hshake;
        setup_data.terminal_settings.hshake=0;
    }
}

```



```

    XMcancel();
    FLUSHCOMBUFF();
    FLUSHCOMBUFF();
    ecntr=0;
    while ( ecntr<XM_MAX_RETRIES && ((XM_RC=XMestablishxmit())!=XM_OK) )
    {
        return(-1);
        ecntr++;
        if (kbhit())
        {
            XMcancel();
            close(handle);
            XM_RC=XM_CANCEL;
            setup_data.terminal_settings.hshake=hs;
            return(-1);
        }
    }
    if (kbhit())
    {
        XMcancel();
        close(handle);
        setup_data.terminal_settings.hshake=hs;
        return(-1);
    }
    if (XM_RC!=XM_OK) {close(handle);setup_data.terminal_settings.hshake=hs;return (-1);}

    while ((dcntr=read(handle,xmdatabuff,XMODEM_DATA_LEN))!=0 && dcntr!=-1)
    {
        for (;dcntr<XMODEM_DATA_LEN;dcntr++) xmdatabuff[dcntr] = (char) NULL;
        blkno++;
    }
    ecntr=0;
    do
    {
        ecntr++;
        XM_RC=XMsenddata(xmdatabuff,blkno);
        if (kbhit())
        {
            XMcancel();
            sleep(2);
            FLUSHCOMBUFF();
            close(handle);
            setup_data.terminal_settings.hshake=hs;
            return(-1);
        }
    }
    if (XM_RC==XM_CANCEL)
    {
        close(handle);
        setup_data.terminal_settings.hshake=hs;
        return(-1);
    }
} while (ecntr<XM_MAX_RETRIES && XM_RC!=XM_ACK);
if (kbhit())
{

```

```

        XMcancel();
        sleep(2);
        FLUSHCOMBUFF();
        close(handle);
        setup_data.terminal_settings.hshake=hs;
        return(-1);
    }
    if (XM_RC!=XM_ACK) {close(handle);setup_data.terminal_settings.hshake=hs;return (-1);}
}

ecntr=0;
while ( ecntr<XM_MAX_RETRIES && ((XM_RC=XMclosexmit())!=XM_ACK) ) ecntr++;
if (XM_RC!=XM_ACK) {close(handle);setup_data.terminal_settings.hshake=hs;return (-1);}

close (handle);
setup_data.terminal_settings.hshake=hs;
return (0);
}
return (-1);
}

/*****
*.....RECEIVE A FILE USING XMODEM PROTOCOL.....*
*****/

XMreceivefile(filename)
unsigned char *filename;
{
    unsigned char blkno=0;
    int handle,ecntr,hs;

    FLUSHCOMBUFF();

    if ((handle=open(filename,O_WRONLY|O_BINARY|O_CREAT|O_TRUNC,S_IREAD|S_IWRITE)) != -1)
    {
        hs=setup_data.terminal_settings.hshake;
        setup_data.terminal_settings.hshake=0;
        ecntr=0;
        while ( ecntr<XM_MAX_RETRIES && ((XM_RC=XMestablishreceive())!=XM_OK) )
        {
            ecntr++;
            if (kbhit())
            {
                XMcancel();
                close(handle);
                setup_data.terminal_settings.hshake=hs;
                return(-1);
            }

```



```

    }
    if (kbhit())
    {
        XMcancel();
        close(handle);
        setup_data.terminal_settings.hshake=hs;
        return(-1);
    }
    if (XM_RC!=XM_OK) {close(handle);setup_data.terminal_settings.hshake=hs;return (-1);}

do
{
    switch (waitcomm(XM_TIMEOUT_INTERVAL))
    {
        case 1: /* Comm port hit */
            if ((XM_RC=X_COMRECEIVECHR())==XM_DATA)
            {
                ecntr=0;
                do
                {
                    ecntr++;
                    XM_RC=XMreceivedata(xmdatabuff,&blkno);
                    if (XM_RC==XM_CANCEL)
                    {
                        close(handle);
                        setup_data.terminal_settings.hshake=hs;
                        return(-1);
                    }
                } while (ecntr<XM_MAX_RETRIES && XM_RC!=XM_OK);
                if (kbhit())
                {
                    XMcancel();
                    close(handle);
                    setup_data.terminal_settings.hshake=hs;
                    return(-1);
                }
            }
            if (kbhit())
            {
                XMcancel();
                close(handle);
                setup_data.terminal_settings.hshake=hs;
                return(-1);
            }
            if (XM_RC==XM_OK)
            {
                lseek(handle,(blkno-1)*XMODEM_DATA_LEN,0);
                write(handle,xmdatabuff,XMODEM_DATA_LEN);
                XM_RC=XM_DATA;
            }
            else
            {
                close(handle);
                setup_data.terminal_settings.hshake=hs;
            }
        }
    }
}

```

```

        return(-1);
    }
}

break;

default:
    XM_RC=XM_TIMEOUT;
    close(handle);
    setup_data.terminal_settings.hshake=hs;
    return(-1);
}

} while (XM_RC==XM_DATA);

if (XM_RC==XM_EOT)
{
    XM_RC=XM_CLOSE_RECEIVE();
    close(handle);
    setup_data.terminal_settings.hshake=hs;
    return(0);
}

return(-1);
}

```

```

arrange_buf(buf);
char *buf;
if (buf!=NULL)
{
    char *ptr=buf+strlen(buf)-1;
    for (i=ptr; i>buf; i--)
    {
        if (*ptr=='\0') *ptr=buf[i];
    }
}

```

```

arrange_buf(buf, len, len);
char *buf;
int len, n_len;
char *ptr=buf+strlen(buf)-1;
int i;
if (buf!=NULL)
{
    n_len=len-strlen(buf);
    sprintf(buf, "%s", buf);
    n_len = strlen(buf);
    if (i>n_len)
    {
        ptr=buf+n_len;
    }
}

```



```

(dpoint=dptr-1;tmpbuff;)
/*****
*****
**
**                                     **
**  if (a_len<len) buff[a_len] DATAMAN.C **
**  for (dcntr=a_len-1;dcntr=dpoint-1;dcntr--;dcntr--) **
**      General Data Types Manipulation Module **
**  if (/chr=tmpbuff[dcntr]); **
**      Creation : 15/3/1991 by Umit KALKAN **
**  **
*****
*****/

for (dcntr=a_len-1;dcntr=dpoint-1;dcntr--;dcntr--)

#include<stdio.h>
#include<nullptr.h>
#include<string.h>
#include<datatype.h>

if (a_len<buff[a_len-1]; buff[a_len-1]='0';
buff[a_len] = (char) 0;

int    days_by_months[13]={0,31,29,31,30,31,30,31,31,30,31,30,31};

rmtrailblanks(buff)
char *buff;
{
if (buff!=NULL_PTR)
{
char*ptr=buff+strlen(buff)-1;
for (;ptr>=buff;ptr--)
if (*ptr==' ') *ptr=(char) NULL; else break;
}
}

arrange_num(buff,len,m_len)
char *buff;
int len,m_len;
{
char tmpbuff[80*25+1],*dptr,chr;
int slen,dpoint,dcntr,scntr;
if (buff!=NULL_PTR && len>=0 && m_len>=0)
{
m_len=m_len>len?len:m_len;
sprintf (tmpbuff,"%s",buff);
slen = strlen(tmpbuff);
if ( (dptr=strchr(tmpbuff,'.'))!=NULL_PTR )

```

```

    {dpoint=dptr-tmpbuff;}
else
    {dpoint=slen;}

char "buff"
if (m_len<len) buff[m_len]='.';
for (dcntr=m_len-1,scntr=dpoint-1;dcntr>=0&&scntr>=0;scntr--)
{
    if ((chr=tmpbuff[scntr])!=' ')
        {buff[dcntr]=chr;dcntr--;}
}
for (;dcntr>=0;dcntr--) buff[dcntr]=' ';

for (dcntr=m_len+1,scntr=dpoint+1;dcntr<len&&scntr<slen;scntr++)
{
    if ((chr=tmpbuff[scntr])!=' ')
        {buff[dcntr]=chr;dcntr++;}
}
for (;dcntr<len;dcntr++) buff[dcntr]='0';
if (m_len>0&&buff[m_len-1]!=' ') buff[m_len-1]='0';
buff[len] = (char) NULL;
}
}

```



```

int check_date(string)
char *string;
{
char *buff=" ";
int month,day,year;
if (strlen(string)>=DATE_LEN)
{
if (string[2]=='/' && string[5]=='/')
{
strncpy(buff,string+3,2);buff[2]=(char) NULL;
if ((month=atoi(buff))>0 && month<13)
{
strncpy(buff,string,2);buff[2]=(char) NULL;
if ((day=atoi(buff))>0 && day<=days_by_months[month])
{
strncpy(buff,string+6,4);buff[4]=(char) NULL;
if ((year=atoi(buff))>0)
{
if ( !((year%4) && month==2 && day>28) )
{return ( 0);}
}
}
}
}
return (-1);
}
}

int check_time(string)
char *string;
{
char *buff=" ";
int hour,minute;
if (strlen(string)>=TIME_LEN)
{
if (string[2]==':')
{
strncpy(buff,string+3,2);buff[2]=(char) NULL;
if ((minute=atoi(buff))>=0 && minute<60)
{
strncpy(buff,string,2);buff[2]=(char) NULL;
if ((hour=atoi(buff))>=0 && hour<24)
{
return ( 0);
}
}
}
}
return (-1);
}

```

```

/*****
*****
**      char init [73];                               **
**      char dial  UMCOMMON.C                          **
**      char qocar[11];                                **
**      Umicomm and Umisrv Programs Common Module      **
**      char conn[11];                                  **
**      Creation : 15/3/1991 by Umit KALKAN            **
**      char conn[11];                                  **
*****
*****
#define DIALDIR_NUM_LEN 31 /* Set including NULL */
#define MAX_DIALDIR_ENTRIES 34

#define XON          DC1
#define XOFF         DC3
#define TEXTATTR (A_BGBLACK|A_FGGREEN);
#define PORT         setup_data.line_settings.port
#define HDX          setup_data.terminal_settings.dupmode
#define ICRXLATION   setup_data.terminal_settings.in_cr_xlation
#define OCRXLATION   setup_data.terminal_settings.out_cr_xlation
#define BSXLATION    setup_data.terminal_settings.bs_xlation
#define LINEWRAP     setup_data.terminal_settings.line_wrap
#define SCROLL       setup_data.terminal_settings.scroll
#define BELLSOUND    setup_data.terminal_settings.bell_sound
#define HSDTR        setup_data.terminal_settings.hshake==1
#define HSONXOFF     setup_data.terminal_settings.hshake==2
#define HSBOTH       setup_data.terminal_settings.hshake==3
#define DIALCMD      setup_data.modem_settings.dial
#define NO_CARRIER setup_data.modem_settings.nocar
#define BUSY         setup_data.modem_settings.busy
#define CONNECT1     setup_data.modem_settings.conn1
#define CONNECT2     setup_data.modem_settings.conn2
#define CONNECT3     setup_data.modem_settings.conn3
#define TERMINALTYPE setup_data.terminal_settings.term_emulation
#define TERMLINES    clineslist[setup_data.terminal_settings.lines]
#define KEYBOARDHANDLER keyboard_handler_procs[TERMINALTYPE]
#define DISPLAYHANDLER display_handler_procs [TERMINALTYPE]

