A COMPUTER COMMUNICATIONS SYSTEM

by

KIM W. JANNE

B.S., Kansas State University, 1980

A MASTER'S PROJECT

submitted in partial fulfillment of the

requirements for the degree

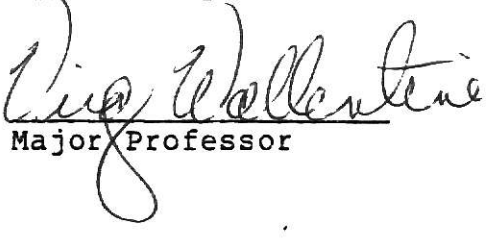
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

Approved by:



Major Professor

ACKNOWLEDGMENTS

I wish to thank Dr. Virgil Wallentine for his help and guidance in completing this project. And a special thanks to my very understanding wife.

TABLE OF CONTENTS

I. Introduction 1

II. Design 7

III. Implementation 14

IV. User Guide 20

A. MAILMAN 21

B. MAILLINK 32

C. INCNVTR / MECNVTR 40

V. Implementors Guide 41

VI. Summary 46

VII. Bibliography 49

VIII. Appendix 51

A. MAILMAN Source Listing 52

B. MAILLINK Source Listing 99

C. INCNVTR Source Listing. 109

D. MECNVTR Source Listing. 113

I. INTRODUCTION

The increase in popularity of personal computers has significantly altered the distribution of computing resources. Traditionally, the "computer center" was the sole source for utilizing these resources. However, with the abundance of microprocessor technology, the costs of traditional computing power have diminished to such an extent that many individuals can now justify the acquisition of personal computer equipment.

This developing trend has brought about new ideas regarding the role of the "computer center". Historically, the trend was to build bigger and bigger systems. Recently this trend has reversed itself to where current implementations have tended to promote decentralized resources. Examples of such configurations are the linking of smaller, more application dependent computers to a single large system acting primarily as a resource environment.

Within this decentralized computing environment, the small systems are able to function independently of the large system. The smaller systems may only require the services of the larger system for specialized resources that generally are not cost effective to be distributed. Such resources could entail high speed line printers, letter quality printers, and expensive high speed, large capacity storage devices.

While the theoretical basis for such distributions are

relatively sound, the implementational aspects of such resource sharing have proven to be a very undeveloped area of current computing technology. Partially due to the extreme variation of available remote computing systems, generalized distributed support environments are indeed rare. The MAILMAN / MAILLINK, (Mail Manager - Mail Manager Communication Link), project was an attempt to provide such an environment for the Kansas State University, Department of Computer Science.

The MAILMAN / MAILLINK systems are not a truly universal solution to the distributed resource problem. They do however, provide to a substantially large base, the ability to communicate and remotely operate in a compatible environment with the Interdata 8/32 system in the Computer Science Department's minicomputer laboratory.

To provide this resource for more than a single architectural system, the need for a heterogeneous computer solution to this communication system was a high requirement. The ability to meet this goal is provided with the UCSD "P" system.[9,16] The University of California at San Diego has developed a non-machine dependent operating system. The essence of this system is to provide a pseudo pascal compiler which generates a universal object "P" code. This "P" code is then executed by a machine dependent run time interpreter. By restricting machine dependent features to the run time

interpreter, source code compatibility is insured for all systems running in a "P" code environment. Because of this source code compatibility, the UCSD "P" system has become a very popular system and therefore, its utilization for the remote systems will provide for a large application base.

To expand upon the distribution concept, the ability to communicate across machines was also desired. Uses for such communication are to transfer information across media incompatibilities. While the source code to a "P" system program is execution compatible with differing machines, the media format on which these source listings may be exchanged are frequently not compatible. With the MAILMAN / MAILLINK system, information can be transmitted in its pure ASCII form, and converted by the target machine into its own compatible format.

Because this distribution potential should not be contingent upon simultaneous connection of the transferring users, some mechanism for information storage and retrieval was desired. The functional description of this run time support closely paralleled that of current electronic mail systems. Therefore, the design also included the ability for the system to act as a stand alone mail system. This implementation would allow users operating locally to utilize its applicable features.

The MAILMAN / MAILLINK environment is composed around a host

Interdata 8/32 computer system. All communication is directed through this system. The host does not differentiate between the various modes of users, communicating with a virtual terminal interface. The local users, users whose connection is only a terminal directly connected to the 8/32, can utilize all the features of the electronic mail support. Remote users, users connected through a computer to computer interface, have the additional feature of the computer communication support.

Figure 1 diagrams this MAILMAN / MAILLINK environment. The dashed lines represent the conceptual virtual communication paths that can be accomplished through this communication support system. The straight solid lines show direct physical connections. The heavy jagged lines illustrate remote dial-in users operating through telephone modems. The connectivity of the devices shown, represent the accessibility of the devices to the various users.

An Example of how the MAILMAN / MAILLINK environment works can be illustrated in the following simplified user scenario:

> User A is a local user who creates a program for use on the Interdata 8/32. Upon hearing about the new approach User A attempted, User B wished to review the program and modify it for his personal computer system. User A then "mailed" a copy addressed to User B. When User B got home and connected his system via the MAILLINK program to the 8/32, he found the desired program in his system mailbox and then could transfer the program to his personal system. After the transfer

was complete, the editor format converter, MECNVTR was utilized to convert the file to a source compatible version to be used with the screen oriented editor rather than the line oriented editor with which this program was initially created.

It is definitly true that more involved uses for the MAILMAN / MAILLINK system can be envisioned, but this example does illustrate the conceptual foundation for the MAILMAN / MAILLINK project.

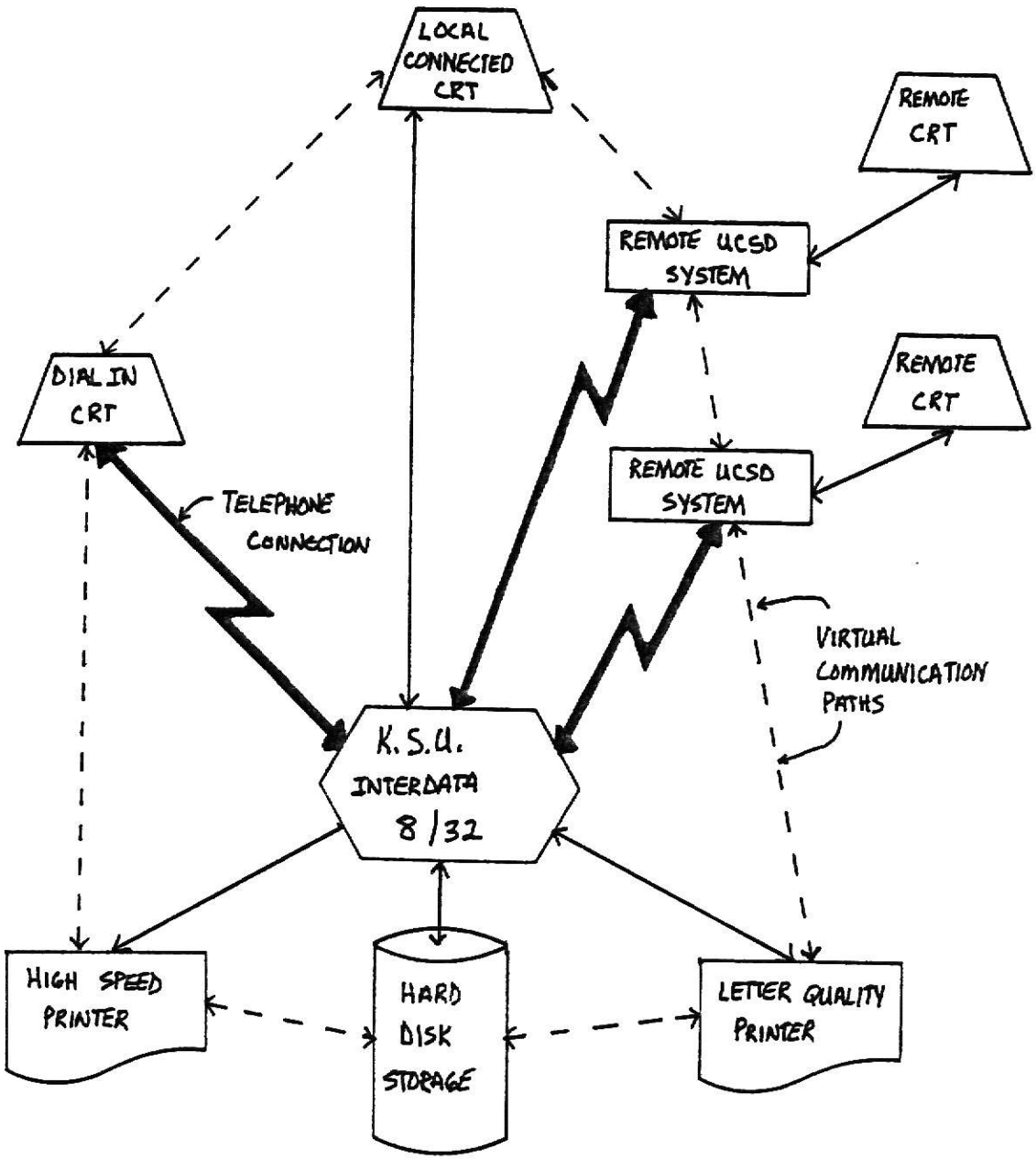


Figure 1 - The MAILMAN / MAILLINK Environment

II. DESIGN

The MAILMAN support section was based upon the functional components of many existing electronic mail systems. Most of the systems reviewed had a relatively similar primitive function foundation. Although the underlying implementational structures differ significantly, the abstract user view of the system's execution environments are generally quite comparable.

It is from this industry pseudo standard that the functional construct of the MAILMAN system was conceived.[1,6,11,12] The basic electronic mail primitives have included the ability to deposit and retrieve a mailed item, forward mail to another user, inquire as to the status of incoming mail, and create and destroy mail items. Additional features such as encryption devices, distribution lists, time stamps, and priority encoding, are the differentiating characteristics to the multitude of electronic mail systems now emerging in the industrial communication markets.

The most non-standard elements to the electronic mail systems currently available, are the naming, routing and addressing mechanisms. Some systems offer physical independence by addressing only in terms of logical users whereas others may be as specific as actual physical port connections. Such trade offs are generally a result of the specific application and to the extent of which the system may function in a network wide

environment. For the purpose of the MAILMAN system, a rather limited scope was defined. Therefore naming and routing restrictions brought about by network addressing were avoided. This inherent characteristic of the MAILMAN system was an architectual foundation for many of the operational intrinsics found in the MAILMAN system.

The issue of user naming conventions was further defined by the expected intent and usage characteristic of the user agents.[12] It was anticipated that the total number of participants within the system would be no greater than what could be handled with the current MTM user ID conventions. This convention limits the user signon ID's to be unique with the current Interdata MTM operating system's user register. Therefore, most users have adopted unique user ID's to distinguish themselves while operating within the MTM environment.

While this naming convention would certainly create problems operating in a network wide environment, the MTM user on the single host system is significantly freed of mail addressing problems. Rather than having to address a particular user registration data base for each item mailed, the user has only to direct the mail with the destination user's commonsignon identification. It can be generally assumed that if mail were

intended to be sent to a particular user, the source user would have knowledge of the destination user's common identification. To supplement this convention, a user identification directory could be made available.

Of the design considerations, the most involved was the support required to accommodate concurrent users of the MAILMAN system. While the system is not a true concurrent environment, the ability for multiple task images to be used by multiple users was a definite requirement. To support such operations, the dynamic mail directory data base had to be constructed to avoid the problems of delete and update anomalies.

The solution presented by the MAILMAN system is to restrict the update and deletion processes from directly accessing the global data resource. When a user enters the MAILMAN system, an image of his current directory is copied to the user's task data storage. Furthermore, a record is made of the user's presence in the MAILMAN environment in the Active User Table. This Active User Table is a directory of potentially inaccurate mail directories.

When a directory image is copied from the master file, the user's associated dirty bit is reset. This bit indicates at exit time whether or not a directory update is required. An update is required when a modification has been made to the user's directory. Such modifications are the result of the commands,

F)orward, P)urge, R)emove, G)et, E)ncrypt, and D)ecrypt.

After the dirty bit is checked the Tempmail file is searched for entries that have been addressed to this user. The Tempmail file is a mechanism by which concurrent users may address mail to each other without resulting in update anomalies. Because each user has an image of the directory data base loaded into the user's own task data space, concurrently operating users cannot manipulate the pointer information for another active user. Therefore when mail is addressed to an active user, the mail item is temporarily deposited within the Tempmail file. When the active user exits the system, and likewise will not further update the pointer table in the users task area, a search is made through the Tempmail file for entries directed to this user. If a match is found, the Tempmail entry is removed from the Tempmail file and deposited in the exiting user's task area pointer table for download to the main system directory table.

