22021

İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

PC LER ARASINDA VERİ İLETİŞİMİNİ SAĞLAYAN BİR YAZILIM

YÜKSEK LİSANS TEZİ

Müh. Osman Nuri Özpınar

Tezin Enstitüye Verildiği Tarih : 22 Haziran 1992
Tezin Savunulduğu Tarih : 7 Temmuz 1992

Tez Danışmanı : Doç.Dr. Mithat Uysal
Diğer Jüri Üyeleri : Doç.Dr. Fikret Balta
Y.Doç.Dr. Celal Tuncer

TEMMUZ 1992

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

ÖNSÖZ

Bu çalışmada bana yol göstererek özgün bir çalışma yapmamı destekleyen değerli tez danışmanım Sayın Doç. Dr. Mithat UYSAL ` a , donanım ile ilgili sorunların çözümündeki katkılarıyla değerli arkadaşım Cüneyt SELEK ` e ve bu çalışmamdaki tüm problemlerin çözümünde bana verdiği bilgiler ve moral için YÖNSİS Ltd. Şti.nden değerli ağabeyim Mustafa ÖZPINAR ` a teşekkürü bir borç bilirim.

Temmuz 1992 O.Nuri Özpınar

İÇINDEKILER

ÖZET

Bu çalışmada, hiç bir ek donanıma gereksinim olmadan birden fazla kişisel bilgisayarın veri alışverişinde bulunabilmesi üzerine geliştirilen bir yazılım sunulmuştur. Bu yazılım, genel olarak, kişisel bilgisayarların birbirleriyle haberleşmesine yönelik olmakla birlikte gözönünde bulundurulması gereken önemli bir nokta da ; bu haberleşme sağlanırken bilgisayarların salt diğer bilgisayarlardan gelen mesajı algılaması ya da cevap vermesi değil, bu işlemleri yaparken kendi ortamındaki mevcut işe devam edebilmesidir. Bu yazılım sayesinde birden fazla kişisel bilgisayar çoklu hizmet özelliğini kullanarak birbirleriyle veri alışverişinde bulunabilecektir.Bu yazılımın geliştirilmesinde amaç olarak küçük işletmelerde en düşük maliyetle bir mini bilgi ağının kurulabilirliği esas alınmıştır.

Bu çalışmadaki yazılımlar, kişisel bilgisayarlarda DOS (Disk Operating System/Disk İşletim Sistemi) ile çalışmaktadırlar. Haberleşme RS-232 seri port vasıtası ile yapılmaktadır. İki makine arasında kullanılan bağlantı elemanları standart, ucuz ve piyasada rahatlıkla bulunabilen malzemelerden (kablo ve konnektörler) ibarettir. Zaten günümüzde pazarlanan tüm PC lerde en az bir tane RS-232 seri port mevcut olmakla beraber bu sayı artırılabilmektedir. Bu çalışmada bir ana makine ve iki kullanıcı makine esas alınmıştır. Ancak iki olan kullanıcı sayısı yine piyasada çok ucuza satılan RS-232 seri portları vasıtasıyla artırılabilir. Tabiiki kullanıcı sayısı arttıkça sistemin verimliliği düşecektir. Bu sistemin ideal çalışma ortamı bir ana makine ve iki kullanıcı makine olarak tasarlanmıştır.

A SOFTWARE ABOUT DATA COMMUNICATION BETWEEN PCs

SUMMARY

Many organization already have a substantial number of computers in operation, often located far apart. For example, a company with many factories may have a computer at each location to keep tracck of inventories, monitor productivity, and do the local payroll. Initially, each of these computers may have worked in isolation from the others, but at some point, management may have decided to connect them to be able to extract and correlate information about the entire company.

Put in slightly more general form, the issue here is resource sharing, and the goal is to make all programs, data, and equipment available to anyone on the network without regard to the physical location of the resource and the user. In other words, the mere fact that a user happens to be 1000 km away from his data should not prevent him from using the data as though they were local. Load sharing is another aspect of resource sharing. This goal may be summarized by saying that it is an attempt to end the "tyranny of geography".

A second goal is to provide high reliability by having alternative sources of supply. For example, all files could be replicated on two or three machines, so if one of them is unavailable (due to a hardware failure), the other copies could be used. In addition, the presence of multiple CPUs means that if one goes down, the others may be able to take over its work, although at reduced performance. For military, banking, air traffic control, and many other applications, the ability to continue operating in the face of hardware problems is of great importance.

Another goal is saving money. Small computers have a much better price / performance ratio than large ones. Mainframes are roughly a factor of ten faster than the fastest single chip microprocessors, but they cost a thousand times more. This imbalance has caused many

systems designers to build systems consisting of powerful personal computers, one per user, with data kept on one or more shared file server machines.

This goal leads to networks with many computers located in the same building. Such a network is called a LAN (Local Area Network) to constrast it with the farflung WAN (Wide Area Network), which is also called a long haul network.

A closely related point is the ability to increase system performance gradually as the workload grows just by adding more processors. With central mainframes, when the system is full, it must be replaced by a larger one, usually at great expense and with even greater disruption to the users.

Yet another goal of setting up a computer network has little to do with technology at all. A computer network can provide a powerful communication medium among widely separated people. Using a network, it is easy for two or more people who live far apart to write a report together. When one author makes a change to the document, which is kept online, the others can see the change immediately, instead of waiting several days for a letter. Such a speedup makes cooperation among far-flung groups of people easy where it previously had been impossible. In the long run, the use of networks to enhance human-to-human communication may prove more important than technical goals such as improved reliability.

In the following figure we give a classification of multiple processor systems arranged by physical size. At the top are data flow machines, highly parallel computers with many functional units all working on the same program. Next come the multiprocessors, systems that communicate by exchanging messages. Finally, the connection of two or more networks is called internetworking.

interprocessor distance	Processors located in same	Example
0.1 m	Circuit Board	Data Flow Machine
1 m	System	Multiprocessor
10 m	Room	Local Network
100 m	Building	
1 km	Campus	
10 km	City	Long haul network
100 km	Country	
1000 km	Continent	interconnection of long haul networks
10,000 km	Planet	

Replacing a single mainframe by workstations on a LAN does not make many new applications possible, although it may improve the reliability and performance. IN contrast, the availability of a (public) WAN makes many new applications feasible. Some of these new applications may have important effects on society as a whole. To give an idea about some important uses of computer networks, we will now briefly look at just three examples; access to remote programs, access to remote databases, and value-added communication facilities.

A company that has produced a model simulating the world economy may allow its clients to log in over the network and run the program to see how various projected inflation rates, interest rates, and currency fluctuations might affect their businesses. This approach is often preferable to selling the program outright, especially if the model is constantly being adjusted or requires an extremely large mainframe computer to run.

Another major area of network use is access to remote datebases. It may soon be easy for the average person sitting ta home to make reservations for airplanes, trains, buses, boats, hotels, restaurant, theatre, and so on, anywhere in the world with instant confirmation. Home banking and the automated newspaper also fall in this category. Present newspaper affer a little bit of everything, but electronic ones can be easily

tailored to each reader's personal taste, for example, everything about computers, the major stories about politics and epidemics.

Next step beyond automated newspaper (plus magazines and scientific journals) is the fully automated library. Depending on the cost, size, and weight of the terminal, the printed word may become obsolete. Skeptics should take note of the effect the printing press had on the medieval illuminated manuscript.

All these applications use networking for economic reasons; calling up a distant computer via a network is cheaper than calling it directly. The lower rate is possible because a normal telephone call ties up an expensive, dedicated circuit for the duration of the call, whereas access via a network ties up long-distance lines only while data are actually being transmitted.

A third category of potential widespread network use is as a communication medium. Computer scientists already take it for granted that they can send electronic mail from their terminals to their colleagues anywhere in the world. In the future,it will be possible for everyone, not just people in the computer business, to send and receive electronic mail. Furthermore, this mail will also be able to contain digitized voice, still picture and possibly even moving television and video images. One can easily imagine children in different countries trying to learn each other's languages by drawing a picture of a child on a shared screen and labeling it girl, jeune fille, or meisje.

Electronic bulletin board systems already exist, but these tend to be used by computer experts, are oriented towards technical topics, and are often limited in geographic scope. Future systems will be national or international, be used by millions of nontechnical people, and cover a much broader range of sunjects. Using a bulletin board may be as common as reading a magazine.

It is sometimes said that there is a race going on between trasportation and communication, and whichever one wins will make the other unnecessary. Using a computer network as a sophisticated communication

system may reduce the amount of traveling done, thus saving energy. Home work may become popular, especially for part-time workers with young children The office and school as we now know them may disappear. Stores may be replaced by electronic mail order catalogs. Cities may disperse, since high quality communication facilities tend to reduce the need for physical proximity. The information revolotuion may change society as much as the industrial revolution did.

The above discussion of computational costs neglects the cost of software. While the art of software design has been improving, the improvement is partly counter-balanced by the increasing cost of good software engineers. When software can be replicated, however, the cost per unit goes down inversely with the number of replicas. Thus, even though software is a major cost of a new computer system, the increasing market decreases its unit cost. Each advance in solid state technology decreases cost and increases the performance of computer systems; this leads to an increase in market, thus generating decreased unit software costs, leading, in a feedback loop to furher increases in market. Each new application, however, requires new specialized software which is initially expensive and which requires a user learning curve. Thus, it is difficult to forecast the details of the growth of the computer market and similarly of the data network market.

The cost of transmitting data on a communication link from one point to another has also been dropping, but at a much slower rate than computational costs. The cost of a data link increases with the available data rate, but the increase is much less than linear in the data rate. Thus, the cost per transmitted binary symbol decreases with the data rate of the link. It is often economically advantageous for many users to share one high-rate data link rather than having separate data links for each.

One result of sharing high-speed communication links is that the cost of sending data from one point to another increases less than linearly with the geogrraphic separation of the points. This occurs because the communication path could include a short link to a shared long-distance, high-speed link and then another short link to the destination.

Estimating the cost of transmission facilities is

highly specialized and complex. The cost of a communication link depends on whether one owns the facility or leases it; with leasing, thee cost depends on the current competitive and regulatory situation. The details of communication cost will be ignored in what follows, but there are two overall effects of these costs that are important.

First, for wide area networks, the cost of a network is presently dominated by transmission costs. Thus, it is desirable to use the the communication links efficiently, perhaps at added computational costs. The sporadic nature of most data communication, along with the high cost of idle communication links, have led to the development of packet data networks which can increase utilization o the links.

Second, for local area networks, the cost of a network iss not dominated by transmission costs. Coaxial cable and even a twisted pair of wires can achieve relatively high-speed communication at modest cost in a small geographic area. The use of such media and the desire to avoid relatively expensive switching have led to a local area network technology in which many nodes share a common high speed communication mediumon a shared multiaccess basis.

The software presented in this study has been developped on a data exchange basis available among several personal computers without the need of an additional hardware. Although this software is generally focused on the communication among personal computers, another point which should be considered is that while this communication is provided, th PC should not only perceive or respond to the message from the other PCs but it should continue the present job which it`s performing. Due to this software several PCs will be able to exchange using their multitasking facility. The aim of developping this software is to establish a mini network of low cost in the small enterprises.

The softwares presented in this study, are being used in personal computers by Disk Operating System (DOS). Communication is provided through serial port RS-232. The tools (such as cable and connector) connecting two machines are standard and cheap equipments easily available in the market. In all personal computers marketed nowadays, there is at least one serial port RS-232, but this number can be increased. Although the concern of this study is one main machine and two user machines, the number of user machines can be increased due to serial port RS-232 which is very cheap in the market. But the increase in the

number of user machines will decrease the system performance. The ideal functionary environment of this system is planned for one main machine and two user machines.

The programs in this study is written by Pascal programming language and basically they run as a sample network logic.

1. BILGISAYAR AĞLARI

İşletmelerin gereksinimlerinin tümünün bir bilgisayar ile karşılanması yerine birden çok bilgisayarın ayrı ayrı yerlerde ama birbirine bağlı olarak kullanımı bilgisayar ağlarını (computer networks) oluşturmuştur. Bilgisayar ağları birden çok bilgisayardan oluşur ve yerleşim olarak belli bir biçimdedir.

Bilgisayar ağlarının ilk uygulamaları 1960 lı yılların sonlarında başlamıştır. Ancak yaygın ve yerel bilgisayar ağlarının yaygınlaşması 1980 li yıllarda başlamış ve gelişmiştir. 1980 yıllarda, kişisel bilgisayarların çoğalması, bilgisayar teknolojisindeki ve iletişim teknolojilerindeki gelişmeler bilgisayar ağlarının daha yararlı olmasını sağlamıştır. Günümüzde yerel bilgisayar ağlarındaki iletişim hızı genelde 1 - 20 Mbit/S dir. Özellikle fiber optik teknolojisindeki gelişmeler bu hızı çok fazla artırmıştır.

Bilgisayar ağı, birbirine bağlı (interconnected) birçok bağımsız bilgisayar anlamına gelir. İki bilgisayarın birbirinin kaynaklarını (diskini ya da diskinde yer alan bilgilerini) paylaşabilmesi ve konuşabilmesi onların birbirine bağlı olduğunu gösterir. Aradaki bu bağ kablo ile olabileceği gibi mikro dalga, laser ve uydular aracılığıyla da olabilir.

Bilgisayar ağlarının belli bir yerleşiminin olduğu belirtilmişti. Ağlar, yanyana duran iki bilgisayar

arasında gerçekleştirilebileceği gibi, ülke ya da dünya üzerine yayılmış birçok bilgisayar arasında da gerçekleştirilebilir. Belli bir alanda (şirket binası, okul, hastane, vb) kurulan bilgisayar ağları, yerel bilgisayar ağları (Local Area Network) olarak adlandırılırlar. Ülke ya da dünya üzerinde hizmet veren büyük ağlar ise yaygın bilgisayar ağları ya da uzun mesafeli bilgisayar ağları (Longhaul Network) alarak adlandırılırlar.

İşletmecilik açısından ağlar, yönetime ve denetime yardımcı olurlar. Bir bankanın ya da üniversitenin çok sayıda bilgisayarı birbirine bağlı olarak kullanması, onları bağımsız olarak kullanmasından daha anlamlı ve verimli olur. Böylece birimler arası iletişim daha kolay sağlanmakta ve bütünleşik (integrated) uygulamalar daha kolay gerçekleştirilmektedir.

Büyük organizasyonların bilgi işlem merkezi uygulamaları merkezcil ve merkezkaç olabilir. Merkezkaç ya da dağıtımlı (decentralized) bilgi işlem uygulamasında, bir organizasyonun bilgi işlem hizmetlerine gereksinim duyan değişik birimleri ayrı ayrı bigisayar sistemleri kullanarak kendi bilgi işlem çalışmalarını kendi sistemlerinde yürütürler. Dağıtımlı bilgi işlemin en büyük özelliği, bölüm ya da birime özel bilgi işlemi sağlamasıdır.

Dağıtımlı ya da birleşik (merkezcil/centralized) bilgi işlem politikası, organizasyonun kendi öz politikası olmakla beraber kurulacak bilgisayar ağlarıyla bu yapılar desteklenir.

Dağıtımlı bilgisayar sistemleri ile bilgisayar ağları

arasında zaman zaman literatürde karışıklık olur.Bir bilgisayar ağı dağıtılmış bir bilgi işlemi sağlayabileceği gibi merkezcil bir bilgi işlemi de sağlayabilir.

1.1. BILGISAYAR AĞLARININ AMAÇLARI

Bilgisayar ağları, özel amaçlı, eğitim amaçlı, ulusal olarak ve halka açık olarak kurulabilir. Yerel bilgisayar ağları (Local Area Network, LAN) ise genellikle bir bina, okul, hastane gibi sınırlı bir alanda ve genellikle kişisel bilgisayarların yeraldığı ağlardır. Yerel bilgisayar ağları, çokluk ofis otomasyonu için kurulur ve firmanın organizasyonuna göre yerleşimi biçimlendirilir.

Merkezleşmiş bilgisayar sistemleri yerine bilgisayar ağlarının kurulmasının iki amacı vardır. Birinci amaç, işletmelerde ayrı ya da aynı yerlerde çok sayıda bilgisayarın ya da kişisel bilgisayarın bulunmasıdır. Örneğin bir şirketin farklı departmanlarında bir çok bilgisayar, yerine göre muhasebe, bordro, stok takibi alanlarında kullanılmaktadır. Ancak tümüne ilişkin kararların verilmesi ya da bilgi elde edilmesi sürecinde bu bilgisayarların çıktılarının birleştirilmesi gerekir. Oysa şirket içinde bir yerel bilgisayar ağı gerçekleştirilmiş olsa bütün bilgisayarlar ve kullanıcıları iletişimin içinde olurlar. Bununla birlikte yönetimde istediği nitelikte raporlar alabilir.

Bilgisayar ağlarının amaçlarının ikincisi ise ölümcül donanım sorunlarının önlenmesidir. Örneğin muhasebe uygulamasının yürütüldüğü bilgisayarda bir arızanın oluşması onun tümüyle kullanılamaması ve muhasebe uygulamasının

kesilmesi anlamına gelir. Oysa yerel bilgisayar ağı üzerinde bir terminalin yerine başka bir iş istasyonu kullanılabilir.

Bilgisayar ağlarının temel amacı, ağ içindeki kullanıcıları iletişir, konuşur hale getirmek ve özgün uygulamalarına destek olmaktır. Yerel bilgisayar ağı (LAN) olarak gerçekleştirilen ağlar, hem bağımsız çalışmaları ve hem de iletişimi ve merkezcil yönetimi de desteklerler.

Bilgisayar ağları kullanıcılarına birçok olanağı da sunarlar; kullanıcılar bilgisayar ağlarına girerek yeni yazılımlar elde edebilirler. Yine bilgisayar destekli eğitimde ya da üniversiteler arası bilgi alışverişlerinde bilgisayar ağları çok yararlı bir eğitim ortamı sağlarlar. Diğer bir olanak da uzak veri tabanlarına erişimdir. Örneğin bir bilgisayar kullanıcısı kendi bilgisayarından uzak veri tabanlarına girerek sermaye piyasası hakkında bilgi alabilir.

Sonuç olarak, ağlarla sağlanan iletişim olanakları onların en büyük amaçlarının oluşturur. Bu olanak merkezcil ya da merkezkaç işletme yönetimini de destekler.

Bugün dünyada değişik amaçlı birçok yaygın bilgisayar ağı hizmet vermektedir. Bunların en bilinenlerinden bazıları ARPANET, SNA, DECNET, EARN, BITNET dir.

Üniversiteler arası ilk bilgisayar ağları Amerikada 1960 lı yılların sonunda geliştirilmiştir. Bu ağlar bilimsel alanda işbirliği, elektronik postalama ve telekonferans gibi hizmetleri sağlayarak çok yararlı olmuşlardır.

Günümüzde üniversiteler arasında kurulu yaygın bilgisayar ağlarından bazıları JANET (İngiltere), SUNNET (İsveç) dir.

1982 yılında çalışmalarına başlanılan ve Avrupa Akademik topluluğunun kendi ağı olan EARN, aralarında ülkemizin de bulunduğu birçok ülkeye hizmet vermektedir. EARN Avrupa üniversitelerine açık bir yaygın ağdır. Kullanıcılarına elektronik posta, kütük aktarımı, telekonferans, gibi olanaklar sağlar. Bunun dışında uzak veri tabanlarına erişim ve sorgulama olanakları da vardır.

2. AĞ YAPILARI

Bütün bilgisayar ağlarında kullanıcıların gereksinimlerini yerine getirmek üzere birçok makine (bilgisayar) kullanılır. Genel görünüm ana makine ya da makineler ve onlara bağlı uçlar biçimindedir.

İlk ağlarda, örneğin ARPANET de bu makinelere IMP (Interface Message Processor) denilmektedir. Diğer ağ uygulamalarında ise ana makine (host) ve iş istasyonlarından söz edilir. Bir diğer tanımlama da alt ağ (subnet) tanımlamasıdır. Ana makineler bir iletişim alt ağı ile kullanıcılara bağlıdır. Alt ağın görevi makineden makineye mesaj taşımaktır. Bu iletim aynı telefon iltişimindeki gibidir.[1]

Ağların standardı ile çeşitli oraganizasyonlar ilgilenmektedir. Bunların başında ISO (International Standards Organization) tarafından önerilen standartlar gelir. Ayrıca Avrupa Bilgisayar Yapımcıları Birliği (ECMA) de bu konuda çalışmalar yapmaktadır.

2.1 PROTOKOL HIYERARŞİLERI

Bilgisayar ağları, tasarımlarında karmaşıklığın azaltılması ve birimselleştirilmesi bakımından bir dizi katman (Layer) ve düzey (Level) den oluşur. Katmanların fonksiyonları değişik ağlarda farklılık gösterebileceği

gibi her düzeyin bir üst düzey için belli bir görevi yerine getirdiği söylenebilir.[2]

Bir makinedeki n düzeyi diğer makinedeki n düzeyi ile konuşur. Bu iletişimin kurallarıda o düzeyin protokolünü oluşturur. Tasarımda diğer bir konuda bilgi gönderim kurallarıdır. bilgi gönderimi yanlız bir yönde yapılıyorsa tek yönlü (Simplex) ya da basit iletişim olarak anılır. Bilgiler iki yönlü olarak aynı anda iletilebiliyorsa tam-çift yönlü (Full-Duplex) iletişimden söz edilir. İki yönlü bilgi iletişimi aynı anda yapılmıyorsa yarım-çift yönlü (Half-Duplex) iletişim söz konusudur.

Fiziksel iletim, iletişim hattından yapılır. İletişim hattı da bir kanal olarak farklı biçimlerde kullanılabilir.

Temelde üç tür kanal vardır:
-Tek yönlü (basit/simplex)
-Yarım çift yönlü (Half Duplex)
-Tam çift yönlü (Full Duplex)

Tek yönlü kanal bilginin yalnızca tek bir yönde akmasına izin verir. Tek yönlü kanal pek kullanılmamaktadır. Yarım çift yönlü kanalda ise aynı anda olmamakla beraber bilgi alışverişi iki yöndede yapılabilir. Yani herhangi bir anda yalnız bir yönde bilgi akar ve sırayla her iki yöndede bilgi iletimi sağlanır.

Tam çift yönlü kanalda bilgiler hem iki yönde hem aynı anda hareket ederler. Bu kanalda iletimde herhangi bir kesiklik sözkonusu değildir.

2.2. ISO AĞ YAPISI

ISO (International Standard Organization) belli bir model ağ önerisine sahiptir. Bu kurallara göre bir ağ yedi düzeyde ele alınır.

1-Fiziksel katman: Bilgilerin iletişim kanalları aracılığıyla gönderilmesini sağlar.

2-Bilgi birleştirme (Veri Bağlantı) katmanı: Bilgisayar ağına bilgi gönderme olanağını kapsar. Girdi bilgi birimlere bölünerek iletilir.

3-Ağ katmanı:Alt ağın iletişimini sağlar.

4-Gönderim katmanı:İdare katmanından gelen bilgiler diğer bir makineye ulaştırılır.

5-Oturum katmanı:Kullanıcı bilgisayar ağına bu kısımda katılır. Kullanıcılar arası iletişim sağlanır ve kullanıcının uzak veri tabanlarına ulaşması sağlanır.

6-Sunuş katmanı:Kullanıcılara çeşitli olanaklar sağlanır.

7-Uygulama Katmanı:Uygulama katmanı kullanıcılara bağlı olarak değişiklik gösterir. Bilgisayar ile kullanıcılar arasındaki iletişim sağlanır.

2.3. VERİ AKTARIMI

Tek bir iletişim kanalıyla birçok kişinin iletişimini düzenle işlemine çoklama (çok düzeyleme/multiplexing) denir. Çoklama iki temel biçimde yapılır; zaman bölümlemeli ve frekans (sıklık/aralık) bölümlemeli.

Zaman bölümlemesinde kullanıcı belli bir zaman aralığına sahiptir. Frekans bölümlemesinde ise tek bir hat mantıksal olarak bölümlenir.

Çoklama yöntemi ile bir çok ileti bir arada iletilir. Her iki yönde de bilgi gönderiminin olanklı olduğu çoklama sistemlerinde çok sayıda aygıtın (terminal, bilgisayar, giriş/çıkış birimi) farklı çalışma hızlarıda uyumlaştırılarak iletişimi sağlanır.

Bilgi gönderiminde sayısal gönderim, örneksel gönderime göre çok daha doğru sonuç verir. Bu amaçla terminallere birer sayısal arabirim (Digital interface) eklenir.

İnsanların telefon aracılığıyla birbiriyle konuşmaları ile bilgisayarların birbirleriyle konuşmaları farklı esaslara sahiptir. İnsanlar telefonla konuşurken beklemelerin belli bir düzeni olmamakla birlikte bilgisayarlar arası iletişimde bu bekleme düzenlidir. Bu bağlamda iletişimi düzenlemek için devre anahtarlama ve paket anahtarlama yöntemleri geliştirilmiştir.

Devre anahtarlama (circuit switching) yönteminde bilgi iletimi hat boyunca belli aralıklarla yapılır. Paket anahtarlama (Packet switching) iletişim yönteminde ise belli miktarlardaki bilgi öğeleri bir anahtarlama ofisinde önce paketlenerek saklanır ve gönderilir. Paket anahtarlama yönteminde birçok paket hat üzerinde ilerleyebilir.

Paket anahtarlamalı yöntemde bilgilerin belli bir düzene sokulması (paketlenmesi) ve açılması işleminde kullanılan birime ya da aygıta paket düzenleyici/açıcı (Packet Assembler/ Disassembler) (PAD) denir.

Çoklayıcılar (Multiplexer) farklı kaynaklardan gelen sinyalleri birbirine karışmamasını sağlarlar. Eğer yayın yapan uçların herbirine belli zaman aralıkları tahsis ediliyorsa bu durumda zaman temelli çoklama yapılıyor demektir. Her verici kanalına ayrı bir frekans bandı tahsis ediliyorsa frekans temelinde çoklama yapılıyor demektir.

Zamanlı çoklayıcıların kapasiteleri sınırlıdır. İletim hızı ve mevcut kapı sayısı ile orantılı olarak bilgi iletimi yapabilir. Oysa frekans çoklayıcılarının kapasitesi çok daha geniştir.

3. YEREL BILGISAYAR AĞLARI

Yerel Bilgisayar Ağları (YBA), özgün adı ile Local Area Network (LAN), sınırlı bir coğrafi alan içinde bulunan bilgisayarlar ve diğer çeşitli aygıtlar arasında oluşturulan bir bilgi iletişim sistemidir. Yerel ağlar genellikle bir bina, binalar grubu, fabrika, üniversite kampüsü gibi alanlarda kurulurlar. Yönetim ya da ticari amaçlı olarak yerel bilgisayar ağları ofis otomasyonunu sağlamada kullanılırlar.

Yerel bilgisayar ağlarının ortak özelliklerini şöyle sıralayabiliriz:

- Sınırlı bir alana yayılmış ve sayıları binlere varabilen bağımsız birimlerin (kişisel bilgisayar ve mini bilgisayar) bağlanabilmesi.
- Yeniden organize edilebilme, bakım ve kurma kolaylığı.
- Başka aygıtların da ağa katılması.
- Büyüyebilirlik.
- Yüksek hız ve düşük hata oranı.
- Sistemin tüm birimlerce verimli kullanımı.
- Kontrolun ağdaki birimlerce paylaşılması.