#define COMM_BUFF_LEN 1024

/*****
*.....DATA STRUCTURES.....*
*****/

typedef struct
{
    int    port;
    int    bps;
    int    parity;
    int    dbits;
    int    sbits;}    _line_settings;

```



```

typedef struct
{
    char init [73];
    char dial [21];
    char nocar[21];
    char busy [21];
    char conn1[21];
    char conn2[21];
    char conn3[21];} _modem_settings;

#define DIALDIR_NAME_LEN 16 /* Not including NULL */
#define DIALDIR_NUM_LEN 21 /* Not including NULL */
#define MAX_DIALDIR_ENTRIES 34
typedef struct
{
    char names [MAX_DIALDIR_ENTRIES][DIALDIR_NAME_LEN+1];
    char numbers[MAX_DIALDIR_ENTRIES][DIALDIR_NUM_LEN+1 ];
} _dialdir_settings;

/*.....SETUP DATA.....*/
typedef struct
{
    int term_emulation;
    int dupmode;
    int hshake;
    int in_cr_xlation;
    int out_cr_xlation;
    int bs_xlation;
    int lines;
    int line_wrap;
    int scroll;
    int bell_sound;} _terminal_settings;

#define OFFSET_LINESETTINGS 0
#define OFFSET_MODEMSETTINGS sizeof(_line_settings)
#define OFFSET_DIALDIRSETTINGS OFFSET_MODEMSETTINGS+sizeof(_modem_settings)
#define OFFSET_TERMINALSETTINGS OFFSET_DIALDIRSETTINGS+sizeof(_dialdir_settings)
typedef struct
{
    _line_settings line_settings;
    _modem_settings modem_settings;
    _dialdir_settings dialdir_settings;
    _terminal_settings terminal_settings;} _setup_data;

/*.....GLOBAL DATA.....*/
char cmdbuff[120];
char resbuff[120];

WINDOW*help_win;

```

```

int HSHAKEHOLD=0;

void interrupt (*comlorigvec)(), interrupt (*com2origvec)();

/*****
 *.....TERMINAL DATA.....*
 *****/

int curx=0,cury=0,curattr=TEXTATTR;

/*****
 *.....LINE SETUP INPUT INFO.....*
 *****/

int cominsrc=0,comindest=0;
char cominbuff[COMM_BUFF_LEN];

/*****
 *.....SETUP DATA.....*
 *****/

_setup_data setup_data={
    {0,0,0,0,0},
    {
        "
        "
        "
        "
        "
    },
    {0,0,0,0,0,0,0,0}
};

/*****
 *.....ERROR BOX COLOR INFO.....*
 *****/

COLOR_INFO error_cinfo={A_FGRED|A_BGWHITE,
    A_FGRED|A_BGWHITE,
    A_FGRED|A_BGWHITE};

/*****
 *.....NORMAL DIALOG BOX COLOR INFO.....*
 *****/

COLOR_INFO normal_dialog_cinfo={A_FGBLUE|A_BGWHITE,
    A_FGMAGENTA|A_BGWHITE,
    A_FGBLUE|A_BGWHITE};

```



```

FIELD_INP_INFO  *com_fid_list[3] = {&com_data,
                                     &stbit,
                                     &stport};

char    cstpbitlist [2] = {COM_1STOP, COM_2STOP};
char    *sstpbitlist[2] = {"1 Bit", "2 Bits"};
FIELD_INP_INFO stpbit  = {DAT_ALPHA,
                           {5,16},
                           {5,8,4},
                           SEL_ONLY,
                           SIMPLE_SEL,
                           {2,sstpbitlist},
                           NULL_PTR,
                           DO_AND_CALL,
                           "  ",
                           &(setup_data.line_settings.sbits));

```

```

char    cdatbitlist [2] = {COM_7BITS, COM_8BITS};
char    *sdatbitlist[2] = {"7 Bits", "8 Bits"};
FIELD_INP_INFO datbit  = {DAT_ALPHA,
                           {4,16},
                           {5,8,4},
                           SEL_ONLY,
                           SIMPLE_SEL,
                           {2,sdatbitlist},
                           NULL_PTR,
                           DO_AND_CALL,
                           "  ",
                           &(setup_data.line_settings.dbits));

```

```

char    comirqlist[2]   = {4,3};
int     comadrlist[2]   = {0x3F8, 0x2F8};
char    comveclist[2]   = {0x0C, 0x0B};
char    cportlist [2]   = {COM1, COM2};
char    *sportlist[2]   = {"COM1", "COM2"};
FIELD_INP_INFO  port    = {DAT_ALPHA,
                           {7,16},
                           {5,8,4},
                           SEL_ONLY,
                           SIMPLE_SEL,
                           {2,sportlist},
                           NULL_PTR,
                           DO_AND_CALL,
                           "  ",
                           &(setup_data.line_settings.port));

```



```

FIELD_INP_INFO  *com_fld_list[5]={&baudrate,
                                &datbit,
                                &stopbit,
                                &parity,
                                &port};

SCR_INP_INFO     line_scr={5,com_fld_list,0};

char  "shshakelist[4] ={"HSH","SHR","SHS/SHF","SHW"};
FIELD_INP_INFO shshake ={"SHS/SHF",
                          {3,27},
                          {5,8,4},
                          SEL_ONLY,
                          SIMPLE_SEL,
                          {3,stermemlist}},

/*****
*.....TERMINAL SETUP INPUT INFO.....*
*****/

char  "shshakelist[4] ={"HSH","SHR","SHS/SHF","SHW"};

extern  vt52_keyboard_handler(),
        vt52_display_handler(),
        vt100_keyboard_handler(),
        vt100_display_handler();
void    (*display_handler_procs[3])()={vt52_display_handler,
                                       vt100_display_handler,
                                       vt52_display_handler};
void    (*keyboard_handler_procs[3])()={vt52_keyboard_handler,
                                       vt100_keyboard_handler,
                                       vt52_keyboard_handler};

char    *stermemlist[3]={"VT-52","VT-100","ADDSVP"};
FIELD_INP_INFO term_emulation={
    DAT_ALPHA,
    {3,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {3,stermemlist},
    NULL_PTR,
    DO_AND_CALL,
    " ",
    &(setup_data.terminal_settings.term_emulation));

char    dupmodlist [2]  ={"COM_FDX","COM_HDX"};
char    *sdupmodlist[2] ={"FDX","HDX"};
FIELD_INP_INFO  dupmode  ={
    DAT_ALPHA,
    {4,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,sdupmodlist},
    NULL_PTR,

```

```

char    *shshakelist[4] ={"NONE","DTR","XON/XOFF","BOTH"};
FIELD_INP_INFO hshake = {DAT_ALPHA,
    {5,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {4,shshakelist},
    NULL_PTR,
    DO_AND_CALL,
    " ",
    &(setup_data.terminal_settings.hshake));

char    *sicrxlationlist[2] ={"CR","CR/LF"};
FIELD_INP_INFO in_cr_xlation={
    DAT_ALPHA,
    {6,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,sicrxlationlist},
    NULL_PTR,
    DO_AND_CALL,
    " ",
    &(setup_data.terminal_settings.in_cr_xlation));

char    *socrxlationlist[2] ={"CR","CR/LF"};
FIELD_INP_INFO out_cr_xlation={
    DAT_ALPHA,
    {7,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,socrxlationlist},
    NULL_PTR,
    DO_AND_CALL,
    " ",
    &(setup_data.terminal_settings.out_cr_xlation));

```



```

char    *sbsxlationlist[2]={"NONDESTR","DESTR"};
FIELD_INP_INFO bs_xlation={
    DAT_ALPHA,
    {8,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,sbsxlationlist},
    NULL_PTR,
    DO_AND_CALL,
    "          ",
    &(setup_data.terminal_settings.bs_xlation));

```

```

char    *clineslist[2] = {24,25};
char    *slineslist[2] = {"24","25"};
FIELD_INP_INFO lines    = {
    DAT_ALPHA,
    {9,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,slineslist},
    NULL_PTR,
    DO_AND_CALL,
    "          ",
    &(setup_data.terminal_settings.lines));

```

```

char    *slinewraplist[2]={"OFF","ON"};
FIELD_INP_INFO line_wrap={
    DAT_ALPHA,
    {10,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,slinewraplist},
    NULL_PTR,
    DO_AND_CALL,
    "          ",
    &(setup_data.terminal_settings.line_wrap));

```

```

char *sscrolllist[2]={"OFF","ON"};
FIELD_INP_INFO scroll = {
    DAT_ALPHA,
    {11,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,sscrolllist},
    NULL_PTR,
    DO_AND_CALL,
    " ",
    &(setup_data.terminal_settings.scroll));

char *sbelsoundlist[2]={"OFF","ON"};
FIELD_INP_INFO bell_sound = {
    DAT_ALPHA,
    {12,27},
    {5,8,4},
    SEL_ONLY,
    SIMPLE_SEL,
    {2,sbelsoundlist},
    NULL_PTR,
    DO_AND_CALL,
    " ",
    &(setup_data.terminal_settings.bell_sound));

FIELD_INP_INFO *term_fld_list[10]={&term_emulation,
    &dupmode,
    &hshake,
    &in_cr_xlation,
    &out_cr_xlation,
    &bs_xlation,
    &lines,
    &line_wrap,
    &scroll,
    &bell_sound};

SCR_INP_INFO term_scr={10,term_fld_list,0};

```



```

/*****
*.....MODEM SETUP INPUT INFO.....*
*****/

FIELD_INP_INFO init_str={DAT_ALPHA,
    {3,18},
    {0,61,0},
    ENTR_ONLY,
    BOXED_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings.init),
    NULL_PTR};

FIELD_INP_INFO dial_str={DAT_ALPHA,
    {4,18},
    {0,20,0},
    ENTR_ONLY,
    BOXED_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings.dial),
    NULL_PTR};

FIELD_INP_INFO nocar_str={DAT_ALPHA,
    {5,18},
    {0,20,0},
    ENTR_ONLY,
    BOXED_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings.nocar),
    NULL_PTR};

FIELD_INP_INFO busy_str={DAT_ALPHA,
    {6,18},
    {0,20,0},
    ENTR_ONLY,
    BOXED_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings.busy),
    NULL_PTR};

```

```

*****
***** DIALING DIRECTORY INPUT INFO *****
FIELD_INP_INFO conn1_str={DAT_ALPHA,
    {7,18},
    {0,20,0},
    ENTR_ONLY,
    BOXED_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings.conn1),
    NULL_PTR};

*****
FIELD_INP_INFO conn2_str={DAT_ALPHA,
    {8,18},
    {0,20,0},
    ENTR_ONLY,
    BOXED_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings.conn2),
    NULL_PTR};

*****
FIELD_INP_INFO conn3_str={DAT_ALPHA,
    {9,18},
    {0,20,0},
    ENTR_ONLY,
    BOXED_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings.conn3),
    NULL_PTR};

*****
FIELD_INP_INFO *modem_fld_list[7]={&init_str,
    &dial_str,
    &nocar_str,
    &busy_str,
    &conn1_str,
    &conn2_str,
    &conn3_str};

*****
SCR_INP_INFO modem_scr={7,modem_fld_list,0};
    DATA_NULL,
    DATA_SEL,
    {0,NULL_PTR},
    NULL_PTR,
    DO_AND_CALL,
    &(setup_data.modem_settings),
    NULL_PTR};

```