While the Tempmail solution significantly reduces the potential for update problems, there still exists the possibility of access right errors while attempting to gain the exclusive access required for updating and reviewing the Tempmail file. While the probability of two users simultaneously exiting the system, and requesting the Tempmail file are indeed minimal, the possibility does exist. Therefore a check is made following the

assignment request for successful access assignment. If assignment was not successful then it can be assumed that another user is exiting the system, and therefore the rejected user will repeatedly attempt the request until the Tempmail file is available.

Another important factor considered in the initial design of the MAILMAN system was the issue of abiding to existing and developing standards in the electronic mail industry. While no such universally accepted standards have been adopted, several emerging protocols have been getting widespread attention. However, due to the user intent of this project and the immense software overhead that would be required to support such conventions, the decision was made to implement useful features of the protocols, but not to emphatically support the entire spectrum of standardizing requirements.

This design of practicality was furthermore carried to the MAILLINK section as well. The International Standards Organization seven layer hierarchical model for file transfer protocols [3,13] would be an unrealistic expectation for implementation on many of the intended target systems. Therefore, significant features, such as a pseudo layering of transmission functions, as well as error and flow control were designed into the MAILLINK construction.

Other design considerations were dictated by the inherent

restrictions imposed by the host Interdata 8/32 MTM operating system. Because the user interfaces were to be considered virtual terminals, thus allowing for remote UCSD computer connections as well as local terminal users, the input interfaces were required to operate through the MTM system. This introduced substantial design constraints due to the limitations of the current interface drivers available. (Perkin-Elmer has claimed that future releases of the MTM system will facilitate to a much greater extent, this kind of input interaction)

The most significant of the MTM imposed restrictions was from the input buffer management. MTM limits the input buffer size to a maximum of 80 characters outstanding.(10) This results in a limitation to the maximum size of frame that can be transmitted. Both the UCSD and 8/32 system file sizes are referenced in terms of 512 byte blocks. However transmission in the same format size was impossible because of the potential overrun and resulting sequence error problems imposed via the MTM input drivers. Thus frame sizes were reduced to a maximum 64 characters per frame with 3 additional overhead management bytes. The 64 character frame data size was chosen by the fact that it is the largest even multiple of the 512 byte record size that could be accommodated in the worst case sequence by the MTM i/o drivers.

The overhead introduced by this MTM restriction does make a

noticeable difference in the overall efficiency of the transmission protocol. With the resulting 67 byte frame size, the ratio of actual data to data plus overhead is approximately 84.9%. Whereas, if the driver restriction were not present, an effective data transmission efficiency approaching 98% could be achieved.

The additional overhead mentioned above includes the frame command code as well as a cyclical redundancy check. This CRC calculating algorithm has been modified to overcome several undocumented problems that occurred with transmission of CRC characters in the ordinal value range of 0 - 31. These control code values generated very unpredictable results when operating on the Western Digital WD / 90 Microengine. Therefore, a modification and shift of the calculated CRC values was performed. While the overall probability of missing an erring frame increased as a result of this modification, the performance monitoring demonstrated that the risk would still be far below acceptable limits.

III. IMPLEMENTATION

The host Interdata 8/32 design configuration proved to be quite an exercise in system level programming. Virtually every function required some degree of supervisory system support. The Pascal supervisory systems call procedure support was a significant help. However, this tremendous amount of machine dependant coding makes the distribution potential for the MAILMAN / MAILLINK system rather restricted. Furthermore, the upward compatiblility for future MTM operating system releases is somewhat in doubt as well.

The first major division of code with the MAILMAN system contains runtime support procedures. Because of the character oriented input/output definitions of the Kansas State University implementation of Pascal,[17] several higher level output procedures were created. These procedures facilitate the transmission of character strings as well as the transformation of numeric values to their charecter string representations. A lower case to upper case converting function is also provided to insure uniformity in input responses.

The following procedural sections deal with the maintenance of the mail directories. To clarify the procedural processes, a further examination of the data file structures will be presented.

The system is maintained within three seperate data files. The Active User File contains a current record of all users currently active within the MAILMAN system environment. When a user loads and starts execution of a MAILMAN task, the user's signon identification is placed in the Active User File in the first available record slot. This table is maintained for the purpose of identifing and isolating potentially inconsistent user directories.

The Tempmail file further isolates the update anomalies associated with multiple versions of similar data residing with seperate owners. Because of the assumption that mail cannot be deleted, or retracted by the source user, (unless the mail source and destination addresses were identical), The only way in which update inconsistencies may occur is by the deposit or addition of a mail item to a user's directory. Therefore, by trapping mail items whose destination address is a current member of the Active User List and maintaining them in a temporary directory until the destination user exits the MAILMAN environment, the directory integrity will be maintained. The only ramification of this being that a user will have no knowledge of mail that was sent to the user while the user was executing within the MAILMAN environment.

When the user exits the MAILMAN system, a check will be made to determine whether an update of the user's directory must be

made. If either a change was made by the user within the user's directory, or mail was sent to this user, indicated by an entry within the Tempmail file, then a directory update is required. Each exit of the system will result in a search of the Tempmail file for entries targetted for the exiting user.

The third data file is the main system directory. This directory is a mapping of individual user directories. Whenever a mail item is created and sent throught the MAILMAN system, the mail item is linked into a destination user's directory. If no directory existed, then a directory will be created, and the new user directory recorded in the main system directory. (see figure 2)

Whereas the main system directory maintains only a mapping address for individual user directories, the user directory record entry contains a full set of envelope attributes. The envelope record contains the following fields: source_id, date mailed, encryption flag, a short subject or topic description, as well as the system assigned mail text address. To insure uniqueness regarding file naming, the MAILMAN system renames all mail files submitted with a cummulating sequence number. A similar mechanism is used for creating individual user directories. The convention being, Mxxxxxxx.SYS for mail files, and Dxxxxxxx.SYS for user directories.

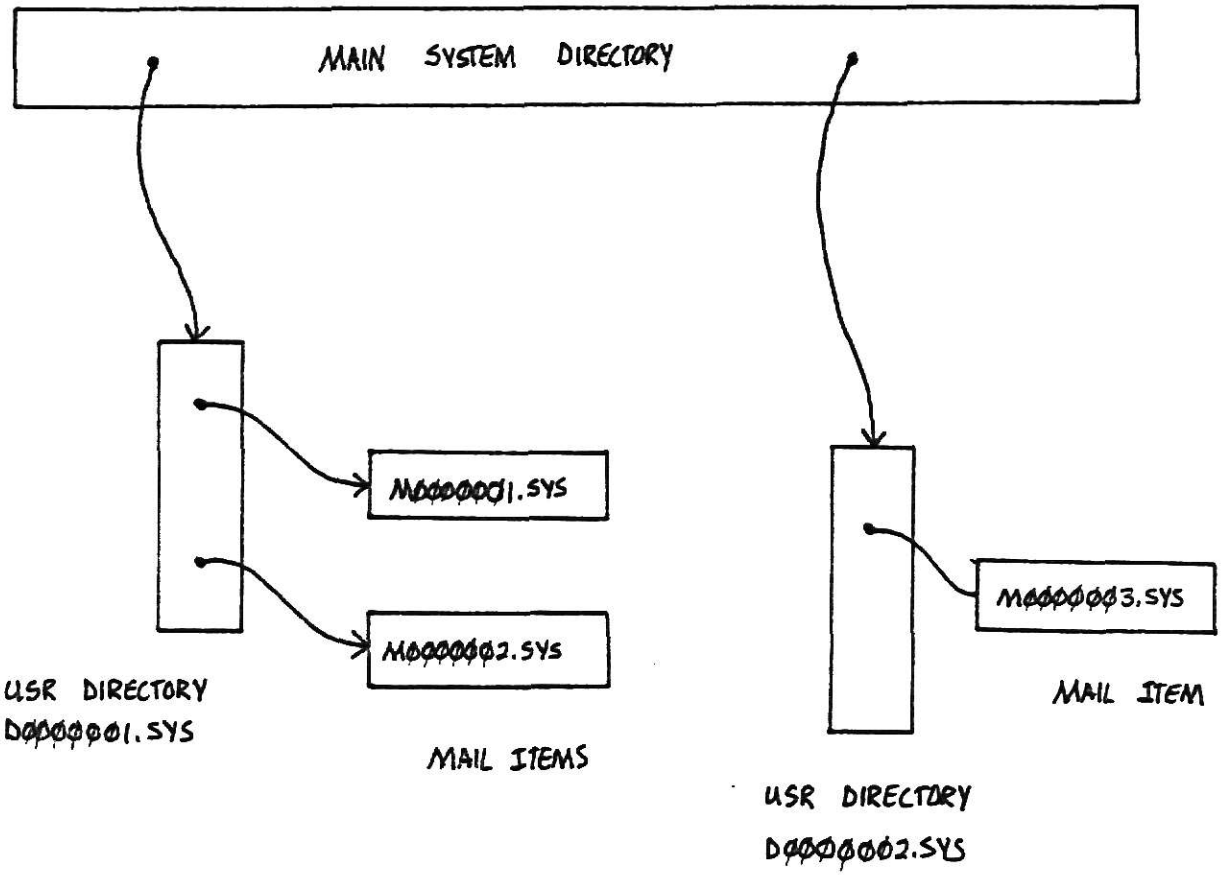


Figure 2 - Mail Directory Structure

The MAILMAN command functions are basically an interaction and manipulation of information within these data files. Mail can be created via the MEDIT support overlay, or composed of previously created files or text. It is then deposited by first checking for existence of the destination user's directory, creating a directory if nonexistent, and then linking in the new mail item. If a new mail directory was created, then a record must be created, indicating the presence of the user's new directory in the main system directory. This record will contain a pointer to the new user directory.

Forwarding mail is the process by which an entry in a user's directory is transferred to another user's directory. Removal and retrieval of mail items result in the removal of a user directory entry. In the case of retrieval, the item is placed in the user's own MTM file directory, whereas a removal will just eliminate the image of that mail item.

The inquire and encryption processes are processing applications for the information maintained in the user directory entry envelope record. The inquire function will just sequentially display the envelopes maintained for this user in the user's directory. The encryption routines will modify the associated envelope flags as well as the desired textual material.

In order to support mail creation within the MAILMAN environment, a form of text editor was required. Because of the common familiarity with the Interdata 8/32 PEDIT line oriented editor, a version of this was incorporated. Modifications were neccessitated because of the overlay invoking procedure. Within an overlaid task, supervisory calls,(SVC's) are prohibited because of the inability of the linking mechansims to correctly establish the overlaid run time environment. Several other enhancements were also included. Among these changes were the inclusion of certain command derivatives, basically shorthand notations of existing commands. (i.e. "ty" -> "t", "dn" -> "d"...)

IV. MAILMAN / MAILLINK USERS GUIDE

The following presentation is a description and user reference document to the MAILMAN / MAILLINK communication system. In the accompanying examples, several illustrating conventions will be observed: All user entered responses are underlined; computer generated prompts will appear in boldface; and the symbol <CR> represents the "RETURN" or "ENTER" key on the user's terminal.

The presentation that follows is divided into two separate sections. The first will describe the local MAILMAN mode. The following section contains a detailed description of the additional features provided to a remote user via the MAILLINK support programs. Note that all the features described in the MAILMAN section apply equally to users operating either locally or remotely. Concluding this document are hard copy loggings of typical user sessions, in both local and remote modes. It is further advised that the user read the manual completely prior to using the MAILMAN system in order that greatest utilization of the features provided may be achieved.

The MAILMAN System

The MAILMAN system can be conceptually described in terms relating to its physical counterpart, the post office. Mail in the MAILMAN context refers to text transmitted, stored and retrieved by electronic means. Text can be a short message, or likewise an entire source listing. The only restriction is that mail will be transmitted and retrieved in 512 byte blocks. Therefore, text whose semantic interpretation is depended upon the physical block representation must compensate for such transmission adjustments.

In later descriptions of the functions of the MAILMAN communication system, comparisons will be frequently made to the postal service. These comparisons are for a purely semantic illustration and are not intended to be a reflection or commentary on either of the compared parties.

The MAILMAN system in its current implementation is resident on the Interdata 8/32 as a Sequential PASCAL program. To execute in the MAILMAN environment, the MAILMAN task must be loaded and the proper logical units assigned. This invoking process has been combined and is available to the user by executing the MAILMAN Command Substitution System file. This sequence is illustrated below:

* MAILMAN <CR>

(Note: the space preceding the <CR> is not required, but used for clarity in all of the following examples)

After a few seconds delay, the lines shown below will be displayed. These lines will be referred to as the main system prompt line. To select a particular routine, type the first character of the routine name and press the <CR>. All responses may be entered as upper or lower case characters unless the documentation explicitly describes otherwise.

```
Mail Manager R00-03     Type "H"elp for command summary
Please report problems to K.W. Janne

Enter command   H)elp, I)nquire, G)et, C)reate, R)emove,
                P)urge, M)ail, F)orward, T)ype, X)fer,
                E)ncrypt, D)ecrypt, S)et_edit, Q)uit
```

(main system prompt line)

The routine actions are generally closely related to the routine names. The specific operations of the routines provided will now be discussed in detail.