Yerel bilgisayar ağları içinde özellikleri birbirinden farklı olan çeşitli aygıtlar bulunabilir. Örneğin büyük bilgisayarlar, mini bilgisayarlar, kişisel bilgisayarlar, yazıcılar, disk sürücüleri, terminaller ve diğer sayısal ya da örneksel aygıtlar ağ içinde yer alabilirler.

Yaygın ya da uzun mesafeli (long haul network) bilgisayar ağlarında gerekli olan iletişim yöntemi kullanılır. Oysa yerel ağlarda bu zorunluluk yoktur ve

istenilen iletişim teknolojisi seçilebilir. Bu özelliği ile de yerel ağları yaygın ağlardan ayırt edebiliriz.

Yerel bilgisayar ağları çokluk topolojilerine (yapı/yerleşim) ve iletişim protokollerine göre sınıflanırlar. Yerel ağlar, kullandıkları iletişim ortamları, bağlantı biçimleri ve protokollerine göre farklılık göstermekle beraber bu alanda bir standartlaşmaya doğru da gidilmektedir.

Yerel bilgisayar ağının topolojisi onu oluşturan dü ğümlerin (node/iş istasyonu/uç ya da terminal), bağlantıların ve diğer birimlerin fiziksel yapısı ya da yerleşimidir. Yerel bilgisayar ağlarında en çok kullanılan topolojiler taşıt (bus) ve halka (ring) dir. Ayrıca yıldız (star) ve hiyerarşik (sıradüzensel) yapılar da kullanılmaktadır. Bildiğimiz gibi ağ üzerindeki düğümler birer iş istasyonu ya da bir uç olarak da nitelenebilir.[3]

Taşıt yapısında, bir omurga üzerinde düğümler (iş istasyonları) yer alır ve bu yapı (topoloji) yaygın kullanılır. Bu omurgayı şekilde görüldüğü gibi bir doğrusal hat olarak tanımlayabiliriz. Ethernet yerel bilgisayar ağı buna bir örnektir. Halka topolojisinde düğümler bir halka oluşturacak biçimde birbirlerine bağlanırlar. Taşıt ve halka topolojilerinin yaygın kullanılmasının nedenlerinden biri de farklı yayın tiplerinin kullanılabilmesidir. Bu yapıdaki ağlarda çoklu yayın (multıcast) ve genel yayın (broadcast) uygulamaları yapılabilir.

3.1 YBA (LAN) PROTOKOLLERİ

Yerel bilgisayar ağlarında nitelikleri ve kullanılış biçimleri farklı aygıtların birbiri ile iletişimde bulunması, gönderilen iletilerin ortak bir anlayışa göre düzenlenmesini ve bazı dönüştürme işlemlerini zorunlu kılar. Bu zorunluluktan ötürü iletişim protokolleri ortaya çıkmıştır.

Uluslararası standartlar organizasyonu (ISO), bu konuda bir standart yayınlamıştır. Bu standarda göre yedi katmandan oluşan bir iletişim protokolu önerilmiştir. En üst katman, aygıta özgü sorunları ele alır. Bu katman kul lanıcının yüzyüze geldiği katmandır. En alt katmanda ise bilgilerin ortak iletişim kanalından iletimi sağlanır. Aradaki katmanlar ise bu katmanlar arasında, işlevlerin sırayla yerine getirilmesini sağlar. Bu arada belirtmek gerekir ki, yerel bilgisayar ağlarının bir çoğunda adı geçen yedi katman tüm işlevleri ile gerçekleştirilmeden (daha az katman ile) temel işlevler sağlanır.

Protokol, iletişim ortamının bilgisayarlarca kullanımını ayrıntılı olarak tanımlayan kurallardır. Bir bilgisayar ağında iletişim ortamını herhangi bir anda hangi düğüm ya da uç tarafından kullanılacağını düzenlemek gerekir. Bu protokollerin başında çarpışmalı protokoller ve andaçlı (token) protokoller gelir. Çarpışmalı protokoller daha çok taşıt yapısındaki ağlarda kullanılır. Bu protokolde düğümler birbirinden bağımsız olarak ortamı kullanmak isteyebilir ve iletime geçebilirler. Aynı anda birden çok düğüm iletime geçebileceğinden çarpışma kaçınılmazdır. Bu nedenle çarpışan iletilerin algılanarak yeniden gönderilmesi gerekir.

Andaçlı protokolde iletişim ortamını kullanma hakkı tüm düğümlere sırayla verilir. Düğümlerin ortamı kullanmaları token (andaç) adı verilen özel bir bit kalıbı ile denetlenir. Andacı o anda elinde bulunduran düğüm iletişim ortamını o anda kullanmaya yetkilidir.

Bunun dışında yine değişik ağ topolojileri için farklı iletişim protokolleri de vardır.

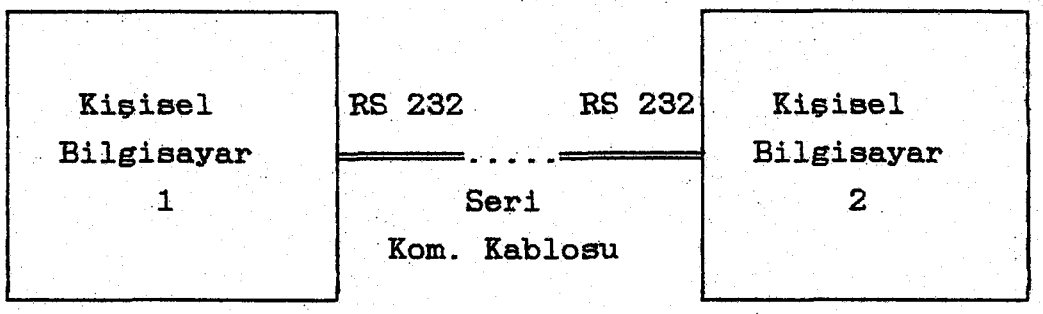
Yerel bir bilgisayar ağının tasarımında önemli olan seçilen topoloji ve iletişim protokülüdür. Birçok topoloji ve farklı iletişim protokollerine dayanarak çeşitli tasarımlarda yerel bilgisayar ağı oluşturmak olasıdır. Bunların başında çarpışmalı protokol (CSMA/CD Carrier Sense Multiple Access with Collision Detection / Çarpışma yakalamalı hat algılama ve çoklu erişim) yöntemi ile çalışan taşıt yapılı yerel bilgisayar ağı ve andaçlı halka (Token Ring) yerel bilgisayar ağı gelir.

4. BASIT BIR AĞ YAZILIMI

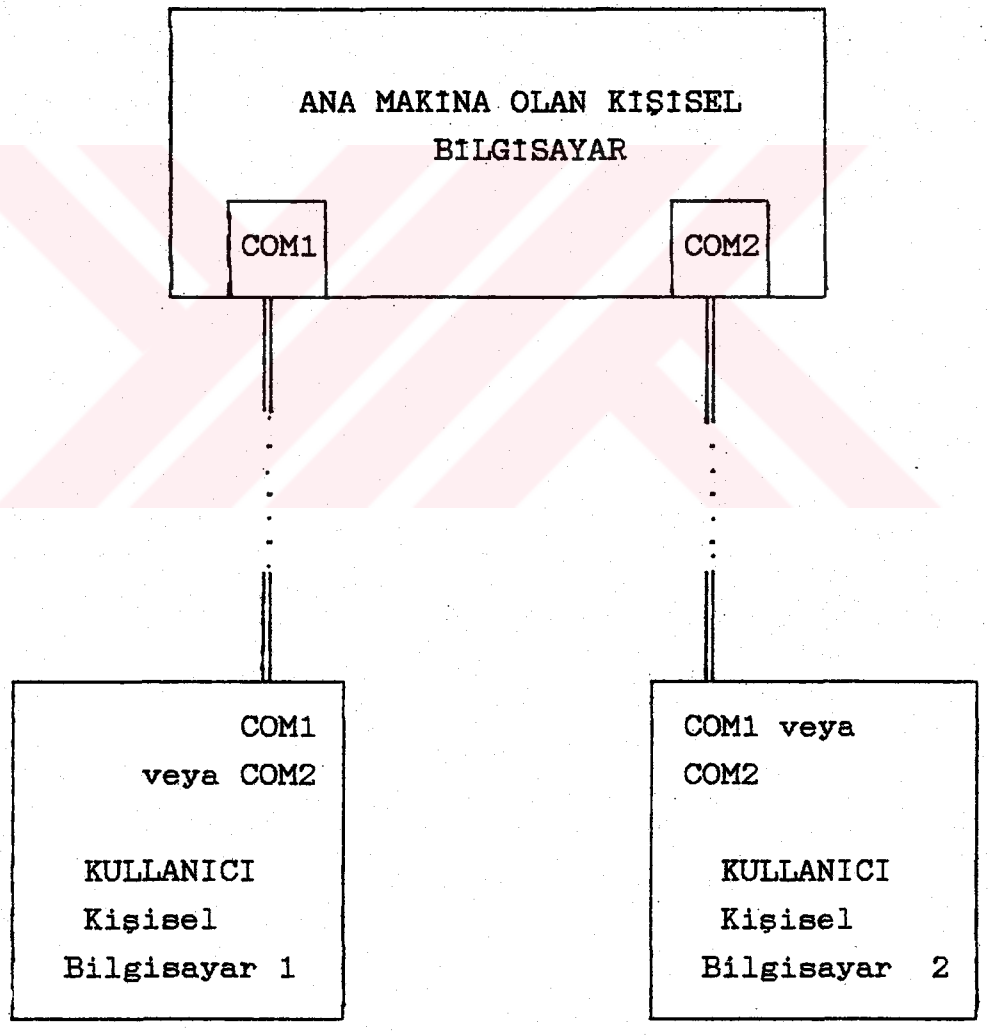
Bu çalışmada, birden fazla kişisel bilgisayar ile ek bir donanıma (Network kartı vb) gereksinim olmadan, basit anlamda bir yerel bilgisayar ağı (LAN) gibi çalışan bir ağ gerçekleştirilmiştir. Her ne kadar günümüzdeki yerel bilgisayar ağları yazılımlarının verdiği kadar olanağa sahip değilse de bu çalışma geliştirilerek çok küçük işletmelerde kullanılabilir.

4.1. SISTEMIN TEKNIK TANITIMI

Bu uygulamada birden fazla kişisel bilgisayar kullanılmıştır. Bu kişisel bilgisayarın her ikisinde de seri komünikasyon portu şartı sağlanmıştır. Günümüzde piyasada pazarlanan her kişisel bilgisayarda standart olarak az bir tane seri port (RS 232) vardır. Zaten bu çalışmada üstünde durulan bir konuda, ek bir donanım yatırımına gereksinim duyulmamasıdır. Uygulamada kullanılacak tek donanım birkaç metre uzunluğunda bir komünikasyon kablosudur. Bu kablonun boyu seri portların sinyal gönderme güçleriyle sınırlıdır. Bu mesafe hiç bir donanıma gereksinim duyulmadan 20 metre civarındadır. Ancak modemler vasıtasıyla telefon hatları kullanılarak mesafe problemi çözümlenebilir. Şekil-1a da uygulama donanım olarak ifade edilmiştir.



Şekil-1a



Şekil-1b

Bu uygulama bir yazılım ile üç kişisel bilgisayarı birbirine bağlayarak ortaklaşa aynı veri tabanını veya veri tabanlarını kullanmayı gerçekleştirmektedir. Yazılım paketindeki programların çoğunluğu Borland şirketinin Turbo Pascal diliyle yazılmış ve derlenmiştir. Yine veri tabanı kullanımı aynı şirketin TACCESS adı verilen bir paketiyle gerçekleştirilmiştir.

Şekil-1b de görüldüğü gibi eğer ana makina olarak kullanılan kişisel bilgisayarda iki tane RS 232 kominikasyon portu varsa bu portlara iki kullanıcı bağlanabilir. Şekil-1b de bu seri portlar COM1 ve COM2 olarak gösterilmiştir.

4.2. UYGULAMANIN ÇALIŞMA ŞEKLİ

Kişisel bilgisayarlardan birisi ana bilgisayar diğeri kullanıcı bilgisayar olarak tanımlanabilir. Ana bilgisayar denilen bilgisayarın bir sabit diski olmalıdır (kullanılan veri tabanı bu diskte yer alacaktır). Kullanıcı makinede sadece veri tabanına ulaşılacak programların bulunduğu bir disk veya disket sürücü olmalıdır.

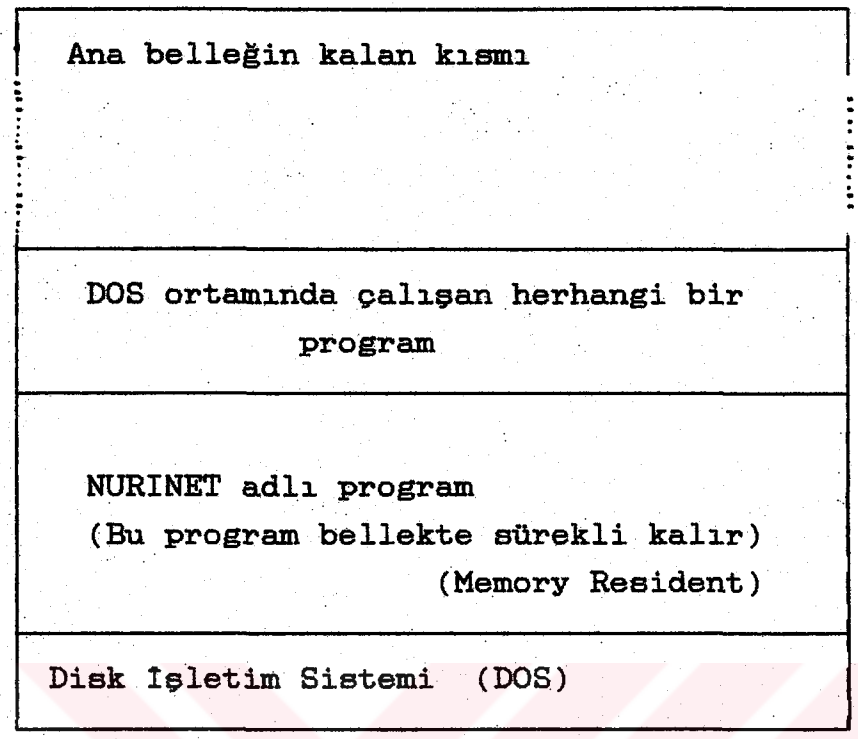
Ana bilgisayar açıldığında öncelikle makinenin seri port tanımlamaları yapılır. Bu tanımlamalar seri haberleşmenin bazı parametrelerini (haberleşme hızı vb) bildirmek anlamına gelir. Bu tanımlamaların aynılarının kullanıcı bilgisayarda da yapılması gerekmektedir. Yoksa birbirlerinin gönderdiği sinyalleri çözmeleri mümkün olamaz. Bu tanımlamalar Disk Işletim Sistemi (DOS / Disk Operating System) nde bulunan MODE adlı programla yapılabileceği gibi kullanıcı programıyla da yapılabilir.

Ana bilgisayarda NURINET adlı program çalıştırılır. Bu program ana bilgisayarın ana belleğinde bilgisayar kapanana kadar kalacak bir programdır (Memory Resident Program). Yanıt adlı program ana bilgisayara seri portdan bir bilgi geldiğinde çalışarak gelen bilgiyi değerlendirip gerekli işlemleri yaparak cevap yollar. Bu mesajı alma ve cevap yollama işi SRUN adlı bir program vasıtası ile yapılır. Ana bilgisayarda veri tabanına erişimi SACC adı verilen program yapar. SACC adlı program bu işi yaparken daha önce adı geçen TACCESS adlı programı kullanır.

Kullanıcı bilgisayarda da seri port tanımlamaları yapıldıktan sonra herhangi bir veri tabanı uygulaması çalıştırıldığında ana bilgisayardaki bilgileri aynı anda kullanmaya başlar. Kullanıcı bilgisayardaki veri tabanı uygulama programında veri tabanına erişim için UACC adlı bir program kullanılır. Bu program daha önce adı geçen SRUN adlı program vasıtasıyla ana bilgisayara isteğini iletir ve yine aynı SRUN adlı program ile isteğine karşı gelen mesajı alır ve değerlendirir. Böylece ana bilgisayardaki bir veri tabanını başka bir bilgisayar da kullanmış olur.

4.3. SİSTEMDEKİ PROGRAMLARIN ÇALIŞMA MANTIĞI

Uygulamanın çalışma mantığını anlayabilmek için ana bilgisayarın ve kullanıcı bilgisayarın ana bellek haritalarına bakmak faydalı olur. Şekil-2 de ana bilgisayarın ana bellek haritası verilmiştir.



Şekil-2

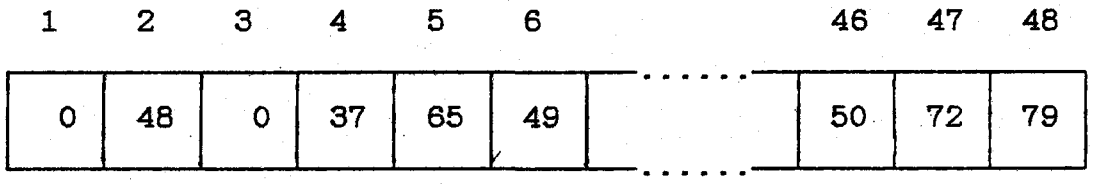
NURINET adlı program seri porta herhangi bir sinyal geldiğinde çalışmaya başlar. Bu programın içinde mesajları alan ve yollayan program olan SRUN da kullanılır. Bu programın çalışma adımları şöyle belirlenebilir;

1 - Seri portdan gelen bir sinyal ile NURINET adlı program çalışmaya başlar.
2 - SRUN programının yardımı ile gelen mesajın tümü bir tampon alana alınır.
3 - Gelen mesaj değerlendirilir. Yani gelen mesaj bir kayıt mı istiyor yoksa bir anahtar alan kontrolü mü yapmak istiyor bu belirlenir. Bu belirleme işi mesajın 3. ve 4. byte ları alarak yapılır.
4 - İsteğe göre SACC programının içinden gerekli fonksiyon çağrılır.
5 - Cevap yine tampon bir mesaj alanına alınarak SRUN programı tarafından kullanıcıya geri yollanır.

Kullanıcı tarafından bir veri tabanı ile ilgili bir uygulama çalıştırıldığında önce UACC adlı program isteği bir mesaj şekline getirmektedir. Bu oluşturulan mesajı yine SRUN adlı program ana bilgisayara yollar. Ana bilgisayarda işlemlenen mesajın cevabı yine bir mesaj olarak SRUN tarafından tampon alana alınır. UACC programı bu mesajı yine program içinde kullanılabilir hale getirir. Böylece ana bilgisayardaki bir bilgi kullanılır.

4.4. BILGI GONDERME VE ALMA İŞLEMI

Bilgi gönderme : Yollanacak bilgi bir tampon alana alınarak bilgi yollama olayı başlatılmış olur. Tampon alana alınan bilgi byte byte diğer makineye yollanır. Tampon alanın büyüklüğü örnek çalışmada 2048 byte olarak alınmıştır. Fakat gerek duyulursa ve eğer kullanılan bilgisayarların hafızası yeterli olursa bu rakamı artırmak mümkündür. Tampon alana alınmış olan bilgiyi şekil-3 deki gibi ifade edebiliriz. Eğer bilginin yolladığına dair doğrulama sinyali belirli bir süre içinde alınamazsa bağlantı hattında bir arıza olduğu gerekçesiyle program kesilir.



Şekil-3

Şekil-3 de görüleceği gibi yollanacak bilginin uzunluğu 48 dir. Tampon bölge bir dizi şeklindedir ve yollanacak bilginin her bir byte ı bu dizinin bir elemanıdır. Bu dizi, programların yazıldığı dil

olan Turbo Pascal programlama dilindeki karakter(char) değişken tipindedir. Bu dizinin ilk iki elemanı olan karakterler ile yollanacak bilginin tüm uzunluğu temsil edilir. Yani uzunluğu ifade etmekte kullanılan karakterlerin uzunluğu da dahildir. Böylece bilgi alıcı taraf olan bilgisayar kaç tane bilgi sinyali alacağını bilebilir.

Kullanıcı bilgisayar tarafından yollanan mesajın yapısı aşağıdaki gibidir.

Başlangıç ve
bitiş byte ları

[1]..[2] : Mesajın uzunluğu
[3]..[4] : Mesaj tanıtım kodu
[5]..[2048] : Mesajın içeriği

Ana bilgisayar tarafından yollanan mesaj yapısı;

Başlangıç ve
bitiş byte ları

[1]..[2] : Mesajın uzunluğu
[3]..[2048] : Mesajın içeriği

Tampon alan olarak kullanılan dizinin elemanları olarak görülen sayılar, gönderilecek bilgilerin karakter olarak ifadesidir. Bu ifade PC lerde DOS adı verilen işletim sisteminde kullanılan ASCII (American Standard Code for Information Interchange) kod sistemidir. Yani örneğin 65 sayısının karşılığı olan karakter `A` dır. Tampon alandaki karakterlerin hepsi, yollanırken sayı olarak yollanır ve alıcı bilgisayarda yine aynı şekilde ASCII karakter tablosu kullanılarak karakter şekline çevrilir.

Yollanacak olan bilginin uzunluğu saptandıktan sonra bu bilgi iki karakter ile ifade edilir. Bu ifade için 256 lık sayı sistemi kullanılır. Bu sayı sistemini kullanarak iki karakterde maksimum 65535 sayısını ifade etmek mümkündür. Yani ifade edebileceğimiz maksimum sayı

$$255 \times 256^{1} + 255 \times 256^{0} = 65535$$

şeklinde gösterilir.

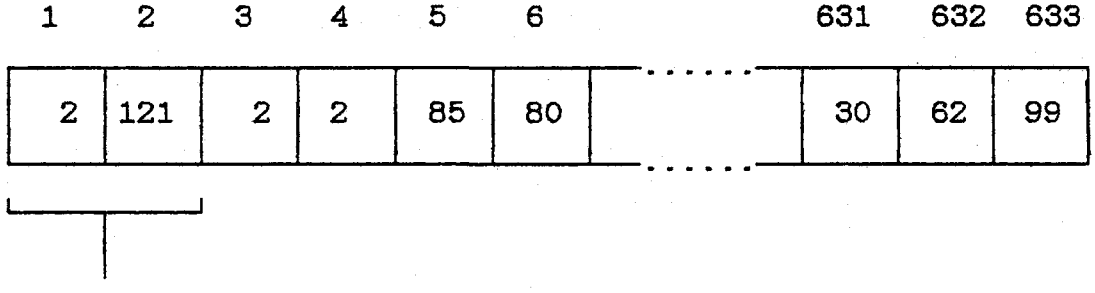
Örneğin 5620 sayısını bu sayı sisteminde ifadesi;

$$21 \times 256^{1} + 244 \times 256^{0} = 5620$$

şeklinde olur.

Eğer 5620 uzunluğundaki bu bilgi alanı yollanmak istenirse, tampon alanın ilk iki bölümündeki karakterler 21 ve 244 sayılarına karşılık gelen karakterler olmalıdır.

Toplam uzunluğu 633 olan bir tampon alanın sembolik görünüşü aşağıdaki gibidir.

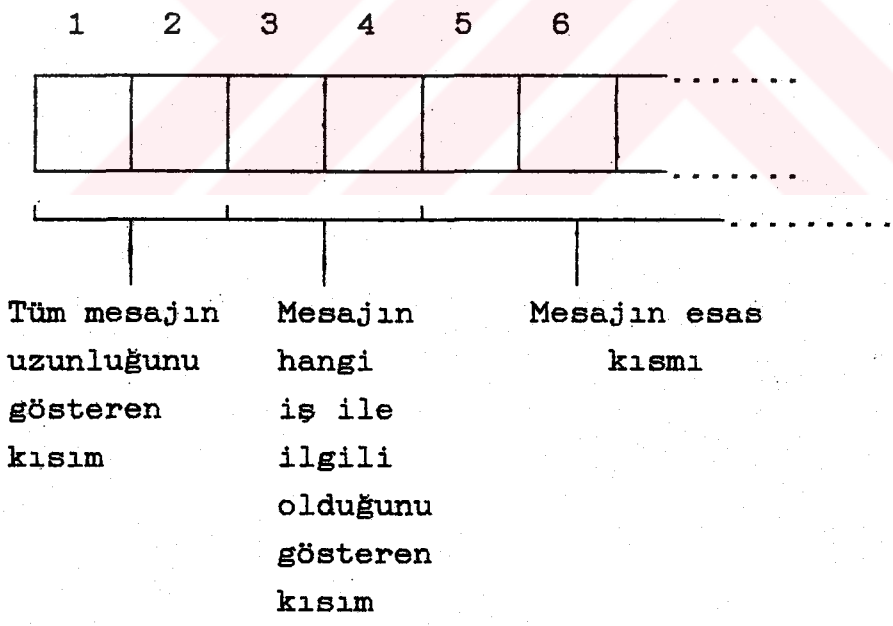


$2 \times 256 + 121 = 633$

Şekil-3a

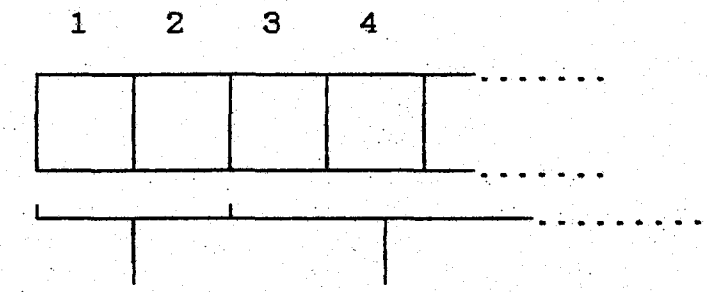
Yollanacak bilgi tampon alana, tampon alanın 3. karakterinden itibaren yerleştirilir. `ADJLEN` adlı bir altprogram vasıtasıyla bu uzunluğa iki ilave edilir (bu, uzunluğun ifade edildiği 2 karakter için) ve elde edilen yeni uzunluk 256 lık sayı düzenine göre belirlenerek tampon alanın 1. ve 2. elemanlarına yerleştirilir.

Gönderilecek bilgi alanının oluşturulması yani tampon alanın oluşturulması, istek yapan bilgisayar ile cevap veren bilgisayarda değişiktir. Her iki taraf (soru soran taraf ile cevap veren taraf) için oluşturulan tampon alanın sembolik gösterimi şekil-3 deki gibidir.



Soru soran tarafda oluşturulan tampon alan deseni

Şekil-3b



Cevap veren tarafda oluşturulan tampon alan deseni

Şekil-4

Soru soran taraf ile cevap veren tarafın yolladığı mesajlardaki bu yapısal değişiklik, soru sorup bekleyen tarafın zaten ne için cevap beklediğini bilmesinden dolayıdır. Bu yüzden cevap mesajına, bu mesajın ne ile ilgili olduğunu belirtir bir ibare koymaya gerek yoktur.