```

/*****
*.....DIALING DIRECTORY INPUT INFO.....*
*****/

FIELD_INP_INFO  dialdir_fld_arr  {MAX_DIALDIR_ENTRIES*2};
FIELD_INP_INFO  *dialdir_fld_list{MAX_DIALDIR_ENTRIES*2};
SCR_INP_INFO    dialdir_scr      =(MAX_DIALDIR_ENTRIES*2,dialdir_fld_list,0);

/*****
*.....XMODEM UPLOAD INPUT INFO.....*
*****/

char XMsendfilename[60] = "NONAME.ULD";

FIELD_INP_INFO XMupldfilename={DAT_ALPHA,
                                {3,12},
                                {0,40,0},
                                ENTR_ONLY,
                                BOXED_SEL,
                                {0,NULL_PTR},
                                NULL_PTR,
                                DO_AND_CALL,
                                XMsendfilename,
                                NULL_PTR};

FIELD_INP_INFO *XMupldfldlist[1]={&XMupldfilename};
SCR_INP_INFO    XMupld_scr      ={1,XMupldfldlist,0};

/*****
*.....XMODEM DOWNLOAD INPUT INFO.....*
*****/

char XMreceivefilename[60] = "NONAME.DLD";

FIELD_INP_INFO XMdownldfilename={DAT_ALPHA,
                                   {3,12},
                                   {0,40,0},
                                   ENTR_ONLY,
                                   BOXED_SEL,
                                   {0,NULL_PTR},
                                   NULL_PTR,
                                   DO_AND_CALL,
                                   XMreceivefilename,
                                   NULL_PTR};

```

```

.....
FIELD_INP_INFO *XMdownldfldlist[1]={&XMdownldfilename};
SCR_INP_INFO    XMdownld_scr    ={1,XMdownldfldlist,0};

int             lsetup_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW*win;
/*****
*.....LINE SETUP CALLBACK PROCEDURE.....*
*****/
int             key,flag;
{
    switch (key)
int             lsetup_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW*win;
SCR_INP_INFO    *sinp_info;
KEY_INFO        *key_info;
COLOR_INFO      *color_info;
int             key,flag;
{
    switch (key)
    {
        case KEY_ESC: /* User typed ESC */
            ask_and_save_data_part(&(setup_data.line_settings),
                                sizeof(_line_settings),
                                OFFSET_LINESSETTINGS );
            return (2); /* Exit */

        default:
            if (umicomm_key(key))
                return(key); /* Exit - Key must be handled by other procedures */
            return (CONTINUE);
    }
}

int             lsetup_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW*win;
SCR_INP_INFO    *sinp_info;
KEY_INFO        *key_info;
COLOR_INFO      *color_info;
int             key,flag;
{
    switch (key)
    {
        case KEY_ESC: /* User typed ESC */
            ask_and_save_data_part(&(setup_data.line_settings),
                                sizeof(_line_settings),
                                OFFSET_LINESSETTINGS );
            return (2); /* Exit */

        default:
            if (umicomm_key(key))

```



```

return(2); /* Exit - key must be handled by other procedures */

/*****
 *.....MODEM SETUP CALLBACK PROCEDURE.....*
 *****/

int      msetup_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW *win;
SCR_INP_INFO  *sinp_info;
KEY_INFO      *key_info;
COLOR_INFO    *color_info;
int           key,flag;
{
    switch (key)
    {
        case KEY_ESC: /* User typed ESC */
            ask_and_save_data_part(&(setup_data.modem_settings),
                                   sizeof(_modem_settings),
                                   OFFSET_MODEMSETTINGS );
            return (2); /* Exit */

        default:
            if (unicomm_key(key))
                return(key); /* Exit - Key must be handled by other procedures */
    }
    return (CONTINUE);
}

/*****
 *.....TERMINAL SETUP CALLBACK PROCEDURE.....*
 *****/

int      tsetup_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW *win;
SCR_INP_INFO  *sinp_info;
KEY_INFO      *key_info;
COLOR_INFO    *color_info;
int           key,flag;
{
    switch (key)
    {
        case KEY_ESC: /* User typed ESC */
            ask_and_save_data_part(&(setup_data.terminal_settings),
                                   sizeof(_terminal_settings),
                                   OFFSET_TERMINALSETTINGS );
            return (2); /* Exit */

        default:
            if (unicomm_key(key))

```

```

        return(key); /* Exit - Key must be handled by other procedures */
    }
    return (CONTINUE);
}

/*-----DIALING DIRECTORY CALLBACK PROCEDURE-----*/
int dialdir_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW*win;
SCR_INP_INFO *sinp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
int key,flag;
{
    switch (key)
    {
        case KEY_ESC: /* User typed ESC */
            ask_and_save_data_part(&(setup_data.dialdir_settings),
                                sizeof(_dialdir_settings),
                                OFFSET_DIALDIRSETTINGS );
            return (2); /* Exit */

        case ALT_D: /* User wants to dial */
            return (3); /* Dial Out */

        default:
            if (umicomm_key(key))
                return(key); /* Exit - Key must be handled by other procedures */
    }
    return (CONTINUE);
}
WINDOW*win;
SCR_INP_INFO *sinp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
int key,flag;
{
    switch (key)
    {
        case KEY_ESC: /* User typed ESC */
            return (2); /* Exit */

        case ALT_D: /* User typed Return */
            return (3); /* Dial Out */

        default:
            if (umicomm_key(key))
                return(key); /* Exit - Key must be handled by other procedures */
    }
    return (CONTINUE);
}
WINDOW*win;
SCR_INP_INFO *sinp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
int key,flag;
{
    switch (key)
    {
        case KEY_ESC: /* User typed ESC */
            return (2); /* Exit */

        case ALT_D: /* User typed Return */
            return (3); /* Dial Out */

        default:
            if (umicomm_key(key))
                return(key); /* Exit - Key must be handled by other procedures */
    }
    return (CONTINUE);
}

```



```

/*****
*.....XMODEM UPLOAD CALLBACK PROCEDURE.....*
*****/

```

```

return (CONTINUE);

int XMupld_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW *win;
SCR_INP_INFO *sinp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
int key,flag;
{
switch (key)
{
case KEY_ESC: /* User typed ESC */
return (2); /* Exit */

case 0: /* User Typed Return */
XMxmitfile(XMsendfilename);
return(2);

default:
if (umicomm_key(key))
return(key); /* Exit - Key must be handeld by other procedures */
}
return (CONTINUE);
}

```

```

/*****
*.....XMODEM DOWNLOAD CALLBACK PROCEDURE.....*
*****/

```

```

int XMdownld_proc(win,sinp_info,key_info,color_info,key,flag)
WINDOW *win;
SCR_INP_INFO *sinp_info;
KEY_INFO *key_info;
COLOR_INFO *color_info;
int key,flag;
{
switch (key)
{
case KEY_ESC: /* User typed ESC */
return (2); /* Exit */

case 0: /* User Typed Return */
XMreceivefile(XMreceivefilename);
return(2);
}
}

```

```

while ((chr==0) || (chr==0x00)) _comsendchr(port,chr);
}
default:
    if (unicomm_key(key))
        return(key); /* Exit - Key must be handled by other procedures */
    }
return (CONTINUE);
}
}

/* comadrlist[port]+3 is 0x00, return 0; */
return (0);
}

_comsendbreak(port)
char port;
{
    char chr;
    unsigned int cnt;
    chr = inportb(comadrlist[port]+3);
    chr |= 0x40;
    outportb(comadrlist[port]+3,chr);
    for (cnt=0;cnt<65000;cnt++) ;
    chr &= 0xBF;
    outportb(comadrlist[port]+3,chr);
}

char _comreceivechr()
{
    return(inportb(comadrlist[PORT]));
}

/* comadrlist[PORT]+1 is 0x00, return 0; */
/* comadrlist[PORT]+4 is 0x00, return 0; */
/* comadrlist[PORT]+5 is 0x00, return 0; */

_comsendchr(port,chr)
char chr,port;
{
    /* while (!comsendok(port)) {;} */
    if (HSBOTH || HSONXOFF) while(HSHAKEHOLD) {;}
    if (HSBOTH || HSDTR ) while((inportb(comadrlist[PORT]+6)&0x20)==0) {;}
    while ( ((inportb(comadrlist[port]+5)&0x020)==0 ) {;}
    outportb(comadrlist[port],chr);
}

_comsendstr(port,buff)
char port,*buff;
{
    char chr,*ptr=buff;

```



```
while ((chr=*ptr++)!=NULL) _comsendchr(port,chr);
}
```

```
char COME_comhit(port)
int port;
{
    if ( (inportb(comadrlist[port]+5)&0x01)!=0 ) return (1);
    return (0);
}
```

```
int FLASHCOMESPP;
```

```
void interrupt COMIRQHANDLER()
{
    char chr;
    outportb(comadrlist[PORT]+1,0x00);
    while (_comhit(PORT))
    {
        chr=_comreceivechr(PORT);
        switch (chr)
        {
            case XON:
                if (HSXONXOFF) {HSHAKEHOLD=0;break;}
            case XOFF:
                if (HSXONXOFF) {HSHAKEHOLD=1;break;}
            default:
                cominbuff[comindest]=chr;
                comindest= rotate(comindest);
        }
    }
    outportb(comadrlist[PORT]+1,0x01);
    outportb(comadrlist[PORT]+4,0x0f);
    outportb(0x020,0x020);
}
```

```
return (dest);
```

```
/* Connection */
```

```
/* COM */
```

```
int COM_OR_KB_HIT()
{
    if (COMHIT()) return(1);
    if (_kbhit()) return(2);
    return(0);
}
```

```

dtron();
{
    char value=inporth(comadrlist[PORT]+4,0x01);
    outporth(comadrlist[PORT]+4,value);
}

int COMHIT()
{
    if (cominsrc!=comindest) return (1); else return (0);
}

dtroff();
{
    char value=inporth(comadrlist[PORT]+4,0x02);
    outporth(comadrlist[PORT]+4,value);
}

int FLUSHCOMBUFF()
{
    disable();
    cominsrc=comindest=0;
    enable();
}

int cnts;
if (key!=0x00_000)
{
    for (cnts=0;cnts<com_kinfo.no_of_buys;cnts++)
        if (com_kinfo.key_list[cnts]==key) return(1);
}

char COMRECEIVECHR()
{
    return (0);
    char chr;
    while (comindest==cominsrc) {handle_kbd();};
    chr = cominbuff[cominsrc];
    cominsrc= rotate(cominsrc);
    return (chr);
}

switch (chr)
{
}

int rotate(dest)
{
    int dest;
    {
        dest= ++dest<COMM_BUFF_LEN?dest:0;
        return (dest);
    }
}

drop_connection()
{
    dtroff();
    sleep(2);
    dtron();
}

```



```

dtron()
{
    char value=inportb(comadrlist[PORT]+4)|0x01;
    outportb(comadrlist[PORT]+4,value);
}

dtroff()
{
    char value=inportb(comadrlist[PORT]+4)&0xFE;
    outportb(comadrlist[PORT]+4,value);
}

int umicomm_key(key)
int key;
{
    int cntr;
    if (key!=KEY_ESC)
    {
        for (cntr=0;cntr<umicomm_kinfo.no_of_keys;cntr++)
            if (umicomm_kinfo.key_list[cntr]==key) return(1);
    }
    return (0);
}

displaychar(chr)
char chr;
{
    switch (chr)
    {
        case CR:
            curx=0;
            gotoyx(cury,curx);
            if (ICRXLATION) increase_line();
            break;

        case LF:
        case FF:
        case VT:
            increase_line();
            break;

        case HT:
            movetotabforward();
            break;

        case BS:
            decrease_col();
            if (BSXLATION) REAL_SCREEN_BASE[cury*COLS+curx]=' '|curattr;
    }
}

```