H)elp

The help command provides to the user a run time explanation of and reference to the available commands. The commands will be

accompanied by a brief explanation of the command's operation. Details regarding a particular command's specific format are not provided in this command summary. Once the help listing has been displayed, the user may return to the main system prompt line by depressing the <CR> key.

I)nquire

The inquire command will produce a summary of mail that has been directed or sent to the current user's ID. Included in this summary is the source's ID as well as the date on which this mail was deposited. The listing also includes a short topic description, (if provided by the source) and an indication as to whether or not the material has been encoded. The address description is the MAILMAN assigned filename for this mail. When invoking routines which require the source or destination address of a current mail item, this address must be used. All system addresses will be of the format: Mxxxxxxx.SYS where the letter "x" represents a unique numerical identifier.

G)et

The get command allows the user to retrieve the user's directed mail from the MAILMAN system. The equivalent comparison would be to remove a letter from a post office box. The GET command removes the mail from the system directory and will

deposit it in the user's currently logged volume. The box address it requests will be the associated file descriptor as displayed in the summary produced by the Inquire command. Once the mail has been retrieved to the user's private account, the mail can be renamed or re-edited following standard 8/32 MTM conventions.

C)reate

The Create routine allows the user to create a mail file while remaining in the MAILMAN environment. The routine will first ask for a destination address. This address is the user ID to whom this mail will be directed. After the user ID is solicited, the prompts will ask for a subject or title description. While this information is not required, (user may simply enter a <CR>), it is significantly useful to help differentiate incoming mail by the destination user. Other information is acquired transparently to the creating user, such as post_mark date - the system clock, source ID - the creators signon ID, and the encryption flag. When mail is created via the create routine, it is assumed that the information is not encoded.

After the mail envelope information has been collected, the editor is overlaid to begin execution. The editor invoked is the current default editor set via the S)et_edit command. In the default state, the MEDIT line oriented editor is called. This

editor is an enhanced and reconfigured version of the 8/32 PEDIT line oriented editor available on the K.S.U. Interdata 8/32. The user can then create information to be transmitted using a basically similar construction sequence as would be available outside of the MAILMAN environment. Using the same completion characters as PEDIT,("EN","QU"), MEDIT will reload the MAILMAN task and depending on the completion request, finish the mail create process. In the case of a "QU" exit from the MEDIT overlay, the attempted mail created file is destroyed and control will be passed to the main system prompt line. If the create was terminated with a "EN" command, the mail is encapsulated in the established envelope and is deposited in the "post office" under the appropriate user identification. After successful deposit, the main system prompt line will appear.

R)emove

Once mail has been deposited to a user's identification box, the mail will remain in the user's directory until it is either retrieved to the user's private account or is removed from the mail directory. The R)emove command facilitates the removal of the unwanted mail from the MAILMAN system.

The system will prompt for a mail address: once again this is the file descriptor assigned by the MAILMAN system to this piece of mail. The descriptor can be determined by first

executing the I)nquire command. Once the proper descriptor box address has been entered, the system will dispose of the mail file, and will remove the entry from the user's mail directory. This process will only remove one file or mail element at a time. To clean out an entire directory, use the following P)urge command.

P)urge

The P)urge command is an extended R)emove command. This command will repeatedly execute the R)emove sequence for all remaining directory or mail entries for this user. **WARNING - THIS COMMAND WILL NOT SOLICIT FOR FILE REMOVAL VERIFICATION BEFORE EXECUTING.** Therefore, if there is a file or mail entry that is not desired to be removed, do not use the P)urge command. Once a directory has been P)urged, the directory cannot be restored.

M)ail

The M)ail command allows the user to deposit a mail file or message into the MAILMAN system. The text must have already been created outside of the MAILMAN environment. The deposit sequence begins with solicitation of the destination user's identification. After the ID, the system will prompt for the subject or topic attributes. With this and the transparent information collected and stored in the deposit envelope, the system will request the source document address. This address is

the full file descriptor. A volume name omitted will be default assigned. If the file address is incorrect or unavailable (i.e. exclusively assigned to another user's task), the message "- **MAIL ERRORS** - " will be displayed and the mail sequence aborted. If all procedures proceed correctly, the indication of successful completion will be displayed prior to the redisplay of the main system prompt line.

It should be noted at this time that due to the underlying structure of the MAILMAN system, even though a source-destination pair are simultaneously executing in the system, mail between them will not be identified until the current directories are updated. This update will only take place upon the exiting of the MAILMAN environment. For further explanation see the Q)uit command.

F)orward

This command will redirect mail that was sent to the current user to a different destination address. The result of this command will remove the original entry from the current user's mail directory and will deposit the mail and it's corresponding envelope information to the solicited new destination address. The original source address will be maintained as will the initial creation or post_mark date.

The process begins by asking for the new destination

address. Next the mail address is determined and if found to be a valid envelope address, the mail is forwarded to the new address.

An additional feature of the F)orward command is the ability to F)orward mail to a printing device. This action will only produce a hard copy image of the mail text and will not remove or alter the mail entry in the user's directory. This feature is invoked by specifying one of the available logical printer addresses for the destination address. The logical device mnemonics are "PR:" and "SPIN:". The first, "PR:" refers to a line oriented printer. This generally will only produce upper case letters. The second, "SPIN:", will be logically associated will a letter quality printing device. (For further information regarding printer assignments, see the MAILMAN / MAILLINK Implementors Guide)

T)ype

The T)ype command will sequentially display all envelopes and their associated mail contents, that currently are listed in the user's mail directory. Included in each display are the source ID, post_mark, and subject information along with the individual box address. If the file has been encrypted, only the envelope information will be displayed along with a message indicating that the file must be decrypted in order to be

displayed via the T)ype command. Otherwise, the mail contents are displayed following the envelope header information. This process is repeated for all undelivered mail in the user's mail directory. To stop the display for detailed viewing, depress the control key and the "S" key simultaneously. To resume the display, press the control key and "R" key simultaneously. Occasionally the <CR> key must also be pressed in order to resume display.

X)fer

This is the command for remote file / mail transfers. The execution intrinsics will be discussed later in the MAILLINK section.

E)ncrypt

This command and its inverse, the D)ecrypt command have not been installed in the present version of the MAILMAN system. These command prompts are therefore, presently null commands. (For installation of Encryption Algorithms see the Installation Guide.)

D)ecrypt

This is the inverse of the E)ncrypt function. See note above in E)ncrypt description.

S)et Edit

The S)et Edit command allows the user to reset the default editor used for the C)reate mail command. The system normally defaults to the MEDIT (PEDIT) line oriented editor. However, the prompts describe and allow for a screen oriented editor as well as a user defined editor. Neither of these optional editors are currently available. (Users wishing to supplement the currently available editors are directed to section "Installation of MAILMAN Editors", in the MAILMAN / MAILLINK Installation Guide.)

Q)uit

The final command on the main system prompt line is the Q)uit command. This command is basically an exit initiator. When this command is executed, the MAILMAN system first checks the dirty bit associated with this user to determine if a directory update is required. This bit would be set if the user has made any changes that have affected his directory. Such changes would be the R)emoval or P)urging of the user directory, as well as F)orwarding of mail.

Once this has been confirmed or ignored, the MAILMAN system then checks the TEMPMAIL file for mail that has been sent to this user while the user was active in the MAILMAN environment. When a user sends mail to another user who is currently executing in the MAILMAN environment, the mail is entered into the Tempmail

directory. This temporary assignment avoids problems associated with the various update and delete anomalies. If the user has been sent mail, then this entry is transferred from the Tempmail file and entered into the user permanent mail directory as part of the Q)uit process.

If a change has been made to the user's directory and an update is required, or likewise an addition needs to be made from the Tempmail file, the message "- DOWNLOAD COMPLETE -", will be displayed following the successful completion of this task. If no message is displayed following the execution of the Q)uit command, the user will know that no changes were made to the permanent directory for this user, either by the user himself, or by another user directing mail to this user.

The MAILLINK System

The MAILLINK support system is an additional set of routines available to the remote user operating through a UCSD system environment. The MAILLINK primarily offers the remote UCSD user a dumb terminal emulation mode, thus allowing the user to operate the MAILMAN system as though the user were a local terminal user. However, in addition to the basic MAILMAN commands, a remote user operating the MAILLINK support program may also transmit mail or other textual information created at the remote site in a detached mode. The X)fer command within the MAILMAN system sends special control codes to the remotely operating MAILLINK process, causing the MAILLINK to switch its executing environment from that of a dumb terminal to a computer to computer link.

The MAILMAN / MAILLINK combination offers the detached user full information transfer control. The protocol transparently controls transmission requests, cyclical redundancy checks, retransmission requests for frames in which errors are detected, and aborting procedures for transfer attempts whose error rate was above allowed limits. The protocol is UCSD 1.5 compatible and will operate on a remote dial-up line up to 1200 baud, with Racal-Vadic 3400 full duplex protocol.

Two additional routines are supplied for the remote UCSD computer site. They are the "INCNVTR" and "MECNVTR" programs.

These two programs offer compatibility between the editor formats of the separate host environments. The "INCNVTR" program will convert a text file created under the UCSD screen oriented editor to a format compatible with the Interdata 8/32 line oriented editor, PEDIT (MEDIT). Likewise, the "MECNVTR" will convert a text file created under the PEDIT (MEDIT) format to a format acceptable with the UCSD screen oriented editor.

To invoke the MAILMAN / MAILLINK communication system from a remote location operating under the UCSD environment, follow the initiation sequence below:

Once the UCSD command line is present, X)ecute the program

```
> X
Execute what file ?  MAILLINK <CR>
```

After a few seconds delay, the MAILLINK header will appear:

```
Western Digital <-> 8/32 Communication Interface
   R00-01     June 1982    K.W. Janne

          type cntrl-q to abort
```

Once this header line is displayed, the communication link between the two machines has been established. The user should then proceed with the normal signon sequence as required by the

host 8/32 system. Once the signon has been validated and the normal system prompt appears, invoke the MAILMAN support program via the MAILMAN css.

```
* MAILMAN <CR>
```

At this point the user is emulating the execution environment of a locally connected user.

All of the commands available to the local user operating the MAILMAN system are now available to this remote user. For descriptions of these commands, see the preceding section. In addition to the locally available commands, the remote user may furthermore utilize the X)fer command. This command will enable the file or textual data transmission to occur.

To initiate this transfer mechanism, the remote user simply executes the X)fer command in the MAILMAN system prompt line. The MAILMAN will generate the appropriate control codes to the MAILLINK process, signaling it to switch from the dumb terminal mode to the transfer protocol.

Upon receiving the control codes to switch from the emulation mode, the following header line will be displayed:

```
UCSD Remote <=> 8/32
File Transfer Control  R00-01
```

Enter X)fer Mode	**1**	**UCSD -> 8/32**
	2	**8/32 -> UCSD**
	3	**ABORT**

From this point, the user directs the system as to the direction of transfer desired. If the user has had second thoughts about the transfer and wishes to abort the transfer, then the third choice would be appropriate. The abort mode will just return the remote user to the dumb terminal emulation mode within the MAILMAN / MAILLINK environment.

Files that are sent to the 8/32 should be format converted prior to the transfer, using the "INCNVTR" routine. Likewise, files that are received from the 8/32 should be reformated via the "MECNVTR" routine prior to use within the UCSD environment editors. Files that are directed towards the 8/32 will be deposited similarly to the C)reate command sequence with the exception that the source file/mail will be transmitted from the remote site first. A user wishing to send the file to himself, may use his own signon ID as the destination address. However, the user must first remove himself from the MAILMAN system and reinvoke the system in order to have the file appear in his own user's mail directory. (See MAILMAN Q)uit command)

Files that are transmited directly to the remote site may be any valid file directory entry including files not entered within the MAILMAN environment.

When the first choice is selected, the UCSD -> 8/32 direction, the following information will be requested:

Enter UCSD source file name .....
Enter destination user ID.....
Enter file / mail subject .....
Is file encrypted (y/n) .....

The source file name should be the output file of the "INCNVTR" program. This will allow the 8/32 editors to review the contents. However, if the eventual destination will be another UCSD system, and the 8/32 MAILMAN is only a routing device, then format conversion is not required.

When all of this information has been collected, an envelope is constructed and transmitted along with the other command frame information. If the transfer establishment succeeds, the transfer will proceed until all blocks have been transmitted. A continuing display of markers will indicate that the transmission is progressing.

If the display should stop for an extended period of time, (several minutes), an unforeseen error has broken the transmission sequence. Because the systems have no concurrent means by which to time-out in an error situation, infinite waits may result. To resolve this situation, warm boot or restart the remote unit and resume processing by issuing several <CR> until the main system prompt line appears. (A common source of such

situations is an unexpected message sent by another 8/32 MTM user via the "ME" command. It is therefore strongly advised that a remote user intending to utilize the remote file transfer mechanisms prevent such message interruption by issuing the "PRE" command upon signon to the system. To reenable message deliveries after exiting the MAILMAN / MAILLINK environment, type "ENA". This will enable other users to once again send MTM messages.)