Mesaj alma : Alınan her byte bir tampon alana yazılarak bir kuyruk oluşturulur. Alınan her byte dan sonra gönderici bilgisayara alınmıştır sinyali yollanır. Böylece bilgi kaybının yüzde yüz engellenmesi mümkün kılınmıştır. Bu mesaj alma ve gönderme kuralları hem ana bilgisayar ve hem de kullanıcı bilgisayar için aynen geçerlidir. Mesaj alan bilgisayar mesajın ilk iki karakterinden kaç bilgi sinyali alacağını bilir ve o kadar uzunlukda mesaj aldığında bekleme işini keser.

4.4.1. MESAJ TANITIM KODLARI

Mesaj tanıtım kodunun ilk byte ı bu mesajın ne ile ilgili olduğunu belirtir. İkinci byte ise ne iş yapılacağını belirtir.

Mesaj tanıtım kodunun ilk byte ı eğer 1 ise bu mesaj bir dosya işlemi ile ilgilidir.

Mesaj tanıtım kodunun ilk byte ı eğer 2 ise bu mesaj bir anahtar alan işlemi ile ilgilidir.

Mesaj tanıtım kodunun ilk byte ı eğer 3 ise bu mesaj bir kayıt işlemi ile ilgilidir.

Mesaj tanıtım kodunun ilk byte ı eğer 4 ise bu mesaj bir haber niteliğinde mesaj işlemi ile ilgilidir.

[1]	[2]	işlem tipi	Açıklaması
1	1	Dosya İşlemi	Veri Dosyasını açma işlemi
1	2	Dosya İşlemi	İndeksli Dosyayı açma işlemi
1	3	Dosya İşlemi	Veri Dosyasını kapatma işlemi
1	4	Dosya İşlemi	İndeksli Dosyayı kapatma işlemi
2	1	İndeks işlemi	İndeks kontrol
2	2	İndeks işlemi	İndeks konumlandırma
2	3	İndeks işlemi	İndeksi bir önceki indekse konumlandırma
2	4	İndeks işlemi	İndeksi bir sonraki indekse konumlandırma
2	5	İndeks işlemi	İndeksi indeks dosyasına ekleme
2	6	İndeks işlemi	İndeksi indeks dosyasından silme
3	1	Kayıt İşlemi	Veri dosyasından kayıt okuma
3	2	Kayıt İşlemi	Veri dosyasına kayıt ekleme
3	3	Kayıt İşlemi	Veri dosyasında var olan kaydın üzerine tekrar yazma işlemi
4	1	Mesaj İşlemi	Ana makinaya herhangi bir haber gönderme

4.4.2. MESAJ YAPILARI

Bütün işlem kodları SRUN adı ile bahsedilen mesaj gönderme ve mesaj alma ünitesi olan program parçacığında aşağıdaki isimlerle tarif edilmiştir. Tüm programlarda bu isimler aynen kullanılmıştır.

Dosya işlemleri:

1 1	(OPENDFILEID)	Veri dosyası açma işlemi
1 2	(OPENIFILEID)	İndeksli dosyayı açma işlemi
1 3	(CLOSEDFILEID)	Veri dosyası kapama işlemi
1 4	(CLOSEIFILEID)	İndeksli dosyayı kapama işlemi

İndeks işlemleri:

2 1	(FINDKEYID)	İndeksli dosyada verilen indeksin olup olmadığını bulma işlemi
2 2	(SEARCHKEYID)	İndeksli dosyada verilen indekse eşit veya büyük olan indekse konumlanma
2 3	(PREVKEYID)	İndeksli dosyada daha önceden konumlanılan indeksden bir önceki indekse konumlanma
2 4	(NEXTKEYID)	İndeksli dosyada daha önceden konumlanılan indeksden bir sonraki indekse konumlanma
2 5	(ADDKEYID)	İndeksli dosyaya yeni bir indeks eklenmesi
2 6	(DELKEYID)	İndeksli dosyadan bir indeks silinmesi

Kayıt işlemleri:

3 1	(GETRECID)	Veri dosyasından bir kayıt okuma
3 2	(ADDRECID)	Veri dosyasına bir kayıt ekleme
3 3	(PUTRECID)	Veri dosyasında var olan bir kaydın üzerine yeniden yazma
3 4	(DELRECID)	Veri dosyasında var olan bir kaydı silme

Mesaj işlemleri:

4 1	(MESSAGEID)	Kullanıcı makinadan ana makinaya haber niteliğinde bir mesaj gönderme

4.4.3. MESAJ YAPILARININ DETAYLARI

Ana makinedeki uygulama programında kullanılan dosya işlemleri zaten TACCESS adı verilen program parçası ile yapılmaktadır. Bu yüzden kullanıcıların bu işlemleri nasıl gerçekleştirdiği yani kullanıcı tarafındaki programların bu işlemleri nasıl ifade ettikleri detaylı şekilde anlatılmıştır.

Her işlem kodu için hem kullanıcı ve hem de ana makine tarafı olmak üzere iki türlü mesaj tipi vardır. Kullanıcı tarafındaki mesaj alanı UACC adı verilen ünite tarafından ana makinedeki yollanacak mesaj alanı ise NURINET adlı program tarafından hazırlanmaktadır.

Tüm mesajlar UACC tarafından hazırlanır SRUN tarafından bu mesaj yollanır. NURINET tarafından alınan mesaj işleme konulur cevap olarak mesaj hazırlanır ve yine SRUN

tarafından kullanıcıya yollanır.

Aşağıda oldukça sık olarak kullanılacak bazı kelimeler ve tanımlamalar için bazı kısaltmalar verilmiştir.

RL	:	(Record Length) Kayıt uzunluğu
KL	:	(Key Length) İndeks alanı uzunluğu
DFS	:	(Data File Structure) Veri dosyası yapısı
IFS	:	(Index File Structure) İndeksli dosya yapısı
DFName	:	(Data File Name) Veri Dosyası adı
IFName	:	(Index File Name) İndeksli Dosya adı
ML	:	(Message Length) Mesaj uzunluğu
MI	:	(Message Identification code) Mesaj Tanıtım kodu
DP	:	(Duplicate Key) Bir indeksli dosyada birden fazla aynı indeksden olup olmadığını belirten değişken
RRN	:	(Relative Record Number) Veri dosyasında kayıt no
PKEY	:	(Processing Key) İşlem yapılan indeks alanı
REC	:	(Record) Elde edilecek veya eldeki kayıt
GRM	:	(GetRec Mode) Kayıt okuma modu. Eğer bu 0 ise kayıt kilitlenmeden okunur. Eğer 1 ise okunan kayıt kilitlenir.

4.4.3.1. DOSYA İŞLEMLERİ İLE İLGİLİ MESAJ YAPILARI

Dört farklı dosya işlemi vardır. Bunlarda ikisi veri dosyalarıyla diğer ikisi de indeksli dosyalarla ilgili olup dosyaları açma ve kapama işlemlerini gerçekleştirirler.

4.4.3.1.a. Veri dosyası açma işlemi (OPENDFILEID)

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UOpenDataFile(DFS,DFName,RL) şeklindedir.

Burada,

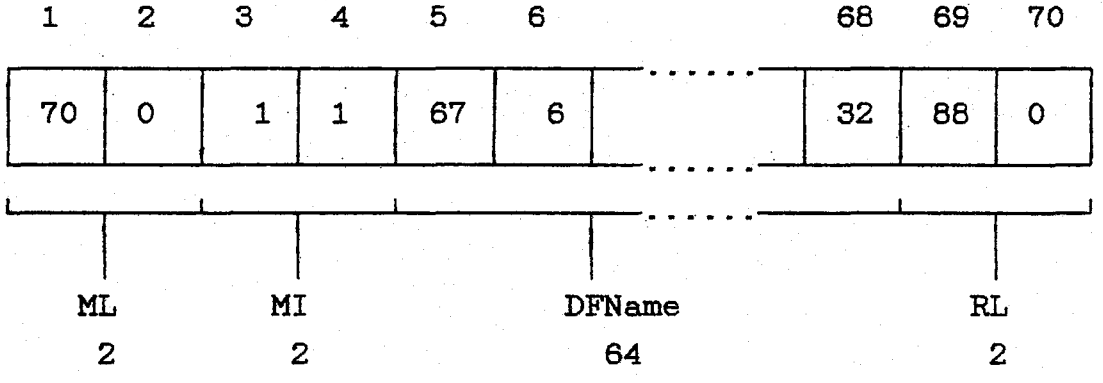
DFS : Veri dosyasının TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 146 byte dır.

DFName : Veri dosyasının ana makinedeki ismi içeren kısım olup uzunluğu maksimum 64 byte dır.

RL : Veri dosyasındaki herbir kaydın uzunluğudur. RL nin uzunluğu ise 2 byte dır.

UACC tarafından DFName ve RL yollanır karşılığında ise veri dosyasının tüm özelliklerini içeren DFS elde edilir.

UACC tarafından yollanan istek mesajın deseni;



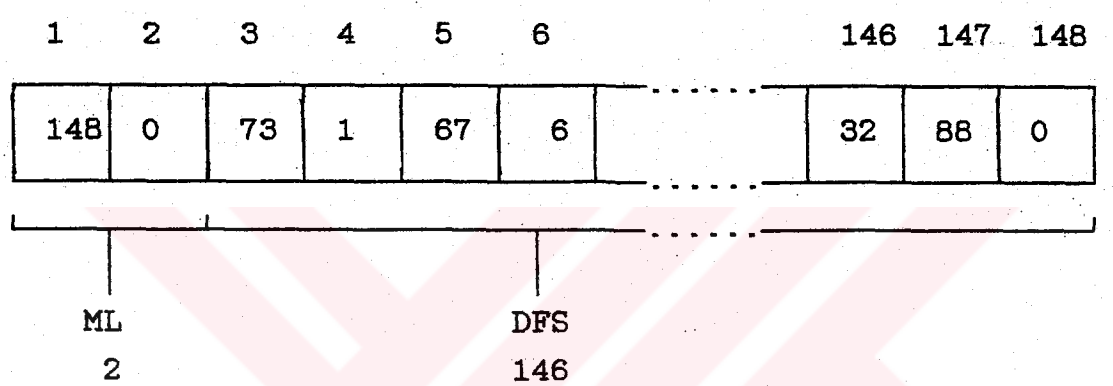
Toplam mesaj uzunluğu = 2 + 2 + 64 + 2 = 70

Şekil-5

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve DFS byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır. SACC de sadece TACCESS de kullanılan OpenFile komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 146 = 148

Şekil-6

4.4.3.1.b. İndeksli dosya açma işlemi (OPENIFILEID)

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UOpenIndexFile(IFS,IFName,KL,DP) şeklindedir.

Burada,

IFS : Veri dosyasının TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

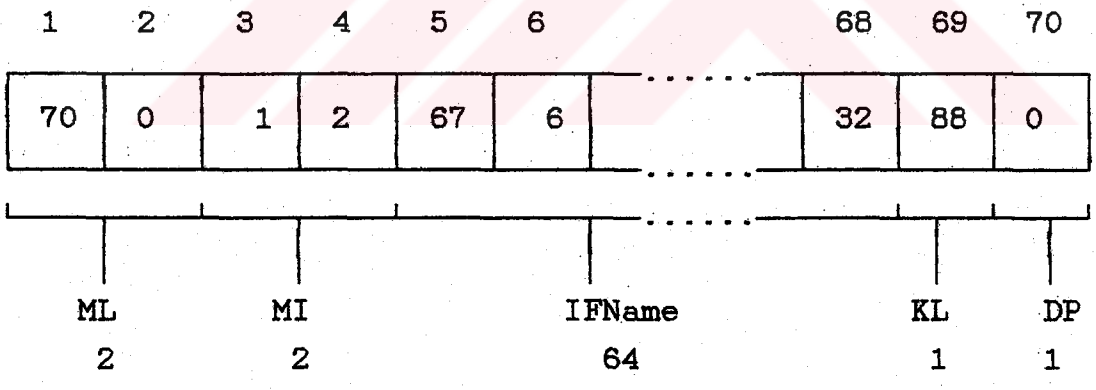
IFName : İndeksli dosyanın ana makinedeki ismi içeren kısım olup uzunluğu maksimum 64 byte dır.

KL : İndeksli dosyadaki indeks alanının uzunluğudur. KL nin uzunluğu ise 1 byte dır.

DP : İndeksli dosyada aynı indeks den birden fazla olup olamayacağını belirten göstergedir. Eğer 1 ise olabilir 0 ise olamaz anlamındadır. Bu ifade TACCESS de de aynene kullanılmaktadır.

UACC tarafından IFName, KL ve DP yollanır karşılığında ise indeksli dosyanın tüm özelliklerini içeren IFS elde edilir.

UACC tarafından yollanan istek mesajın deseni;



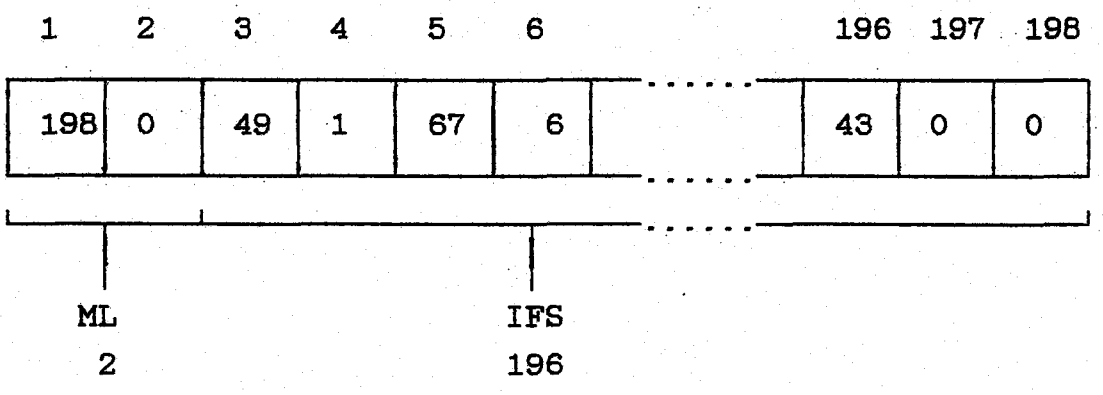
Toplam mesaj uzunluğu = 2 + 2 + 64 + 1 + 1 = 70

Şekil-7

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır. SACC de sadece TACCESS de kullanılan OpenIndex komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 196 = 198

Şekil-8

4.4.3.1.c. Veri dosyası kapama işlemi (CLOSEDFILEID)

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

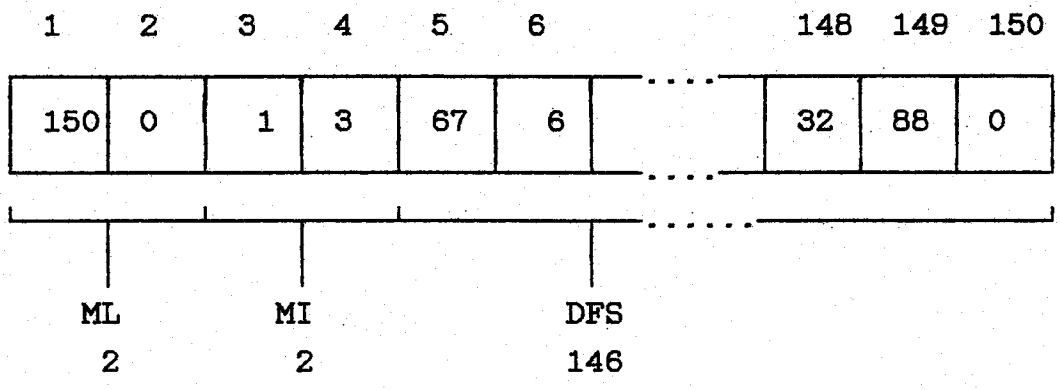
UCloseDataFile(DFS) şeklindedir.

Burada,

DFS : Veri dosyasının TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 146 byte dır.

UACC tarafından DFS yollanır karşılığında ise yine veri dosyasının tüm özelliklerini içeren DFS elde edilir.

UACC tarafından yollanan istek mesajın deseni;



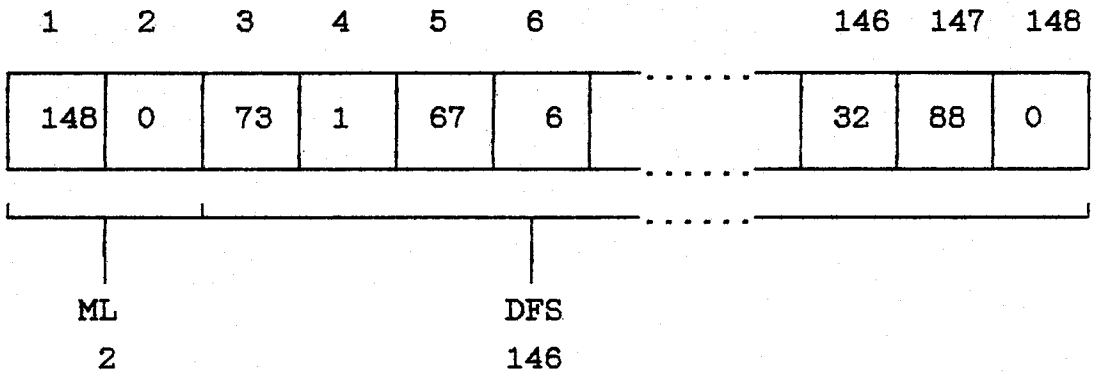
Toplam mesaj uzunluğu = 2 + 2 + 146 = 150

Şekil-9

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve DFS byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır. SACC de sadece TACCESS de kullanılan CloseFile komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 146 = 148

Şekil-10

4.4.3.1.d. İndeksli dosya kapama işlemi (CLOSEIFILEID)

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

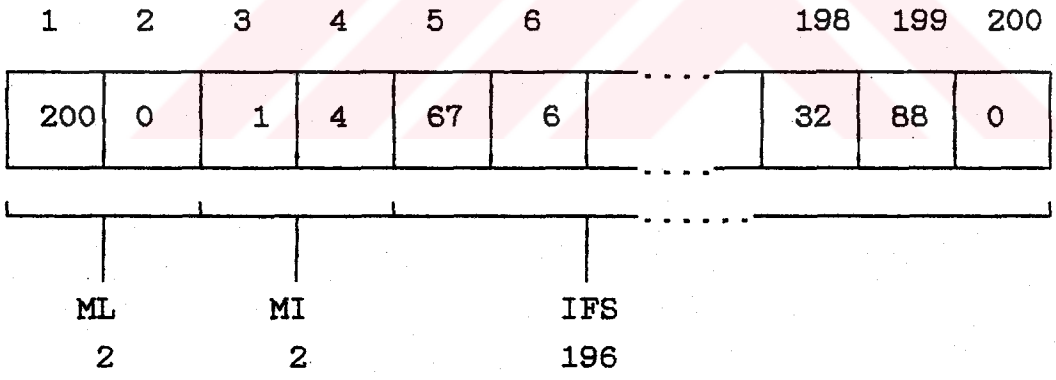
UCloseIndexFile(IFS) şeklindedir.

Burada,

IFS : İndeksli dosyanın TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

UACC tarafından IFS yollanır karşılığında ise yine indeksli dosyanın tüm özelliklerini içeren IFS elde edilir.

UACC tarafından yollanan istek mesajın deseni;



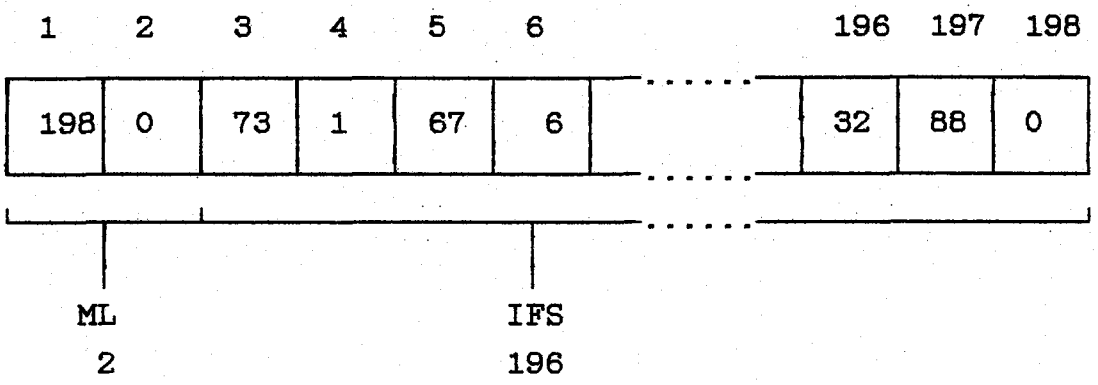
Toplam mesaj uzunluğu = 2 + 2 + 196 = 200

Şekil-11

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır. SACC de sadece TACCESS de kullanılan CloseIndex komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 196 = 198

Şekil-12

4.4.3.2. INDEKS IŞLEMLERI ILE ILGILI MESAJ YAPILARI

Altı farklı indeks işlemi vardır. Bunlar indeksli dosyayı kullanarak kayıtlara erişme, silme vb. işlemler için kullanılırlar.

4.4.3.2.a. İndeks kontrol işlemi (FINDKEYID)

Bu işlem ile verilen indeksin indeksli dosyada var olup olmadığı kontrol edilir. Eğer aranılan indeks yok ise kayıt numarası 0 olur.

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UFindKey(IFS,RRN,PKEY) şeklindedir.

Burada,

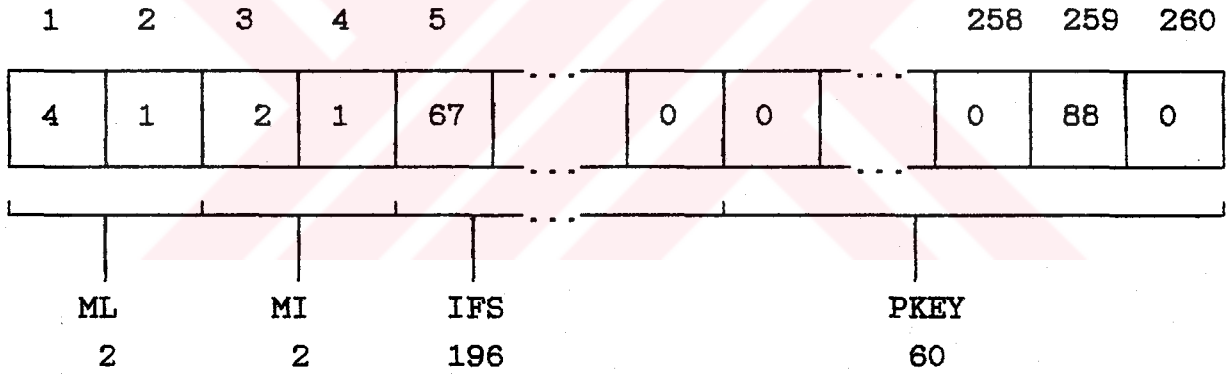
IFS : İndeksli dosyanın TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

RRN : Veri dosyasındaki kayıt numarasıdır. Eğer kayıt bulunursa bu alan kaydın numarasını alır. Bu kısmın uzunluğu 4 byte dır.

PKEY : Bu alana aranması istenilen indeks yüklenir. Bu kısmın uzunluğu 60 byte dır.

UACC tarafından IFS ve PKEY yollanır karşılığında ise veri dosyasının tüm özelliklerini içeren IFS ve RRN elde edilir.

UACC tarafından yollanan istek mesajın deseni;



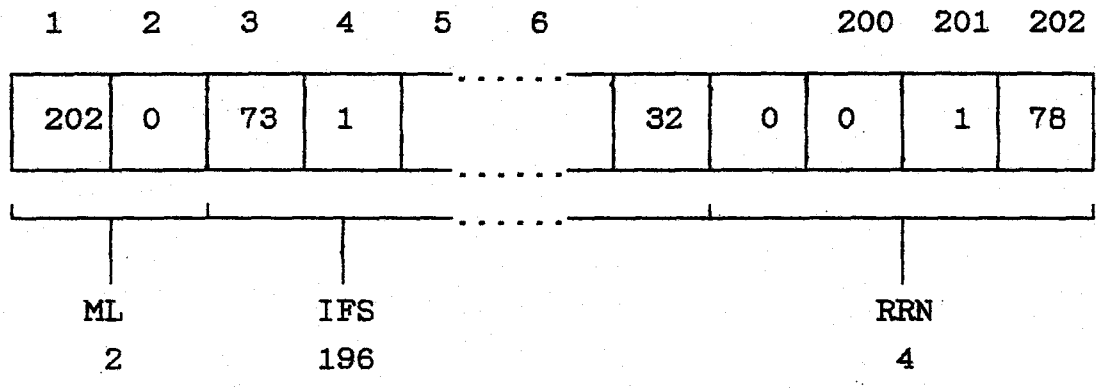
Toplam mesaj uzunluğu = 2 + 2 + 196 + 60 = 260

Şekil-13

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS ile RRN byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır.
SACC de sadece TACCESS de kullanılan FındKey komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 196 + 4 = 202

Şekil-14

4.4.3.2.b. İndeksi konumlandırma işlemi (SEARCHKEYID)

Bu işlem ile indeksli dosyada verilen indekse eşit veya büyük olan indekse konumlanılır. Eğer indeksli dosyanın sonuna gelinmiş ise kayıt numarası 0 olur.

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

USearchKey(IFS,RRN,PKEY) şeklindedir.

Burada,

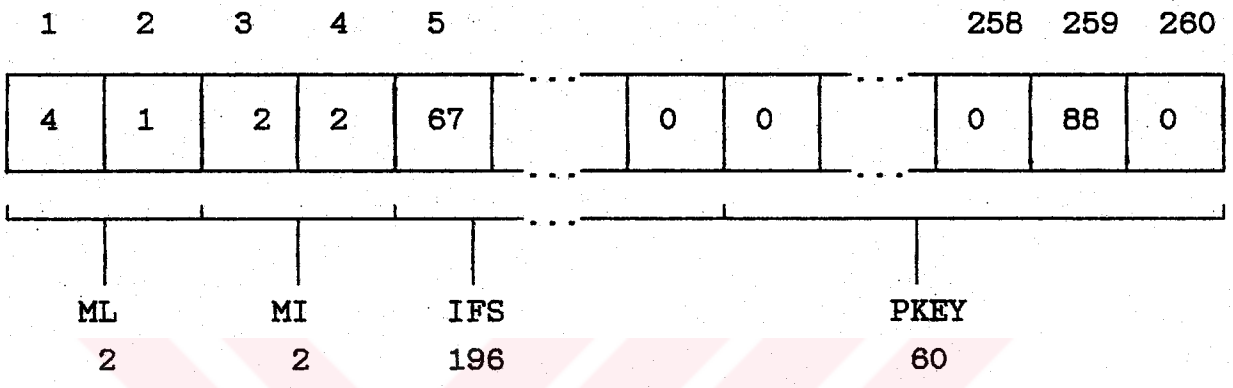
IFS : İndeksli dosyanın TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

RRN : Veri dosyasındaki kayıt numarasıdır. Eğer uygun kayıt bulunursa bu alan kaydın numarasını alır. Bu kısmın uzunluğu 4 byte dır.

PKEY : Bu alana aranması istenilen indeks yüklenir. Bu kısmın uzunluğu 60 byte dır.

UACC tarafından IFS ve PKEY yollanır karşılığında ise veri dosyasının tüm özelliklerini içeren IFS, RRN ve PKEY elde edilir.