```

break;

case BEL:
    if (BELLSOUND) beep();
    break;

default:
    if (chr>=' '&&chr<127)
    {
        REAL_SCREEN_BASE[cury*COLS+curx]=chr|curattr;
        increase_col();
    }
}
}

```

```

movetotabforward()
{
    curx=((curx+8)/8)*8;
    if (curx>=COLS) {curx-=COLS;increase_line();}
    gotoyx(cury,curx);
}

```

```

movetotabbackward()
{
    if (curx%8)
    {
        curx-=curx%8;
    }
    else
    {
        curx-=8;
    }
    if (curx<0)
    {
        curx=COLS-8;
        decrease_line();
    }
}

```

```

scroll_screen()
{
    int i;
    for (i=0;i<LINES-1;i++)
        REAL_SCREEN_BASE[i+1]=REAL_SCREEN_BASE[i];
    for (i=LINES-1;i<LINES-1+COLS;i++)
        REAL_SCREEN_BASE[i]=0;
}

```



```

increase_col()
{
    if (++curx>=COLS)
    {
        if (LINEWRAP) {curx=0;increase_line();} else curx=COLS-1;
    }
    gotoyx(cury,curx);
}

int cntx;
for (cntx=TERMLINES*COLS-1;cntx>=0;cntx-=COLS;cntx--)
    REAL_SCREEN_BASE[cntx]=REAL_SCREEN_BASE[cntx-COLS];
for (i=0;i<COLS-1;i++) REAL_SCREEN_BASE[i]=TERMINATOR;

decrease_col()
{
    if (--curx<0)
    {
        if (LINEWRAP) {curx=COLS-1;decrease_line();} else curx=0;
    }
    gotoyx(cury,curx);
}

int cntx;
for (cntx=TERMLINES*COLS-1;cntx>=(cury*COLS)*COLS;cntx--)

increase_line()
{
    if (cury*COLS+curx+1>TERMLINES*COLS)
    {
        if (cury*COLS+curx+1>TERMLINES*COLS) REAL_SCREEN_BASE[cntx]=TERMINATOR;
        if (++cury>=TERMLINES)
        {
            if (SCROLL) {cury=TERMLINES-1;scroll_scrn();} else cury=0;
        }
        gotoyx(cury,curx);
    }
}

delchr(count);
int count;

int cntx;

decrease_line()
{
    if (cury*COLS+curx+1>TERMLINES*COLS)
    {
        REAL_SCREEN_BASE[cntx]=REAL_SCREEN_BASE[cntx-count];
        if (--cury<0) REAL_SCREEN_BASE[cntx]=TERMINATOR;
        {
            gotoyx(cury,curx);
            cury=TERMLINES-1;
        }
        gotoyx(cury,curx);
    }
}

scroll_scrn()
{
    int cntx;
    for (cntx=0;cntx<(TERMLINES-1)*COLS;cntx++)
        REAL_SCREEN_BASE[cntx]=REAL_SCREEN_BASE[cntx+COLS];
    for (cntx=(TERMLINES-1)*COLS;cntx<TERMLINES*COLS;cntx++)

```

```

    REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

instchr(count)
{
    int cntr;
    for (cntr=cury*TERMLINES+curx;cntr<(TERMLINES-count)*TERMLINES+curx+count;cntr++)
        REAL_SCREEN_BASE[cntr]=REAL_SCREEN_BASE[cntr-count];
    instchr(count);
    for (;cntr>=cury*TERMLINES+curx;cntr--) REAL_SCREEN_BASE[cntr]=' '|TEXTATTR;
    gotoyx(cury,curx);
}

instline(count)
{
    int count;
    {
        int cntr;
        for (cntr=TERMLINES*COLS-1;cntr>=(cury+count)*COLS;cntr--)
            REAL_SCREEN_BASE[cntr]=REAL_SCREEN_BASE[cntr-(count*COLS)];
        for (;cntr>=cury*COLS;cntr--) REAL_SCREEN_BASE[cntr]=' '|TEXTATTR;
        gotoyx(cury,curx);
    }
}

delchr(count)
{
    int count;
    {
        int cntr;
        for (cntr=cury*COLS+curx;cntr<TERMLINES*COLS-count;cntr++)
            REAL_SCREEN_BASE[cntr]=REAL_SCREEN_BASE[cntr+count];
        for (;cntr<TERMLINES*COLS;cntr++) REAL_SCREEN_BASE[cntr]=' '|TEXTATTR;
        gotoyx(cury,curx);
    }
}

```



```

Umdelline(count)
int count;
{
    int cntr;
    int cntr;
    for (cntr=cury*COLS;cntr<(TERMLINES-count)*COLS;cntr++)
        REAL_SCREEN_BASE[cntr]=REAL_SCREEN_BASE[cntr+count*COLS];
    for (;cntr<TERMLINES*COLS;cntr++) REAL_SCREEN_BASE[cntr]=' '|TEXTATTR;
    gotoyx(cury,curx);
}

etoeod()
{
    int cntr;
    for (cntr=cury*COLS+curx;cntr<TERMLINES*COLS;cntr++)
        REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

etoeol()
{
    int cntr;
    for (cntr=cury*COLS+curx;cntr<(cury+1)*COLS;cntr++)
        REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

etobod()
{
    int cntr;
    for (cntr=cury*COLS+curx;cntr>=0;cntr--)
        REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

```

```

etobol()
{
    int cntr;
    for (cntr=cury*COLS+curx;cntr>=cury*COLS;cntr--)
        REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

```

```

escl()
{
    int cntr;
    for (cntr=0;cntr<TERMLINES*COLS;cntr++)
        REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

```

```

eline()
{
    int cntr;
    for (cntr=cury*COLS;cntr<(cury+1)*COLS;cntr++)
        REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

```

```

echars(count)
    int count;
{
    int cntr;
    for (cntr=cury*COLS;cntr<cury*COLS+count&&cntr<TERMLINES*COLS;cntr++)
        REAL_SCREEN_BASE[cntr]=' '|curattr;
    gotoyx(cury,curx);
}

```

```

do
{
    FLAGCONBUFF();
    _cursesendwin(PAGE,curline);
    _cursesendwin(PAGE,CR);
    if (!CONSOLE())
    {
        _cursesendwin(PAGE,curline);
        _cursesendwin(PAGE,CR);
    }
}
while (1)

```



```

/*****
*.....WAIT FOR EVENT.....*
*****/

```

```

int waitevent(duration,proc)
int    duration;
int    (*proc)();
{
    long  ctime,etime;
    int    rc;
    etime=time(&ctime)+duration;
    while (time(&ctime)<etime)
    {
        if ((rc=proc())!=0) return(rc);
    }
    return (0);
}

```

```

/*****
*.....DIAL OUT.....*
*****/

```

```

dialout(num)
char    *num;
{
    int    rc;
    char    *errmsg;
    sprintf(cmdbuff,"%s",DIALCMD);
    strcat (cmdbuff,num);

    while (1)
    {
        do
        {
            FLUSHCOMBUFF();
            _comsendstr(PORT,cmdbuff);
            _comsendchr(PORT,CR);
            if ( !(COMHIT()) )
            {
                errmsg=" Modem does not respond! Retry? (Y/N) ";
                break;
            }
        }
        else

```

```

{
    getresultstr(resbuff);
}

if (strcmp(resbuff,cmdbuff)!=0)
{
    errmsg=" Bad echo from Modem! Retry? (Y/N) ";
    break;
}

rc=waitevent(120,COM_OR_KB_HIT);

if (rc==1) /* COM Port Hit */
{
    getresultstr(resbuff);
    if (strcmp(resbuff,NO_CARRIER)==0)
    {
        errmsg=" No Carrirer! Retry? (Y/N) ";
        break;
    }
    if (strcmp(resbuff,BUSY )==0)
    {
        errmsg=" Busy! Retry? (Y/N) ";
        break;
    }
    if (strcmp(resbuff,CONNECT1)==0 ||
        strcmp(resbuff,CONNECT2)==0 ||
        strcmp(resbuff,CONNECT3)==0 ) return(0);
}

if (rc==2) /* KEYBOARD Hit */
{
    errmsg=" Connection Interrupted by User! Retry? (Y/N) ";
    _comsendchr(PORT,0);
    break;
}

errmsg=" Connection timed out! Retry? (Y/N) ";
} while(0);
if (promptforyesno(errmsg,"Yy","Nn",&error_cinfo))
{
    FLUSHCOMBUFF();
    return;
}
}
}

```



```

/*****
*.....GET RESULT STRING.....*
*****/

wattrset  (setup_win,A_BOLD);
waddch    (setup_win,'|');
waddch    (setup_win,' ');

getresultstr(buff)
char      *buff;
{
    char    chr;
    while ( (chr=COMRECEIVECHR())==LF || chr==CR ) {;;}
    *buff++ = chr;
    while ( (chr=COMRECEIVECHR())!=LF && chr!=CR ) *buff++ = chr;
    *buff++ = (char) NULL;
}

/*****
*.....LINE SETUP.....*
*****/

#define      SETUP_SCR_SIZE_Y 9
#define      SETUP_SCR_SIZE_X 40
#define      SETUP_SCR_POS_Y   8
#define      SETUP_SCR_POS_X   20

line_setup()
{
    WINDOW*setup_win;
    int      cpos_x, cpos_y, csize_b, csize_e, cntr, retcode,
            scr_buff[SETUP_SCR_SIZE_Y*SETUP_SCR_SIZE_X];

    savescrn  (SETUP_SCR_SIZE_Y,SETUP_SCR_SIZE_X,
              SETUP_SCR_POS_Y ,SETUP_SCR_POS_X ,scr_buff);
    getcurpos  (&cpos_y,&cpos_x);
    get_cursor_size (&csize_b,&csize_e);
    set_cursor_size (-1,-1);

    setup_win = newwin(SETUP_SCR_SIZE_Y,SETUP_SCR_SIZE_X,
                      SETUP_SCR_POS_Y ,SETUP_SCR_POS_X );
    fillwin    (setup_win, ' '|A_BGCYAN);
    wattrset   (setup_win,A_BGCYAN|A_FGBLUE);
    box        (setup_win,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);
    mvwaddch   (setup_win,2,0,SLE_TEE);
    mvwaddch   (setup_win,2,SETUP_SCR_SIZE_X-1,SRI_TEE);

```