To invoke the inverse process, the second direction choice must be selected. This choice will display the header and solicit the information listed below:

```
                    8/32 => UCSD
                File Transfer  R00-01

Enter the UCSD dest file name  (must use full name)....
Enter the 8/32 source volume name .....................
Enter the source file   ( 8 chars max )................
Enter the source extn .................................
Enter the source account  ( p/g ) .....................
```

The UCSD file name should include the volume prefix and should not include the extension if the file is of the 8/32 line oriented format. The extension will be appended upon completion of the "MECNVTR" reformater program. If however, the file was previously a UCSD format file transmitted via the MAILMAN system,

the extension ".text" should be included in the UCSD destination name.

The source descriptors in this case will represent files currently residing on the host 8/32 system. The volume name should include only the letters and digits and not the ":" delimiter. Likewise, the extension should only include up to three characters and not the ".".

As in the UCSD -> 8/32 transfer, the inverse transfer will similarly indicate transfer progression by a continued display of markers. And the same error recover sequence should be invoked in similar circumstances.

The transfer mechanisms will detect line errors and other related problems and will try several times to generate a correct transfer of data. If however the problem persists, the transfer will be aborted and a failure message will be displayed prior to the redisplay of the main system prompt line. Likewise, when the transfer is completed successfully, a successful completion message will be displayed and control resume to the main system prompt line and the remote unit resort to operating in the dumb terminal emulation mode again.

When the user wishes to exit from the MAILMAN / MAILLINK environment, the user should first Q)uit the MAILMAN system. This will restore and update the users mail directories. After

this has been accomplished and the system is back in the MTM command mode, the user should signoff the 8/32 using the standard signoff sequence. Once the signoff message has been displayed, the user may then finish the termination sequence by depressing the control key and the "Q" key simultaneously. This will return the remote user to the UCSD operating system prompt line. At any point in the MAILMAN / MAILLINK connection the user may remove himself from the connected environment by depressing the control - "Q" keys. However, this is not recommended because of the undefined state that this might leave the user's mail directory in.

INCNVTR / MECNVTR

These two support programs operate outside of the MAILMAN / MAILLINK environment. Their operations are similar, simply inverses of each other, and therefore, will be discussed together. To run either of these programs, the user should X)ecute from the UCSD prompt line the desired program code file. The source file name will then be solicited as well as the destination name. When the source files have been found and the destination file created, the conversion will proceed. A continuing marker display will monitor the conversion progress and a completion code result will be displayed following the conversion.

Several convensions should be followed regarding naming of the source and destination file. Files that are 8/32 format should be referred to within the UCSD environment as ".DATA" files and UCSD file should be categorized as ".TEXT" files. In order to achieve this convention, 8/32 files should not be referenced with an extension attribute, (i.e. #vol:fn) whereas UCSD files should always be referenced with the extension ".TEXT" (i.e. "#vol:fn.TEXT") This will help to establish a consistent pattern to the conversion process and file directory maintenance.

V. IMPLEMENTOR'S GUIDE

This first section is provided for the user desiring to implement the remote node UCSD environment MAILLINK support system for the MAILMAN /MAILLINK communication system. All references to the MAILLINK section will assume that the user is operating a UCSD compatible environment of revision 1.5 or greater.

The primary implementation characteristic of the MAILLINK system is the input / output dependent features. In the initial installation configuration, the primary communication link to the MAILMAN system is through the logical REMOTE: device port. If the target machine has several serial ports available, this port assignment may be redefined. The unit assignment is made in the program's CONST section. To assign the remote link to another available serial port, the user should set the new constant declaration for the REMOTE: device. Care must also be taken to assure that the new assignment is not in conflict with other predefined logical to physical mappings.

The MAILMAN / MAILLINK system has been designed to operate at a maximum date transmission rate of 1200 baud asynchrononsly. This is the current maximum modem rate available to dial in users. However, systems with slow UCSD interpreters may find that certain optimization within the character acquisition

procedures necessary. Low level, machine dependent input / output routines may need to be constructed for such systems to operate at the 1200 baud rate in the dumb terminal emulation mode. This modification may be avoided by simply operating at the lower connection speed of 300 baud.

In discussing the implementation of the MAILMAN system on the INTERDATA 8/32 system, several areas are available for user modification. The primary user modifiable area would deal with the printer naming and assignment values associated with the F)orward command. As described in the "Users Guide" section, the printers may be selected as the destination address of a Forward command. Such assignment is based upon the character matching of the printer device mnemonics. The default values are "PRINTER" for a high speed line printer, and the "SPINWTR" value for a letter quality printing device. To reassign to different logical devices, the CONST declarations in the program prefix section should be altered to reflect the changes desired.

To initially load the MAILMAN system, a command substitution system file has been created to automate this procedure. The load and initialize values assume that a maximum number of potential uses is 255. To extend beyond that value would require significant modification to the user directory structure because of the fact that the largest directory maintainable within MTM is 255 records. Therefore, increases in the initial size

declarations should be avoided. However, decreases will not effect the structure and will improve the performance characteristic of the MAILMAN system.

In the MAILMAN system, there currently is no encryption and decryption algorithms; only the envelope flag is modified. The system has been constructed such that encoding algorithms may be easily incorporated. The E)ncrypt and D)ecrypt command are processed by the Cypher procedure which will assign the desired target file to logical unit 5. If the assignment is succesful, the proper algorithm procedure is then called. The user incorporated algorithms should be placed in the procedures Encrypt_Alg, and Decrypt_Alg. No other assignments will need to be made. The Cypher procedure will restore the logical unit to its previous assignment following completion of the encoding.

A similar sequence has been established for incorporating other editors for use with the C)reate command. It generally is assumed that the input file will be null and the output file the generated mail. The current editor utilizes the logical units 1 and 2 for input and output respectively, where logical unit 1 is a null file. The user installed editors may reassign any logical unit for the purpose of working with the editor so long as the original unit assignments are re-established upon completion.

The foundation has been installed for a screen oriented

editor to supplement the present line oriented editor. An addition link has been provided for another editor, labeled "other editor". These three are the available choices for selection via the S)et_Edit command.

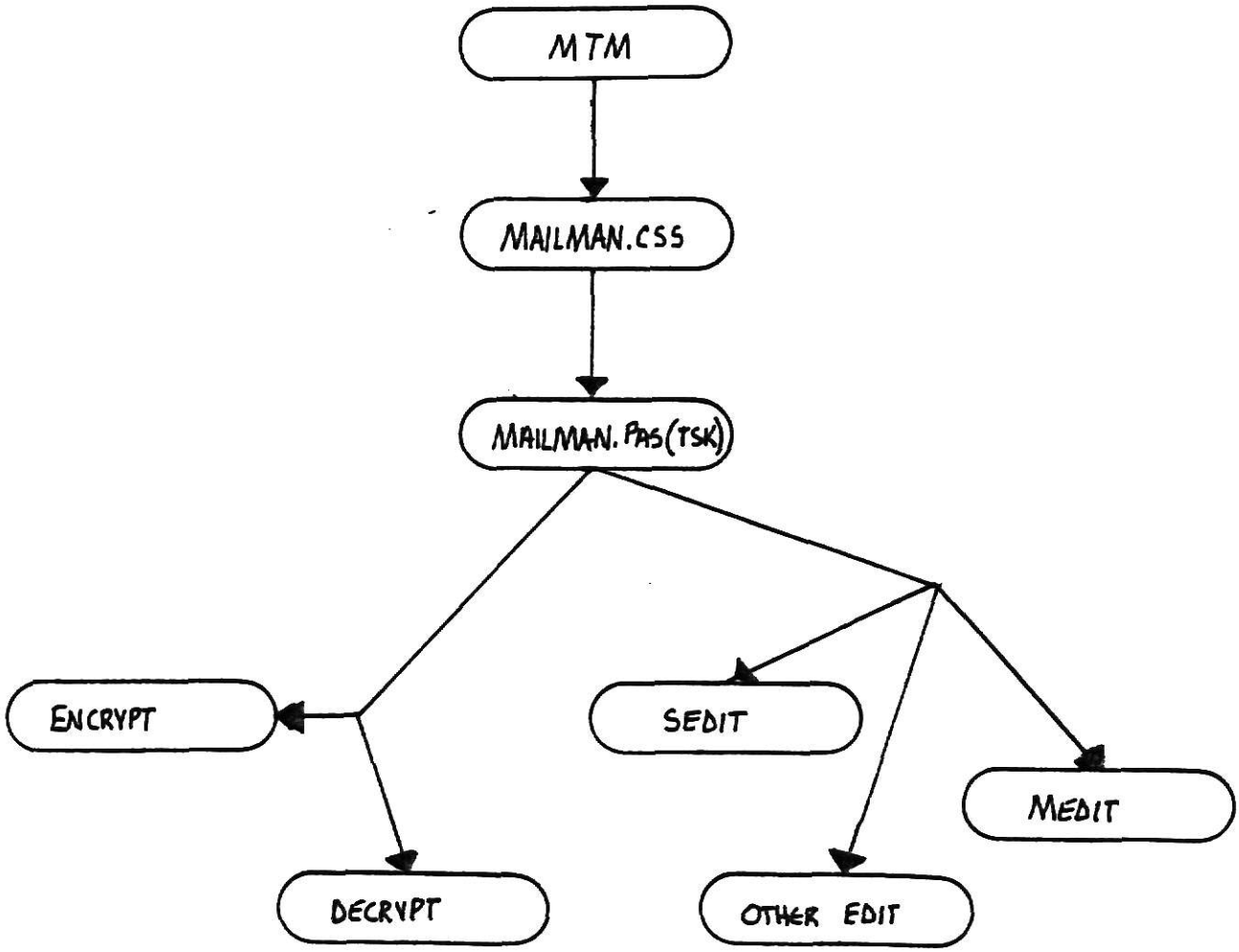


Figure 3 - Program Invocation Sequence

VI. SUMMARY

In reviewing the initial intent of this project, one can not help but notice that the support environment for remote system interaction became a far less significant component than was originally anticipated. Although the combination of transfer mechanisms within the electronic mail environment are natural extensions, the degree to which the systems had to interact to accomodate architectural discrepancies and dependencies proved to be a compromising factor in terms of individual efficiency.

It now appears that the overall intent could have been better served by separating the mail and transfer entities to a greater degree. In evaluating use patterns of the MAILMAN / MAILLINK system, observation has shown that combining the elements frequently introduces added confusion. Many times the desired application is simply a transfer of source code between machines. In this case, the MAILMAN system puts significant overhead into the transfer process. The user must "mail" the information rather than just specifying for a transfer to take place.

Similar conclusions can be made regarding the electronic mail support. In actual applications, a probable majority of users will never utilize the transfer mechanisms, using the MAILMAN system for its mail functions independently.

Therefore, future revisions would probably entail the isolation and separation of remote file transfer mechanism from the electronic mail support. This realization is not however, a confession of misguided intent. It is rather an indication of the fact that the electronic mail aspect of the project became substantially greater than expected. This investment thus produced a very viable entity within itself.

In final review of the MAILLINK system support, once again the issue of compromise appears. The question remains as to whether or not the degradation of features and efficiency, by restricting revision compatibility, is justified by the extensions of distribution bases.

From the system design point of view, the compatibility rationale has strong merits. While it can be assumed users of the MAILLINK support system will undoubtably be critical of various components within the system, the compatible base does provide the adventurous programmer the neccessary framework for constructing more efficent, machine and revision dependent implementations.

The MAILMAN system has been implemented with very basic command functions. It has a very high potential for future extensions. Various public key encryption techniques are finding their way into the electronic mail system environments. Signature

and untraceable mail [4] are several aspects of electronic mail that have developed recently; which in this project we made no attempt to address. Electronic mail environments are quickly becoming an office mechansim, instead of a corporate toy, and uses as well as abuses are bound to be yet discovered.

SELECTED BIBLIOGRAPHY

1. Birrell, Andrew D., Levin, R., Needham, R.M., and Schroeder, Michael D., "Grapevine: An Exercise in Distributed Computing",Communications of the A.C.M., Vol. 25, No. 4, pp. 260-274.

2. Birrell, Andrew D., Levin, R., Needham, R.M., and Schroeder, Michael D., "Grapevine: An Exercise in Distributed Computing, A Summary",Proceeding of the Eighth Symposium on Operating Systems Principles, Vol. 15, No. 5, Dec. 81, pp 178-179.

3. Blanc, Robert and Cotton, Ira, Computer Networking, I.E.E.E. Press, New York 1976.

4. Chaum, David L., "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", Communications of the A.C.M., Vol. 24, No. 2, pp. 84-88.

5. Davies, D.W., Barber, D.L., Price, W.L., and Solomonides, C., Computer Networks and Their Protocols, John Wiley & Sons, New York, 1979.

6. "EMS News",Computer World, March 22, 1982 p 43.