UACC tarafından yollanan istek mesajın deseni;



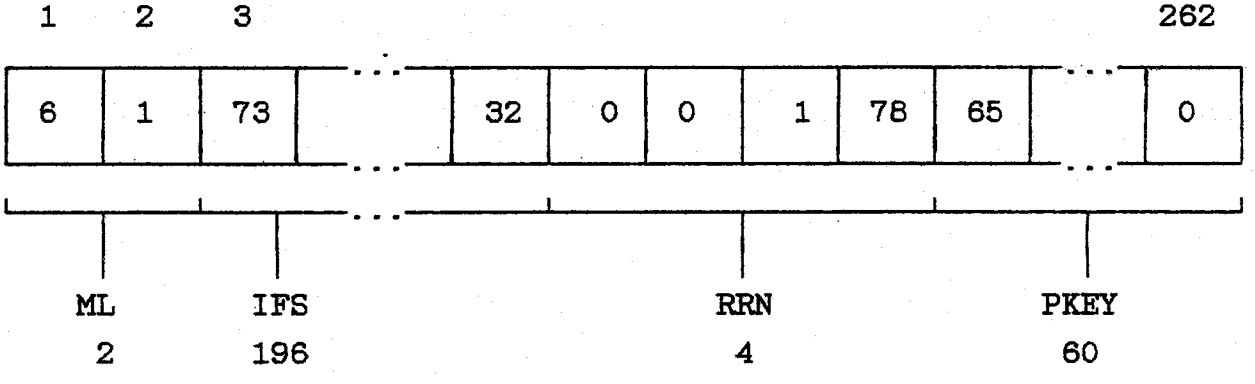
Toplam mesaj uzunluğu = 2 + 2 + 196 + 60 = 260

Şekil-15

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS,RRN,PKEY byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır.
SACC de sadece TACCESS de kullanılan SearchKey komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Şekil-16

Toplam mesaj uzunluğu = 2 + 196 + 4 + 60 = 262

4.4.3.2.c. İndeksi bir öncekine konumlandırma (PREVKEYID)

Bu işlem ile indeksli dosyada verilen indeksden küçük olan indekse konumlanılır. Eğer indeksli dosyanın başına gelinmiş ise kayıt numarası 0 olur.

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UPrevKey(IFS,RRN,PKEY) şeklindedir.

Burada,

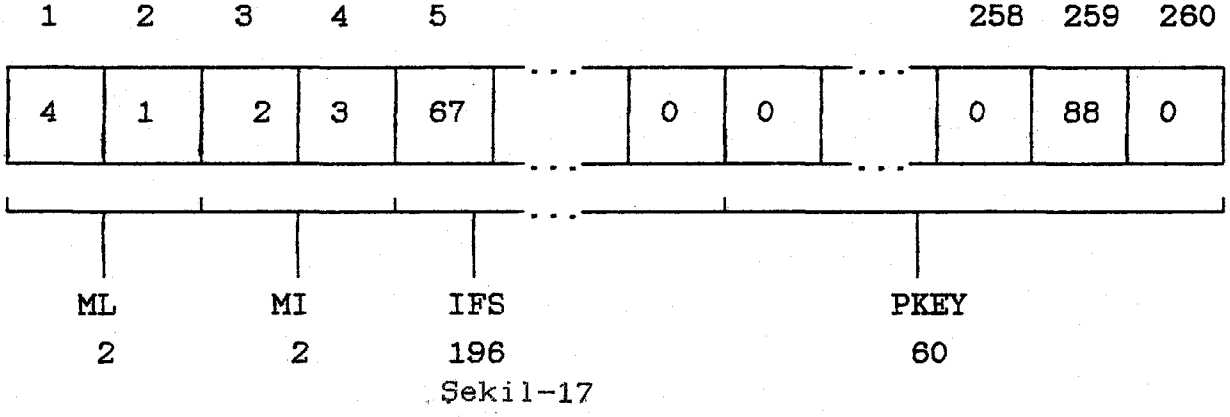
IFS : İndeksli dosyanın TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

RRN : Veri dosyasındaki kayıt numarasıdır. Eğer uygun kayıt bulunursa bu alan kaydın numarasını alır. Bu kısmın uzunluğu 4 byte dır.

PKEY : Bu alana bir önceki istenilen indeks yüklenir. Bu kısmın uzunluğu 60 byte dır.

UACC tarafından IFS ve PKEY yollanır karşılığında ise veri dosyasının tüm özelliklerini içeren IFS, RRN ve PKEY elde edilir.

UACC tarafından yollanan istek mesajın deseni;



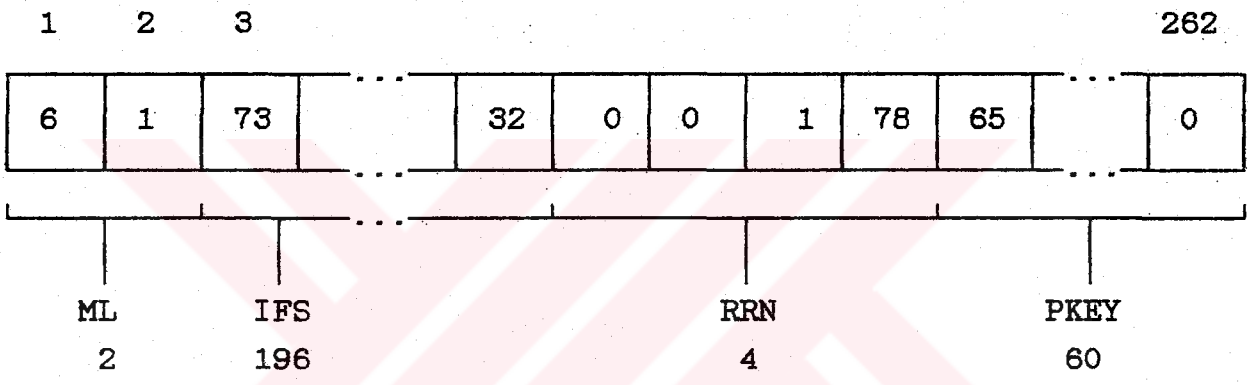
Şekil-17

Toplam mesaj uzunluğu = 2 + 2 + 196 + 60 = 260

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS,RRN,PKEY byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır.
SACC de sadece TACCESS de kullanılan PrevKey komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 196 + 4 + 60 = 262

Şekil-18

4.4.3.2.d. İndeksi bir sonrakine konumlandırma (NEXTKEYID)

Bu işlem ile indeksli dosyada verilen indeksden büyük olan indekse konumlanılır. Eğer indeksli dosyanın sonuna gelinmiş ise kayıt numarası 0 olur.

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UNextKey(IFS,RRN,PKEY) şeklindedir.

Burada,

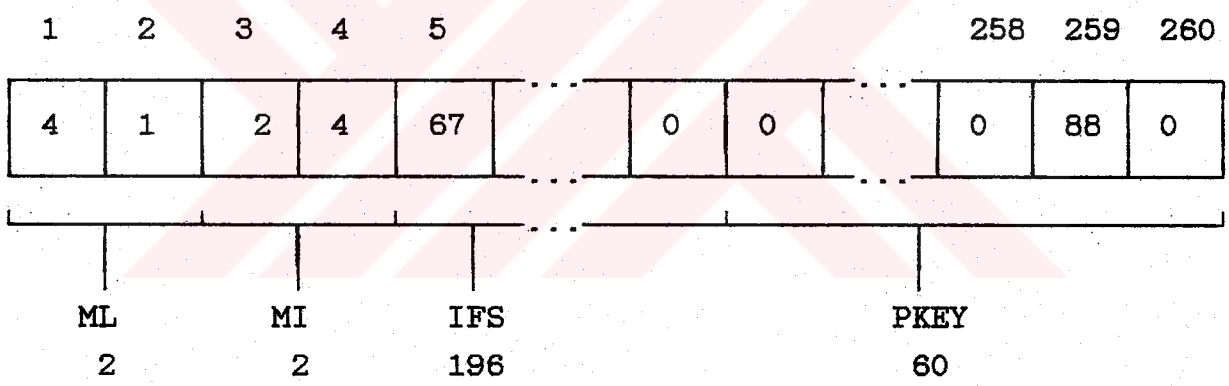
IFS : İndeksli dosyanın TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

RRN : Veri dosyasındaki kayıt numarasıdır. Eğer uygun kayıt bulunursa bu alan kaydın numarasını alır. Bu kısmın uzunluğu 4 byte dır.

PKEY : Bu alana bir sonraki istenilen indeks yüklenir. Bu kısmın uzunluğu 60 byte dır.

UACC tarafından IFS ve PKEY yollanır karşılığında ise veri dosyasının tüm özelliklerini içeren IFS, RRN ve PKEY elde edilir.

UACC tarafından yollanan istek mesajın deseni;



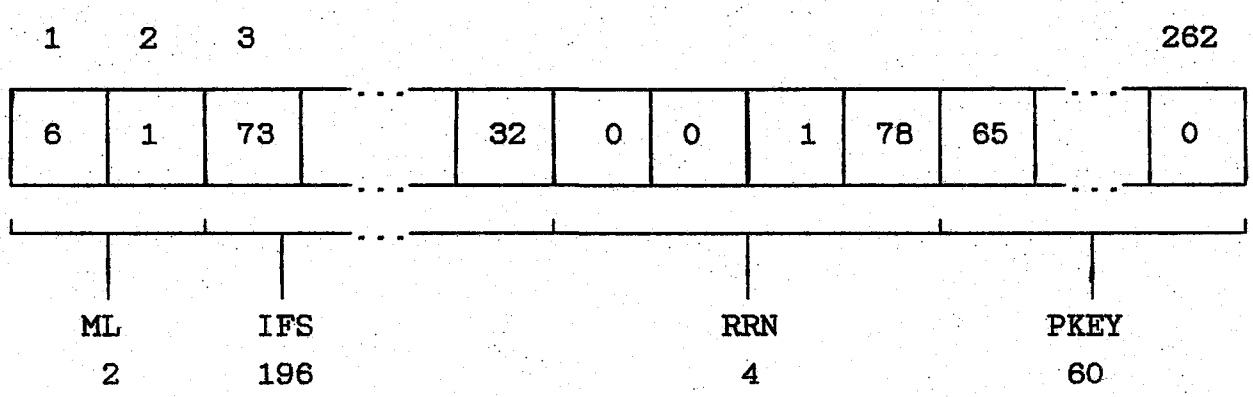
Toplam mesaj uzunluğu = 2 + 2 + 196 + 60 = 260

Şekil-19

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS,RRN,PKEY byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır.
SACC de sadece TACCESS de kullanılan NextKey komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 196 + 4 + 60 = 262

Şekil-20

4.4.3.2.e. İndeks ekleme işlemi (ADDKEYID)

Bu işlem ile verilen indeks indeksli dosyaya eklenir.

Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UAddKey(IFS,RRN,PKEY) şeklindedir.

Burada,

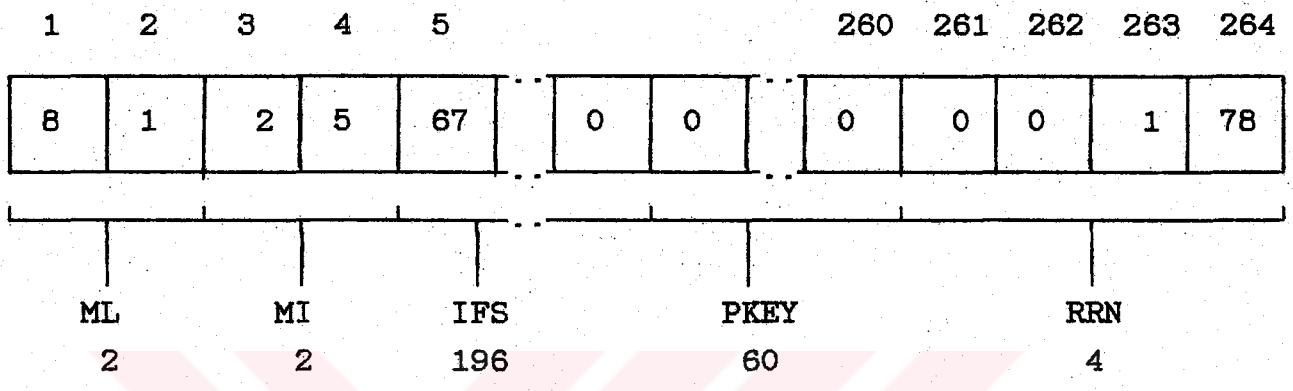
IFS : İndeksli dosyanın TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

RRN : Veri dosyasındaki ilgili kayıt numarasıdır. Eğer indeks indeksli dosyaya ilave edilemezse bu alan 0 olarak geri döner.

PKEY : Bu alana indeksli dosyaya ilavesi istenilen indeks yüklenir. Bu kısmın uzunluğu 60 byte dır.

UACC tarafından IFS, PKEY ve RRN yollanır karşılığında ise veri dosyasının tüm özelliklerini içeren IFS ve RRN elde edilir. Eğer RRN 0 olarak bulunursa verilen indeks indeksli dosyaya ilave edilememiş demektir.

UACC tarafından yollanan istek mesajın deseni;



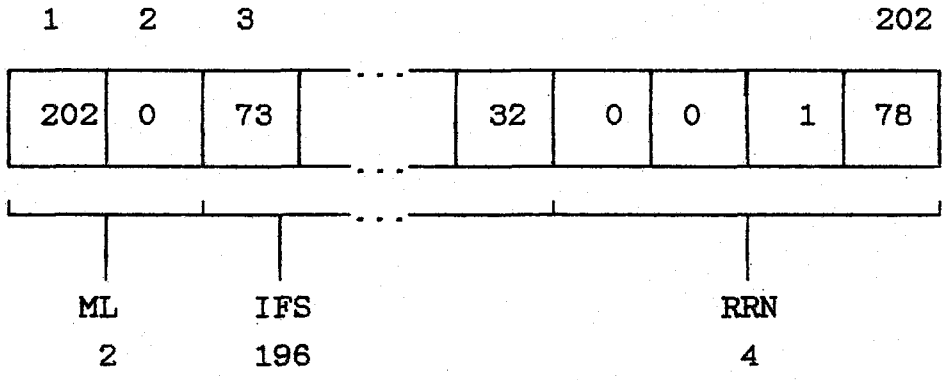
Şekil-21

Toplam mesaj uzunluğu = 2 + 2 + 196 + 60 + 4 = 264

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS ile RRN byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır.
SACC de sadece TACCESS de kullanılan AddKey komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Şekil-22

Toplam mesaj uzunluğu = 2 + 196 + 4 = 202

4.4.3.2.f. İndeks silme işlemi (DELKEYID)

Bu işlem ile verilen indeks indeksli dosyadan silinir.
Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UDeleteKey(IFS,RRN,PKEY) şeklindedir.

Burada,

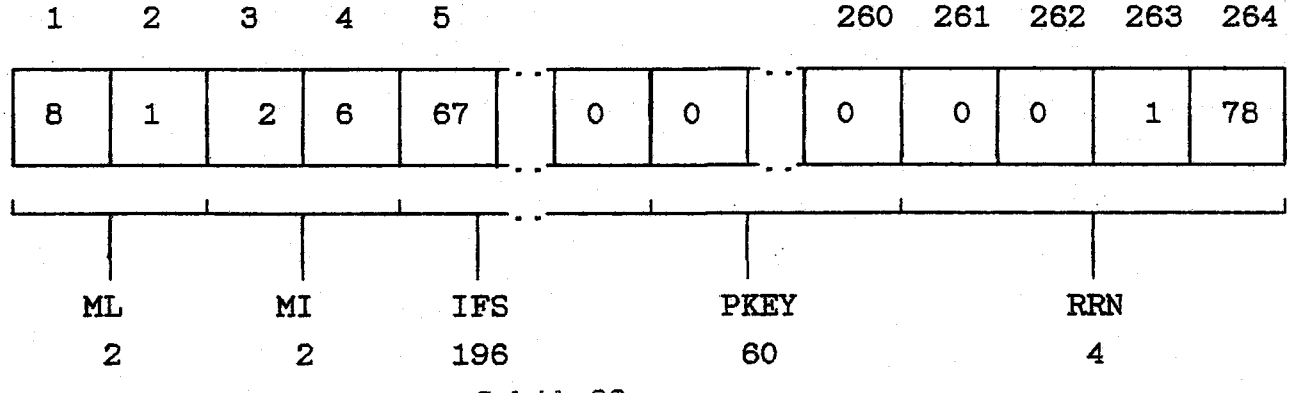
IFS : İndeksli dosyanın TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 196 byte dır.

RRN : Veri dosyasındaki ilgili kayıt numarasıdır. Eğer indeks indeksli dosyadan silinemezse bu alan 0 olarak geri döner.

PKEY : Bu alana indeksli dosyadan silinmesi istenilen indeks yüklenir. Bu kısmın uzunluğu 60 byte dır.

UACC tarafından IFS, PKEY ve RRN yollanır karşılığında ise veri dosyasının tüm özelliklerini içeren IFS ve RRN elde edilir. Eğer RRN 0 olarak bulunursa verilen indeks indeksli dosyadan silinememiş demektir.

UACC tarafından yollanan istek mesajın deseni;



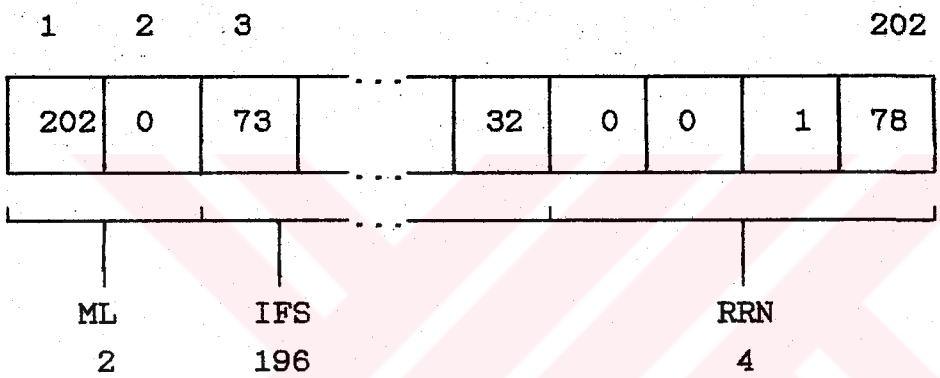
Şekil-23

Toplam mesaj uzunluğu = 2 + 2 + 196 + 60 + 4 = 264

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve IFS ile RRN byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır.
SACC de sadece TACCESS de kullanılan DeleteKey komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 196 + 4 = 202

Şekil-24

4.4.3.3. KAYIT İŞLEMLERİ İLE İLGİLİ MESAJ YAPILARI

Dört farklı kayıt işlemi vardır. Bunlar veri dosyasından kayıt alma, kayıt yazma ve silme işlemlerini yerine getirirler.

4.4.3.3.a. Kayıt okuma işlemi (GETRECID)

Bu işlem ile verilen kayıt numarasına ilişkin kayıt veri dosyasından okunur.

Kullanıcı tarafındaki yapılar ve işlemler ;
Program tarafından bildirimi

UGetRec(DFS,RRN,REC,GRM) şeklindedir.

Burada,

DFS : Veri dosyasının TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 146 byte dır.

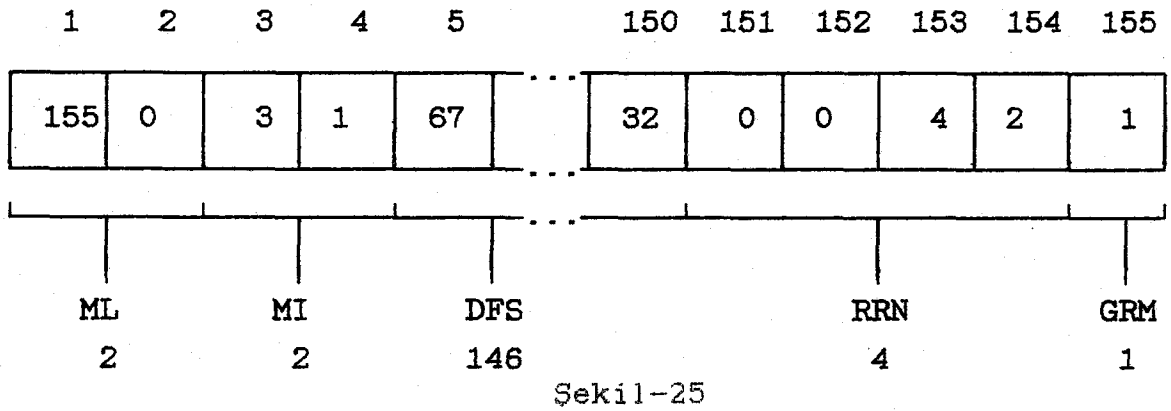
RRN : Veri dosyasındaki ilgili kayıt numarasıdır.

REC : Eğer kayıt okunabilirse bu değişkenin içine aktarılır. Bu alanın uzunluğuda RL ile ifade edilir.

GRM : Eğer okunulan kayıt kilitlenmek istenirse bu alanın değeri 1 olmalıdır ve eğer kilitli olsun istenmiyorsa bu alanın değeri 0 olmalıdır. Bu alanın geri dönüş değeri eğer 0 olursa işlem başarısız anlamındadır. Bu alanın geri dönüş değeri eğer 1 olursa işlem başarılı anlamındadır.

UACC tarafından DFS, RRN ve GRM yollanır karşılığında GRM ve eğer işlem başarılı ise REC gelir. Yani eğer GRM 0 olarak geri dönerse REC kısmı gelmez. Eğer GRM 1 olarak geri dönerse REC kısmı da gelir.

UACC tarafından yollanan istek mesajın deseni;



Şekil-25

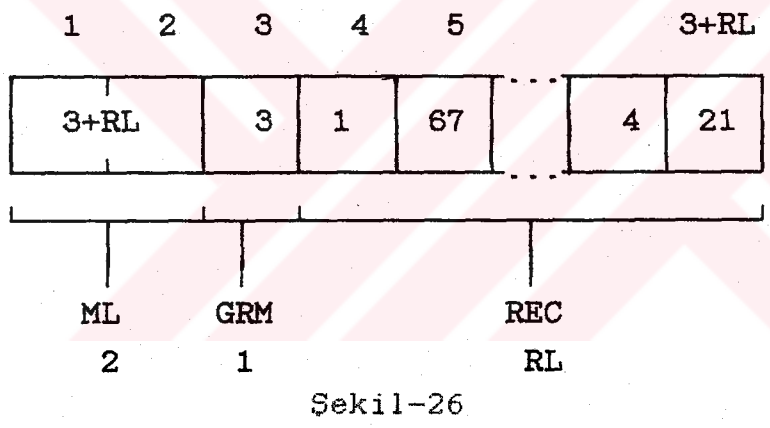
Toplam mesaj uzunluğu = 2 + 2 + 146 + 4 + 1 = 155

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir. GRM ve eğer işlem başarılı oldu ise REC byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır. SACC de sadece TACCESS de kullanılan GetRec komutu kullanılmaktadır. Aynı zamanda SACC de kayıt kilitleme olayı bir DOS fonksiyonuyla gerçekleştirilmektedir.

NURINET tarafından yollanan cevap mesajın desenleri;

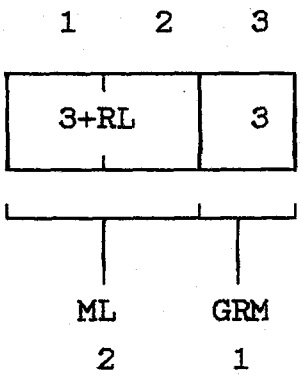
Eğer GRM=1 ise mesaj deseni



Şekil-26

Toplam mesaj uzunluğu = 2 + 1 + RL = 3 + RL

Eğer GRM=0 ise mesaj deseni



Şekil-27

Toplam mesaj uzunluğu = 2 + 1 = 3

4.4.3.3.b. Kayıt ekleme işlemi (ADDRECID)

Bu işlem ile verilen kayıt veri dosyasına ilave edilir.
Kullanıcı tarafındaki yapılar ve işlemler ;

Program tarafından bildirimi

UAddRec(DFS,RRN,REC) şeklindedir.

Burada,

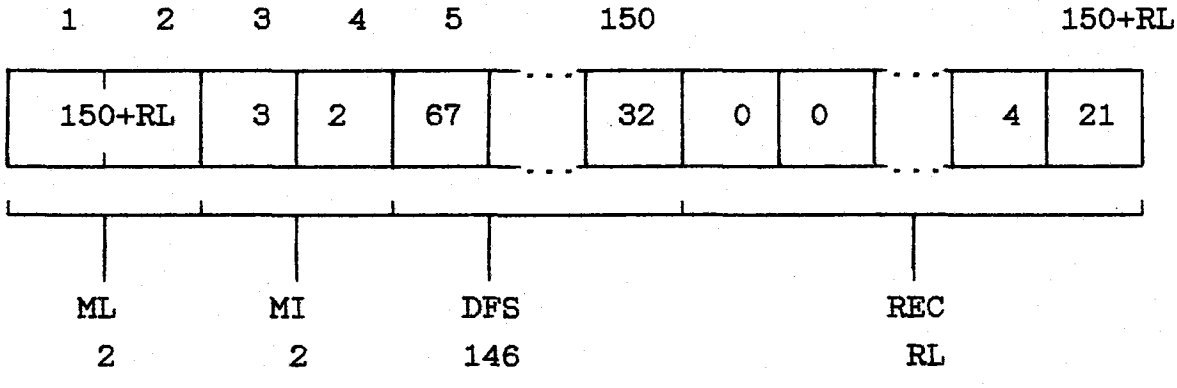
DFS : Veri dosyasının TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 146 byte dır.

RRN : Veri dosyasındaki ilgili kayıt numarasıdır. Bu alan giderken boştur ama dönüşte bir değer alır ve eğer bu değer 0 olursa kaydı yazma işlemi başarısız olmuştur. Eğer bu değer 0 dan farklı bir rakam ile dönerse işlem başarılmıştır ve bu rakam da ilgili kaydın numarasıdır.

REC : Veri dosyasına yazılacak kayıt bu alanın içindedir. Bu alanın uzunluğuda RL ile ifade edilir.

UACC tarafından DFS ve REC yollanır karşılığında RRN gelir.

UACC tarafından yollanan istek mesajın deseni;



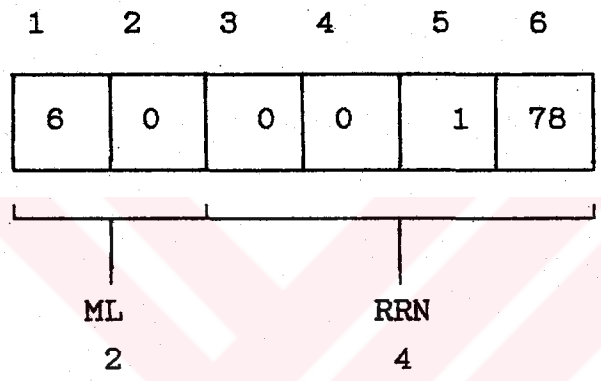
Şekil-28

Toplam mesaj uzunluğu = 2 + 2 + 146 + RL = 150+RL

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve RRN byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır. SACC de sadece TACCESS de kullanılan AddRec komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Toplam mesaj uzunluğu = 2 + 4 = 6

Şekil-29

4.4.3.3.c. Kayıt üzerine yaniden yazma işlemi (PUTRECID)

Bu işlem ile verilen kayıt veri dosyasına yazılır.