```

for (cntr=1;cntr<SETUP_SCR_SIZE_X-1;cntr++)
{
    mvwaddch (setup_win,2,cntr,SHO_BAR);
}

wattrset (setup_win,A_BGBLUE|A_FGYELLOW);
mvwaddstr (setup_win,1,12," Line Setup ");
wattrset (setup_win,A_BGCYAN|A_FGMAGENTA);
mvwaddstr (setup_win,3,4,"Baudrate  :");
mvwaddstr (setup_win,4,4,"Data Bits  :");
mvwaddstr (setup_win,5,4,"Stop Bits  :");
mvwaddstr (setup_win,6,4,"Parity    :");
mvwaddstr (setup_win,7,4,"Com Port  :");
wrefresh (setup_win);

retcode=screen_input (setup_win,
    &line_scr,
    &umicomm_kinfo,
    &setup_cinfo,
    &setup_proc);

if (retcode==2) prog_reset();

restorescrn (SETUP_SCR_SIZE_Y,SETUP_SCR_SIZE_X,
    SETUP_SCR_POS_Y ,SETUP_SCR_POS_X ,scr_buff);
gotoyx (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
delwin (setup_win);
return (retcode);
}

/*****
*.....MODEM SETUP.....*
*****/

#define M_SETUP_SCR_SIZE_Y 11
#define M_SETUP_SCR_SIZE_X 80
#define M_SETUP_SCR_POS_Y 8
#define M_SETUP_SCR_POS_X 0

modem_setup()
{
    WINDOW *setup_win;
    int cpos_x,cpos_y,csize_b,csize_e,cntr,retcode,
        scr_buff[M_SETUP_SCR_SIZE_Y*M_SETUP_SCR_SIZE_X];

    savescrn (M_SETUP_SCR_SIZE_Y,M_SETUP_SCR_SIZE_X,
        M_SETUP_SCR_POS_Y ,M_SETUP_SCR_POS_X ,scr_buff);
    getcurpos (&cpos_y,&cpos_x);
    get_cursor_size (&csize_b,&csize_e);

```



```

set_cursor_size (-1,-1);

setup_win = newwin(M_SETUP_SCR_SIZE_Y,M_SETUP_SCR_SIZE_X,
                  M_SETUP_SCR_POS_Y ,M_SETUP_SCR_POS_X );
fillwin      (setup_win,' '|A_BGCYAN);
wattrset     (setup_win,A_BGCYAN|A_FGBLUE);
box          (setup_win,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);
mvwaddch     (setup_win,2,0,SLE_TEE);
mvwaddch     (setup_win,2,M_SETUP_SCR_SIZE_X-1,SRI_TEE);
for (cntr=1;cntr<M_SETUP_SCR_SIZE_X-1;cntr++)
{
    mvwaddch (setup_win,2,cntr,SHO_BAR);
}
wattrset     (setup_win,A_BGBLUE|A_FGYELLOW);
mvwaddstr    (setup_win,1,37," Modem Setup ");
wattrset     (setup_win,A_BGCYAN|A_FGMAGENTA);
mvwaddstr    (setup_win,3,4,"Init String :");
mvwaddstr    (setup_win,4,4,"Dial Command:");
mvwaddstr    (setup_win,5,4,"No Carrier  :");
mvwaddstr    (setup_win,6,4,"Busy      :");
mvwaddstr    (setup_win,7,4,"Connected1 :");
mvwaddstr    (setup_win,8,4,"Connected2 :");
mvwaddstr    (setup_win,9,4,"Connected3 :");
wrefresh     (setup_win);

retcode=screen_input(setup_win,
                     &modem_scr,
                     &umicomm_kinfo,
                     &setup_cinfo,
                     &msetup_proc);

if (retcode==2) prog_reset();

restorescrn (M_SETUP_SCR_SIZE_Y,M_SETUP_SCR_SIZE_X,
            M_SETUP_SCR_POS_Y ,M_SETUP_SCR_POS_X ,scr_buff);
gotoyx      (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
delwin      (setup_win);
return      (retcode);
}

```

```

/*****
*.....TERMINAL SETUP.....*
*****/

getyx      (cpos_y, cpos_x);

#define     T_SETUP_SCR_SIZE_Y      14
#define     T_SETUP_SCR_SIZE_X      40
#define     T_SETUP_SCR_POS_Y       8
#define     T_SETUP_SCR_POS_X       20

terminal_setup()
{
    WINDOW *setup_win;

    int      cpos_x, cpos_y, csize_b, csize_e, cntr, retcode,
             scr_buff[T_SETUP_SCR_SIZE_Y*T_SETUP_SCR_SIZE_X];

    savescrn (T_SETUP_SCR_SIZE_Y, T_SETUP_SCR_SIZE_X,
              T_SETUP_SCR_POS_Y, T_SETUP_SCR_POS_X, scr_buff);

    getcurpos (&cpos_y, &cpos_x);
    get_cursor_size (&csize_b, &csize_e);
    set_cursor_size (-1, -1);

    setup_win = newwin(T_SETUP_SCR_SIZE_Y, T_SETUP_SCR_SIZE_X,
                      T_SETUP_SCR_POS_Y, T_SETUP_SCR_POS_X);

    fillwin   (setup_win, ' ' | A_BGCYAN);
    wattrset  (setup_win, A_BGCYAN | A_FGBLUE);
    box       (setup_win, SHO_BAR, SVE_BAR, SUL_COR, SUR_COR, SLL_COR, SLR_COR);
    mvwaddch  (setup_win, 2, 0, SLE_TEE);
    mvwaddch  (setup_win, 2, T_SETUP_SCR_SIZE_X-1, SRI_TEE);
    for (cntr=1; cntr<T_SETUP_SCR_SIZE_X-1; cntr++)
    {
        mvwaddch (setup_win, 2, cntr, SHO_BAR);
    }
    wattrset  (setup_win, A_BGBLUE | A_FGYELLOW);
    mvwaddstr (setup_win, 1, 10, " Terminal Setup ");
    wattrset  (setup_win, A_BGCYAN | A_FGMAGENTA);
    mvwaddstr (setup_win, 3, 4, "Emulation      :");
    mvwaddstr (setup_win, 4, 4, "Duplex       :");
    mvwaddstr (setup_win, 5, 4, "Handshake    :");
    mvwaddstr (setup_win, 6, 4, "CR Translation (IN) :");
    mvwaddstr (setup_win, 7, 4, "CR Translation (OUT) :");
    mvwaddstr (setup_win, 8, 4, "BS Translation  :");
    mvwaddstr (setup_win, 9, 4, "Lines        :");
    mvwaddstr (setup_win, 10, 4, "Line Wrap     :");
    mvwaddstr (setup_win, 11, 4, "Scroll        :");
    mvwaddstr (setup_win, 12, 4, "Bell Sound    :");
    wrefresh  (setup_win);

    retcode=screen_input (setup_win,
                          &term_scr,
                          &umicomm_kinfo,
                          &setup_cinfo,
                          tsetup_proc);
}

```



```

restorecrn (T_SETUP_SCR_SIZE_Y,T_SETUP_SCR_SIZE_X,
            T_SETUP_SCR_POS_Y ,T_SETUP_SCR_POS_X ,scr_buff);
gotoxy      (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
delwin      (setup_win);
return      (retcode);
}

/*.....DIALING DIRECTORY.....*/
*****/

#define DIALDIR_SCR_SIZE_Y      23
#define DIALDIR_SCR_SIZE_X      80
#define DIALDIR_SCR_POS_Y      1
#define DIALDIR_SCR_POS_X      0

dialing_directory()
{
WINDOW*dialing_win;
int      cpos_x,cpos_y,csize_b,csize_e,cntr,fld,rc,
scr_buff[DIALDIR_SCR_SIZE_Y*DIALDIR_SCR_SIZE_X];

savescrn  (DIALDIR_SCR_SIZE_Y,DIALDIR_SCR_SIZE_X,
           DIALDIR_SCR_POS_Y ,DIALDIR_SCR_POS_X ,scr_buff);
getcurpos (&cpos_y,&cpos_x);
get_cursor_size (&csize_b,&csize_e);
set_cursor_size (-1,-1);

dialing_win = newwin(DIALDIR_SCR_SIZE_Y,DIALDIR_SCR_SIZE_X,
                    DIALDIR_SCR_POS_Y ,DIALDIR_SCR_POS_X );
fillwin     (dialing_win,' '|A_BGCYAN);
wattrset    (dialing_win,A_BGCYAN|A_FGBLUE);
box         (dialing_win,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);
mvwaddch    (dialing_win,2,0 ,SLE_TEE);
mvwaddch    (dialing_win,2,DIALDIR_SCR_SIZE_X-1,SRI_TEE);
for (cntr=1;cntr<DIALDIR_SCR_SIZE_X-1;cntr++)
{
mvwaddch    (dialing_win,2,cntr,SHO_BAR);
}
wattrset    (dialing_win,A_BGBLUE|A_FGYELLOW);
mvwaddstr   (dialing_win,1,30 , " Dialing Directory ");
mvwaddstr   (dialing_win,3,1 , " Name ");
mvwaddstr   (dialing_win,3,DIALDIR_NAME_LEN+2 , " Phone Number ");
mvwaddstr   (dialing_win,3,41 , " Name ");
mvwaddstr   (dialing_win,3,40+DIALDIR_NAME_LEN+2," Phone Number ");
}

```



```

rc=screen_input      (dialing_win,
for (ctrl=1;ctrl<256;ctrl++)
    &dialdir_scr,
    &umicomm_kinfo,
    &setup_cinfo,
    dialdir_proc);

wattrset (A_BOLD,A_COLOR);

restorescrn (DIALDIR_SCR_SIZE_Y,DIALDIR_SCR_SIZE_X,
             DIALDIR_SCR_POS_Y ,DIALDIR_SCR_POS_X ,scr_buff);
gotoyx      (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
delwin      (dialing_win);

if (rc==3) /* Dial Out */
{
    fld = dialdir_scr.current_field;
    if ( !(fld%2) ) fld++;
    dialout (dialdir_scr.field_infos[fld]->out_buff);
    rc=2;
}
return (rc);
}

/*****
*.....XMODEM FILE UPLOAD.....*
*****/

/*****
*.....XMODEM FILE DOWNLOAD.....*
*****/

#define XMUPLD_SCR_SIZE_Y      5
#define XMUPLD_SCR_SIZE_X      53
#define XMUPLD_SCR_POS_Y      10
#define XMUPLD_SCR_POS_X      13

int XMfileupload()
{
    WINDOW *XMwin;
    int cpos_x,cpos_y,csize_b,csize_e,cntr,fld,rc,
        scr_buff[XMUPLD_SCR_SIZE_Y*XMUPLD_SCR_SIZE_X];

    savescrn      (XMUPLD_SCR_SIZE_Y,XMUPLD_SCR_SIZE_X,
                   XMUPLD_SCR_POS_Y ,XMUPLD_SCR_POS_X ,scr_buff);
    getcurpos      (&cpos_y,&cpos_x);
    get_cursor_size (&csize_b,&csize_e);
    set_cursor_size (-1,-1);

    XMwin = newwin(XMUPLD_SCR_SIZE_Y,XMUPLD_SCR_SIZE_X,
                  XMUPLD_SCR_POS_Y ,XMUPLD_SCR_POS_X );
    fillwin        (XMwin,' '|A_BGCYAN);
    wattrset        (XMwin,A_BGCYAN|A_FGBLUE);
    box             (XMwin,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);
    mvwaddch        (XMwin,2,0 ,SLE_TEE);

```