7. Denning, Peter J., "Electronic Junk",Communications of the A.C.M., Vol. 25, No. 3, pp. 163-165.

8. Helmers, Carl, "The Grass Roots Electronic Post Office",Byte, Vol. 5, No. 6, pp. 6-11.

9. North Star Pascal System Reference Manual, North Star Computers, Inc., Ver. 1.0 Berkeley, CA. 1979.

10. O.S. 32 Programmers Reference Guide, Publ. No. S29-613, Rev. 02, Perkin-Elmer Interdata Division, 2 Crescent Place, Ocean Port, New Jersey. 1979.

11. Panko, R.R. and Panko, R.U., "A Survey of EMS Users at DARCOM",Computer Networks, Vol. 5, No. 1, pp. 19-35.

12. Schicker, P., "The Computer Based MAil Environment - An Overview", Computer Networks, Vol. 5, No. 6, pp. 435-444.

13. Tanenbaum, Andrew S., Computer Networks, Prentice Hall, New Jersey, 1981.

14. Tomanek, Gerald, "Implementing Electronic Mail in a Telephone System", Proceedings, National Computer Conference, 1980, pp. 527-531.

15. Western Digital WD / 90 Pascal Microengine Reference Manual, The Microengine Company, 1979.

16. Wirth, Niklaus and Jensen, Kathleen, Pascal Users Manual and Report, Spring-Verley, New York 1978.

17. Young, Robert and Wallentine, Virgil, Pascal / 32 Language Definition, Dept. of Computer Science, K.S.U., 1978.

Appendix

A. MAILMAN Source Listing

B. MAILLINK Source Listing

C. INCNVTR Source Listing

D. MECNVTR Source Listing

```
"PER BRINCH HANSEN                          *   AS MODIFIED FOR THE INTERDATA
                                            *      8/32 UNDER OS/32-MT AT
 INFORMATION SCIENCE                        *   DEPARTMENT OF COMPUTER SCIENCE
 CALIFORNIA INSTITUTE OF TECHNOLOGY         *      KANSAS STATE UNIVERSITY
                                            *
 UTILITY PROGRAMS FOR                       *
 THE SOLO SYSTEM                            *
                                            *
 18 MAY 1975                                *            1 DEC 1976"


"############
#  PREFIX  #
############"


CONST NL = '(:10:)';   FF = '(:12:)';   CR = '(:13:)';   EM = '(:25:)';

CONST PAGELENGTH = 512;
TYPE PAGE = ARRAY (.1..PAGELENGTH.) OF CHAR;

CONST LINELENGTH = 132;
TYPE LINE = ARRAY (.1..LINELENGTH.) OF CHAR;

CONST IDLENGTH = 12;
TYPE IDENTIFIER = ARRAY (.1..IDLENGTH.) OF CHAR;

TYPE FILE = 1..7;

TYPE FILEKIND = (EMPTY, SCRATCH, ASCII, SEQCODE, CONCODE);

TYPE FILEATTR = RECORD
                  KIND: FILEKIND;
                  ADDR: INTEGER;
                  PROTECTED: BOOLEAN;
                  NOTUSED: ARRAY (.1..5.) OF INTEGER
                END;

TYPE IODEVICE = 0..255;   "LOGICAL DEVICE NUMBERS"

TYPE IOOPERATION = (INPUT, OUTPUT, MOVE, CONTROL);

TYPE IOARG = (WRITEEOF, REWIND, UPSPACE, BACKSPACE);

TYPE IORESULT =
  (COMPLETE, INTERVENTION, TRANSMISSION, FAILURE,
   ENDFILE, ENDMEDIUM, STARTMEDIUM);

TYPE IOPARAM = RECORD
```

```
                OPERATION: IOOPERATION;
                STATUS: IORESULT;
                ARG: IOARG
              END;

TYPE TASKKIND = (INPUTTASK, JOBTASK, OUTPUTTASK);

TYPE ARGTAG =
  (NILTYPE, BOOLTYPE, INTTYPE, IDTYPE, PTRTYPE);

TYPE POINTER = @BOOLEAN;

TYPE ARGTYPE = RECORD
                 CASE TAG: ARGTAG OF
                   NILTYPE, BOOLTYPE: (BOOL: BOOLEAN);
                   INTTYPE: (INT: INTEGER);
                   IDTYPE: (ID: IDENTIFIER);
                   PTRTYPE: (PTR: POINTER)
               END;

CONST MAXARG = 10;
TYPE ARGLIST = ARRAY (.1..MAXARG.) OF ARGTYPE;

TYPE ARGSEQ = (INP, OUT);

TYPE PROGRESULT =
  (TERMINATED, OVERFLOW, POINTERERROR, RANGEERROR, VARIANTERROR,
   HEAPLIMIT, STACKLIMIT, CODELIMIT, TIMELIMIT, CALLERROR);


"*****************************************************************"
"                                                                 "
"          OS/32MT3 SVC INTERFACE ROUTINE TYPES                   "
"                                                                 "
"*****************************************************************"


"MISCELLANEOUS DATA TYPES"

TYPE CHAR1 = PACKED ARRAY [1..1] OF CHAR;
TYPE CHAR3 = PACKED ARRAY [1..3] OF CHAR;
TYPE CHAR8 = PACKED ARRAY [1..8] OF CHAR;
TYPE CHAR4 = PACKED ARRAY [1..4] OF CHAR;
TYPE CHAR16 = ARRAY [1..16] OF CHAR;
TYPE CHAR28 = ARRAY [1..28] OF CHAR;

"SVC1 PARAMETER BLOCK"

TYPE SVC1_BLOCK = RECORD
      SVC1_FUNC: BYTE;          "FUNCTION CODE"
```

```
        SVC1_LU: BYTE;              "LOGICAL UNIT NUMBER"
        SVC1_STAT: BYTE;            "DEV-INDEP STATUS"
        SVC1_DEV_STAT: BYTE;        "DEV-DEPENDENT STATUS"
        SVC1_BUFSTART: INTEGER;     "ADDRESS(BUFFER)"
        SVC1_BUFEND: INTEGER;       "ADDRESS(BUFFER)+SIZE(BUFFER)-1"
        SVC1_RANDOM_ADDR: INTEGER;  "RANDOM ADDRESS FOR DASD"
        SVC1_XFER_LEN: INTEGER;     "TRANSFER LENGTH"
        SVC1_RESERVED: INTEGER;     "RESERVED FOR ITAM USE"
       END;

"SVC 1 FUNCTION CODES"

CONST SVC1_DATA_XFER = #00;      SVC1_COMMAND   = #80;
      SVC1_READ      = #40;      SVC1_WRITE     = #20;
      SVC1_TESTSET   = #60;      SVC1_TESTIO    = #00;
      SVC1_ASCII     = #00;      SVC1_BINARY    = #10;
      SVC1_PROCEED   = #00;      SVC1_WAIT      = #08;
      SVC1_SEQL      = #00;      SVC1_RANDOM    = #04;
      SVC1_CWAIT     = #00;      SVC1_UNC_PROC  = #02;
      SVC1_FORMAT    = #00;      SVC1_IMAGE     = #01;

      SVC1_REW       = #40;      SVC1_BSR       = #20;
      SVC1_FSR       = #10;      SVC1_WFM       = #08;
      SVC1_FSF       = #04;      SVC1_BSF       = #02;
      SVC1_RESV_FN   = #01;

"SVC 1 DEVICE-INDEPENDENT STATUS CODES"

CONST SVC1_OK        = #00;      SVC1_ERROR     = #80;
      SVC1_ILGFN     = #40;      SVC1_DU        = #20;
      SVC1_EOM       = #10;      SVC1_EOF       = #08;
      SVC1_UNRV      = #04;      SVC1_RECV      = #02;
      SVC1_ILGLU     = #01;      SVC1_DEVBUSY   = #7F;


"SVC2 PEEK PARM BLOCK"

    TYPE SVC2_PEEK_PARM
      = RECORD
          SVC2_OP : BYTE;        "OPTION   X 00 "
          SVC2_CODE : BYTE;      "SVC-2 CODE"
          SVC2_NLU : BYTE;       "MAXIMUM NO OF LOGICAL UNITS"
          SVC2_MPRI : BYTE;      "HIGHEST PRIORITY TASK MAY EXECUTE"
          SVC2_OSID : CHAR8;     "OPERATING SYSTEM NAME"
          SVC2_TASK : CHAR8;     "USER TASK NAME"
          SVC2_CTSW : INTEGER;   "CURRENT TASK STATUS WORD"
          SVC2_OPT : SHORTINTEGER; "TASK OPTIONS"
          SVC2_RESRV : SHORTINTEGER; "SYSTEM RESERVED"

        END;
```

```
"FILE DESCRIPTOR FOR SVC7 REQUESTS"

TYPE FD_TYPE = PACKED RECORD
      VOLN: CHAR4;                "VOLUME NAME"
      FN: CHAR8;                  "FILE NAME"
      EXTN: CHAR3;                "EXTENSION"
      ACCT: CHAR;                 "ACCOUNT NUMBER CODE"
     END;

"SVC 7 PARAMETER BLOCK"

TYPE SVC7_BLOCK = RECORD
      SVC7_CMD: BYTE;             "COMMAND"
      SVC7_MOD: BYTE;             "MODIFIER/DEVICE TYPE"
      SVC7_STAT: BYTE;            "STATUS"
      SVC7_LU: BYTE;              "LOGICAL UNIT NUMBER"
      SVC7_KEYS: SHORTINTEGER;    "READ/WRITE KEYS"
      SVC7_RECLEN: SHORTINTEGER;"LOGICAL RECORD LENGTH"
      SVC7_FD: FD_TYPE;           "FILE DESCRIPTOR"
      SVC7_SIZE: INTEGER;         "FILE(/INDEX) SIZE"
     END;

"SVC 7 COMMAND CODES"


CONST SVC7_ALLOC        = #80;      SVC7_ASSIGN     = #40;
      SVC7_CHAP         = #20;      SVC7_RENAME     = #10;
      SVC7_REPROT       = #08;      SVC7_CLOSE      = #04;
      SVC7_DELETE       = #02;      SVC7_CHECKPT    = #01;
      SVC7_FETCH_ATTR   = #00;

"SVC 7 MODIFIER CODES - ACCESS PRIVILEGES"

CONST SVC7_AP_SRO       = #00;      SVC7_AP_ERO     = #20;
      SVC7_AP_SWO       = #40;      SVC7_AP_EWO     = #60;
      SVC7_AP_SRW       = #80;      SVC7_AP_SREW    = #A0;
      SVC7_AP_ERSW      = #C0;      SVC7_AP_ERW     = #E0;

"SVC 7 MODIFIER CODES - BUFFERING/FILE TYPE"

CONST SVC7_BUF_DEFAULT  = #00;      SVC7_BUF_PHYS   = #08;
      SVC7_BUF_LOG      = #10;      SVC7_BUF_SVC15  = #18;
      SVC7_FTYPE_CONTIG = #00;      SVC7_FTYPE_CHAIN= #01;
      SVC7_FTYPE_INDEX  = #02;      SVC7_FTYPE_ITAM = #07;

"SVC 7 STATUS ERROR CODES"

CONST SVC7_OK  = 0;
```

```
PROCEDURE READ(VAR C: CHAR);
PROCEDURE WRITE(C: CHAR);

PROCEDURE OPEN(F: FILE; ID: IDENTIFIER; VAR FOUND: BOOLEAN);
PROCEDURE CLOSE(F: FILE);
PROCEDURE GET(F: FILE; P: INTEGER; VAR BLOCK: UNIV PAGE);
PROCEDURE PUT(F: FILE; P: INTEGER; VAR BLOCK: UNIV PAGE);
FUNCTION LENGTH(F: FILE): INTEGER;

PROCEDURE MARK(VAR TOP: INTEGER);
PROCEDURE RELEASE(TOP: INTEGER);

PROCEDURE IDENTIFY(HEADER: LINE);
PROCEDURE ACCEPT(VAR C: CHAR);
PROCEDURE DISPLAY(C: CHAR);

PROCEDURE READPAGE(VAR BLOCK: UNIV PAGE; VAR EOF: BOOLEAN);
PROCEDURE WRITEPAGE(BLOCK: UNIV PAGE; EOF: BOOLEAN);
PROCEDURE READLINE(VAR TEXT: UNIV LINE);
PROCEDURE WRITELINE(TEXT: UNIV LINE);
PROCEDURE READARG(S: ARGSEQ; VAR ARG: ARGTYPE);
PROCEDURE WRITEARG(S: ARGSEQ; ARG: ARGTYPE);

PROCEDURE LOOKUP(ID: IDENTIFIER; VAR ATTR: FILEATTR; VAR FOUND: BOOLEAN);

PROCEDURE IOTRANSFER
  (DEVICE: IODEVICE; VAR PARAM: IOPARAM; VAR BLOCK: UNIV PAGE);