Kullanıcı tarafındaki yapılar ve işlemler ;
Program tarafından bildirimi

UPutRec(DFS,RRN,REC) şeklindedir.

Burada,

DFS : Veri dosyasının TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 146 byte dır.

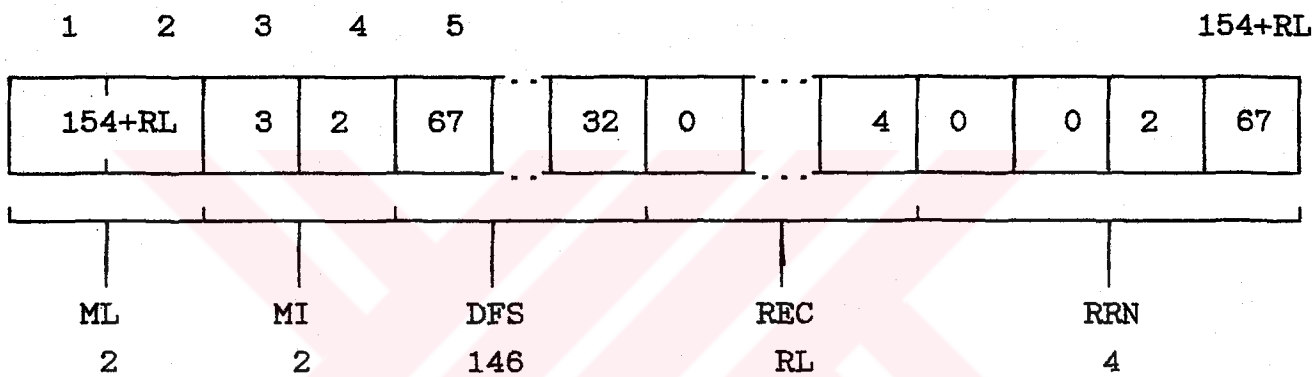
RRN : Veri dosyasındaki ilgili kayıt numarasıdır. Bu alan giderken kayıt numarasını taşır ama dönüşte eğer bu değer 0 olursa kaydı yazma işlemi

başarısız olmuştur. Eğer bu değer 0 dan farklı bir rakam ile dönerse işlem başarılmıştır ve bu rakam da ilgili kaydın numarasıdır.

REC : Veri dosyasına yazılacak kayıt bu alanın içindedir. Bu alanın uzunluğuda RL ile ifade edilir.

UACC tarafından DFS, REC ve RRN yollanır karşılığında RRN gelir.

UACC tarafından yollanan istek mesajın deseni;



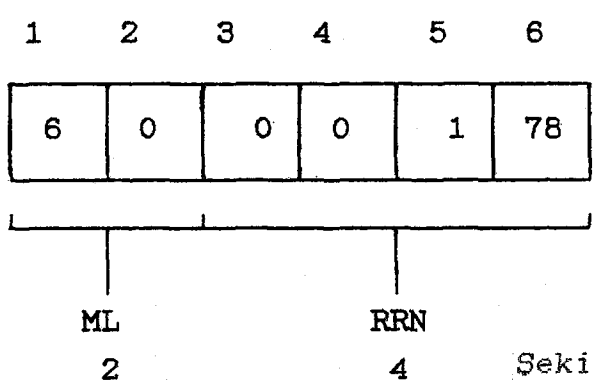
Şekil-30

Toplam mesaj uzunluğu = 2 + 2 + 146 + RL +4 = 154+RL

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir ve RRN byte byte mesaj alanına doldurularak istek yapan kullanıcıya yollanır. SACC de sadece TACCESS de kullanılan PutRec komutu kullanılmaktadır.

NURINET tarafından yollanan cevap mesajın deseni;



Şekil-31

Toplam mesaj uzunluğu = 2 + 4 = 6

4.4.3.3.d. Kayıt silme işlemi (DELRECID)

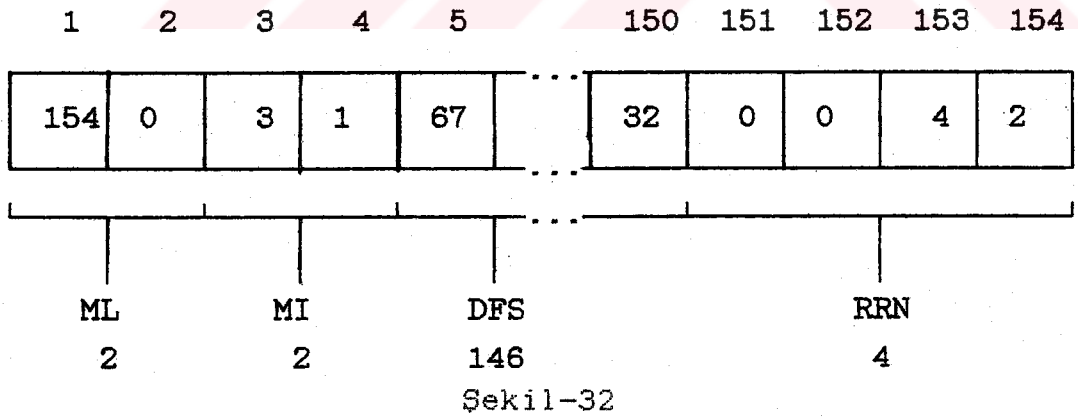
Bu işlem ile verilen kayıt numarasına ilişkin kayıt veri dosyasından silinir.

Kullanıcı tarafındaki yapılar ve işlemler ;
Program tarafından bildirimi

UDelRec(DFS,RRN) şeklindedir.

Burada,
DFS : Veri dosyasının TACCESS de kullanıldığı şekliyle yapısıdır. Bu kısmın uzunluğu 146 byte dır.
RRN : Silinmesi istenilen kayıtın veri dosyasındaki kayıt numarasıdır.

UACC tarafından DFS ve RRN yollanır karşılığında ise bilgi olarak hiçbir şey alınmaz.
UACC tarafından yollanan istek mesajın deseni;



Şekil-32

Toplam mesaj uzunluğu = 2 + 2 + 146 + 4 = 154

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir SACC vasıtasıyla yerine getirilir. SACC de sadece TACCESS de kullanılan DelRec komutu kullanılmaktadır. NURINET tarafından kullanıcıya kullanacağı bir mesaj yollanmaz.

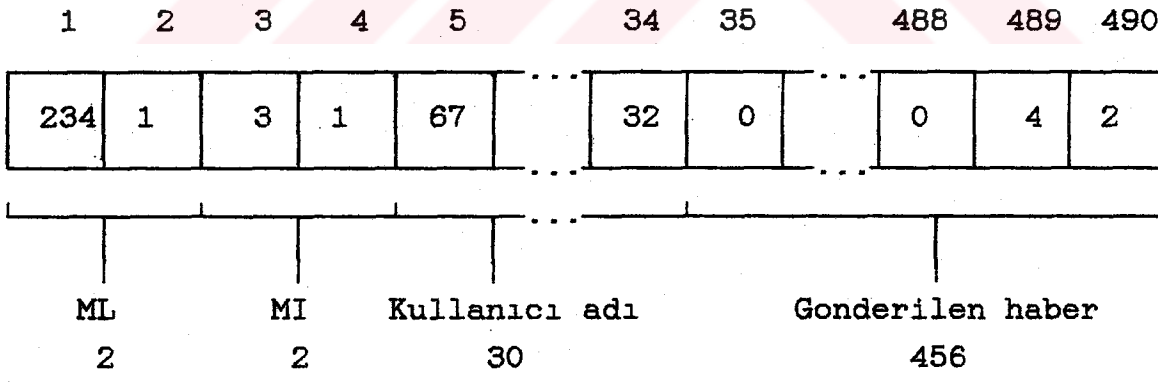
4.4.3.4. MESAJ IŞLEMLERI ILE ILGILI MESAJ YAPILARI

Sistemde bir tane mesaj işlemi vardır. Bu işlem kullanıcı makinalardan ana makinaya haber niteliğinde mesaj yollamak için kullanılır. Bu işlem sistemde tek taraflıdır. Yani ana makinadan kullanıcı makinaya haber niteliğinde mesaj yollamak mümkün değildir. Ana makinadan kullanıcılara istekleri dışında herhangi bir bilgi yollanmaz.

4.4.3.4.a. Haber yollama işlemi (MESSAGEID)

Bu işlem ile ana makinaya mesaj yollanır.

Kullanıcı tarafındaki yapılar ve işlemler ;



Şekil-33

Toplam mesaj uzunluğu = 2 + 2 + 30 + 456 = 490

Ana makine tarafındaki yapılar ve işlemler ;

Ana makinede alınan emir yerine getirilir fakat cevap olarak kullanıcıya herhangi birşey yollanmaz.

5. SISTEMI OLUŞTURAN YAZILIMLAR

Bu çalışmada muhtelif işleri paylaşan yazılımlar yeralmıştır. Örneğin mesaj yollama ve alma işlerini sadece SRUN adlı program yapar. Bu yazılımlar INST, UACC, SRUN, NURINET, SACC diye sıralanabilir. Çünkü kullanıcı makinedeki bir program önce UACC kullanır, dolayısıyla UACC de SRUN adlı programı kullanır ve mesaj ana makineye yollanır. Yine SRUN ve NURINET programlarıyla ana makinede mesaj alınır ve gerekli işlem SACC sayesinde yapılarak cevap hazırlanarak kullanıcı makineye yollanır. Kullanıcı makinedeki program yine UACC programı ile cevap olarak gelen mesajı SRUN isimli program vasıtasıyla alır.

5.1. "INST" YAZILIMI

Bu yazılım sistemdeki kullanıcıların olan makinaların kullanılacak seri portları tanımlamaları için hazırlanmıştır. Kullanıcı bu program vasıtasıyla kendi üzerinde ana makina ile haberleşmek için hangi seri portu kullanacağını, ana makinada hangi sürücüyü hangi dizini kullanacağını belirler. Bu programın ana makinada çalıştırılması gerekmez. Kişisel bilgisayarlarda Seri portların adresleri (2 tane olduğu varsayılırsa) COM1 için 3F8h ve COM2 için 2F8h dir [4]. Örneğin COM1 den bilgi göndermek istenirse 3F8h portuna gönderilecek bilgiyi yazmak yeterlidir. Aynı şekilde bilgi okumak istendiğinde aynı porta gelen bilgiyi okumak yeterlidir.

5.2. "SRUN" YAZILIMI

Bu sistemde iletişim açısından en önemli parçayı bu yazılım oluşturmaktadır. Makinaların hangi seri portlardan haberleşeceği bu program içindeki "ParametreOku" adlı alt program parçacığı yapar. Bu alt program ile INST adlı programın oluşturduğu "PORT.DFT" dosyasından kullanılacak seri portu ve ana makinada kullanılacak olan sürücüyü ve dizini okur. Bundan sonra kullanıcının uygulama programında bu parametreler kullanılır.

Iletişimde bir aksaklık olursa örneğin "SRUN" adlı programda, parametre olarak verilen sayıdan fazla denemeye rağmen hala daha mesaj yollanamıyorsa bir donanım sorunu var demektir. Bu durumda program kesilir. Aynı şekilde eğer kullanıcı tarafından ana makineye sorulan soruya verilmesi gereken cevap belirli bir süre içinde gelmez ise program yine kesilir.

Ayrıca bazı programlarda ortaklaşa kullanılan tanımlamalar da "SRUN" adlı program içinde tanımlanmıştır. Çünkü "SRUN" adlı program her iki taraftada yani hem kullanıcı ve hem de ana makina tarafında kullanılmaktadır.

5.3. "UACC" YAZILIMI

Bu yazılım, kullanıcının uygulama programındaki isteğini seri port ile yollanabilir hale getirir ve kontrolu "SRUN" a devreder. Mesaj yollama işi bittiğinde yinde kontrolu alır ve cevap mesajını beklemek üzere yine kontrolu "SRUN" a verir. Beklenen cevap geldikten sonra bu cevabı yine uygulama programında kullanılacak hale getirir.

5.4. "NURINET" YAZILIMI

Kişisel bilgisayar yazılımları arasında "Memory Resident" adı verilen hafızada kalarak geri planda çalışan bir programdır. Bu Turbo Pascal dilindeki "Keep" komutu ile gerçekleştirilmektedir.

Yine bu yazılım içinde seri portların kullandığı kesmeler (Interrupt) yerine bazı program parçacıkları konulmuştur. Bu program parçacıkları seri porta bir sinyal geldiğinde çalışmaya başlarlar. Böylece bilgisayarın tüm zamanını almamış olur. Bu işlemde yine Turbo Pascal ın GetIntVec ve SetIntVec adlı komutları ile yapılmıştır.

5.5. "SACC" YAZILIMI

Kullanıcıdan gelen mesaj "NURINET" programıyla alınıp kullanılabilir hale getirildikten sonra "SACC" adlı yazılım istenen işlemi yapar. Bu işlemlerin çoğu "TACCESS" adlı program tarafından yapılır. Fakat kayıt kilitleme, kayıt serbest bırakma ve kayıtın kilitli olup olmadığı kontrolleride "SACC" programı içinde küçük birer program parçacığı olarak yer almıştır. Kayıt kilitleme işlemi ve kayıt serbest bırakma işlemi bir DOS fonksiyonu yardımıyla yapılmaktadır. Fakat bu fonksiyonun çalışabilmesi için yine DOS ortamında bulunan "SHARE" adlı programın "NURINET" adlı programdan önce çalıştırılması gerekmektedir.

5.6. "MESAJ" YAZILIMI

Bu program kullanıcı makinaların ana makinaya haber niteliğinde mesaj yollamasını sağlar. Mesaj 30 uzunluğunda kullanıcı adını ve 456 uzunluğundaki mesajı içermektedir. Kullanıcı adı, kullanıcının girebileceği bir alandır. "NURINET" programıyla mesajın tümü alındıktan sonra bu alınan mesaj, mesajın geldiği tarih ve mesajın geldiği zaman "MESAJLAR" adlı bir dosyaya yazılır. Ana makinada bu dosya bir DOS komutu olan TYPE ile de görülebilir.