```

mvwaddch      (XMwin,2,XMUPLD_SCR_SIZE_X-1,SRI_TEE);
for (cntr=1;cntr<XMUPLD_SCR_SIZE_X-1;cntr++)
{
    mvwaddch      (XMwin,2,cntr,SHO_BAR);
}

wattrset      (XMwin,A_BGBLUE|A_FGYELLOW);
mvwaddstr      (XMwin,1,16," XMODEM File Upload ");
wattrset      (XMwin,A_BGCYAN|A_FGBLUE);
mvwaddstr      (XMwin,3,1," File Name : ");

rc=screen_input      (XMwin,
    &XMupld_scr,
    &umicomm_kinfo,
    &setup_cinfo,
    XMupld_proc);

restorescrn      (XMUPLD_SCR_SIZE_Y,XMUPLD_SCR_SIZE_X,
    XMUPLD_SCR_POS_Y ,XMUPLD_SCR_POS_X ,scr_buff);
gotoyx      (cpos_y,cpos_x);
set_cursor_size      (csize_b,csize_e);
delwin      (XMwin);
return      (rc);
}

/*****
*.....XMODEM FILE DOWNLOAD.....*
*****/

#define      XMDLD_SCR_SIZE_Y      5
#define      XMDLD_SCR_SIZE_X      53
#define      XMDLD_SCR_POS_Y      10
#define      XMDLD_SCR_POS_X      13

int XMfiledownload()
{
    WINDOW *XMwin;
    int      cpos_x,cpos_y,csize_b,csize_e,cntr,fld,rc,
        scr_buff[XMDLD_SCR_SIZE_Y*XMDLD_SCR_SIZE_X];

    savescrn      (XMDLD_SCR_SIZE_Y,XMDLD_SCR_SIZE_X,
        XMDLD_SCR_POS_Y ,XMDLD_SCR_POS_X ,scr_buff);
    getcurpos      (&cpos_y,&cpos_x);
    get_cursor_size      (&csize_b,&csize_e);
    set_cursor_size      (-1,-1);

    XMwin = newwin(XMDLD_SCR_SIZE_Y,XMDLD_SCR_SIZE_X,
        XMDLD_SCR_POS_Y ,XMDLD_SCR_POS_X );
    fillwin      (XMwin,' '|A_BGCYAN);

```

```

wattrset      (XMwin,A_BGCYAN|A_FGBLUE);
box           (XMwin,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);
mvwaddch      (XMwin,2,0,SLE_TEE);
mvwaddch      (XMwin,2,XMDLD_SCR_SIZE_X-1,SRI_TEE);
for (cntr=1;cntr<XMDLD_SCR_SIZE_X-1;cntr++)
{
    mvwaddch (XMwin,2,cntr,SHO_BAR);
}
wattrset      (XMwin,A_BGBLUE|A_FGYELLOW);
mvwaddstr      (XMwin,1,16," XMODEM File Download ");
wattrset      (XMwin,A_BGCYAN|A_FGBLUE);
mvwaddstr      (XMwin,3,1," File Name : ");

rc=screen_input (XMwin,
                 &XMdownld_scr,
                 &umicomm_kinfo,
                 &setup_cinfo,
                 XMdownld_proc);

restorescrn    (XMDLD_SCR_SIZE_Y,XMDLD_SCR_SIZE_X,
                 XMDLD_SCR_POS_Y,XMDLD_SCR_POS_X,scr_buff);
gotoyx         (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
delwin         (XMwin);
return         (rc);
}

/*.....PROGRAM RESET.....*/

prog_reset()
{
    bioscom(COM_SETPARM,
            (cdatbitlist[setup_data.line_settings.dbits ]|
             cparitylist[setup_data.line_settings.parity]|
             cstpbitlist[setup_data.line_settings.sbits ]|
             cbaudlist [setup_data.line_settings.bps   ]),
            setup_data.line_settings.port);
    setvect   (comveclist[0],comlorigvec);
    setvect   (comveclist[1],com2origvec);
    mask_irq   (comirqlist[PORT]);
    setvect   (comveclist[PORT],&COMIRQHANDLER);
    outportb   (comadrlist[PORT]+1,0x01);
    outportb   (comadrlist[PORT]+4,0x0f);
    unmask_irq (comirqlist[PORT]);

    rmtrailblanks(&(setup_data.modem_settings.init ));
    rmtrailblanks(&(setup_data.modem_settings.dial ));

```



```

rmtrailblanks(&(amp;setup_data.modem_settings.nocar));
rmtrailblanks(&(amp;setup_data.modem_settings.busy ));
rmtrailblanks(&(amp;setup_data.modem_settings.conn1));
rmtrailblanks(&(amp;setup_data.modem_settings.conn2));
rmtrailblanks(&(amp;setup_data.modem_settings.conn3));
_comsendstr (PORT,&(setup_data.modem_settings.init));
_comsendchr (PORT,CR);
sleep (1);
FLUSHCOMBUFF ();
}

/* Display help */
int display_help()
{
    int x, y, size_x, size_y, cnt, cnt2;

    /*****
    * .....HELP SETUP.....
    *****/

    HELP_SCR_POS_Y = HELP_SCR_POS_X = scr_buff;

    getopts ("fpos: y: lpos: x:");

#define HELP_SCR_SIZE_Y 15
#define HELP_SCR_SIZE_X 40
#define HELP_SCR_POS_Y 7
#define HELP_SCR_POS_X 20

    help_setup()
    {
        int cnt;
        help_win = newwin(HELP_SCR_SIZE_Y,HELP_SCR_SIZE_X,
                        HELP_SCR_POS_Y,HELP_SCR_POS_X );

        fillwin (help_win,'|A_BGCYAN);
        wattrset (help_win,A_BGCYAN|A_FGBLUE);
        box (help_win,SHO_BAR,SVE_BAR,SUL_COR,SUR_COR,SLL_COR,SLR_COR);
        mvwaddch (help_win,2,0 ,SLE_TEE);
        mvwaddch (help_win,2,HELP_SCR_SIZE_X-1,SRI_TEE);
        for (cnt=1;cnt<HELP_SCR_SIZE_X-1;cnt++)
        {
            mvwaddch (help_win,2,cnt,SHO_BAR);
        }

        wattrset (help_win,A_BGBLUE|A_FGYELLOW);
        mvwaddstr (help_win,1,12," Online Help ");
        wattrset (help_win,A_BGCYAN|A_FGBLACK);
        mvwaddstr (help_win,3 ,4,"Help.....: ALT-H");
        mvwaddstr (help_win,4 ,4,"Line Setup.....: ALT-S");
        mvwaddstr (help_win,5 ,4,"Modem Setup.....: ALT-M");
        mvwaddstr (help_win,6 ,4,"Terminal Setup.....: ALT-T");
        mvwaddstr (help_win,7 ,4,"Dial Directory.....: ALT-D");
        mvwaddstr (help_win,8 ,4,"Dial (In Directory)....: ALT-D");
        mvwaddstr (help_win,9 ,4,"Save Setup.....: ALT-K");
        mvwaddstr (help_win,10,4,"Exit.....: ALT-X");
        mvwaddstr (help_win,11,4,"Send BREAK.....: ALT-B");
        mvwaddstr (help_win,12,4,"XMODEM Upload File..... PGUP ");
        mvwaddstr (help_win,13,4,"XMODEM DOWNLOAD File.... PGDN ");
    }
}

```

```

*****
*.....DISPLAY HELP .....*
*****

int display_help()
{
    int cpos_x, cpos_y, csize_b, csize_e, cntr, retcode,
    scr_buff[HELP_SCR_SIZE_Y*HELP_SCR_SIZE_X];

    savescrn (HELP_SCR_SIZE_Y, HELP_SCR_SIZE_X,
              HELP_SCR_POS_Y, HELP_SCR_POS_X, scr_buff);
    getcurpos (&cpos_y, &cpos_x);
    get_cursor_size (&csize_b, &csize_e);
    set_cursor_size (-1, -1);

    touchwin (help_win);
    wrefresh (help_win);

    retcode = wgetch(help_win);

    restorescrn (HELP_SCR_SIZE_Y, HELP_SCR_SIZE_X,
                 HELP_SCR_POS_Y, HELP_SCR_POS_X, scr_buff);
    gotoxy (cpos_y, cpos_x);
    set_cursor_size (csize_b, csize_e);
    return (retcode);
}

*****
*.....LOGO SCREEN.....*
*****

logo_screen()
{
    WINDOW *logo_win;
    int cpos_x, cpos_y, csize_b, csize_e;

    getcurpos (&cpos_y, &cpos_x);
    get_cursor_size (&csize_b, &csize_e);

```



```

set_cursor_size (-1,-1);

logo_win = newwin(0,0,0,0);
fillwin      (logo_win,' '|A_BGWHITE);
wattrset     (logo_win,A_BGWHITE|A_FGBLUE);
box          (logo_win,DHO_BAR,DVE_BAR,DUL_COR,DUR_COR,DLL_COR,DLR_COR);
wattrset     (logo_win,A_FGWHITE|A_BGBLUE);
mvwaddstr    (logo_win,9,24,"    U M I C O M M    ");
wattrset     (logo_win,A_BGWHITE|A_FGGREEN);
mvwaddstr    (logo_win,11,30," By Umit Kalkan ");
wattrset     (logo_win,A_BGWHITE|A_FGRED);
mvwaddstr    (logo_win,15,23," Press ALT-H When You Need Help ");
wattrset     (logo_win,A_BGWHITE|A_FGBLACK);
mvwaddstr    (logo_win,17,23," Press Any Key To Continue ");
wrefresh     (logo_win);

wgetch       (logo_win);

gotoyx       (cpos_y,cpos_x);
set_cursor_size (csize_b,csize_e);
delwin       (logo_win);
}

for (cnt=0;cnt<MAX_DIALDIR_ENTRIES;cnt++) {
    dialdir[cnt].fld_list[cnt] = (dialdir_fld_arr[cnt]);
}

/*****
*.....PROGRAM INIT.....*
*****/

prog_init()
{
    int cnt, _posx, _posy, _sizx;
    char *_buffptr;

    com1origvec = getvect(comveclist[0]);
    com2origvec = getvect(comveclist[1]);

    load_data();

    for (cnt=0;cnt<MAX_DIALDIR_ENTRIES*2;cnt++)
    {
        dialdir_fld_list[cnt]=&(dialdir_fld_arr[cnt]);

        if (cnt/2>=MAX_DIALDIR_ENTRIES/2)
        { _posy=cnt/2-MAX_DIALDIR_ENTRIES/2+5; _posx=41;}
        else
        { _posy=cnt/2+5; _posx=1;}

        if (cnt%2)
        {
            _buffptr=&(setup_data.dialdir_settings.numbers[cnt/2][0]);

```

```

    _posx+= (DIALDIR_NAME_LEN+1);
    _sizx=  DIALDIR_NUM_LEN;
}
else
{
    _buffptr=&(setup_data.dialdir_settings.names [cntr/2][0]);
    _sizx=  DIALDIR_NAME_LEN;
}

dialdir_fld_list[cntr]->data_type  = DAT_ALPHA;
dialdir_fld_list[cntr]->position.y = _posy;
dialdir_fld_list[cntr]->position.x = _posx;
dialdir_fld_list[cntr]->size.y    = 1;
dialdir_fld_list[cntr]->size.x    = _sizx;
dialdir_fld_list[cntr]->inp_style  = ENTR_ONLY;
dialdir_fld_list[cntr]->usr_info   = NULL_PTR;
dialdir_fld_list[cntr]->call_flag  = DO_AND_CALL;
dialdir_fld_list[cntr]->out_buff   = _buffptr;
dialdir_fld_list[cntr]->selection  = NULL_PTR;
}

help_setup();
logo_screen();
for (cntr=0;cntr<LINES*COLS;cntr++) *(REAL_SCREEN_BASE+cntr)=' '|TEXTATTR;
}