PROCEDURE IOMOVE(DEVICE: IODEVICE; VAR PARAM: IOPARAM);

FUNCTION TASK: TASKKIND;

PROCEDURE RUN(ID: IDENTIFIER; VAR RETURN_LINE: LINE;
              VAR LINE: INTEGER; VAR RESULT: PROGRESULT);

PROCEDURE RESET(LU: IODEVICE);

PROCEDURE BREAKPT ( LN : INTEGER );

"$EJECT"


"                 ******************************************
                  *                                        *
                  *          MAIL - MANAGER  R00-03        *
                  *                  06/82                 *
                  *                K.W. JANNE              *
```

```
                    *                                              *
                    ************************************************"


PROGRAM MAILMAN3;

CONST   LOWERA = '(:97:)';              LOWERZ = '(:122:)';
        OK = TRUE;                      NOK = FALSE;
        UC_TO_LC = 32;
        MAX_ACTIV_USERS = 255;          SYS_DIR_ENTRY_LENGTH = 24;
        ENVELOPE_SIZE = 62;

    "PRINTER DEVICES"

        PRINTER = 'PR:     ';
        SPINWTR = 'SP18:   ';
        PRINTER_LU = 7;
        SPINWTR_LU = 8;

    "XFER CONSTS"

        ACK           = '1';                "XFER_COMD REQUESTS"
        NAK           = '2';
        ABORT_XFER    = '3';
        REQ_SEND      = '4';  "CON: -> 8/32"
        REQ_REC       = '5';  "8/32 -> CON"
        EOM_FLAG      = '6';
        TRANSFER      = '7';
        FRAME_LENGTH = 67;           "LENGHT OF TRANSFER FRAME "
        FRAME_DATA_LENGTH = 64;      "VALID DATA CHARS IN FRAME"

        XFER_IN_LU  = 1;
        XFER_OUT_LU = 2;

        STX = '(:02:)';           "SIGNALS START OF FILE TRNASFER"
        ETX = '(:03:)';           "SIGNALS END OF FILE TRANSFER  "

        XFER_SUBJECT_OFF = 8;              "USED IN REQ_FRAME"
        XFER_DEST_OFF    = 0;            "OFFSETS WITHIN FRAME_DATA "
        XFER_ENCRYPT_OFF = 24;
        XFER_VOLN_OFF    = 0;
        XFER_FN_OFF      = 4;
        XFER_EXTN_OFF    = 12;
        XFER_ACCT_OFF    = 15;

        NAK_LIMIT = 3;                     "NUMBER OF NAKS BEFORE ABORT"


    "DELAY TIME CONSTANTS"
```

```
        SHORT   = 10;
        MEDIUM  = 1000;
        LONG    = 10000;


TYPE RUN_TYPE = ( DEBUG, USER, EXEC );

     ENVELOPE_PTR = ^ ENVELOPE;

     ENVELOPE = RECORD
        DEST_NAME   : CHAR8;
        SOURCE_NAME : CHAR8;
        SUBJECT     : CHAR16;
        POST_MARK   : CHAR8;
        BOX_ADDR    : FD_TYPE;
        NEXT_LETTER : ENVELOPE_PTR;
        ENCRYP : BOOLEAN;
        END; "RECORD"

    MAIL_COMMANDS = ( HELP, INQUIRE, GET, CREATE, REMOVE, MAIL_,
                      FORWARD_M,SET_EDIT, NVIRONMENT, PURGE, BREAK,
                      ENCRYPT, DECRYPT, TYPE_M, XFER, LIST_STAT,
                      QUIT_CMD );

    COUNTERS = ( TRAFFIC, DIRECTORY, NEITHER );

    EDITORS = ( MEDIT, SEDIT, OTHEREDIT );

    FRAME_DATA = PACKED ARRAY [ 1..FRAME_DATA_LENGTH ] OF CHAR;

    FRAME =  PACKED RECORD
             XFER_COMD : CHAR;      "COMMAND CODE CONST"
             BLOCK_NO : CHAR;       "ASCII CHAR CODE MOD 256"
             DATA : FRAME_DATA;
             CRC : CHAR;
            END; "FRAME RECORD"

    CYPHER_DIR = ( ENCODE, DECODE );

CONST MAX_CMDS = QUIT_CMD;
      MIN_CMDS = HELP;


VAR RUN_MODE          : RUN_TYPE;
    MAIL_PTR,MAIL_HEAD,MAIL_TAIL,MAIL_TRAIL,FREE_PTR : ENVELOPE_PTR;
    MAIL : ENVELOPE;
    DIRECT_CT, TRAFFIC_CT : INTEGER;
    COMMAND : ARRAY [MAIL_COMMANDS] OF CHAR;
    ABORT : BOOLEAN;
    DIRTY : BOOLEAN; "SET IF NEED TO REVISE DISK DATA BASE"
```

```
    USER_CMD : MAIL_COMMANDS;
    LOGON_ID : CHAR8;
    USRS_DIR_NAME : FD_TYPE;
    FILE_NAME : FD_TYPE;
    EXEC_STATUS : BOOLEAN;
    MAIL_EDITOR : EDITORS;
    ERROR_LINE : INTEGER;
    RET_LINE : LINE;
    RUN_RESULT : PROGRESULT;       "USED FOR RUN OVERLAY RETURN PARAM"
    OVERLAY_NAME : IDENTIFIER;


   "XFER GLOBAL VARS"

    XFER_SVC1_IN,XFER_SVC1_OUT : SVC1_BLOCK;
    BLOCK : PAGE;
    XFER_FRAME : FRAME;
    FRAME_CT, FRAME_OFFSET, BLOCK_CT : INTEGER;
    EOF_FLAG : BOOLEAN;


"  ************************************************************************
   *                                                                      *
   *                        UTILITY PROCEDURES                            *
   *                                                                      *
   ************************************************************************"



PROCEDURE SVC1(VAR BLOCK:SVC1_BLOCK);EXTERN;
PROCEDURE SVC2FDAT(VAR MMDDYY : CHAR8); EXTERN;
PROCEDURE SVC2TODW(TOD : INTEGER); EXTERN;
PROCEDURE SVC2FTIME (VAR TIME:INTEGER; VAR HHMMSS : CHAR8); EXTERN;
PROCEDURE SVC2PEEK ( VAR SVC2_PARM : SVC2_PEEK_PARM ); EXTERN;
PROCEDURE SVC3 ( RETURN_CODE : INTEGER ); EXTERN;
PROCEDURE SVC7(VAR BLOCK:SVC7_BLOCK);EXTERN;


PROCEDURE WRITETEXT(TEXT: LINE);
"   THIS PROCEDURE OUTPUTS A LINE OF TEXT TO THE CONSOLE.
    THE CHARACTER NL ('(:10:)') SIGNIFIES THE END OF THE LINE.
    A NULL ('(:0:)') ALSO SIGNIFIES END OF TEXT."
VAR I: INTEGER;
   C: CHAR;
BEGIN
   I:= 1;
   REPEAT
       C:= TEXT(.I.);
       DISPLAY(C);
```

```
        I:= SUCC(I);
    UNTIL (C = NL) OR (C = '(:0:)');
END;  "WRITETEXT"


PROCEDURE WRINT ( INT : INTEGER );
VAR REM, DIGIT, I: INTEGER; NUMBER: ARRAY[1..6] OF CHAR;
BEGIN
    DIGIT:= 0;
    REM:= INT;
    REPEAT
       DIGIT:= SUCC(DIGIT);
       NUMBER[DIGIT]:= CHR(REM MOD 10 + ORD('0'));
       REM:= REM DIV 10;
    UNTIL REM = 0;
    FOR I:= 6 DOWNTO DIGIT+1 DO DISPLAY(' ');
    FOR I:= DIGIT DOWNTO 1 DO DISPLAY(NUMBER[I]);
END;  "WRINT"


FUNCTION UC (C:CHAR):CHAR;
BEGIN
  IF ( C>= LOWERA ) AND ( C<= LOWERZ )
     THEN UC := CHR ( ORD (C) - UC_TO_LC )
     ELSE UC := C;
END; "LOWER TO UPPER CASE CONVERS"


PROCEDURE ASSIGN_FILE (FILE_N : FD_TYPE; LU : INTEGER;
    ACCESS_RIGHT : BYTE; VAR OK_STATUS : BOOLEAN); FORWARD;


PROCEDURE ASSIGN_CON ( LU : INTEGER ); FORWARD;


PROCEDURE UPDATE_CT ( WHICH_ONE : COUNTERS );
VAR DIR_NAME : FD_TYPE;
    SVC1_PARM : SVC1_BLOCK;
    STATUS : BOOLEAN;
BEGIN
  WITH DIR_NAME DO BEGIN
    VOLN := 'SYS3';
    FN := 'MAILDIR ';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH"
  ASSIGN_FILE ( DIR_NAME,3,SVC7_AP_SREW + SVC7_FTYPE_INDEX,STATUS);
  IF STATUS = OK THEN BEGIN
    WITH SVC1_PARM DO BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
```

```
        SVC1_LU := 3;
        SVC1_BUFSTART := ADDRESS ( DIRECT_CT );
        SVC1_BUFEND := SVC1_BUFSTART + 3;
        SVC1 ( SVC1_PARM );
        SVC1_BUFSTART := ADDRESS ( TRAFFIC_CT );
        SVC1_BUFEND := SVC1_BUFSTART + 3;
        SVC1 ( SVC1_PARM );
        CASE WHICH_ONE OF
          TRAFFIC : TRAFFIC_CT := SUCC ( TRAFFIC_CT );
          DIRECTORY : DIRECT_CT := SUCC ( DIRECT_CT );
          NEITHER : ;
          END; "CASE"
        SVC1_FUNC := SVC1_COMMAND + SVC1_BSR;
        SVC1 (SVC1_PARM);          "RESET TO BEGINING OF FILE AGAIN"
        SVC1 (SVC1_PARM);
        SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
        SVC1_LU := 3;
        SVC1_BUFSTART := ADDRESS (DIRECT_CT);
        SVC1_BUFEND := SVC1_BUFSTART + 3;
        SVC1 (SVC1_PARM);
        SVC1_BUFSTART := ADDRESS (TRAFFIC_CT);
        SVC1_BUFEND := SVC1_BUFSTART + 3;
        SVC1 (SVC1_PARM);
        END; "WITH"
        CLOSE (1);  "LU -3"
      END "STATUS OK"
     ELSE WRITETEXT('ASSIGN ERROR  MAILDIR.SYS (:10:)');
END; "UPDATE CT"


PROCEDURE GET_NEW_PTR ( VAR PTR : ENVELOPE_PTR );
BEGIN
  IF FREE_PTR = NIL
    THEN NEW (PTR)
    ELSE BEGIN
      PTR := FREE_PTR;
      FREE_PTR := FREE_PTR^.NEXT_LETTER;
      END; "ELSE"
END;"GET_NEW_PTR"


PROCEDURE RECYCLE_PTR ( VAR PTR : ENVELOPE_PTR );
BEGIN
  PTR^.NEXT_LETTER := FREE_PTR;
  FREE_PTR := PTR;
END;"RECYCLE_PTR"


PROCEDURE PACK_INT ( INT:INTEGER; VAR STR: CHAR8 );
VAR REM, DIGIT, I : INTEGER;
```

```
BEGIN
  DIGIT := 9;
  REM := INT;
  REPEAT
    DIGIT := PRED(DIGIT);
    STR [DIGIT] := CHR (REM MOD 10 + ORD ('0'));
    REM := REM DIV 10;
    UNTIL REM = 0;
  FOR I:= 2 TO DIGIT - 1 DO STR [ I ] := '0';
END; "PACK_INT"


PROCEDURE DELAY_IT ( WAIT_LENGTH : INTEGER );
VAR I,J : INTEGER;
BEGIN
  FOR I := 1 TO WAIT_LENGTH DO J := J + 0;
END;


 " ***************************************************************
   *                                                             *
   *              FILE MANIPULATION PROCEDURES                   *
   *                                                             *
   ***************************************************************"



PROCEDURE CREATE_FILE (FILE_N : FD_TYPE; REC_LENGTH : INTEGER;
                       VAR OK_STATUS : BOOLEAN);
VAR SVC7_PARM : SVC7_BLOCK;
BEGIN
  WITH SVC7_PARM DO BEGIN
    SVC7_CMD := SVC7_ALLOC;
    SVC7_MOD := SVC7_FTYPE_INDEX;
    SVC7_RECLEN := REC_LENGTH;
    SVC7_FD := FILE_N;
    SVC7_SIZE := 1;
    SVC7_KEYS := 0;
    SVC7 ( SVC7_PARM );
    IF SVC7_STAT = SVC7_OK THEN OK_STATUS := TRUE