5.7. YAZILIMLARIN DÖKÜMLERI

```
(*****************************************************************)
(*                            INST                               *)
(*                                                               *)
(*      KULLANICILARIN PORT,SÜRÜCÜ VE DIZIN BELIRLEME PROGRAMI   *)
(*                                                               *)
(*****************************************************************)
program INST;
uses crt,dos;
var
   c       : char;
   sut,
   sat,i   : byte;
   aa      : string;
   w1      : word;
   secim,
   Ysecim  : byte;
   SS      : array[1..4] of string[4];
   konumX  : array[1..4] of byte;
   konumY  : array[1..4] of byte;
   BaseWord : word;
   Cik,
   tamam     : Boolean;
   TF        : text;
   Str4      : string[4];
   str64     : string[64];
const NormalRenk = 7;
      SecimRenk  = 127;
      retset     : set of  char = [return,esc];
{------------}
procedure ekr01;
var renk1 : byte;
begin
   renk1:=7;
   outstr('Ana makinede kullanilacak dosya :',4,1,renk1);
   outstr('KULLANILACAK SERI PORT TANITIMI',sat-1,16,renk1);
   outstr('                              ',sat,sut,renk1);
   outstr('  COM1 :                      ',sat+1,sut,renk1);
   outstr('  COM2 :                      ',sat+2,sut,renk1);
   outstr('  COM3 :                      ',sat+3,sut,renk1);
   outstr('  COM4 :                      ',sat+4,sut,renk1);
   outstr('                              ',sat+5,sut,renk1);
end;
{------------}
function WTOSTR(var KK : word) : string;
var Bolum: byte;
    s1 : string[2];
    s2 : string[1];
    sonstr : string[4];
    harf : array [10..15] of string[1];
begin
   harf[10]:='A';  harf[11]:='B';  harf[12]:='C';
   harf[13]:='D';  harf[14]:='E';  harf[15]:='F';
```

```
   s1:='';  s2:='';
   sonstr:='';
   bolum:=0;
   WTOSTR:='';
   if KK > 255  then
      begin
         bolum:=KK div 256;
         KK:=KK-(bolum*256);
         if bolum < 10 then begin str(bolum:0,s1);
         s2:=s1; end;
         if bolum > 9  then s2:=harf[bolum];
         sonstr:=sonstr+s2;
      end;
   if KK > 15  then
      begin
         bolum:=KK div 16;
         KK:=KK-(bolum*16);
         if bolum < 10 then begin str(bolum:0,s1);
         s2:=s1; end;
         if bolum > 9  then s2:=harf[bolum];
         sonstr:=sonstr+s2;
      end;
   if KK > 0  then
      begin
         bolum:=KK div 1;
         KK:=KK-(bolum*1);
         if bolum < 10 then begin str(bolum:0,s1);
           s2:=s1; end;
         if bolum > 9  then s2:=harf[bolum];
         sonstr:=sonstr+s2;
      end;
    if sonstr='' then sonstr:='YOK ';
    sonstr:=sonstr+' ';
    WTOSTR:=sonstr;
end;
{------------}
Procedure FiledanOku;
begin
   Reset(TF);
   readln(TF,Str4);
   if Not(Eof(TF)) then readln(TF,Str64) else str64:=space(64);
   Close(TF);
   end
   else
   begin
     str4:='';
     str64:=space(64);
   end;
end;
{------------}
procedure AraBul;
begin
  if str4<>'' then
    for i:=1 to 4 do
    begin
      if SS[I]=str4 then secim:=i;
```

```
    end;
end;
{------------}
procedure FileaYaz;
begin
   rewrite(TF);
   str4:=SS[secim]+'     ';
   str4[0]:=chr(4);
   writeln(TF,SS[secim]);
   writeln(TF,str64);
   close(TF);
   gotoxy(10,22);
   writeln('Tanimlama isleminiz tamamlanmistir !...');
end;
{------------}

begin
   FiledanOku;
   Str64:=str64+space(64);
   clrscr;
   sat:=10; sut:=20;
   KonumX[1]:=sat+1; KonumY[1]:=sut+9;
   KonumX[2]:=sat+2; KonumY[2]:=sut+9;
   KonumX[3]:=sat+3; KonumY[3]:=sut+9;
   KonumX[4]:=sat+4; KonumY[4]:=sut+9;
   ekr01;
   baseword:=$400;
   for i:=1 to 4 do
    begin
        w1:=memw[0:baseword];
        SS[I]:=WTOSTR(w1);
        outstr(SS[I],konumX[i],konumY[i],NormalRenk);
        BaseWord:=BaseWord+$002;
    end;
   cik:=False;
   secim:=1;
   AraBul;
   tamam:=false;
   repeat
     Readstr(str64,5,1,secimrenk,NormalRenk,retset,ret,' ',' ');
     if ret in [return,esc]  then Tamam:=True;
   until tamam;
   Ysecim:=Secim;
if ret<>esc then
begin
   repeat
      outstr(SS[secim],konumX[secim],konumY[secim],SecimRenk);
      c:=readkey;
      if c=#0 then
        begin
           c:=readkey;
           if c='P' then
             if secim = 4 then Ysecim:=1 else Ysecim:=Secim+1;
           if c='H' then
             if secim = 1 then Ysecim:=4 else Ysecim:=Secim-1;
        end;
```

```
      outstr(SS[secim],konumX[secim],konumY[secim],NormalRenk);
      Secim:=Ysecim;
      if (c=#13) and (SS[secim]='YOK ') then
      begin
          gotoxy(20,20);
          write('Bu seciminiz anlamsiz !...');
          c:=readkey;
          gotoxy(20,20);
          write('                          ');
          c:=#0;
      end;
   until (c=#27) or (c=#13);
   if c=#13 then FileaYaz;
end;
end.
```

```
(*****************************************************************)
(*                          SRUN                                 *)
(*                                                               *)
(*      ANA MAKINEDE VE KULLANICI MAKINELERDE SERI PORTLARDAN    *)
(*                  ULAŞIMI SAĞLAYAN PROGRAM                     *)
(*                                                               *)
(*****************************************************************)
unit SRUN;
{$F+}
INTERFACE
uses crt,dos;
const
    Rcv_Wait_Time : longint = 1000;
    Wait_Time : integer = 1000;
    sb        : longint = 256;
    MKeylen   = 60;
    FNameLen  = 64;
{    MsgID    sabitleri     }
    OPENDFILEID =  #1#1;    {  MsgId = #1#1  File islemi    Open Data }
    OPENIFILEID =  #1#2;    {  MsgId = #1#2  File islemi    Open Index}
    CLOSEDFILEID=  #1#3;    {  MsgId = #1#3  File islemi    Close Data}
    CLOSEIFILEID=  #1#4;    {  MsgId = #1#4  File islemi    Close Index}
    FINDKEYID   =  #2#1;    {  MsgId = #2#1  Key islemi     Find      }
    SEARCHKEYID =  #2#2;    {  MsgId = #2#2  Key islemi     Search    }
    PREVKEYID   =  #2#3;    {  MsgId = #2#3  Key islemi     Previous  }
    NEXTKEYID   =  #2#4;    {  MsgId = #2#4  Key islemi     Next      }
    ADDKEYID    =  #2#5;    {  MsgId = #2#5  Key islemi     Add       }
    DELKEYID    =  #2#6;    {  MsgId = #2#6  Key islemi     Add       }
    GETRECID    =  #3#1;    {  MsgId = #3#1  Record islemi  GetRec    }
    ADDRECID    =  #3#2;    {  MsgId = #3#2  Record islemi  AddRec    }
    PUTRECID    =  #3#3;    {  MsgId = #3#3  Record islemi  PutRec    }
    DELRECID    =  #3#4;    {  MsgId = #3#4  Record islemi  DelRec    }
    MESSAGEID   =  #4#1;    {  MsgId = #4#1  Mesaj  islemi            }
var
  Regs    : Registers;
  ChrSR   : char;
  bs      : array [0..7] of byte;
  bsw     : array [0..15] of byte;
  MsgID   : string[2];
  MsgBuf  : array [1..2048] of char;
  MsgLen  : Integer;
  IndxFileName : string[64];
  DataFileName : string[64];
  IndxKey      : string[MKeyLen];
  RecordLength : word;
  GetRecMode   : Byte;
  PathName     : string;
  XF8       : word ;
  XFD       : word ;
Procedure Abort(ErrorStr:string);
Procedure AdjLen(uzunluk:integer);
function LITOCH(var LI : longint ):string;
function CHTOLI(var str4 : string ):longint;
function RECLENTOCH(sayi : word ) : string;
Procedure ayristir(sayi:byte);
Procedure WordAyristir(sayi:Word);
```

```
Function YU : Boolean; { Serial Port Yollamaya Uygun mu ? }
Function AU : Boolean; { Serial Port Almaya Uygun mu ? }
Function OA : boolean; { Serial porta gonderilen char alinmismi onayi }
Procedure SendChar(var ChrSR:char);
Procedure RcvChar(var ChrSR : char);
Procedure SendMsg(var Msg);
Procedure RcvMsg(var Msg);
IMPLEMENTATION
{------------}
procedure Abort(errorstr : string);
begin
  gotoxy(10,30);
  write(errorstr);
  Halt(1);
end;
{--------------------------------}
Procedure AdjLen(uzunluk:integer);
var
  tmpuz1,
  tmpuz2,
  kalan:integer;
begin
    inc(uzunluk,2);
    if uzunluk>2048 then Abort('Mesaj uzunlugu 2048 i asiyor');
    tmpuz1:=trunc(uzunluk/sb);
    tmpuz2:=tmpuz1*sb;
    kalan :=uzunluk-tmpuz2;
    MsgBuf[1]:=chr(kalan);
    MsgBuf[2]:=chr(tmpuz1);
end;
{--------------------------------}
function LITOCH(var LI : longint ):string;
var
   TMPLI,
   kalan,
   I1,
   sonuc,lp: longint;
   HA     : array [0..4] of char;
   i      : byte;
   s : string;
begin
     lp:=LI;
     for i:=0 to 4 do ha[i]:=#0;
     i:=1;
     repeat
        i1:=trunc(LI/256);
        TMPLI:=i1*256;
        Kalan:=LI-TMPLI;
        ha[i]:=chr(kalan);
        li:=i1;
        inc(i);
     until li<256;
     ha[i]:=chr(li);
     if lp<sb then ha[0]:=chr(1) else ha[0]:=chr(i);
     s:='';
     for i:=1 to ord(ha[0]) do s:=s+ha[i];
```

```
      s:=s+#0+#0+#0+#0;
      s[0]:=chr(4);
      LI:=lp;
      LITOCH:=S;
end;  { end of function LITOCH }
{-------------------------------------------------------}
function CHTOLI(var str4 : string ):longint;
begin
   CHTOLI:=ord(str4[1])+sb*ord(str4[2])+sb*sb*ord(str4[3])+sb*sb*sb*ord(str4[4
end;
{-------------------------------------------------------}
function RECLENTOCH(sayi : word ) : string;
var
   s2 : string[2];
   i1 : byte;
begin
   i1:=trunc(sayi/sb);
   sayi:=sayi-(i1*sb);
   s2:=chr(sayi)+chr(i1);
   RECLENTOCH:=s2;
end;
{-------------------------------------------------------}
procedure WordAyristir(sayi:word);
var i:byte;
begin
  for i:=0 to 15 do bsw[i]:=0;
  if sayi > 32767 then begin bsw[15]:=1; sayi:=sayi-32768; end;
  if sayi > 16383 then begin bsw[14]:=1; sayi:=sayi-16384; end;
  if sayi > 8191 then begin bsw[13]:=1; sayi:=sayi-8192; end;
  if sayi > 4095 then begin bsw[12]:=1; sayi:=sayi-4096; end;
  if sayi > 2047 then begin bsw[11]:=1; sayi:=sayi-2048; end;
  if sayi > 1023 then begin bsw[10]:=1; sayi:=sayi-1024; end;
  if sayi > 511 then begin bsw[9]:=1; sayi:=sayi-512; end;
  if sayi > 255 then begin bsw[8]:=1; sayi:=sayi-256; end;
  if sayi > 127 then begin bsw[7]:=1; sayi:=sayi-128; end;
  if sayi > 63  then begin bsw[6]:=1; sayi:=sayi-64;  end;
  if sayi > 31  then begin bsw[5]:=1; sayi:=sayi-32;  end;
  if sayi > 15  then begin bsw[4]:=1; sayi:=sayi-16;  end;
  if sayi > 7   then begin bsw[3]:=1; sayi:=sayi-8;   end;
  if sayi > 3   then begin bsw[2]:=1; sayi:=sayi-4;   end;
  if sayi > 1   then begin bsw[1]:=1; sayi:=sayi-2;   end;
  if sayi > 0   then begin bsw[0]:=1; sayi:=sayi-1;   end;
end; { end of  WordAyristir }
{-------------------------------------------------------}
procedure ayristir(sayi:byte);
var i:byte;
begin
  for i:=0 to 7 do bs[i]:=0;
  if sayi > 127 then begin bs[7]:=1; sayi:=sayi-128; end;
  if sayi > 63  then begin bs[6]:=1; sayi:=sayi-64;  end;
  if sayi > 31  then begin bs[5]:=1; sayi:=sayi-32;  end;
  if sayi > 15  then begin bs[4]:=1; sayi:=sayi-16;  end;
  if sayi > 7   then begin bs[3]:=1; sayi:=sayi-8;   end;
  if sayi > 3   then begin bs[2]:=1; sayi:=sayi-4;   end;
  if sayi > 1   then begin bs[1]:=1; sayi:=sayi-2;   end;
  if sayi > 0   then begin bs[0]:=1; sayi:=sayi-1;   end;
```

```
end; { end of  ayristir }
{-------------}
function YU : Boolean; { Serial Port Yollamaya Uygun mu ? }
var b : integer;
   DYU : boolean;
begin
   b:=0;
   DYU:=false;
   repeat
     ayristir(port[XFD]);
     if bs[6]=1 then DYU:=true;
     inc(b);
   until (DYU or (b=Wait_time));
   YU:=DYU;
end;
{-------------}
function AU : Boolean; { Serial Port Almaya Uygun mu ? }
var b : integer;
   DAU : boolean;
begin
   b:=0;
   DAU:=false;
   repeat
     ayristir(port[XFD]);
     if bs[0]=1 then DAU:=true;
     inc(b);
   until (DAU or (b=Wait_time));
   AU:=DAU;
   if not DAU then Abort('Seri Port Mesaj almaya uygun degil');
end;
{-------------}
function OA : boolean; { Serial porta gonderilen char alinmismi onayi }
var b : integer;
    DOA : boolean;
begin
   b:=0;
   DOA:=false;
   repeat
      ayristir(port[XFD]);
      if ((bs[0]=1) and (port[XF8]=15)) then DOA:=true;
      inc(b);
   until (DOA or (b=Wait_Time));
   OA:=DOA;
end;
{-----------------------------------------}
procedure SendChar(var ChrSR:char);
begin
  if YU then
  begin
     port[XF8]:=ord(ChrSR);
     if Not OA then Abort('Seri Port Mesaj gondermeye uygun degil');
  end;
end;  { end of SendChar  }
{-------------}
Procedure RcvChar(var ChrSR : char);
var k : char;
```

```
begin
  if AU then
  begin
     ChrSR:=chr(port[XF8]);
     if YU then Port[XF8]:=15;
  end;
end;  { end of RcvChar  }
{-----------------------------------------}
procedure SendMsg(var  Msg);
var i       : integer;
    ch      : char;
    uzunluk : integer;
type  bs = array[1..2048] of char;
begin
  uzunluk:=ord(bs(msg)[1]);
  uzunluk:=uzunluk+sb*ord(bs(msg)[2]);
   for i:=1 to uzunluk do
   begin
      ch:=bs(msg)[i];
      SendChar(ch);
   end;
end;
{-----------------------------------------}
function Rcv_AU : Boolean; { Serial Port Almaya Uygun mu ? }
var b : integer;
   Rcv_DAU : boolean;
begin
   b:=0;
   Rcv_DAU:=false;
   repeat
     ayristir(port[XFD]);
     if bs[0]=1 then Rcv_DAU:=true;
     inc(b);
   until (Rcv_DAU or (b=Wait_time));
   Rcv_AU:=Rcv_DAU;
end;
{-------------}
procedure RcvMsg(Var Msg);
var i : longint;
    k : char;
type  bs = array[1..2048] of char;
begin
   i:=0;
   repeat
     inc(i);
   until ((Rcv_AU) or (i=Rcv_Wait_Time));
   if Not Rcv_AU then Abort('Sorumun cevabi gelmedi amca !....');
   RcvChar(k);
   bs(msg)[1]:=k;
   RcvChar(k);
   bs(msg)[2]:=k;
   Msglen:=ord(bs(msg)[1])+sb*ord(bs(msg)[2]);
   for i:=1 to Msglen-2 do
   begin
      RcvChar(k);
      bs(msg)[i+2]:=k;
```

```
   end;
end;
{-------------------------}
Procedure ParametreOku;
var str4 : string[4];
    str64: string[64];
    FileVar : Boolean;
    TF : text;
    i : byte;
begin
   {$I-}
   assign(TF,'PORT.DFT');
   Reset(TF);
   Close(TF);
   {$I+}
   FileVar:=False;
   if IOresult=0 then FileVar:=True;
   if Filevar then
   begin
      Reset(TF);
      readln(TF,Str4);
      if not(Eof(TF)) then readln(TF,Str64);
      Close(TF);
   end
   else
   begin
      str4:='3F8 ';
      str64:='';
      for i:=1 to 64 do str64:=str64+' ';
   end;
   if str4='3F8 ' then
    begin
      XF8:=$3F8;
      XFD:=$3FD;
    end;
   if str4='2F8 ' then
    begin
      XF8:=$2F8;
      XFD:=$2FD;
    end;
    FileVar:=False;
    i:=64;
    repeat
      if str64[i] <>  ' ' then Filevar:=true else dec(i);
    until ((Filevar) or (i=0));
    str64[0]:=chr(i);
    PathName:=str64;
    PathName[0]:=chr(i);
end; { ParametreOku }
{-----------------------------------------}
begin
   ParametreOku;
end.
{$F-}
```

```
(*****************************************************************)
(*                          UACC                                 *)
(*                                                               *)
(*   KULLANICININ VERI TABANINA ULAŞIMINI SAĞLAYAN PROGRAM       *)
(*                                                               *)
(*****************************************************************)
UNIT UACC;
INTERFACE
uses  CRT, DOS, SRUN;
const
     spc64 : string[64] ='
     spc60 : string[64] ='
     UMaxHeight      =    5;
type
     StrStr = string[255];
     UFileName = string[64];
type
  UDataFile  =  record
                  UF          : file;
                  UFirstFree,
                  UNumberFree,
                  UInt1       : LongInt;
                  UItemSize   : word;
                  UNumRec     : LongInt;
                end;
  UTaSearchStep =
                record
                  UPageRef,UItemArrIndex : LongInt;
                end;
  UTaPath     =  array[1..UMaxHeight] of UTaSearchStep;
  UIndexFile =  record
                  UDataF         : UDataFile;
                  UAllowDuplKeys : Boolean;
                  UKeyL          : byte;
                  URR,UPP        : LongInt;
                  UPath          : UTaPath;
                end;
var
     i  : integer;
     ss : array[1..400] of char;
     adr1,
     adr2 : word;
type bs = array [1..2048] of char;
     bs1 = array [1..2048] of char;
     bskey = array [1..64] of char;
{--------------------------------}
procedure UOpenDataFile(var UDatF      : UDataFile;
                        UFName     : UFileName;
                        URecordLen : word);
procedure UCloseDataFile(var UDatF      : UDataFile);
procedure UOpenIndexFile(var UIdxF   : UIndexFile;
                             UFName  : UFileName;
                             UKeyLen,
                             US      : byte);
procedure UCloseIndexFile(var UIdxF   : UIndexFile);
procedure SetMsgId;
```

```
procedure UFindKey(var UIdxF       : UIndexFile;
                   var UDataRecNo  : LongInt;
                   var UProcKey                 );
procedure UGetRec (var UDatF       : UDataFile;
                   var UDataRecNo : LongInt;
                   var URecord              ;
                   var UGetRecMode: Byte      );
procedure USearchKey  (var UIdxF       : UIndexFile;
                       var UDataRecNo  : LongInt;
                       var UProcKey                  );
procedure UPrevKey  (var UIdxF       : UIndexFile;
                     var UDataRecNo  : LongInt;
                     var UProcKey                );
procedure UNextKey  (var UIdxF       : UIndexFile;
                     var UDataRecNo  : LongInt;
                     var UProcKey                );
procedure UAddKey   (var UIdxF       : UIndexFile;
                     var UDataRecNo  : LongInt;
                     var UProcKey                );
procedure UAddRec (var UDatF       : UDataFile;
                   var UDataRecNo : LongInt;
                   var URecord                 );
procedure UPutRec (var UDatF       : UDataFile;
                   var UDataRecNo : LongInt;
                   var URecord                 );
procedure UDeleteKey (var UIdxF       : UIndexFile;
                      var UDataRecNo  : LongInt;
                      var UProcKey                 );
procedure UDeleteRec (var UDatF       : UDataFile;
                      var UDataRecNo : LongInt    );
implementation
{-------------------------------}
procedure SetMsgId;
begin
  MsgBuf[3]:=MsgId[1];
  MsgBuf[4]:=MsgId[2];
end;
{-------------------------------}
procedure UOpenDataFile(var UDatF   : UDataFile;
                        UFName  : UFileName;
                        URecordLen : word);
var
   s2 : string[2];
begin
   UFname:=UFName+spc64;
   MsgID:=OPENDFILEID;
   s2:=RECLENTOCH(URecordLen);
   for i:=1 to FNameLen do MsgBuf[4+i]:=UFName[i];
   MsgBuf[Fnamelen+5]:=s2[1];
   MsgBuf[Fnamelen+6]:=s2[2];
   AdjLen(length(UFName)+length(s2)+2);
   SetMsgId;
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   adr1:=seg(UDatF);
   adr2:=ofs(UDatF);
```

```
   for i:=0 to sizeof(UDatF)-1
       do mem[adr1:adr2+i]:=ord(MsgBuf[i+3]);
end;  { End of UOpenDataFile  }
{-------------------------------}
procedure UCloseDataFile(var UDatF      : UDataFile);
begin
   MsgID:=CLOSEDFILEID;
   AdjLen(SizeOf(UDatF)+2);
   SetMsgId;
   adr1:=seg(UDatF);
   adr2:=ofs(UDatF);
   for i:=0 to sizeof(UDatF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   adr1:=seg(UDatF);
   adr2:=ofs(UDatF);
   for i:=0 to sizeof(UDatF)-1
       do mem[adr1:adr2+i]:=ord(MsgBuf[i+3]);
end;  { End of UCloseDataFile  }
{-------------------------------}
procedure UOpenIndexFile(var UIdxF    : UIndexFile;
                             UFName   : UFileName;
                             UKeyLen,
                             US       : byte);
var
   s2 : string[2];
begin
   UFname:=UFName+spc64;
   s2:=chr(UKeyLen)+chr(US);
   MsgID:=OPENIFILEID;
   for i:=1 to FNameLen do MsgBuf[4+i]:=UFName[i];
   MsgBuf[Fnamelen+5]:=s2[1];
   MsgBuf[Fnamelen+6]:=s2[2];
   AdjLen(length(UFName)+length(s2)+2);
   SetMsgId;
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=ord(MsgBuf[i+3]);
end;  { End of UOpenIndexFile  }
{-------------------------------}
procedure UCloseIndexFile(var UIdxF    : UIndexFile);
begin
   MsgID:=CLOSEIFILEID;
   AdjLen(sizeof(UIdxF)+2);
   SetMsgId;
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   adr1:=seg(UIdxF);
```

```
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=ord(MsgBuf[i+3]);
end;  { End of UCloseIndexFile  }
{-------------------------------}
procedure UFindKey(var UIdxF       : UIndexFile;
                   var UDataRecNo  : LongInt;
                   var UProcKey                  );
var s1   : string[60];    { Prockey               }
    str4 : string[4];
begin
   s1:=StrStr(UProcKey)+spc60;
   s1[0]:=chr(MKeyLen);
   MsgID:=FINDKEYID;
   AdjLen(sizeof(UIdxF)+length(s1)+2);
   SetMsgId;
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to MkeyLen  do MsgBuf[4+sizeOf(UIdxF)+i]:=s1[i];
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=Ord(MsgBuf[3+i]);
   str4:='';
   for i:=1 to 4 do str4:=str4+bs(msgbuf)[i+2+sizeof(UIdxF)];
   UDataRecNo:=CHTOLI(str4);
end; {  UFindKey   }
{-------------------------------}
procedure UGetRec(var UDatF      : UDataFile;
                  var UDataRecNo : LongInt;
                  var URecord              ;
                  var UGetRecMode: Byte       );
var s4 : string[4];
begin
   MsgID:=GETRECID;
   s4:=LITOCH(UDataRecNo)+#0+#0+#0+#0;
   AdjLen(SizeOf(UDatF)+Length(s4)+1+2);
   SetMsgId;
   adr1:=seg(UDatF);
   adr2:=ofs(UDatF);
   for i:=0 to sizeof(UDatF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to 4 do MsgBuf[i+4+sizeOf(UDatF)]:=s4[i];
   MsgBuf[9+SizeOf(UDatF)]:=chr(UGetRecMode);
   SendMsg(MsgBuf);
   RcvMsg(MsgBuf);
   Msglen:=ord(bs(msgbuf)[1])+ord(bs(msgbuf)[2])*sb;
   for i:=4 to MsgLen do bs(URecord)[i-3]:=bs(msgbuf)[i];
   UGetRecMode:=ord(MsgBuf[3]);
end;  {  UGetRec   }
@{-------------------------------}
procedure USearchKey (var UIdxF       : UIndexFile;
                      var UDataRecNo  : LongInt;
                      var UProcKey                 );
```

```
var s1   : string[60];
    str4 : string[4];
begin
   s1:=StrStr(UProcKey)+spc60;
   s1[0]:=chr(MKeyLen);
   MsgID:=SEARCHKEYID;
   AdjLen(sizeof(UIdxF)+length(s1)+2);
   SetMsgId;
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to MkeyLen  do MsgBuf[4+sizeOf(UIdxF)+i]:=s1[i];
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=Ord(MsgBuf[3+i]);
   str4:='';
   for i:=1 to 4 do str4:=str4+bs(msgbuf)[i+2+sizeof(UIdxF)];
   UDataRecNo:=CHTOLI(str4);
   s1:='';
   for i:=1 to MKeyLen do s1:=s1+MsgBuf[6+i+sizeof(UIdxF)];
   s1[0]:=chr(UIdxF.UKeyL);
   strstr(uprockey):=s1;
end;  { USearchKey }
{-----------------------------------}
procedure UPrevKey (var UIdxF       : UIndexFile;
                    var UDataRecNo  : LongInt;
                    var UProcKey                  );
var s1   : string[60];
    str4 : string[4];
begin
   s1:=StrStr(UProcKey)+spc60;
   s1[0]:=chr(MKeyLen);
   MsgID:=PREVKEYID;
   AdjLen(sizeof(UIdxF)+length(s1)+2);
   SetMsgId;
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to MkeyLen  do MsgBuf[4+sizeOf(UIdxF)+i]:=s1[i];
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=Ord(MsgBuf[3+i]);
   str4:='';
   for i:=1 to 4 do str4:=str4+bs(msgbuf)[i+2+sizeof(UIdxF)];
   UDataRecNo:=CHTOLI(str4);
   s1:='';
   for i:=1 to MKeyLen do s1:=s1+MsgBuf[6+i+sizeof(UIdxF)];
   s1[0]:=chr(UIdxF.UKeyL);
   strstr(uprockey):=s1;
end;  { UPrevKey }
{-----------------------------------}
procedure UNextKey (var UIdxF       : UIndexFile;
```

```
                    var UDataRecNo  : LongInt;
                    var UProcKey                    );
var s1   : string[60];
    str4 : string[4];
begin
   s1:=StrStr(UProcKey)+spc60;
   s1[0]:=chr(MKeyLen);
   MsgID:=NEXTKEYID;
   str4:=LITOCH(UDataRecNo);
   AdjLen(sizeof(UIdxF)+length(s1)+length(str4)+2);
   SetMsgId;
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to MkeyLen
       do MsgBuf[4+sizeOf(UIdxF)+i]:=s1[i];
   for i:=1 to 4
       do MsgBuf[4+sizeOf(UIdxF)+MKeyLen+i]:=str4[i];
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=Ord(MsgBuf[3+i]);
   str4:='';
   for i:=1 to 4 do str4:=str4+bs(msgbuf)[i+2+sizeof(UIdxF)];
   UDataRecNo:=CHTOLI(str4);
   s1:='';
   for i:=1 to MKeyLen do s1:=s1+MsgBuf[6+i+sizeof(UIdxF)];
   s1[0]:=chr(UIdxF.UKeyL);
   strstr(uprockey):=s1;
end;  { UNextKey }
{-------------------------------}
procedure UAddKey  (var UIdxF       : UIndexFile;
                    var UDataRecNo  : LongInt;
                    var UProcKey                    );
var s1   : string[60];    { Prockey                  }
    str4 : string[4];
begin
   s1:=StrStr(UProcKey)+spc60;
   s1[0]:=chr(MKeyLen);
   MsgID:=ADDKEYID;
   str4:=LITOCH(UDataRecNo);
   AdjLen(sizeof(UIdxF)+length(s1)+Length(str4)+2);
   SetMsgId;
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to MkeyLen
       do MsgBuf[4+sizeOf(UIdxF)+i]:=s1[i];
   for i:=1 to 4
       do MsgBuf[4+sizeOf(UIdxF)+Length(s1)+i]:=str4[i];
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=Ord(MsgBuf[3+i]);
```

```
   str4:='';
   for i:=1 to 4
       do str4:=str4+bs(msgbuf)[i+2+sizeof(UIdxF)];
   UDataRecNo:=CHTOLI(str4);
end; {  UAddKey   }
{-----------------------------------}
procedure UAddRec (var UDatF      : UDataFile;
                   var UDataRecNo : LongInt;
                   var URecord                );
var str4 : string[4];
begin
   MsgID:=ADDRECID;
   AdjLen(SizeOf(UDatF)+UDatF.UItemSize+2);
   SetMsgId;
   adr1:=seg(UDatF);
   adr2:=ofs(UDatF);
   for i:=0 to sizeof(UDatF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to UDatF.UItemSize
     do bs(msgbuf)[4+SizeOf(UDatf)+i]:=bs1(URecord)[i];
   SendMsg(MsgBuf);
   RcvMsg(MsgBuf);
   str4:='';
   for i:=1 to 4 do str4:=str4+bs(msgbuf)[i+2];
   UDataRecNo:=CHTOLI(str4);
end;  {  UAddRec   }
{-----------------------------------}
procedure UPutRec (var UDatF      : UDataFile;
                   var UDataRecNo : LongInt;
                   var URecord                );
var str4 : string[4];
begin
   MsgID:=PUTRECID;
   str4:=LITOCH(UDataRecNo);
   AdjLen(SizeOf(UDatF)+UDatF.UItemSize+Length(str4)+2);
   SetMsgId;
   adr1:=seg(UDatF);
   adr2:=ofs(UDatF);
   for i:=0 to sizeof(UDatF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to UDatF.UItemSize
       do bs(msgbuf)[4+SizeOf(UDatf)+i]:=bs1(URecord)[i];
   for i:=1 to 4
       do MsgBuf[4+SizeOf(UDatF)+UDatF.UItemSize+i]:=str4[i];
   SendMsg(MsgBuf);
   RcvMsg(MsgBuf);
   str4:='';
   for i:=1 to 4 do str4:=str4+bs(msgbuf)[i+2];
   UDataRecNo:=CHTOLI(str4);
end;  {  UPutRec   }
{-----------------------------------}
procedure UDeleteKey (var UIdxF       : UIndexFile;
                      var UDataRecNo  : LongInt;
                      var UProcKey                 );
var s1   : string[60];    { Prockey                 }
    str4 : string[4];
```

```
begin
   s1:=StrStr(UProcKey)+spc60;
   s1[0]:=chr(MKeyLen);
   MsgID:=DELKEYID;
   str4:=LITOCH(UDataRecNo);
   AdjLen(sizeof(UIdxF)+length(s1)+Length(str4)+2);
   SetMsgId;
   adr1:=seg(UIdxF);
   adr2:=ofs(UIdxF);
   for i:=0 to sizeof(UIdxF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to MkeyLen
       do MsgBuf[4+sizeOf(UIdxF)+i]:=s1[i];
   for i:=1 to 4
       do MsgBuf[4+sizeOf(UIdxF)+Length(s1)+i]:=str4[i];
   SendMsg(MsgBuf);
   rcvMsg(Msgbuf);
   for i:=0 to sizeof(UIdxF)-1
       do mem[adr1:adr2+i]:=Ord(MsgBuf[3+i]);
   str4:='';
   for i:=1 to 4
       do str4:=str4+bs(msgbuf)[i+2+sizeof(UIdxF)];
   UDataRecNo:=CHTOLI(str4);
end; {  UDeleteKey   }
{-----------------------------------}
procedure UDeleteRec (var UDatF      : UDataFile;
                      var UDataRecNo : LongInt    );
var str4 : string[4];
begin
   MsgID:=DELRECID;
   str4:=LITOCH(UDataRecNo);
   AdjLen(SizeOf(UDatF)+Length(str4)+2);
   SetMsgId;
   adr1:=seg(UDatF);
   adr2:=ofs(UDatF);
   for i:=0 to sizeof(UDatF)-1
       do MsgBuf[i+5]:=chr(mem[adr1:adr2+i]);
   for i:=1 to 4 do MsgBuf[4+SizeOf(UDatF)+i]:=str4[i];
   SendMsg(MsgBuf);
   RcvMsg(MsgBuf);
end;  {  UDeleteRec   }
{-----------------------------------}
begin
end.
```

```
(*****************************************************************)
(*                                                               *)
(*                          NURINET                              *)
(*          RS-232 KOMINIKASYON YONETICI PROGRAM                 *)
(*                                                               *)
(*****************************************************************)
{$M $8000,0,200000}
{$F+}
program NURINET;
uses Crt, Dos , SRUN , SACC;
var
  MsgCount : integer;
  MsgCount2: integer;
  IntVec0C : Procedure;
  IntVec0B : Procedure;
  r        : registers;
  MsgVar   : Boolean;
  MsgVar2  : Boolean;
  c,k : char;
  i : integer;
  KL,DP : byte;
  RRN   : longint;
  str4  : string;
  str2  : string[2];
  recordtoSend : array [1..2000] of char;
  regs : registers;
  bs :   array [1..2048] of char;
  adr1,
  adr2 : word;
  tt : byte;
procedure PrepMesID;
begin
   MsgID:=MsgBuf[3]+MsgBuf[4];
end;
procedure PrepMesID2;
begin
   MsgID2:=MsgBuf2[3]+MsgBuf2[4];
end;
procedure NewInt0C; interrupt;
var
    gelen  : char;
    i      : integer;
begin
  if not(MsgVar) then
  begin
    inc(MsgCount);
    RcvChar(gelen);
    MsgBuf[MsgCount]:=gelen;
    if Msgcount=1 then MsgLen:=ord(MsgBuf[1]);
    if Msgcount=2 then
    begin
       MsgLen:=MsgLen+ord(MsgBuf[2])*sb;
    end;
    if MsgCount=MsgLen then MsgVar:=True;
  end;
    port[$20]:=$20;
```

```
inline ($9C);
IntVec0C;
if MsgVar then
 begin
    PrepMesID;
    if MsgID = OPENDFILEID then
    begin
        DataFileName:='';
        str2:='';   { Record Length }
        for i:=5 to (FNameLen+4) do DataFileName:=DataFileName+MsgBuf[i];
        for i:=1 to 2 do str2:=str2+MsgBuf[i+FNameLen+4];
        RecordLength:=ord(str2[1])+sb*ord(str2[2]);
        SOpenDataFile(SDatF,DataFileName,RecordLength);
        AdjLen(sizeof(SDatf));
        adr2:=ofs(SdatF);
        adr1:=seg(SdatF);
        for i:=0 to sizeof(Sdatf)-1
            do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
        SendMsg(MsgBuf);
    end; { end of OPENDFILEID }
    if MsgID = CLOSEDFILEID then
    begin
        adr2:=ofs(SdatF);
        adr1:=seg(SdatF);
        for i:=0 to sizeof(Sdatf)-1
            do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
        SCloseDataFile(SDatF);
        AdjLen(sizeof(SDatf));
        adr2:=ofs(SdatF);
        adr1:=seg(SdatF);
        for i:=0 to sizeof(Sdatf)-1
            do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
        SendMsg(MsgBuf);
    end; { end of CLOSEDFILEID }
    if MsgID = OPENIFILEID then
    begin
        IndxFileName:='';
        for i:=5 to (FNameLen+4)
            do IndxFileName:=IndxFileName+MsgBuf[i];
        KL:=ord(MsgBuf[5+FNameLen]);
        DP:=ord(MsgBuf[6+FNameLen]);
        SOpenIndexFile(SIdxF,IndxFileName,KL,DP);
        AdjLen(sizeof(SIdxF));
        adr1:=seg(SIdxF);
        adr2:=ofs(SIdxF);
        for i:=0 to sizeof(SIdxF)-1
            do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
        SendMsg(MsgBuf);
    end; { end of OPENIFILEID }
    if MsgID = CLOSEIFILEID then
    begin
        adr2:=ofs(SIdxF);
        adr1:=seg(SIdxF);
        for i:=0 to sizeof(SIdxf)-1
            do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
        SCloseIndexFile(SIdxF);
```

```
    AdjLen(sizeof(SIdxF));
    adr2:=ofs(SIdxF);
    adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxF)-1
        do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
    SendMsg(MsgBuf);
end; { end of CLOSEIFILEID }
if MsgID = FINDKEYID then
begin
    adr2:=ofs(SIdxF);
    adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf[4+sizeOf(SIdxF)+i];
    RRN:=0;
    SFindKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf[2+sizeof(SIdxf)+i]:=str4[i];
    AdjLen(4+SizeOf(SIdxF));
    SendMsg(MsgBuf);
end; { end of FINDKEYID }
if MsgID = GETRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF);
    adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    for i:=1 to 4
        do str4:=str4+MsgBuf[4+SizeOf(SDatF)+i];
    RRN:=CHTOLI(str4);
    SGetRecMode:=ord(MsgBuf[9+SizeOf(SDatF)]);
    SGetRec(SDatF,RRN,RecordtoSend,SGetRecMode);
    MsgBuf[3]:=chr(SGetRecMode);
    AdjLen(1);
    if SGetRecMode=1 then
     begin
       for i:=1 to SdatF.ItemSize
           do MsgBuf[3+i]:=RecordTosend[i];
       AdjLen(1+SDatF.ItemSize);
     end;
    SendMsg(MsgBuf);
    i:=0;
end; { end of GETRECID }
if MsgID = SEARCHKEYID then
begin
    adr1:=seg(SIdxF);
    adr2:=ofs(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
```

```
        do IndxKey:=IndxKey+MsgBuf[4+sizeOf(SIdxF)+i];
    RRN:=0;
    SSearchKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf[2+sizeof(SIdxf)+i]:=str4[i];
    for i:=1 to MKeyLen
        do MsgBuf[6+sizeof(SIdxf)+i]:=IndxKey[i];
    AdjLen(4+MKeyLen+SizeOf(SIdxF));
    SendMsg(MsgBuf);
end; { end of SEARCHKEYID }
if MsgID = PREVKEYID then
begin
    adr2:=ofs(SIdxF);
    adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf[4+sizeOf(SIdxF)+i];
    RRN:=0;
    SPrevKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf[2+sizeof(SIdxf)+i]:=str4[i];
    for i:=1 to MKeyLen
        do MsgBuf[6+sizeof(SIdxf)+i]:=IndxKey[i];
    AdjLen(4+MKeyLen+SizeOf(SIdxF));
    SendMsg(MsgBuf);
end; { end of PREVKEYID }
if MsgID = NEXTKEYID then
begin
    adr2:=ofs(SIdxF);
    adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf[4+sizeOf(SIdxF)+i];
    str4:='';
    for i:=1 to 4
        do str4:=str4+MsgBuf[4+SizeOf(SIdxF)+MKeyLen+i];
    RRN:=CHTOLI(str4);
    SNextKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf[2+sizeof(SIdxf)+i]:=str4[i];
    for i:=1 to MKeyLen
        do MsgBuf[6+sizeof(SIdxf)+i]:=IndxKey[i];
    AdjLen(4+MKeyLen+SizeOf(SIdxF));
    SendMsg(MsgBuf);
end; { end of NEXTKEYID }
if MsgID = ADDKEYID then
```

```
begin
    adr2:=ofs(SIdxF);
    adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf[4+sizeOf(SIdxF)+i];
    str4:='';
    for i:=1 to 4
        do str4:=str4+MsgBuf[4+SizeOf(SIdxF)+MKeylen+i];
    RRN:=CHTOLI(str4);
    SAddKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4
        do MsgBuf[2+sizeof(SIdxf)+i]:=str4[i];
    AdjLen(4+SizeOf(SIdxF));
    SendMsg(MsgBuf);
end; { end of ADDKEYID }
if MsgID = ADDRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF);
    adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    for i:=1 to SdatF.ItemSize
        do Recordtosend[i]:=MsgBuf[4+sizeOf(Sdatf)+i];
    SAddRec(SDatF,RRN,RecordtoSend);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf[2+i]:=str4[i];
    AdjLen(4);
    SendMsg(MsgBuf);
    i:=0;
end; { end of ADDRECID }
if MsgID = PUTRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF);
    adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    for i:=1 to SdatF.ItemSize
        do Recordtosend[i]:=MsgBuf[4+sizeOf(Sdatf)+i];
    for i:=1 to 4
        do str4:=str4+MsgBuf[4+SizeOf(SDatF)+SDatF.ItemSize+i];
    RRN:=CHTOLI(str4);
    SPutRec(SDatF,RRN,RecordtoSend);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf[2+i]:=str4[i];
    AdjLen(4);
    SendMsg(MsgBuf);
end; { end of PUTRECID }
if MsgID = DELKEYID then
```

```
begin
    adr2:=ofs(SIdxF);
    adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf[4+sizeOf(SIdxF)+i];
    str4:='';
    for i:=1 to 4
        do str4:=str4+MsgBuf[4+SizeOf(SIdxF)+MKeylen+i];
    RRN:=CHTOLI(str4);
    SDeleteKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf[2+sizeof(SIdxf)+i]:=str4[i];
    AdjLen(4+SizeOf(SIdxF));
    SendMsg(MsgBuf);
end; { end of DELKEYID }
if MsgID = DELRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF);
    adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf[i+5]);
    for i:=1 to 4 do str4:=str4+MsgBuf[4+SizeOf(SDatF)+i];
    RRN:=CHTOLI(str4);
    SDeleteRec(SDatF,RRN);
    AdjLen(0);
    SendMsg(MsgBuf);
end; { end of DELRECID }
if MsgID  = MESSAGEID then
begin
    kullanici:='';
    for i:=1 to 30 do kullanici:=kullanici+MsgBuf[4+i];
    for j:=1 to 6 do
     begin
      for i:=1 to 76 do xmsg[j][i]:=MsgBuf[34+((j-1)*76)+i];
      xmsg[j][0]:=chr(76);
     end;
    {$I-}
    assign(tf,MessageFileName);
    append(tf);
    close(TF);
    {$I+}
    if IOresult<>0 then
     begin
       assign(tf,MessageFileName);
       rewrite(tf);
       close(TF);
     end;
    assign(tf,MessageFileName);
    append(tf);
    writeln(tf,cizgi80);
```

```
            writeln(tf,'TARiH ... :',MesajTarihi);
            writeln(tf,'SAAT .... :',mesajsaati);
            writeln(tf,'KULLANICI :',kullanici);
            for i:=1 to 6 do writeln(tf,xmsg[i]);
            writeln(tf,cizgi80);
            close(tf);
            AdjLen(0);
            SendMsg(MsgBuf);
        end; { end of MESSAGEID }
        MsgVar:=False;
        MsgCount:=0;
    end;
end;
{*********************************************************************}
procedure NewIntOB; interrupt;
var
    gelen  : char;
    i      : integer;
begin
  if not(MsgVar2) then
  begin
    inc(MsgCount2);
    RcvChar(gelen);
    MsgBuf2[MsgCount2]:=gelen;
    if Msgcount2=1 then MsgLen2:=ord(MsgBuf2[1]);
    if Msgcount2=2 then
    begin
       MsgLen2:=MsgLen2+ord(MsgBuf2[2])*sb;
    end;
    if MsgCount2=MsgLen2 then MsgVar2:=True;
  end;
    port[$20]:=$20;
    inline ($9C);
    IntVecOB;
    if MsgVar2 then
     begin
        PrepMesID2;
        if MsgID2  = OPENDFILEID then
        begin
            DataFileName:='';
            str2:='';    { Record Length }
            for i:=5 to (FNameLen+4)
                do DataFileName:=DataFileName+MsgBuf2[i];
            for i:=1 to 2 do str2:=str2+MsgBuf2[i+FNameLen+4];
            RecordLength:=ord(str2[1])+sb*ord(str2[2]);
            SOpenDataFile(SDatF,DataFileName,RecordLength);
            AdjLen(sizeof(SDatf));
            adr2:=ofs(SdatF);  adr1:=seg(SdatF);
            for i:=0 to sizeof(Sdatf)-1
                do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
            SendMsg(MsgBuf2);
        end; { end of OPENDFILEID }
        if MsgID2  = CLOSEDFILEID then
        begin
            adr2:=ofs(SdatF);  adr1:=seg(SdatF);
            for i:=0 to sizeof(Sdatf)-1
```

```
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    SCloseDataFile(SDatF);
    AdjLen(sizeof(SDatf));
    for i:=0 to sizeof(Sdatf)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    SendMsg(MsgBuf2);
end; { end of CLOSEDFILEID }
if MsgID2  = OPENIFILEID then
begin
    IndxFileName:='';
    for i:=5 to (FNameLen+4)
        do IndxFileName:=IndxFileName+MsgBuf2[i];
    KL:=ord(MsgBuf2[5+FNameLen]);
    DP:=ord(MsgBuf2[6+FNameLen]);
    SOpenIndexFile(SIdxF,IndxFileName,KL,DP);
    AdjLen(sizeof(SIdxF));
    adr1:=seg(SIdxF); adr2:=ofs(SIdxF);
    for i:=0 to sizeof(SIdxF)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    SendMsg(MsgBuf2);
end; { end of OPENIFILEID }
if MsgID2  = CLOSEIFILEID then
begin
    adr2:=ofs(SIdxF);  adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    SCloseIndexFile(SIdxF);
    AdjLen(sizeof(SIdxF));
    adr2:=ofs(SIdxF);  adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxF)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    SendMsg(MsgBuf2);
end; { end of CLOSEIFILEID }
if MsgID2  = FINDKEYID then
begin
    adr2:=ofs(SIdxF); adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf2[4+sizeOf(SIdxF)+i];
    RRN:=0;
    SFindKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf2[2+sizeof(SIdxf)+i]:=str4[i];
    AdjLen(4+SizeOf(SIdxF));
    SendMsg(MsgBuf2);
end; { end of FINDKEYID }
if MsgID2  = GETRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF); adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
```

```
    for i:=1 to 4 do str4:=str4+MsgBuf2[4+SizeOf(SDatF)+i];
    RRN:=CHTOLI(str4);
    SGetRecMode:=ord(MsgBuf2[9+SizeOf(SDatF)]);
    SGetRec(SDatF,RRN,RecordtoSend,SGetRecMode);
    MsgBuf2[3]:=chr(SGetRecMode);
    AdjLen(1);
    if SGetRecMode=1 then
     begin
       for i:=1 to SdatF.ItemSize
           do MsgBuf2[3+i]:=RecordTosend[i];
       AdjLen(1+SDatF.ItemSize);
     end;
    SendMsg(MsgBuf2);
    i:=0;
end; { end of GETRECID }
if MsgID2  = SEARCHKEYID then
begin
    adr1:=seg(SIdxF); adr2:=ofs(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf2[4+sizeOf(SIdxF)+i];
    RRN:=0;
    SSearchKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf2[2+sizeof(SIdxf)+i]:=str4[i];
    for i:=1 to MKeyLen
        do MsgBuf2[6+sizeof(SIdxf)+i]:=IndxKey[i];
    AdjLen(4+MKeyLen+SizeOf(SIdxF));
    SendMsg(MsgBuf2);
end; { end of SEARCHKEYID }
if MsgID2  = PREVKEYID then
begin
    adr2:=ofs(SIdxF); adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf2[4+sizeOf(SIdxF)+i];
    RRN:=0;
    SPrevKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf2[2+sizeof(SIdxf)+i]:=str4[i];
    for i:=1 to MKeyLen
        do MsgBuf2[6+sizeof(SIdxf)+i]:=IndxKey[i];
    AdjLen(4+MKeyLen+SizeOf(SIdxF));
    SendMsg(MsgBuf2);
end; { end of PREVKEYID }
if MsgID2  = NEXTKEYID then
begin
    adr2:=ofs(SIdxF);
```

```
    adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1 do
     begin
        mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
     end;
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf2[4+sizeOf(SIdxF)+i];
    str4:='';
    for i:=1 to 4
        do str4:=str4+MsgBuf2[4+SizeOf(SIdxF)+MKeyLen+i];
    RRN:=CHTOLI(str4);
    SNextKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4
        do MsgBuf2[2+sizeof(SIdxf)+i]:=str4[i];
    for i:=1 to MKeyLen
        do MsgBuf2[6+sizeof(SIdxf)+i]:=IndxKey[i];
    AdjLen(4+MKeyLen+SizeOf(SIdxF));
    SendMsg(MsgBuf2);
end; { end of NEXTKEYID }
if MsgID2  = ADDKEYID then
begin
    adr2:=ofs(SIdxF); adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf2[4+sizeOf(SIdxF)+i];
    str4:='';
    for i:=1 to 4
        do str4:=str4+MsgBuf2[4+SizeOf(SIdxF)+MKeylen+i];
    RRN:=CHTOLI(str4);
    SAddKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf2[2+sizeof(SIdxf)+i]:=str4[i];
    AdjLen(4+SizeOf(SIdxF));
    SendMsg(MsgBuf2);
end; { end of ADDKEYID }
if MsgID2  = ADDRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF);  adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    for i:=1 to SdatF.ItemSize
        do Recordtosend[i]:=MsgBuf2[4+sizeOf(Sdatf)+i];
    SAddRec(SDatF,RRN,RecordtoSend);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf2[2+i]:=str4[i];
    AdjLen(4);
    SendMsg(MsgBuf2);
```

```
end; { end of ADDRECID }
if MsgID2  = PUTRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF); adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    for i:=1 to SdatF.ItemSize
       .do Recordtosend[i]:=MsgBuf2[4+sizeOf(Sdatf)+i];
    for i:=1 to 4
        do str4:=str4+MsgBuf2[4+SizeOf(SDatF)+SDatF.ItemSize+i];
    RRN:=CHTOLI(str4);
    SPutRec(SDatF,RRN,RecordtoSend);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf2[2+i]:=str4[i];
    AdjLen(4);
    SendMsg(MsgBuf2);
end; { end of PUTRECID }
if MsgID2  = DELKEYID then
begin
    adr2:=ofs(SIdxF); adr1:=seg(SIdxF);
    for i:=0 to sizeof(SIdxf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    IndxKey:='';
    for i:=1 to MKeylen
        do IndxKey:=IndxKey+MsgBuf2[4+sizeOf(SIdxF)+i];
    str4:='';
    for i:=1 to 4
        do str4:=str4+MsgBuf2[4+SizeOf(SIdxF)+MKeylen+i];
    RRN:=CHTOLI(str4);
    SDeleteKey(SIdxF,RRN,IndxKey);
    for i:=0 to sizeof(SIdxf)-1
        do MsgBuf2[3+i]:=chr(mem[adr1:adr2+i]);
    str4:=LITOCH(RRN);
    for i:=1 to 4 do MsgBuf2[2+sizeof(SIdxf)+i]:=str4[i];
    AdjLen(4+SizeOf(SIdxF));
    SendMsg(MsgBuf2);
end; { end of DELKEYID }
if MsgID2  = DELRECID then
begin
    str4:='';   { RRN }
    adr2:=ofs(SdatF); adr1:=seg(SdatF);
    for i:=0 to sizeof(Sdatf)-1
        do mem[adr1:adr2+i]:=ord(MsgBuf2[i+5]);
    for i:=1 to 4
        do str4:=str4+MsgBuf2[4+SizeOf(SDatF)+i];
    RRN:=CHTOLI(str4);
    SDeleteRec(SDatF,RRN);
    AdjLen(0);
    SendMsg(MsgBuf2);
end; { end of DELRECID }
if MsgID2  = MESSAGEID then
begin
    kullanici:='';
    for i:=1 to 30 do kullanici:=kullanici+MsgBuf2[4+i];
    for j:=1 to 6 do
```

```
                begin
                 for i:=1 to 76
                     do xmsg2[j][i]:=MsgBuf2[34+((j-1)*76)+i];
                 xmsg2[j][0]:=chr(76);
                end;
               {$I-}
               assign(tf,MessageFileName);
               append(tf);
               close(TF);
               {$I+}
               if IOresult<>0 then
                begin
                  assign(tf,MessageFileName);
                  rewrite(tf);
                  close(TF);
                end;
               assign(tf,MessageFileName);
               append(tf);
               writeln(tf,cizgi80);
               writeln(tf,'TARiH ... :',MesajTarihi);
               writeln(tf,'SAAT .... :',mesajsaati);
               writeln(tf,'KULLANICI :',kullanici);
               for i:=1 to 6 do writeln(tf,xmsg2[i]);
               writeln(tf,cizgi80);
               close(tf);
               AdjLen(0);
               SendMsg(MsgBuf2);
           end; { end of MESSAGEID }
           MsgVar2:=False;
           MsgCount2:=0;
       end;
end;
{$F-}
{**********************************************************************}
begin
  clrscr;
  MsgCount:=0;
  MsgVar:=False;
  Msglen:=0;
  MsgCount2:=0;
  MsgVar2:=False;
  Msglen2:=0;
  GetIntVec($0C,@IntVec0C);
  SetIntVec($0C,Addr(NewInt0C));
  GetIntVec($0B,@IntVec0B);
  SetIntVec($0B,Addr(NewInt0B));
  c:=#0;
  regs.ah:=$0;
  regs.dx:=$0;
  regs.al:=$E3;
  intr($14,regs);
  i:=port[$3f8];
  i:=port[$2f8];
  Keep(0);
end.
```

```
(*****************************************************************)
(*                            SACC                               *)
(*                                                               *)
(*      ANA MAKINEDE VERI TABANINA ULAŞIMI SAĞLAYAN PROGRAM      *)
(*                                                               *)
(*****************************************************************)
UNIT SACC;
INTERFACE
uses  CRT, DOS, SRUN , TACCESS;
type
     StrStr = string[255];
var SDatF : DataFile;
    SIdxF : IndexFile;
type
  SDataFile  =  record
                  SF          : file;
                  SFirstFree,
                  SNumberFree,
                  SInt1       : LongInt;
                  SItemSize   : word;
                  SNumRec     : LongInt;
                end;
var
     IFile : IndexFile;
     DFile : DataFile;
     IFRRN : longint;
     SGetRecMode : Byte;
     c  : char;
     i  : integer;
     Reg: Registers;
procedure SOpenDataFile( var SDatFF : DataFile;
                         var SDataFName  ;
                         var SRecLen     : word );
procedure SCloseDataFile( var SDatFF : DataFile);
procedure SOpenIndexFile(var IdxF   : IndexFile;
                             FName  : FileName;
                             KL,
                             DP     : byte);
procedure SCloseIndexFile( var SIdxFF : IndexFile);
procedure SFindKey (var SIdxFF       : IndexFile;
                    var SDataRecNum  : LongInt;
                    var SProcKey                 );
procedure SAddKey  (var SIdxFF       : IndexFile;
                    var SDataRecNum  : LongInt;
                    var SProcKey                 );
procedure SGetRec (var SDatFF       : DataFile;
                   var SDataRecNum : LongInt ;
                   var SRecord               ;
                   var SGetRecMode : Byte      );
procedure SSearchKey  (var SIdxFF       : IndexFile;
                       var SDataRecNum  : LongInt;
                       var SProcKey                 );
procedure SPrevKey  (var SIdxFF       : IndexFile;
                     var SDataRecNum  : LongInt;
                     var SProcKey                 );
procedure SNextKey  (var SIdxFF       : IndexFile;
```

```
                    var SDataRecNum  : LongInt;
                    var SProcKey                );
procedure SAddRec (var SDatFF      : DataFile;
                   var SDataRecNum : LongInt ;
                   var SRecord                 );
procedure SPutRec (var SDatFF      : DataFile;
                   var SDataRecNum : LongInt ;
                   var SRecord                 );
procedure SDeleteKey (var SIdxFF       : IndexFile;
                      var SDataRecNum  : LongInt;
                      var SProcKey               );
procedure SDeleteRec (var SDatFF      : DataFile;
                      var SDataRecNum : LongInt  );
implementation
{-------------------------------}
procedure  SOpenIndex(Var IndexFileName;
                      var KL,DP : byte);
var s2    : string[64];
begin
   s2:=StrStr(IndexFileName);
   OpenIndex(IFile,s2,KL,DP);
end;
{-------------------------------}
procedure  SOpenFile(Var DataFileName;
                     var RecLen : word);
var s2    : string[64];
begin
   s2:=StrStr(DataFileName);
   OpenFile(DFile,s2,RecLen);
end;
{-------------------------------}
procedure SOpenDataFile( var SDatFF : DataFile;
                         var SDataFName  ;
                         var SRecLen     : word );
var s2    : string[64];
begin
   s2:=StrStr(SDataFName);
   OpenFile(SDatFF,S2,SrecLen);
end;  { SOpenDataFile }
{-------------------------------}
procedure SCloseDataFile( var SDatFF : DataFile);
begin
   CloseFile(SDatFF);
end;  { SCloseDataFile }
{-------------------------------}
procedure SOpenIndexFile(var IdxF   : IndexFile;
                             FName  : FileName;
                             KL,
                             DP     : byte);
begin
    OpenIndex(IdxF,Fname,KL,DP);
end; { SOpenIndexFile }
{-------------------------------}
procedure SCloseIndexFile( var SIdxFF : IndexFile);
begin
   CloseIndex(SIdxFF);
```

```
end; { SCloseIndexFile }
{-----------------------------------}
procedure SFindKey (var SIdxFF       : IndexFile;
                    var SDataRecNum  : LongInt;
                    var SProcKey                );
begin
     FreshIndex(SIdxFF);
     FindKey(SIdxFF,SDataRecNum,SProcKey);
     if Not(OK) then SDataRecNum:=0;
end; { SFindKey   }
{-----------------------------------}
procedure SAddKey  (var SIdxFF       : IndexFile;
                    var SDataRecNum  : LongInt;
                    var SProcKey                );
begin
     AddKey(SIdxFF,SDataRecNum,SProcKey);
     FlushIndex(SIdxFF);
     if Not(OK) then SDataRecNum:=0;
end; { SAddKey   }
{-----------------------------------}
Function KayitKilit ( FileHandle : Byte;
                      RecordLength: integer;
                      RecordNo   : Longint   ) : word;
begin
   KayitKilit:=0;
   Reg.AH:=$5C;
   Reg.AL:=$00;
   Reg.BX:=FileHandle;
   Reg.CX:=$00;
   Reg.DX:=((RecordNo-1)*RecordLength)+1;
   Reg.SI:=0;
   Reg.DI:=RecordLength;
   MsDos(Reg);
   Wordayristir(Reg.Flags);
   if bsw[0]=1 then KayitKilit:=Reg.Ax;
end;
{-----------------------------------}
Procedure KayitBirak ( FileHandle : Byte;
                       RecordLength: integer;
                       RecordNo   : Longint   ) ;
begin
   Reg.AH:=$5C;
   Reg.AL:=$01;
   Reg.BX:=FileHandle;
   Reg.CX:=$00;
   Reg.DX:=((RecordNo-1)*RecordLength)+1;
   Reg.SI:=0;
   Reg.DI:=RecordLength;
   MsDos(Reg);
end;
{-----------------------------------}
Function KayitKilitlimi( FileHandle : Byte;
                         RecordLength: integer;
                         RecordNo   : Longint   ) : word;
var sonuc : word;
begin
```

```
    sonuc:=KayitKilit(FileHandle,RecordLength,RecordNo);
    KayitKilitlimi:=sonuc;
    if Sonuc=0 then KayitBirak(FileHandle,
                               RecordLength,RecordNo);
end;
{-----------------------------------}
procedure SGetRec(var SDatFF      : DataFile;
                  var SDataRecNum : LongInt ;
                  var SRecord               ;
                  var SGetRecMode : Byte      );
var Sonuc : word;
begin
   Sonuc:=KayitKilitlimi ( FileRec(SDatFF.f).Handle ,
                           SdatF.ItemSize , SDataRecNum );
   if sonuc=0 then
    begin
       if SgetRecMode=1 then
         begin
           sonuc:=KayitKilit ( FileRec(SDatFF.f).Handle ,
                               SdatF.ItemSize , SDataRecNum );
           if sonuc=0 then SGetRecMode:=1 else SGetRecMode:=0;
         end
         else SGetRecMode:=1;
       GetRec(SDatFF,SDataRecNum,SRecord);
    end else SGetRecMode:=0;
end;
{-----------------------------------}
procedure SSearchKey  (var SIdxFF       : IndexFile;
                       var SDataRecNum  : LongInt;
                       var SProcKey                );
begin
     SearchKey(SIdxFF,SDataRecNum,SProcKey);
     if Not(OK) then SDataRecNum:=0;
end; { SSearchKey   }

{-----------------------------------}
procedure SPrevKey  (var SIdxFF       : IndexFile;
                     var SDataRecNum  : LongInt;
                     var SProcKey               );
begin
     PrevKey(SIdxFF,SDataRecNum,SProcKey);
     if Not(OK) then SDataRecNum:=0;
end; { SPrevKey   }
{-----------------------------------}
procedure SNextKey  (var SIdxFF       : IndexFile;
                     var SDataRecNum  : LongInt;
                     var SProcKey               );
var s1 : string[60];
begin
     s1:=StrStr(SProcKey);
     NextKey(SIdxFF,SDataRecNum,S1);
     if Not(OK) then SDataRecNum:=0;
     strstr(sProcKey):=s1;
end; { SNextKey   }
{-----------------------------------}
procedure SAddRec (var SDatFF      : DataFile;
```

```
                    var SDataRecNum : LongInt ;
                    var SRecord                );
begin
    AddRec(SDatFF,SDataRecNum,Srecord);
    FlushFile(SDatFF);
end; { SAddRec  }
{-----------------------------------}
procedure SPutRec (var SDatFF      : DataFile;
                   var SDataRecNum : LongInt ;
                   var SRecord                );
begin
    KayitBirak(FileRec(SDatFF.f).Handle ,
               SdatF.ItemSize , SDataRecNum );
    PutRec(SDatFF,SDataRecNum,Srecord);
end; { SPutRec  }
{-----------------------------------}
procedure SDeleteKey (var SIdxFF       : IndexFile;
                      var SDataRecNum  : LongInt;
                      var SProcKey               );
begin
     DeleteKey(SIdxFF,SDataRecNum,SProcKey);
     if Not(OK) then SDataRecNum:=0;
end; { SDeleteKey   }
{-----------------------------------}
procedure SDeleteRec (var SDatFF      : DataFile;
                      var SDataRecNum : LongInt   );
begin
    DeleteRec(SDatFF,SDataRecNum);
end; { SDeleteRec  }
{-----------------------------------}
begin
end.
```

```
(*************************************************************************)
(*                              MESAJ                                    *)
(*                                                                       *)
(*              ANA MAKINEYE MESAJ YOLLAYAN PROGRAM                      *)
(*                                                                       *)
(*************************************************************************)
program MESAJ;
uses crt,UACC,SRUN,util01;
const
   retset : set of  char = [return,esc,up,down,F10];
   renk1  = 126;
   renk2  = 4;
var
   c : char;
   i : integer;
   cik1,
   cik2 : boolean;
   AN : byte;
   Kullanici : string[30];
   xmsg   : array [1..6] of string[76];
   cevap : array [1..6] of string[76];
{-------------}
procedure ekr01;
begin
   clrscr;
   CreateWindow(1,1,80,3,2,2);
   CreateWindow(1,4,80,14,2,2);
   outstr('ANA MAKINAYA MESAJ YOLLAMA',2,26,3);
   outstr('Kullanici ismi.....:',5,4,5);
   outstr('Gonderilecek mesaj',7,20,5);
   for i:=8 to 13 do
   begin
      outstr(chr(16),i, 2,5);
      outstr(chr(17),i,79,5);
   end;
end; { ekr01 }
{-------------}
procedure MesajGonder;
var k : byte;
begin
   MsgID:=MESSAGEID;
   AdjLen(30+(sizeof(xmsg)-6)+2);
   SetMsgId;
   for i:=1 to 30 do Msgbuf[4+i]:=Kullanici[i];
   for k:=1 to 6 do
      for i:=1 to 76 do MsgBuf[34+((k-1)*76)+i]:=xmsg[k][i];
   SendMsg(MsgBuf);
   RcvMsg(MsgBuf);
   Msglen:=ord(msgbuf[1])+ord(msgbuf[2])*sb;
   Outstr('Mesaj Yollandi !...',15,30,5);
end; { MesajGonder }
{-------------}
begin
   ekr01;
   for i:=1 to 7 do xmsg[i]:=space(76);
   kullanici:=space(30);
```

```
   cik1:=false;
   AN:=1;
   repeat
     case AN of
       1 : begin
           Readstr(Kullanici,5,26,renk1,renk2,retset,ret,' ',' ');
           end;
       2..8 :
           begin
           Readstr(xmsg[AN-1],6+AN,3,renk1,renk2,retset,ret,' ',' ');
           end;
     end; { end of case AN }
     case ret of
        esc,
        F10    : cik1:=True;
        return,
        down   : if AN < 7 then inc(AN) else AN:=1;
        up     : if AN > 1 then dec(AN) else AN:=7;
     end; { end of case ret }
   until cik1;
   if ret=F10 then MesajGonder;
end.
```