/*.....SAVE DATA.....*/
/*.....*/

int save_data()
{
    int handle;
    while (1)
    {
        if ((handle=open(DATAFILE,O_CREAT|O_WRONLY|O_BINARY|O_TRUNC,S_IREAD|S_IWRITE)) != -1)
        {
            write (handle,&setup_data,sizeof(_setup_data));
            close (handle);
            return(0);
        }
        if (promptforyesno(" Error while writing data! Retry? (Y/N) ",
                           "Yy","Nn",&error_cinfo)) return(-1);
    }
}

```



```

/*****
*.....ASK & SAVE DATA PART.....*
*****/

```

```

int ask_and_save_data_part(buff,count,offset)
char *buff;
int count;
int offset;
{
if (promptforyesno(" Do you want to save changes? (Y/N) ",
    "Yy","Nn",&normal_dialog_cinfo)) return;
save_data_part(buff,count,offset);
}

```

```

/*****
*.....LOAD DATA.....*
*****/

```

```

/*****
*.....SAVE DATA PART.....*
*****/

```

```

int save_data_part(buff,count,offset)
char *buff;
int count;
int offset;
{
int handle;
while (1)
{
do
{
if ((handle=open(DATAFILE,O_WRONLY|O_BINARY))!= -1)
{
if ((handle=open(DATAFILE,O_CREAT|O_WRONLY|O_BINARY|O_TRUNC,S_IRREAD|S_IWRITE)) != -1)
{
int cntr;
char null_chr=0;
for (cntr=0;cntr<sizeof(_setup_data);cntr++)
{
write(handle,&null_chr,1);
}
}
else
{
break;
}
}
}
}

```

```

}
if (lseek(handle,offset,0)!= -1)
{
    write (handle,buff,count);
    close (handle);
    return(0);
}
} while (0);
if (promptforyesno(" Error while writing data! Retry? (Y/N) ",
                    "Yy","Nn",&error_cinfo)) return(-1);
}
}

/*-----*/
/*****
 *.....LOAD DATA.....*
 *****/

load_data()
{
    int handle;
    if ((handle=open(DATAFILE,O_RDONLY|O_BINARY)) != -1)
    {
        read (handle,&setup_data,sizeof(_setup_data));
        close (handle);
        return(0);
    }
    return (-1);
}

char * getch();
if (strcmp(h,"") != 0) { printf("%s\n",h); }
if (strcmp(h,"quit"))
{
    do
    {
        switch (getch())
        {
            case '\n':
                if (promptforyesno(" Do you want to quit? (Y/N) ",
                                    "Yy","Nn",&error_cinfo))
                {
                    printf("Error while writing data! Retry? (Y/N) ",
                            "Yy","Nn",&error_cinfo);
                }
                break;
            case '\q':
                printf("Quitting...\n");
                break;
            default:
                break;
        }
    } while (1);
}

```



```

/*****
*****
*****  TERMINAL EMULATION HANDLER PROCEDURES  *****
*****
*****/

#include <termemul.c>

break;

case ALT_F:
chrterminal_setup();

break;

/*****
*****
*****  XMODEM FILE TRANSFER PROCEDURES  *****
*****
*****/

#include <xmodem.c>

case ALT_P:
chrxtilexload();

break;

case ALT_Q:
chrxtilexload();

break;

/*****
*.....HANDLE KEYBOARD.....*
*****/

while (chr!=7)

handle_kbd()
{
int chr;
if ( _kbhit() )
{
chr = wgetch();
if (chr==ALT_B) {_comsendbreak(PORT);return;}
if (umicomm_key(chr))
{
do
{
switch (chr)
{
case ALT_X:
if (promptforyesno(" Exit to DOS? (Y/N) ",
"Nn","Yy",&normal_dialog_cinfo))
{
setvect (comveclist[0],comlorigvec);
setvect (comveclist[1],com2origvec);
endwin ();
exit ();
}
chr=2;
break;

```



```

/*****
*****
**
**          COMM.H
**
**          Communication Module
**
**          Creation : 15/3/1991 by Umit KALKAN
**
*****
*****/

```

```
#ifndef __COMMINC
```

```
#define __COMMINC
```

```
#define COM1 0
```

```
#define COM2 1
```

```
#define COM_SETPARM 0
```

```
#define COM_SNDCHAR 1
```

```
#define COM_RCVCHAR 2
```

```
#define COM_GETSTAT 3
```

```
#define COM_7BITS 0x02
```

```
#define COM_8BITS 0x03
```

```
#define COM_1STOP 0x00
```

```
#define COM_2STOP 0x04
```

```
#define COM_NOPAR 0x00
```

```
#define COM_ODPAR 0x08
```

```
#define COM_EVPAR 0x18
```

```
#define COM_110 0x00
```

```
#define COM_150 0x20
```

```
#define COM_300 0x40
```

```
#define COM_600 0x60
```

```
#define COM_1200 0x80
```

```
#define COM_2400 0xA0
```

```
#define COM_4800 0xC0
```

```
#define COM_9600 0xE0
```

```
#define COM_FDX 0
```

```
#define COM_HDX 1
```

```
#define COM_TIMEOUT 0x8000
```

```
#define COM_TXSHREGETY 0x4000
```

```
#define COM_TXHOREGETY 0x2000
```

```
#define COM_BREAKDET 0x1000
```

```
#define COM_FRAMINGERR 0x0800
```

```
#define COM_PARITYERR 0x0400
```

```
#define COM_OVERRUNERR 0x0200
```

```
#define COM_DATAREADY 0x0100
```

```

#define COM_CD 0x0080
#define COM_RI 0x0040
#define COM_DSR 0x0020
#define COM_CTS 0x0010
#define COM_DCD 0x0008
#define COM_DRI 0x0004
#define COM_DDSR 0x0002
#define COM_DCTS 0x0001

```

```

#endif

```

```

/* Set the state indicator being passed to user procedures */

```

```

#define STATE_WORDS 8
#define STATE_WORDS 1

```

```

/* Set the state indicator being passed to user procedures */

```

```

#define STATE_WORDS 8
#define STATE_WORDS 1

```

```

typedef struct {
    int num_of_fields;
    int field_index;
    int current_field;
    int num_of_fields;
}

```

```

/* Set the state indicator being passed to user procedures */

```

```

/* Set the state indicator being passed to user procedures */

```

```

/* Set the state indicator being passed to user procedures */

```

```

/* Set the state indicator being passed to user procedures */

```

```

/* Set the state indicator being passed to user procedures */

```

```

/* Set the state indicator being passed to user procedures */

```

```

/* Set the state indicator being passed to user procedures */

```



```

/*****
*****
**
**          CURSES.H
**
**          CURSES Module Include Header File
**
**          Creation : 15/3/1991 by Umit KALKAN
**
*****/

#define

#define

#ifndef __CURSESINC
#define __CURSESINC
#define A_BLINK 0x8700
#define A_UNDERLINE 0x0100
#define A_NORMAL 0x0700
#define A_REVERSE 0x7000
#define A_BOLD 0x0f00

#define A_FGBLACK 0x0000
#define A_FGBLUE 0x0100
#define A_FGGREEN 0x0200
#define A_FGCYAN 0x0300
#define A_FGRED 0x0400
#define A_FGMAGENTA 0x0500
#define A_FGBROWN 0x0600
#define A_FGWHITE 0x0700
#define A_FGGRAY 0x0800
#define A_FGYELLOW 0x0E00

#define A_BGBLACK 0x0000
#define A_BGBLUE 0x1000
#define A_BGGREEN 0x2000
#define A_BGCYAN 0x3000
#define A_BGRED 0x4000
#define A_BGMAGENTA 0x5000
#define A_BGBROWN 0x6000
#define A_BGWHITE 0x7000

#define A_CBLINK 0x8000
#define A_CALITE 0x0800
#define A_FGCOLOR 0x0700
#define A_BGCOLOR 0x7000

#define A_ATTRIBUTE 0xff00
#define A_CHARTEXT 0x00ff

#define LINES 25
#define COLS 80

```



```

#define NUL 0
#define EOT 4
#define ACK 6
#define BEL 7
#define BS 8
#define HT 9
#define LF 10
#define VT 11
#define FF 12
#define CR 13
#define SO 14
#define SI 15
#define DC1 17
#define DC3 19
#define NAK 21
#define CAN 24
#define SUB 26
#define ESC 27
#define DEL 127

#define DUL_COR 201
#define DUR_COR 187
#define DLL_COR 200
#define DLR_COR 188
#define DVE_BAR 186
#define DHO_BAR 205
#define DLE_TEE 204
#define DRI_TEE 185
#define DUP_TEE 203
#define DLO_TEE 202
#define D_CROSS 206

#define SUL_COR 218
#define SUR_COR 191
#define SLL_COR 192
#define SLR_COR 217
#define SVE_BAR 179
#define SHO_BAR 196
#define SLE_TEE 195
#define SRI_TEE 180
#define SUP_TEE 194
#define SLO_TEE 193
#define S_CROSS 197

#define KEY_F1 0x013b
#define KEY_F2 0x013c
#define KEY_F3 0x013d
#define KEY_F4 0x013e
#define KEY_F5 0x013f
#define KEY_F6 0x0140
#define KEY_F7 0x0141
#define KEY_F8 0x0142
#define KEY_F9 0x0143

```

```

#define KEY_F10 0x0144

#define KEY_SF1 0x0154
#define KEY_SF2 0x0155
#define KEY_SF3 0x0156
#define KEY_SF4 0x0157
#define KEY_SF5 0x0158
#define KEY_SF6 0x0159
#define KEY_SF7 0x015a
#define KEY_SF8 0x015b
#define KEY_SF9 0x015c
#define KEY_SF10 0x015d

#define KEY_CF1 0x015e
#define KEY_CF2 0x015f
#define KEY_CF3 0x0160
#define KEY_CF4 0x0161
#define KEY_CF5 0x0162
#define KEY_CF6 0x0163
#define KEY_CF7 0x0164
#define KEY_CF8 0x0165
#define KEY_CF9 0x0166
#define KEY_CF10 0x0167

#define KEY_aF1 0x0168
#define KEY_aF2 0x0169
#define KEY_aF3 0x016a
#define KEY_aF4 0x016b
#define KEY_aF5 0x016c
#define KEY_aF6 0x016d
#define KEY_aF7 0x016e
#define KEY_aF8 0x016f
#define KEY_aF9 0x0170
#define KEY_aF10 0x0171

#define KEY_UP 0x0148
#define KEY_DOWN 0x0150
#define KEY_LEFT 0x014b
#define KEY_RIGHT 0x014d
#define KEY_cLEFT 0x0173
#define KEY_cRIGHT 0x0174

#define KEY_HOME 0x0147
#define KEY_END 0x014f
#define KEY_PGUP 0x0149
#define KEY_PGDN 0x0151
#define KEY_cHOME 0x0177
#define KEY_cEND 0x0175
#define KEY_cPGUP 0x0184
#define KEY_cPGDN 0x0176