                           ELSE OK_STATUS := FALSE;
  END; "WITH"
END; "CREATE_FILE"


PROCEDURE CLOSE_FILE ( LU : INTEGER );
VAR SVC7_PARM : SVC7_BLOCK;
BEGIN
  WITH SVC7_PARM DO BEGIN
```

```
    SVC7_CMD := SVC7_CLOSE;
    SVC7_LU := LU;
    SVC7 (SVC7_PARM);
    END; "WITH"
END; "CLOSE FILE "


PROCEDURE ASSIGN_FILE "( FILE_N : FD_TYPE; LU : INTEGER ;
                         ACCESS_RIGHT :BYTE; VAR OK_STATUS : BOOLEAN)";
VAR SVC7_PARM : SVC7_BLOCK;
BEGIN
  CLOSE_FILE ( LU );        "MAKE SURE ITS CLOSED FIRST"
  WITH SVC7_PARM DO BEGIN
    SVC7_CMD := SVC7_ASSIGN;
    SVC7_LU := LU;
    SVC7_FD := FILE_N;
    SVC7_MOD := ACCESS_RIGHT ;
    SVC7 ( SVC7_PARM );
    IF SVC7_STAT <> 0 THEN OK_STATUS := FALSE
                      ELSE OK_STATUS := TRUE;
    END; "WITH"
  RESET (LU);
END; "ASSIGN_FILE"


PROCEDURE DELETE_FILE ( DELETE_FD :FD_TYPE; VAR OK_STATUS : BOOLEAN );
VAR SVC7_PARM : SVC7_BLOCK;
BEGIN
  WITH SVC7_PARM DO BEGIN
    SVC7_CMD := SVC7_DELETE;
    SVC7_FD := DELETE_FD;
    SVC7 (SVC7_PARM);
    OK_STATUS := (SVC7_STAT = SVC7_OK);
  END; "WITH"
END; "DELETE_FILE"


PROCEDURE RENAME_FILE ( OLD_FD : FD_TYPE; NEW_FD : FD_TYPE;
                        VAR OK_STATUS : BOOLEAN );
VAR SVC7_PARM : SVC7_BLOCK;
    STATUS : BOOLEAN;
BEGIN
  IF OLD_FD <> NEW_FD THEN BEGIN
    ASSIGN_FILE (OLD_FD,3,SVC7_AP_ERW+SVC7_FTYPE_INDEX,STATUS);
    IF STATUS = OK THEN BEGIN
      WITH SVC7_PARM DO BEGIN
        SVC7_CMD := SVC7_RENAME;
        SVC7_LU := 3;
        SVC7_FD := NEW_FD;
        SVC7 ( SVC7_PARM );
```

```
        IF SVC7_STAT = SVC7_OK THEN OK_STATUS := TRUE
                               ELSE OK_STATUS := FALSE;
        CLOSE (1); "LU -3"
       END; "WITH"
      END "STATUS OK"
     ELSE WRITETEXT('ASSIGN ERRORS OLD FD (:10:)');
   END " <> "
   ELSE OK_STATUS := TRUE;   "KLUDGE FOR USE ON PRIVATE ACCOUNT"
END; "RENAME_FILE"


PROCEDURE FIND_FILE ( F_DESC : FD_TYPE; VAR PTR : ENVELOPE_PTR;
                                        VAR FOUND : BOOLEAN );
BEGIN
  FOUND := FALSE;
  PTR := MAIL_HEAD;
  WHILE (FOUND=FALSE) AND (PTR<>NIL) DO BEGIN
    IF PTR^.BOX_ADDR = F_DESC THEN FOUND := TRUE
                              ELSE PTR := PTR^.NEXT_LETTER;
  END; "WHILE"
END; "FIND_FILE"


PROCEDURE TYPE_FILE ( FILE_LU : INTEGER );
VAR DATA_BUF : PAGE;
    I : INTEGER;
    SVC1_IN_PAGE : SVC1_BLOCK;
BEGIN
  WITH SVC1_IN_PAGE DO BEGIN
   SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
   SVC1_BUFSTART := ADDRESS (DATA_BUF);
   SVC1_BUFEND := SVC1_BUFSTART + PAGELENGTH -1;
   SVC1_LU := FILE_LU;
   REPEAT
     SVC1 (SVC1_IN_PAGE);
     I := 0;
     REPEAT
       I := SUCC(I);
       DISPLAY (DATA_BUF[I]);
       UNTIL (I=PAGELENGTH) OR (DATA_BUF[I]=EM);
     UNTIL (DATA_BUF[I]=EM);
  END; "WITH"
END;"TYPE_FILE"


PROCEDURE COPY_FILE ( IN_LU : INTEGER; OUT_LU : INTEGER );
VAR COPY_IN,COPY_OUT : SVC1_BLOCK;
    BUFFER : PAGE;
BEGIN
  COPY_IN.SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
```

```
  COPY_OUT.SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
  COPY_IN.SVC1_LU := IN_LU;
  COPY_OUT.SVC1_LU := OUT_LU;
  COPY_IN.SVC1_BUFSTART := ADDRESS (BUFFER);
  COPY_IN.SVC1_BUFEND := COPY_IN.SVC1_BUFSTART + PAGELENGTH - 1;
  COPY_OUT.SVC1_BUFSTART := COPY_IN.SVC1_BUFSTART;
  COPY_OUT.SVC1_BUFEND := COPY_IN.SVC1_BUFEND;
  REPEAT
    SVC1 (COPY_IN);
    SVC1 (COPY_OUT);
   UNTIL COPY_IN.SVC1_STAT <> SVC1_OK;
END; "COPY_FILE"


PROCEDURE PRINT_MAIL ( WHAT : FD_TYPE; WHERE : CHAR8 );
VAR STATUS1, STATUS2 : BOOLEAN;
    C : CHAR;
    PRT_FD : FD_TYPE;
BEGIN
  ASSIGN_FILE ( WHAT , 1, SVC7_AP_SRO+SVC7_FTYPE_INDEX, STATUS1 );
  IF STATUS1 = OK THEN BEGIN
    WITH PRT_FD DO BEGIN
      IF WHERE = PRINTER THEN VOLN := 'PR  ' ELSE VOLN := 'SP18';
      FN := '        ';
      EXTN := '   ';
      ACCT := ' ';
     END; "WITH"
    ASSIGN_FILE (PRT_FD, 2, SVC7_AP_SREW, STATUS2);
    IF (STATUS1 AND STATUS2) = OK THEN
      REPEAT
         READ(C); WRITE(C);
      UNTIL C = EM;
    END;
  ASSIGN_CON (1);
  ASSIGN_CON (2);
END; "PRINT_FILE"


PROCEDURE ASSIGN_CON "( LU : INTEGER )";
VAR CON_FD : FD_TYPE;
    CON_STATUS : BOOLEAN;
BEGIN
  WITH CON_FD DO BEGIN
    VOLN := 'CON ';
    FN := '        ';
    EXTN := '   ';
    ACCT := ' ';
   END; "WITH"
  CLOSE_FILE (LU);
  ASSIGN_FILE ( CON_FD, LU, SVC7_AP_SRW, CON_STATUS );
```

```
  RESET (LU);
END; "ASSIGN_CON"


"  ************************************************************************
   *                                                                      *
   *                          STATUS UTILITIES                            *
   *                                                                      *
   ************************************************************************"


PROCEDURE GET_USERNAME (VAR NAME_STR : CHAR8);
VAR I,J : INTEGER;
    CH : CHAR;
BEGIN
  I := 0;
  WRITETEXT('enter destination name(:07:)(:10:)');
  REPEAT
    I := SUCC(I);
    READ (CH);
    NAME_STR [I] := UC(CH);
    UNTIL (I=8) OR (CH=NL);
  IF CH = NL THEN NAME_STR[I] := ' ';
  FOR J:=I+1 TO 8 DO NAME_STR[J] := ' ';    "PAD TO END OF ARRAY"
  WHILE CH <> NL DO READ (CH);  "READ EXTRANEOUS CHARS "
END; "GET_USERNAME"


PROCEDURE GET_LOGON_ID (VAR ID : CHAR8);
VAR SVC2_IN : SVC2_PEEK_PARM;
BEGIN
  SVC2PEEK (SVC2_IN);
  ID := SVC2_IN.SVC2_TASK;
END;


FUNCTION MAIL_WAITING ( ID : CHAR8 ): BOOLEAN;
TYPE DIR_ENTRY = RECORD
                  DEST_ID : CHAR8;
                  USER_DIR : FD_TYPE;
                 END; "RECORD"
VAR SVC1_IN : SVC1_BLOCK;
    SLOT : DIR_ENTRY;
    OK_STATUS : BOOLEAN;
    SYS_DIR : FD_TYPE;
BEGIN
  WITH SYS_DIR DO BEGIN
    FN := 'MAILDIR ';
    VOLN := 'SYS3';
    EXTN := 'SYS';
```

```
    ACCT := 'P';
  END; "WITH"
  ASSIGN_FILE (SYS_DIR,3,SVC7_AP_SREW+SVC7_FTYPE_INDEX,OK_STATUS);
  IF OK_STATUS THEN  BEGIN
    WITH SVC1_IN DO BEGIN
      SVC1_FUNC := SVC1_COMMAND + SVC1_FSR;
      SVC1_LU := 3;
      SVC1 (SVC1_IN);        "SKIP PAST COUNTERS IN FILE BEGINING"
      SVC1 (SVC1_IN);
      SVC1_FUNC := SVC1_READ + SVC1_WAIT + SVC1_IMAGE;
      SVC1_BUFSTART := ADDRESS (SLOT);
      SVC1_BUFEND := SVC1_BUFSTART + SYS_DIR_ENTRY_LENGTH -1;
      REPEAT
        SVC1 (SVC1_IN);
        UNTIL (SLOT.DEST_ID=ID) OR (SVC1_IN.SVC1_STAT <> SVC1_OK);
      IF SLOT.DEST_ID=ID  THEN MAIL_WAITING := TRUE
        ELSE MAIL_WAITING := FALSE;
      CLOSE (1);   "CLOSES LU -3"
    END; "WITH"
  END "OK" ELSE BEGIN
    WRITETEXT('ASSIGN ERROR - MAILDIR.SYS (:10:)');
    ABORT := TRUE;
  END; "NOT OK_STATUS"
END; "MAIL_WAITING"


FUNCTION ACTIVE_USR ( ID : CHAR8 ): BOOLEAN;
VAR OK_STATUS : BOOLEAN;
    I : INTEGER;
    FILE_N : FD_TYPE;
    USR_ENTRY : CHAR8;
    SVC1_IN : SVC1_BLOCK;
BEGIN
  WITH FILE_N DO BEGIN
    VOLN := 'SYS3';
    FN := 'ACTIVUSR';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH "
  ASSIGN_FILE (FILE_N,3,SVC7_AP_SRO+SVC7_FTYPE_INDEX,OK_STATUS);
  IF OK_STATUS THEN BEGIN
    WITH SVC1_IN DO BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
      SVC1_LU := 3;
      SVC1_BUFSTART := ADDRESS ( USR_ENTRY );
      SVC1_BUFEND := SVC1_BUFSTART + 7;
      END; "WITH"
    REPEAT
      SVC1 (SVC1_IN)
      UNTIL (USR_ENTRY=ID) OR (SVC1_IN.SVC1_STAT <> SVC1_OK);
```

```
    IF USR_ENTRY=ID THEN ACTIVE_USR := TRUE
                    ELSE ACTIVE_USR := FALSE;
    CLOSE (1);  "LU -3"
    END "OK_STATUS"
   ELSE WRITETEXT('ASSIGN ERROR - ACTIVUSR.SYS (:10:)');
END; "ACTIVE_USR"


FUNCTION TEMP_MAIL ( ID : CHAR8 ): BOOLEAN;
VAR TEMP_DIR : FD_TYPE;
    STATUS1 : BOOLEAN;
    SVC1_IN : SVC1_BLOCK;
    TEMP_ENVELOPE : ENVELOPE;
BEGIN
  WITH TEMP_DIR DO BEGIN
    VOLN := 'SYS3';
    FN := 'TEMPMAIL';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH"
  ASSIGN_FILE (TEMP_DIR,3,SVC7_AP_SRO+SVC7_FTYPE_INDEX,STATUS1);
  IF STATUS1 = OK THEN BEGIN
    WITH SVC1_IN DO BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
      SVC1_LU := 3;
      SVC1_BUFSTART := ADDRESS (TEMP_ENVELOPE);
      SVC1_BUFEND := SVC1_BUFSTART + ENVELOPE_SIZE - 1;
      REPEAT
        SVC1 (SVC1_IN);
        UNTIL (TEMP_ENVELOPE.DEST_NAME=ID) OR (SVC1_STAT<>SVC1_OK);
      IF TEMP_ENVELOPE.DEST_NAME=ID THEN TEMP_MAIL := TRUE
                                    ELSE TEMP_MAIL := FALSE;
      CLOSE (1);
      END; "WITH"
    END "STATUS1 OK"
   ELSE BEGIN
      WRITETEXT('ASSIGN ERRORS - TEMPMAIL (:10:)');
      TEMP_MAIL := FALSE;
      END; "ELSE"
END; "TEMP_MAIL"


"  ************************************************************************
   *                                                                      *
   *                        SUPPORT PROCEDURES                            *
   *                                                                      *
   ************************************************************************"


PROCEDURE GET_FILE_NAME (VAR FILE_N : FD_TYPE );
```

```
VAR I,INDEX,J : INTEGER;
    C : CHAR;