6. ÖRNEK UYGULAMA PROGRAMLARI

Bu ağ yazılımını test eden programlar iki farklı çeşitte yapılmıştır. Yani ana makinadaki uygulama programı ile kullanıcı makinadaki uygulama programı farklıdır. Çünkü birinde veri tabanı doğrudan kullanılırken diğerinde seri portlar vasıtasıyla kullanılmaktadır. Dolayısıyla kullanıcı makinadaki uygulama programında "UACC" yazılımındaki komutlar kullanılmıştır. Ana makinadaki uygulama yazılımında ise "TACCESS" adlı yazılımın komutları doğrudan kullanılmıştır.

6.1. ANA MAKINADAKI UYGULAMA PROGRAMI

Bu program küçük bir adres programıdır. Dört hane alfanümerik bir kodla isim adres ve telefon numarasını "KART.DAT" adlı bir veri dosyasında tutar. "KART.IND" adlı dosya indeksli dosya olarak kullanılır. Bu program ile yeni bir kayıt yapılabilir, daha önceden girilmiş bir kaydı düzeltilebilir veya bu kayıt iptal edilebilir. Yine aynı program ile küçük bir çerçevede daha önce girilmiş kodlar ve isimler ekrana getirilerek bunlardan biri seçilebilir. Düzeltme yapmak üzere ekrana bir kayıt getirildiğinde o kayıt düzeltme işlemi bitene kadar kilitli olarak kalır. Başka bir kullanıcı bu durumdayken aynı kayıtı kullanamaz.