#define KEY_INS 0x0152
#define KEY_DEL 0x0153

```



```

#define KEY_CROSS 0x014C /* keypad "5" */
#define KEY_RETURN 0x000d
#define KEY_SPACE 0x0020
#define KEY_BACKSPACE 0x0008
#define KEY_TAB 0x0009
#define KEY_DOT '.'
#define KEY_ESC 0x001b
#define CTRL_DEL 0x0193
#define ALT_B 0x0130
#define ALT_D 0x0120
#define ALT_H 0x0123
#define ALT_K 0x0125
#define ALT_M 0x0132
#define ALT_X 0x012D
#define ALT_S 0x011F
#define ALT_T 0x0114
#define ALT_PGUP 0x0199
#define ALT_PGDN 0x01A1
#define ALT_DEL 0x01A3

struct _win_st
{
    int *_buffer;
    int _maxx;
    int _maxy;
    int _begx;
    int _begy;
    int _curx;
    int _cury;
    int _ubegx;
    int _ubegy;
    int _umaxx;
    int _umaxy;
    int _attr;
};

typedef struct _win_st WINDOW;

```

```

extern int      *REAL_SCREEN_BASE,color_set,initial_cur_beg,initial_cur_end;

extern  initscr(),endwin(),clrscrn(),waddch(),wdelch(),mvwdelch(),
/*      winstch(),mvinstch(),wmove(),mvwaddch(),waddstr(),mvwaddstr(),
/*      clrwin(),fillwin(),box(),wattrset(),wrefresh(),wnoutrefresh(),
/*      touchwin(),gotoyx(),getcurpos(),set_cursor_size(),get_cursor_size(),
/*      delwin(),beep(),savescrn(),restorescrn();

extern  int      winch(),mvwinch(),wgetch();

extern  WINDOW   *newwin();

```

```

#endif

```



```

*****
**
**
** FIELDIO.H
**
**
** FIELDIO Module Include Header File
**
**
** Creation : 15/3/1991 by Umit KALKAN
**
*****
*****
*****
#define __FIELDIOINC

#include <datatype.h>

#define SEL_REQ_KEY      ALT_S

#define GETKEY(A)        wgetch(A)

typedef enum {ENTR_ONLY,
              SEL_ONLY,
              DISP_ONLY,
              ENTR_AND_SEL,
              NO_OF_INP_STYLES}  INP_STYLE;

typedef enum {SIMPLE_SEL,
              BOXED_SEL,
              NO_OF_SEL_STYLES}  SEL_STYLE;

typedef enum {JUST_DO,
              JUST_CALL,
              DO_AND_CALL,
              CALL_AND_DO,
              CALL_DO_CALL}      CALL_FLAG;

typedef struct
{int          no_of_keys;
 int          *key_list;}      KEY_INFO;

typedef struct
{int          box_color;
 int          text_color;
 int          sel_color;}      COLOR_INFO;

typedef struct
{int          y;
 int          x;}              POSITION_INFO;

typedef struct

```

```

{int      y;
 int      x;
 int      mant_len;}      SIZE_INFO;

```

```
typedef struct
```

```

{int      no_of_sel;
 char      **sel_list;}      SEL_INFO;

```

```
typedef struct
```

```

{DATA_TYPE      data_type;
 POSITION_INFO position;
 SIZE_INFO      size;
 INP_STYLE      inp_style;
 SEL_STYLE      sel_style;
 SEL_INFO      sel_info;
 void          *usr_info;
 CALL_FLAG      call_flag;
 char          *out_buff;
 int          *selection;}      FIELD_INP_INFO;

```

```

extern int      alpha_input(),
                num_input(),
                date_input(),
                time_input(),
                simple_selection(),
                boxed_selection();

```

```

extern          alpha_display(),
                num_display(),
                date_display(),
                time_display();

```



```
extern int (*inp_procs[NO_OF_DATA_TYPES})();
extern int (*sel_procs[NO_OF_SEL_STYLES})();
extern (*disp_procs[NO_OF_DATA_TYPES})();
```

```
#endif
```

```
/*
```

```
*/
```

```
/*
```

```
*/
```

```
*/
```

```
*/
```

```
*/
```

```

/*****
*****
**
**                               **
**      NULLPTR.H                **
**                               **
**      Definition of System Based NULL pointer Value      **
**                               **
**      Creation : 15/3/1991 by Umit KALKAN                **
**                               **
*****
*****/

```

```

#ifndef __NULLPTRINC

```

```

#define __NULLPTRINC

```

```

#ifndef MK_FP

```

```

    #include <dos.h>

```

```

#endif

```

```

#define NULL_PTR(MK_FP(0L,0L))

```

```

#endif

```



```

/*****
*****
**
**          DATATYPE.H          **
**
**          Definitions of Common Data Types          **
**
**          Creation : 15/3/1991 by Umit KALKAN          **
**
*****
*****/

#ifndef __DATATYPEINC
#define __DATATYPEINC

/*
 * Muzayyi: aramadan önce bir saniye bekler.
 * T      Çift Tonlu Arama yapar.
 * P      Darbeli Arama yapar.
 * K      Sadece gönderme yapabilen modemi arama.
 */
typedef enum {DAT_ALPHA,
              DAT_NUM,
              DAT_DATE,
              DAT_TIME,
              NO_OF_DATA_TYPES} DATA_TYPE;

/*
 * Cevir sesi için bekler.
 */
#define TIME_LEN 5
#define DATE_LEN 10

#endif

/*
 * B<n>   Telefona cevap vermez.
 * S<n>   Telefona bir zilde cevap verir.
 * S<n>   Telefona "n" zilde cevap verir.
 * B<n>   Telefona zil bekleme süresi cevap verir.
 * B<n>   Çalışma standardını belirler (0:CCITT, 1:GSM).
 * F<n>   Eko kontrol (0:Eko yok, 1:Eko var).
 * B<n>   0:Half duplex, 1:Full duplex.
 * B<n>   0:Hatı bırak, 1:Hatı al.
 * O      Hat bağlantısına göre döndürür.
 * Q<n>   Sonuç kodu kontrolü.
 * S      Başlangıç değerlerine döndürür.
 */

```

EK : HAYES Akıllı Modem Komutları

Aşağıdaki komutlar, Hayes akıllı modem komut setinin bazı komutlarıdır ve "AT" karakterlerinden sonra peşpeşe kullanılabilirler. Daha ayrıntılı bilgi için bu komut setini kullanan herhangi bir akıllı modemin kullanım kılavuzuna baş vurunuz.

D <num> Numarayı Arar.
 , Numarayı aramadan önce bir saniye bekler.
 T Çift Tonlu Arama yapar.
 P Darbeli Arama yapar.
 R Sadece gönderme yapabilen modemi arama.
 A/ Son komutu tekrarlar.
 ; Aramadan sonra tekrar komut durumuna döner.
 ! Hattı kapatıp açar. (Flaş).
 / 1/8 saniye bekler.
 @ Sessizlik algılamak için bekler.
 W Çevir sesi için bekler.
 S0=0 Telefona cevap vermez.
 S0=1 Telefona bir zilde cevap verir.
 S0=<n> Telefona "n" zilde cevap verir.
 A Telefona zil beklemeksizin cevap verir.
 B<n> Çalışma standardını belirler (0:CCITT, 1:BELL).
 E<n> Eko kontrol (0:Eko yok, 1:Eko var).
 F<n> 0:Half duplex, 1:Full duplex.
 H<n> 0:Hattı bırak, 1:Hattı al.
 O Hat bağlantısına geri döndürür.
 Q<n> Sonuç kodu kontrolü.
 Z Başlangıç değerlerine döndürür.

(8) MS-DOS Version 3.30 Programmers Reference Manual. (Microsoft Corp.).

(9) VT-220 Terminal Programmers Guide. (Digital Corp.).

KAYNAKLAR

- (1) Data Communications, Computer Networks and OSI. (Fred HALSHALL).
- (2) Electronic Communication Systems. (George KENNEDY).
- (3) Computer Networks. (Andrew S. TANENBAUM).
- (4) Turbo C Programmers Guide and Reference Manual. (Borland International Corp.).
- (5) IBM PC Technical Reference Manual. (IBM Corp.).
- (6) Commodore PC 10-20-II Technical Reference Manual. (Commodore Corp.).
- (7) Inside The IBM PC. (Peter NORTON).
- (8) MS-DOS Version 3.30 Programmers Reference Manual. (Microsoft Corp.).
- (9) VT-220 Terminal Programmers Guide. (Digital Corp.).

1983 yılında T222K'de DMS (Digital Multiplexed System) telefon santralleri üzerinde yazılım destek mühendisi olarak görev aldı. Bu sırada Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsünde Elektronik dalında yüksek lisans eğitimi almaya başladı. Yüksek Lisans Tezi olarak yine kişisel bilgisayarlar için netaş (Netaş) ile modem veri iletişimi ve birji arşim programının tasarımı ve gerçekleştirilmesi üzerine KALKAN, bu projede Doç. Dr. Filiz GÜNEŞ ile birlikte çalıştı.

Özgeçmiş

Umit KALKAN, 19 Ocak 1967 tarihinde İstanbul ilinin Fatih ilçesinde doğdu. İlk öğreniminin ilk iki yılını Kadırga İlkokulunda yapan KALKAN, ailesiyle beraber 1976 yılında Kartal-Küçükalya yerleşerek ilk öğreniminin geri kalan bölümünü İdealtepe İlkokulunda tamamladı. 1978 yılında ilkokuldan mezun olduktan sonra orta öğrenimini İnönü Ortaokulundan birincilik derecesiyle, lise öğrenimini ise 50. Yıl Küçükalya Lisesinden ikincilik derecesiyle mezun olarak tamamladı.

1984 yılında İstanbul Yıldız Üniversitesine girdi ve bu üniversitenin Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği bölümünden 1988 yılında mezun olana kadar, aynı zamanda bilgisayar endüstrisinde hizmet veren özel bir kuruluştaki teknik görevli olarak çalıştı. Bitirme ödevi olarak üzerine aldığı IBM PC uyumlu kişisel bilgisayarlar için ROM Disk projesinde Prof. Dr. Sezgin ALSAN ile beraber çalıştı ve bu projeyi başarıyla tamamladı. Uygulamalı staj eğitimini Netaş A.Ş. de tamamladıktan sonra, bu kuruluşun Araştırma ve

Geliştirme bölümünde DMS (Digital Multiplexed System) telefon santralleri üzerinde sistem yazılımı destek mühendisi olarak görev aldı. Bu sırada Yıldız Üniversitesi Fen Bilimleri Enstitüsünde Elektronik dalında yüksek lisans eğitimi yapmaya başladı. Yüksek Lisans Tezi olarak yine kişisel bilgisayarlar için hata düzeltmeli bir modem veri iletişim ve bilgi erişim programının tasarımı ve gerçekleştirilmesini üstlenen KALKAN, bu projede Doç. Dr. Filiz GÜNEŞ ile birlikte çalıştı.

Üniversite eğitiminin tamamı boyunca, endüstrideki uygulamalı mühendislik bilgileriyle üniversiteden edindiği teorik bilgileri pekiştirmiş olan Ümit KALKAN şu anda TÖBANK A.Ş. de uzman yazılım geliştirme mühendisi olarak görev yapmaktadır.