6.1. KULLANICILARDAKI UYGULAMA PROGRAMI

Bu program ana makinada yazılan aslından çok az farklıdır. Sadece kayıt, indeks, ve dosya işlemleri ile ilgili komutlar "U" harfiyle başlamaktadır. Böylelikle ana makina için yazılmış bir uygulama programını kullanıcı makina için uyarlamak hayli kolay bir iş haline getirilmiştir. Yine aynı ana makinada olduğu gibi düzeltme için bir kayıt ekrana getirildiğinde başka bir kullanıcı bu kayda ulaşamaz.

6.3. UYGULAMA PROGRAMLARI DÖKÜMLERI

```
(*****************************************************************)
(*                          ORNEK                                *)
(*                                                               *)
(* ANA MAKINADA VERI TABANI KULLANAN BIR ORNEK UYGULAMA PROGRAMI *)
(*                                                               *)
(*****************************************************************)
program ORNEK;
uses dos,crt,sacc,taccess,util01,srun;
const
   retset : set of  char = [return,esc,up,down];
   renk1  = 126; renk2  = 4;
var
   reg : registers;
    c : char;
    i : byte;
    ret : char;
    GetRecMode: byte;
    adr1 , adr2, adr3, adr4 : word;
    KilitliRecordNo : Longint;
Type  RehberKayit =  Record
                       stat     : longint;
                       Kod      : string[4];
                       Ad       : string[20];
                       Adres1   : string[20];
                       Adres2   : string[20];
                       Tel      : string[15];
                     end;
var
   DF1    : DataFile;
   DF2    : DataFile;
   IF1    : IndexFile;
   REHREC : rehberKayit;
   REHRN  : longint;
   REHKEY : string[4];
   DFName,IFName : string[12];
   AN     : Byte;
   KonX, KonY   : Array[1..5] of byte;
   TamamenCIk,
   KayitVar : Boolean;
   TMP_K    : array [1..9] of string[4];
   TMP_A    : array [1..9] of string[20];
   TMP_R    : array [1..9] of LongInt;
procedure FileAc;
begin
   DFName:='KART.dat'; IFName:='KART.ind';
   openfile(DF1,DFName,SizeOf(REHREC));
   if ok then openindex(IF1,IFName,4,0)
      else
      begin
```

```
         makefile (DF1,DFName,SizeOf(REHREC));
         makeindex(IF1,IFName,4,0);
      end;
end; { FileAc }
{--------------------}
Procedure ekran01;
begin
   clrscr;
   CreateWindow(1,4,80,20,2,2);
   CreateWindow(27,1,48,3,2,2);
   gotoxy(29,2);
   write('KARTOTEKS PROGRAMI');
   NormVideo;
   gotoxy(10,6);  write('KOD.....:');
   gotoxy(10,9);  write('AD SOYAD.:');
   gotoxy(10,11); write('ADRES.1..:');
   gotoxy(10,13); write('ADRES 2..:');
   gotoxy(10,15); write('TELEFON..:');
end;
{--------------------}
procedure konumlar;
var i : byte;
begin
 for i:=1 to 5 do KonX[i]:=21;
 KonY[1]:=6;
 KonY[2]:=9;
 KonY[3]:=11;
 KonY[4]:=13;
 KonY[5]:=15;
end;  { konumlar }
{--------------------}
procedure kayitbosla;
begin
   with Rehrec do
   begin
      stat  := 0;
      Kod   :=space(4);
      Ad    :=space(20);
      Adres1:=space(20);
      Adres2:=space(20);
      Tel   :=space(15);
   end;
end;
{--------------------}
procedure IptalTusu;
begin
    Gotoxy(2,23);
    if KayitVar then write('F6 = iptal')
                else write('          ');
end; {  IptalTusu  }
{--------------------}
procedure KayitGoster;
begin
   With RehRec do
   begin
      outstr(Ad,KonY[2],KonX[2],renk2);
```

```
      outstr(Adres1,KonY[3],KonX[3],renk2);
      outstr(Adres2,KonY[4],KonX[4],renk2);
      outstr(Tel,KonY[5],KonX[5],renk2);
   end;
end;
{-----------------------------------}
procedure SSGetRec(var SDatFF      : DataFile;
                   var SDataRecNum : LongInt ;
                   var SRecord               ;
                   var SGetRecMode : Byte      );
var Sonuc : word;
begin
   Sonuc:=KayitKilitlimi ( FileRec(SDatFF.f).Handle ,
                           SdatFF.ItemSize , SDataRecNum );
   if sonuc=0 then
    begin
       if SgetRecMode=1 then
         begin
           sonuc:=KayitKilit ( FileRec(SDatFF.f).Handle ,
                               SdatFF.ItemSize , SDataRecNum );
           if sonuc=0 then SGetRecMode:=1 else SGetRecMode:=0;
         end;
       FreshFile(SDatFF);
       GetRec(SDatFF,SDataRecNum,SRecord);
    end else SGetRecMode:=0;
end;
{-----------------------------------}
procedure SerbestBirakKilitliyi;
begin
   KayitBirak( FileRec(DF1.f).Handle ,
               DF1.ItemSize , KilitliRecordNo);
   KilitliRecordNo:=0;
end;
{-----------------------------------}
procedure mesaj9;
begin
   outstr('Bu kayit baska bir kullanici tarafindan kullaniliyor !...',
          21,2,renk2);
   c:=readkey;
   outstr(space(70),21,2,renk2);
   AN:=1;
   ret:=#0;
end;
{--------------------}
procedure kayitbak;
begin
   retset:=retset-[F6];
   Kayitvar:=False;
   REHKEY:=RehRec.Kod;
   FreshIndex(IF1);
   FindKey(IF1,REHRN,REHKEY);
   if OK then
     begin
       KayitVar:=True;
       GetRecMode:=1;
       SSGetRec(DF1,REHRN,REHREC,GetRecMode);
```

```
        if GetRecMode=1 then
         begin
            KilitliRecordno:=REHRN;
            retset:=retset+[F6];
            KayitGoster;
            IptalTusu;
         end
         else mesaj9;
      end
      else begin KayitBosla; RehRec.Kod:=REHKEY; end;
end;
{--------------------}
procedure mesaj1;
begin
   gotoxy(10,22);
   if Kayitvar then
            write('Duzeltilen bilgiler dogru mu ? [E/H]...[ ]')
      else  write('Girilen bilgiler dogru mu  ?  [E/H]....[ ]');
   gotoxy(50,22);
end;
{--------------------}
procedure mesaj2;
begin
   gotoxy(10,22);
   write('Silmek istiyor musunuz ?   [E/H].......[ ]');
   gotoxy(50,22);
end;
{--------------------}
procedure MesajTemizle;
begin
   gotoxy(10,22);
   write('                                                  ');
   AN:=1;
end;
{--------------------}
procedure KayitYap;
begin
  if KayitVar then
   begin
     KayitBirak( FileRec(DF1.f).Handle , DF1.ItemSize , REHRN);
     PutRec(DF1,REHRN,RehRec);
   end
   else { Eger kayit yoksa }
   begin
     AddRec(DF1,REHRN,RehRec);
     KayitBirak( FileRec(DF1.f).Handle , DF1.ItemSize , REHRN);
     AddKey(IF1,REHRN,REHKEY);
     KAyitvar:=True;
   end;
   KayitBosla;
   AN:=1;
   Ekran01;
end;  { KayitYap }
{--------------------}
procedure KayitSil;
begin
```

```
   DeleteKey(IF1,REHRN,REHKEY);
   if OK then DeleteRec(DF1,REHRN)
      else
      begin
         gotoxy(10,22);
         write('Silme islemi basarisiz oldu !....');
         c:=readkey;
      end;
    mesajtemizle;
    kayitbosla;
    KayitGoster;
    AN:=1;
    Kayitvar:=False;
    IptalTusu;
end;
{--------------------}
procedure SecEkran1;
begin
   CreateWindow(50,6,76,18,7,2);
   outstr('╟──────────────────────╢',8,50,7);
   gotoxy(51,7); write('KODU      ADI    SOYADI    ');
   for i:=9 to 17 do
     begin
       gotoxy(51,i);
       write('                          ');
     end;
end;
{--------------------}
procedure TmpTemizle;
begin
   for i:=1 to 9 do
    begin
       TMP_K[i]:=space(4);
       TMP_A[i]:=space(20);
       TMP_R[i]:=0;
    end;
end;
{--------------------}
function ileriDoldur(REHKEY : string) : byte;
begin
   TmpTemizle;
   i:=0;
   SearchKey(IF1,REHRN,REHKEY);
   if OK then
    repeat
      inc(i);
      GetRec(DF1,REHRN,REHREC);
      TMP_K[i]:=RehRec.Kod;
      TMP_A[i]:=RehRec.Ad;
      TMP_R[i]:=REHRN;
      NextKey(IF1,REHRN,REHKEY);
    until (not(OK) or (i=9));
   ileriDoldur:=i;
end;
{--------------------}
function GeriDoldur(REHKEY : string) : byte;
```

```
begin
   TmpTemizle;
   i:=10;
   SearchKey(IF1,REHRN,REHKEY);
   if OK then
    repeat
      dec(i);
      GetRec(DF1,REHRN,REHREC);
      TMP_K[i]:=RehRec.Kod;
      TMP_A[i]:=RehRec.Ad;
      TMP_R[i]:=REHRN;
      PrevKey(IF1,REHRN,REHKEY);
    until (not(OK) or (i=1));
   if i > 1 then i:=ileriDoldur(TMP_K[i]);
   if i = 10 then i:=0;
   GeriDoldur:=i;
end;
{--------------------}
procedure Goruntule;
begin
   for i:=1 to 9 do
    begin
      outstr(TMP_K[i]+' '+TMP_A[i],i+8,51,renk2);
    end;
end; { goruntule }
{--------------------}
procedure islemle;
var cc   : char;
    sec1 : byte;
begin
   Goruntule;
   sec1:=1;
   repeat
     outstr(TMP_K[sec1]+' '+TMP_A[sec1],sec1+8,51,renk1);
     readtus(cc,[esc,return,up,down,pgup,pgdn]);
     if ((cc = down) and (sec1 < 9)) then
      begin
         inc(sec1);
         if TMP_K[sec1]<>space(4) then
         begin
            outstr(TMP_K[sec1-1]+' '+TMP_A[sec1-1],
                   sec1-1+8,51,renk2);
            outstr(TMP_K[sec1]+' '+TMP_A[sec1],
                   sec1+8,51,renk1);
         end else dec(sec1);
      end;
     if ((cc = up) and (sec1 > 1)) then
      begin
         dec(sec1);
         if TMP_K[sec1]<>space(4) then
         begin
            outstr(TMP_K[sec1+1]+' '+TMP_A[sec1+1],
                   sec1+1+8,51,renk2);
            outstr(TMP_K[sec1]+' '+TMP_A[sec1],
                   sec1+8,51,renk1);
         end;
```

```
      end;
     if cc = return then RehRec.Kod:=TMP_K[sec1];
     if cc = esc    then RehRec.Kod:=space(4);
     if cc = PgDn   then
        begin
           if TMP_K[9] <> space(4) then
            begin
              if ileriDoldur(TMP_K[9]) <> 0 then
               begin
                  goruntule;
                  sec1:=1;
               end;
            end;
        end; { PgDn }
     if cc = PgUp   then
        begin
           if TMP_K[1] <> space(4) then
            begin
              if GeriDoldur(TMP_K[1]) <> 0 then
               begin
                  goruntule;
                  sec1:=1;
               end;
            end;
        end; { PgDn }
   until ((cc=esc) or (cc=return));
end; { islemle }
{--------------------}
procedure EkranaListe;
begin
   GetWhere(50,6,76,18);
   SecEkran1;
   REHKEY:=space(4);
   TmpTemizle;
   if ileriDoldur(REHKEY)<>0 then islemle;
   PutWhere(50,6,76,18);
end;  { Goruntule }
{--------------------}
{**********************************************************************}
begin
   FileAc;
   Ekran01;
   Konumlar;
   AN:=1;
   KayitBosla;
   TamamenCik:=False;
   KilitliRecordno:=0;
   repeat   { TamamenCik }
    Case AN of
      1: begin
          if KilitliRecordno<>0 then SerbestBirakKilitliyi;
          retset:=retset+[F8];
          Readstr(RehRec.Kod,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
          if RehRec.Kod<>'     ' then Kayitbak;
          if ((RehRec.Kod<>'     ') and (ret=F8)) then ret:=return;
```

```
          retset:=retset-[F8];
        end;
      2: begin
          Readstr(RehRec.Ad,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
        end;
      3: begin
          Readstr(RehRec.Adres1,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
        end;
      4: begin
          Readstr(RehRec.Adres2,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
        end;
      5: begin
          Readstr(RehRec.Tel,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
        end;
      6: begin
           mesaj1;
           readtus(ret,['E','H']);
           if ret='E' then KayitYap;
           MesajTemizle;
        end;
    end; { Case AN }
    case ret of
      up    : begin
                if AN > 1 then dec(AN);
              end;
      down,
      return: begin
                if AN < 6 then inc(AN);
              end;
      F6  : begin  { iptal }
              mesaj2;
              readtus(ret,['E','H']);
              MesajTemizle;
              if ret='E' then KayitSil;
            end;
      F8  : EkranaListe;
      Esc : begin
              if AN=1 then TamamenCIk:=true
                 else
                 begin
                   KayitBosla;
                   Ekran01;
                   AN:=1;
                 end;
            end;
     end; { case ret }
   until TamamenCik;
   CloseFile (DF1);
   CloseIndex(IF1);
end.
```

```
(*****************************************************************)
(*                          UORNEK                               *)
(*                                                               *)
(*      KULLANICI MAKINADA VERI TABANI KULLANAN BIR ÖRNEK        *)
(*                   UYGULAMA PROGRAMI                           *)
(*                                                               *)
(*****************************************************************)
program UORNEK;
uses crt,UACC,SRUN,util01;
const
   retset : set of  char = [return,esc,up,down];
   renk1  = 126;
   renk2  = 4;
var
    RL : word; { Record Length }
    KL : Byte; { Key Length    }
    DP : Byte; { Duplicate or not }
    c : char;
    i : byte;
    ret : char;
    GetMode:byte;
    DF11 : UDataFile;
    IF11 : UIndexFile;
Type  RehberKayit =  Record
                       stat     : longint;
                       Kod      : string[4];
                       Ad       : string[20];
                       Adres1   : string[20];
                       Adres2   : string[20];
                       Tel      : string[15];
                     end;
var
   REHREC : rehberKayit;
   REHRN  : longint;
   REHKEY : string[4];
   DFName,
   IFName : string[60];
   AN     : Byte;
   KonX,
   KonY   : Array[1..5] of byte;
   TamamenCIk,
   KayitVar : Boolean;
   TMP_K    : array [1..9] of string[4];
   TMP_A    : array [1..9] of string[20];
   TMP_R    : array [1..9] of LongInt;
procedure FileAc;
begin
   DFName:=PathName+'\KART.DAT';
   IFName:=PathName+'\KART.IND';
   RL:=SizeOf(REHREC);
   KL:=4;
   DP:=0;
   UopenDataFile(DF11,DFName,RL);
   UOpenIndexFile(IF11,IFName,KL,DP);
end; { FileAc }
{-------------------}
```

```
Procedure ekran01;
begin
   clrscr;
   CreateWindow(1,4,80,20,2,2);
   CreateWindow(27,1,48,3,2,2);
   gotoxy(29,2);
   write('KARTOTEKS PROGRAMI');
   NormVideo;
   gotoxy(10,6);  write('KOD.....:');
   gotoxy(10,9);  write('AD SOYAD.:');
   gotoxy(10,11); write('ADRES.1..:');
   gotoxy(10,13); write('ADRES 2..:');
   gotoxy(10,15); write('TELEFON..:');
end;
{--------------------}
procedure konumlar;
var i : byte;
begin
 for i:=1 to 5 do KonX[i]:=21;
 KonY[1]:=6;
 KonY[2]:=9;
 KonY[3]:=11;
 KonY[4]:=13;
 KonY[5]:=15;
end;  { konumlar }
{--------------------}
procedure kayitbosla;
begin
   with Rehrec do
   begin
      stat  := 0;
      Kod   :=space(4);
      Ad    :=space(20);
      Adres1:=space(20);
      Adres2:=space(20);
      Tel   :=space(15);
   end;
end;
{--------------------}
procedure IptalTusu;
begin
    Gotoxy(2,23);
    if KayitVar then write('F6 = iptal')
                else write('          ');
end; {  IptalTusu  }
{--------------------}
procedure KayitGoster;
begin
   With RehRec do
   begin
      outstr(Ad,KonY[2],KonX[2],renk2);
      outstr(Adres1,KonY[3],KonX[3],renk2);
      outstr(Adres2,KonY[4],KonX[4],renk2);
      outstr(Tel,KonY[5],KonX[5],renk2);
   end;
end;
```

```
{--------------------}
procedure mesaj9;
begin
 outstr('Bu Kayit baska bir kullanici tarafindan kullanilmaktadir ...',
         21,2,renk2);
 c:=readkey;
 outstr(space(70),21,2,renk2);
 ret:=#0;
 AN:=1;
end;
{--------------------}
procedure kayitbak;
begin
   retset:=retset-[F6];
   Kayitvar:=False;
   REHKEY:=RehRec.Kod;
   UFindKey(IF11,REHRN,REHKEY);
   if REHRN<>0 then
     begin
        KayitVar:=True;
        GetMode:=1;
        UGetRec(DF11,REHRN,REHREC,GetMode);
        if GetMode=1 then
           begin
              KayitGoster;
              iptalTusu;
              retset:=retset+[F6];
           end
           else
           begin
            mesaj9;
           end;
     end;
end;
{--------------------}
procedure mesaj1;
begin
   gotoxy(10,22);
   if Kayitvar then
            write('Duzeltilen bilgiler dogru mu ? [E/H]...[ ]')
      else  write('Girilen bilgiler dogru mu  ?  [E/H]....[ ]');
   gotoxy(50,22);
end;
{--------------------}
procedure mesaj2;
begin
   gotoxy(10,22);
   write('Silmek istiyor musunuz ?   [E/H].......[ ]');
   gotoxy(50,22);
end;
{--------------------}
procedure MesajTemizle;
begin
   gotoxy(10,22);
   write('                                                ');
   AN:=1;
```

```
end;
{--------------------}
procedure KayitYap;
var kk : string[60];
begin
  if KayitVar then
   begin
     UPutRec(DF11,REHRN,RehRec);
   end
   else { Eger kayit yoksa }
   begin
     UAddRec(DF11,REHRN,RehRec);
     kk:=rehkey+space(56);
     UAddKey(IF11,REHRN,REHKEY);
     KAyitvar:=True;
   end;
   KayitBosla;
   AN:=1;
   Ekran01;
end;  { KayitYap }
{--------------------}
procedure KayitSil;
begin
   UDeleteKey(IF11,REHRN,REHKEY);
   if REHRN<>0 then UDeleteRec(DF11,REHRN)
      else
      begin
         gotoxy(10,22);
         write('Silme islemi basarisiz oldu !....');
         c:=readkey;
      end;
    mesajtemizle;
    kayitbosla;
    KayitGoster;
    AN:=1;
    Kayitvar:=False;
    IptalTusu;
end;
{--------------------}
procedure SecEkran1;
begin
   CreateWindow(50,6,76,18,7,2);
   outstr('╟──────────────────────────╢',8,50,7);
   gotoxy(51,7); write('KODU      ADI   SOYADI   ');
   for i:=9 to 17 do
     begin
       gotoxy(51,i);
       write('                          ');
     end;
end;
{--------------------}
procedure TmpTemizle;
begin
   for i:=1 to 9 do
    begin
       TMP_K[i]:=space(4);
```

```
        TMP_A[i]:=space(20);
        TMP_R[i]:=0;
      end;
end;
{--------------------}
function ileriDoldur(REHKEY : string) : byte;
begin
   TmpTemizle;
   i:=0;
   USearchKey(IF11,REHRN,REHKEY);
   if REHRN<>0 then
    repeat
      inc(i);
      GetMode:=1;
      UGetRec(DF11,REHRN,REHREC,GetMode);
      TMP_K[i]:=RehRec.Kod;
      TMP_A[i]:=RehRec.Ad;
      TMP_R[i]:=REHRN;
      REHKEY:=REHREC.Kod;
      UNextKey(IF11,REHRN,REHKEY);
    until ((REHRN=0) or (i=9));
   ileriDoldur:=i;
end;
{--------------------}
function GeriDoldur(REHKEY : string) : byte;
begin
   TmpTemizle;
   i:=10;
   USearchKey(IF11,REHRN,REHKEY);
   if REHRN<>0 then
    repeat
      dec(i);
      GetMode:=1;
      UGetRec(DF11,REHRN,REHREC,GetMode);
      TMP_K[i]:=RehRec.Kod;
      TMP_A[i]:=RehRec.Ad;
      TMP_R[i]:=REHRN;
      REHKEY:=REHREC.Kod;
      UPrevKey(IF11,REHRN,REHKEY);
    until ((REHRN=0) or (i=1));
   if i > 1 then i:=ileriDoldur(TMP_K[i]);
   if i = 10 then i:=0;
   GeriDoldur:=i;
end;
{--------------------}
procedure Goruntule;
begin
   for i:=1 to 9 do
    begin
      outstr(TMP_K[i]+' '+TMP_A[i],i+8,51,renk2);
    end;
end; { goruntule }
{--------------------}
procedure islemle;
var cc   : char;
    sec1 : byte;
```

```
begin
   Goruntule;
   sec1:=1;
   repeat
     outstr(TMP_K[sec1]+' '+TMP_A[sec1],sec1+8,51,renk1);
     readtus(cc,[esc,return,up,down,pgup,pgdn]);
     if ((cc = down) and (sec1 < 9)) then
      begin
         inc(sec1);
         if TMP_K[sec1]<>space(4) then
         begin
            outstr(TMP_K[sec1-1]+' '+TMP_A[sec1-1],
                   sec1-1+8,51,renk2);
            outstr(TMP_K[sec1]+' '+TMP_A[sec1],
                   sec1+8,51,renk1);
         end else dec(sec1);
      end;
     if ((cc = up) and (sec1 > 1)) then
      begin
         dec(sec1);
         if TMP_K[sec1]<>space(4) then
         begin
            outstr(TMP_K[sec1+1]+' '+TMP_A[sec1+1],
                   sec1+1+8,51,renk2);
            outstr(TMP_K[sec1]+' '+TMP_A[sec1],
                   sec1+8,51,renk1);
         end;
      end;
     if cc = return then RehRec.Kod:=TMP_K[sec1];
     if cc = esc    then RehRec.Kod:=space(4);
     if cc = PgDn   then
        begin
           if TMP_K[9] <> space(4) then
            begin
              if ileriDoldur(TMP_K[9]) <> 0 then
               begin
                  goruntule;
                  sec1:=1;
               end;
            end;
        end; { PgDn }
     if cc = PgUp   then
        begin
           if TMP_K[1] <> space(4) then
            begin
              if GeriDoldur(TMP_K[1]) <> 0 then
               begin
                  goruntule;
                  sec1:=1;
               end;
            end;
        end; { PgDn }
   until ((cc=esc) or (cc=return));
end; { islemle }
{--------------------}
procedure EkranaListe;
```

```
begin
   GetWhere(50,6,76,18);
   SecEkran1;
   REHKEY:=space(4);
   TmpTemizle;
   if ileriDoldur(REHKEY)<>0 then islemle;
   PutWhere(50,6,76,18);
end;  {  EkranaListe }
{---------------------}
begin
   FileAc;
   Ekran01;
   Konumlar;
   AN:=1;
   KayitBosla;
   TamamenCik:=False;
   repeat   { TamamenCik }
    Case AN of
      1: begin
          retset:=retset+[F8];
          Readstr(RehRec.Kod,KonY[AN],KonX[AN],renk1,renk2,retset,ret,' ',' ');
          if RehRec.Kod<>'     ' then Kayitbak;
          if ((RehRec.Kod<>'     ') and (ret=F8)) then ret:=return;
          retset:=retset-[F8];
         end;
      2: begin
          Readstr(RehRec.Ad,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
         end;
      3: begin
          Readstr(RehRec.Adres1,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
         end;
      4: begin
          Readstr(RehRec.Adres2,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
         end;
      5: begin
          Readstr(RehRec.Tel,KonY[AN],KonX[AN],
                  renk1,renk2,retset,ret,' ',' ');
         end;
      6: begin
           mesaj1;
           readtus(ret,['E','H']);
           if ret='E' then KayitYap;
           MesajTemizle;
         end;
    end; { Case AN }
    case ret of
      up    : begin
                if AN > 1 then dec(AN);
              end;
      down,
      return: begin
                if AN < 6 then inc(AN);
              end;
```

```
      F6  : begin  { iptal }
              mesaj2;
              readtus(ret,['E','H']);
              MesajTemizle;
              if ret='E' then KayitSil;
            end;
      F8  : EkranaListe;
      Esc : begin
              if AN=1 then TamamenCIk:=true
                 else
                 begin
                   KayitBosla;
                   Ekran01;
                   AN:=1;
                 end;
            end;
     end; { case ret }
   until TamamenCik;
   UCloseDataFile(DF11);
   UCloseIndexFile(IF11);
end.
```

SONUÇLAR VE ÖNERİLER

Kişisel bilgisayarlar arasında veri iletişimini sağlayan bu çalışma daha da geliştirilebilir. Örneğin; aynı ağ içinde üçden fazla kişisel bilgisayar konuşur hale getirilebilir. Tek taraflı çalışan haber niteliğinde mesaj gönderme işlemi iki taraflı çalışır hale getirilebilir. Sistem sadece veri tabanındaki verileri kullanmakla kalmayıp işletim sisteminin bazı komutlarıda kullanır hale getirilebilir. Modem bağlantıları kontrolleri konularak da uzak yerlerdeki kişisel bilgisayarlarla haberleşme sağlanabilir.

KAYNAKLAR

[1] TANENBAUM, A.S. Computer Networks (Second Ed.)
Prentice-Hall 1989

[2] BERTSEKAS, D. Data Networks
Prentice-Hall 1987

[3] HALSALL, F. Data Communication, Computer Networks and OSI
Addison-Wesley Publishing Company 1988

[4] IBM , IBM PC and AT Technical Referans manual
IBM 1984

ÖZGEÇMİŞ

Osman Nuri Özpınar 1964 yılında Ankara doğmuştur. 1981 yılında orta öğrenimini İstanbul Fenerbahçe Lisesinde tamamlamıştır. Aynı yıl İstanbul Teknik Üniversitesi Temel Bilimler Fakültesi Matematik Mühendisliği bölümüne girmiştir. 1987 yılında Matematik Mühendisi ünvanı alarak aynı okuldan mezun olmuştur. Aynı yıl İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü tarafından açılan yüksek lisans sınavlarında başarılı olarak Mühendislik Bilimleri Sistem Analizi yüksek lisans programını kazanmıştır.

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