    NAME_BUF : ARRAY [1..80] OF CHAR;
    VOL_NAME,EXT_NAME : BOOLEAN;
BEGIN
  WRITETEXT('enter envelope name (:07:)(:10:)');
  I:= 0;
  REPEAT
    READ (C);
    C := UC(C);
    I := SUCC(I);
    NAME_BUF [I] := C;
    UNTIL (C=NL);
  VOL_NAME := FALSE;
  EXT_NAME := FALSE;
  INDEX := 0;
  FOR J:= 1 TO I DO IF NAME_BUF [J] = ':' THEN VOL_NAME := TRUE;
  FOR J:= 1 TO I DO IF NAME_BUF [J] = '.' THEN EXT_NAME := TRUE;
  IF VOL_NAME
     THEN BEGIN
            FOR J:= 1 TO 4 DO BEGIN
                INDEX := SUCC (INDEX);
                FILE_N.VOLN [J] := NAME_BUF [INDEX];
                END; "FOR J"
            INDEX := INDEX + 2; "SKIP PAST ':' IN NAME_BUF"
          END "VOL"
     ELSE FILE_N.VOLN := 'SYS3';
  J:= 1;
  IF INDEX < 1 THEN INDEX := 1;
  WHILE (NAME_BUF [INDEX] <> NL) AND ( NAME_BUF [INDEX] <> '.')
    AND (J<=8) DO BEGIN
        FILE_N.FN [J] := NAME_BUF [INDEX];
        J := SUCC(J);
        INDEX := SUCC(INDEX);
        END; "FILE DESC"
  FOR I:= J TO 8 DO FILE_N.FN[I] := ' ';  "PAD REST OF FD "
  IF EXT_NAME
     THEN BEGIN
            INDEX := SUCC(INDEX); "SKIP PAST '.' "
            FOR J := 1 TO 3 DO
                BEGIN
                  FILE_N.EXTN[J] := NAME_BUF[INDEX];
                  INDEX := SUCC (INDEX);
                  END;
                END
     ELSE FILE_N.EXTN := 'TXT';
  FILE_N.ACCT := 'P';   "ASSUME GET_FILE CALLS THAT NEED PRIVATE CHAGE LATER"
END; "GET_FILE_NAME"
```

```
PROCEDURE GET_SUBJECT (VAR TOPIC : CHAR16);
VAR I,J : INTEGER;
    CH : CHAR;
BEGIN
  WRITETEXT('enter mail subject  (16 chars)(:07:)(:10:)');
  I := 0;
  REPEAT
    I := SUCC(I);
    READ (CH);
    TOPIC [I] := CH;
    UNTIL (I=16) OR (CH=NL);
  IF CH=NL THEN TOPIC [I] := ' ';
  FOR J := I+1 TO 16 DO TOPIC [J] := ' ';     "PAD TO END OF ARRAY"
  WHILE CH<>NL DO READ (CH);                  "READ EXTRANECUS CHARS"
END; "GET_SUBJECT"


PROCEDURE DISPLAY_ENVELOPE_HEAD;
VAR I : INTEGER;
BEGIN
   DISPLAY (NL);
   DISPLAY (NL);
   DISPLAY (NL);
   WRITETEXT(' -- topic --     -- source --     -- post_m -- (:0:)');
   WRITETEXT('      -- name --     -- enc --(:10:)');
   DISPLAY (NL);
END; "DISPLAY_ENVELOPE_HEAD"


PROCEDURE DISPLAY_ENVELOPE (ENV_PTR : ENVELOPE_PTR);
VAR I : INTEGER;
BEGIN
  WITH ENV_PTR^ DO BEGIN
   FOR I := 1 TO 16 DO DISPLAY (SUBJECT [I]);
   FOR I := 1 TO 3 DO DISPLAY (' ');              "SPACING"
   FOR I := 1 TO 8 DO DISPLAY ( SOURCE_NAME [I] );
   FOR I := 1 TO 9 DO DISPLAY ( ' ' );
   FOR I := 1 TO 8 DO DISPLAY ( POST_MARK [I] );
   FOR I := 1 TO 8 DO DISPLAY ( ' ' );
   FOR I := 1 TO 8 DO DISPLAY (BOX_ADDR.FN [I]);
   DISPLAY ('.');
   FOR I := 1 TO 3 DO DISPLAY (BOX_ADDR.EXTN [I]);
   FOR I := 1 TO 8 DO DISPLAY ( ' ' );
   IF ENCRYP THEN DISPLAY ( '*' ) ;
   DISPLAY ( NL );
  END; "WITH"
  END; " DISPLAY_ENVELOPE "


PROCEDURE DISPLAY_FILES;
```

```
VAR OK_STATUS : BOOLEAN;
    I : INTEGER;
    FILE_DESC : FD_TYPE;
    PTR : ENVELOPE_PTR;
BEGIN
  PTR := MAIL_HEAD;
  WHILE PTR <> NIL DO BEGIN
      FILE_DESC := PTR^.BOX_ADDR;
      ASSIGN_FILE (FILE_DESC,3,SVC7_AP_SRO+SVC7_FTYPE_INDEX,OK_STATUS);
      IF OK_STATUS THEN BEGIN
        WRITETEXT('SOURCE: (:0:)');
        FOR I:= 1 TO 8 DO DISPLAY (PTR^.SOURCE_NAME[I]);
        WRITETEXT('     POST MARK: (:0:)');
        FOR I:= 1 TO 8 DO DISPLAY (PTR^.POST_MARK[I]);
        WRITETEXT('     ENVELOPE: (:0:)');
        FOR I := 1 TO 8 DO DISPLAY (PTR^.BOX_ADDR.FN [I]);
        DISPLAY ('.');
        FOR I := 1 TO 3 DO DISPLAY (PTR^.BOX_ADDR.EXTN [I]);
        DISPLAY (NL);
        WRITETEXT('TOPIC : (:0:)');
        FOR I := 1 TO 16 DO DISPLAY (PTR^.SUBJECT [I]);
        DISPLAY (NL);
        DISPLAY (NL);
        IF PTR^.ENCRYP
          THEN WRITETEXT('FILE ENCODED - MUST DECRYPT FIRST(:10:)')
          ELSE TYPE_FILE (3);
        DISPLAY (NL);
        DISPLAY (NL);
        DISPLAY (NL);
        END "OK_STATUS"
             ELSE WRITETEXT('TYPE FILE ERROR ON LU-3(:10:)');
     CLOSE (1);          "FILE ID 1=LU 3"
    PTR := PTR^.NEXT_LETTER;
   END; "WHILE"
  END; "DISPLAY_FILE"


PROCEDURE DISPLAY_USERS;
VAR SVC1_PARM : SVC1_BLOCK;
    DIR_NAME : FD_TYPE;
    OK_STATUS : BOOLEAN;
    I,J,LINE_NAMES : INTEGER;
    ACTIVUSR_ENTRY : CHAR8;
BEGIN
  WITH DIR_NAME DO BEGIN
    VOLN := 'SYS3';
    FN := 'ACTIVUSR';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH"
```

```
   ASSIGN_FILE ( DIR_NAME,3,SVC7_AP_SRO+SVC7_FTYPE_INDEX,OK_STATUS );
   IF OK_STATUS THEN BEGIN
     WITH SVC1_PARM DO BEGIN
       SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
       SVC1_LU := 3;
       SVC1_BUFSTART := ADDRESS (ACTIVUSR_ENTRY);
       SVC1_BUFEND := SVC1_BUFSTART + 7;
       DISPLAY (NL);
       DISPLAY (NL);
       LINE_NAMES := 0;
       WRITETEXT('Current users -      (:0:)');
       FOR I := 1 TO MAX_ACTIV_USERS DO BEGIN
         SVC1 (SVC1_PARM);
         IF ACTIVUSR_ENTRY <> 'UNASSIGN' THEN BEGIN
           FOR J := 1 TO 8 DO DISPLAY (ACTIVUSR_ENTRY[J]);
           DISPLAY (' ');
           DISPLAY (' ');
           LINE_NAMES := SUCC(LINE_NAMES);
           IF LINE_NAMES = 5 THEN BEGIN
              LINE_NAMES := 0;
              DISPLAY (NL);
              FOR J := 1 TO 20 DO DISPLAY (' ');
              END;
           END; " ACTIVE USER NAME "
       END; "LOOP - FOR"
      DISPLAY (NL);
    END; "WITH";
    CLOSE (1);      "LU -3"
    END "OK_STATUS"
    ELSE WRITETEXT('ASSIGN ERROR - ACTIVUSR.SYS (:10:)');
END; "DISPLAY USRS"


PROCEDURE COMMAND_SUMMARY;
VAR DUMMY_CHAR : CHAR;
BEGIN
  DISPLAY (NL);
  DISPLAY (NL);
  WRITETEXT('          Mail Manager R00-03   Command Summary(:10:)');
  DISPLAY(NL);
  WRITETEXT('HELP      produces this command summary (:10:)');
  WRITETEXT('INQUIRE   will report the status of all mail directed to(:0:)');
  WRITETEXT(' your account.(:10:)');
  WRITETEXT('GET       retrieves mail from the system mailbox to private(:10:)');
  WRITETEXT('          user account. Will also remove this (:0:)');
  WRITETEXT('entry from the mail table(:10:)');
  WRITETEXT('CREATE    used to create mail ( PEDIT superset ) it will (:0:)');
  WRITETEXT('solicit (:10:)');
  WRITETEXT('          destination info and use your(:0:)');
  WRITETEXT('signon ID as the source ID(:10:)');
```

```
  WRITETEXT('MAIL     used to send a previously created file or (:0:)');
  WRITETEXT('message to another user(:10:)');
  WRITETEXT('TYPE     will type all user ID directed mail on (:0:)');
  WRITETEXT('console but will not(:10:)');
  WRITETEXT('         remove mail entry from the mail table(:10:)');
  WRITETEXT('REMOVE   will remove undelivered mail (:0:)');
  WRITETEXT('directed to you from the (:0:)');
  WRITETEXT('mail system(:10:)');
  WRITETEXT('PURGE    will "REMOVE" all remaining mail entries from(:0:)');
  WRITETEXT(' your mail box.(:10:)');
  WRITETEXT('FORWARD  will redirect  mail to additional users(:10:)');
  WRITETEXT('ENCRYPT  will encode a mail file with a user specified key(:10:)');
  WRITETEXT('DECRYPT  will decode a mail file given the proper code key(:10:)');
  WRITETEXT('SET_EDIT  allows the user to modify the default editor used(:10:)');
  WRITETEXT('          for the CREATE command(:10:)');
  WRITETEXT('XFER     a computer-computer file transfer mechanism(:10:)');
  WRITETEXT('QUIT     will return to 8/32 user command mode(:10:)');
  DISPLAY (NL);
  WRITETEXT('         type  <CR>  to continue(:10:)');
  READ(DUMMY_CHAR);
END; "DISPLAY_SUMMARY"


"  ************************************************************************
   *                                                                      *
   *                  MAIL DIRECTORY MAINTENANCE                          *
   *                                                                      *
   ************************************************************************"


PROCEDURE CREATE_DIR ( ID : CHAR8; VAR STATUS : BOOLEAN );
TYPE DIR_ENTRY = RECORD
                 DEST_ID : CHAR8;
                 USER_DIR : FD_TYPE;
                 END; "RECORD"
VAR FILE_N , SYS_DIR : FD_TYPE;
    SVC1_PARM : SVC1_BLOCK;
    STATUS2 : BOOLEAN;
    SLOT : DIR_ENTRY;
BEGIN
  WITH FILE_N DO BEGIN
    VOLN := 'SYS3';
    UPDATE_CT (DIRECTORY);
    PACK_INT ( DIRECT_CT, FN);
    FN[1] := 'D';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH"
  CREATE_FILE ( FILE_N,ENVELOPE_SIZE,STATUS );
  IF STATUS = OK THEN BEGIN
```

```
    WITH SYS_DIR DO BEGIN
      VOLN := 'SYS3';
      FN := 'MAILDIR ';
      EXTN := 'SYS';
      ACCT := 'P';
      END; "WITH"
    ASSIGN_FILE (SYS_DIR,3,SVC7_AP_SREW+SVC7_FTYPE_INDEX,STATUS2);
    IF STATUS2 = OK THEN BEGIN
    WITH SVC1_PARM DO BEGIN
      SVC1_FUNC := SVC1_COMMAND + SVC1_FSR;
      SVC1_LU := 3;
      END; "WITH";
    SVC1 (SVC1_PARM);          "SKIP PAST DIRECT & TRAFFIC COUNTS"
    WITH SVC1_PARM DO BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
      SVC1_BUFSTART := ADDRESS (SLOT);
      SVC1_BUFEND := SVC1_BUFSTART + SYS_DIR_ENTRY_LENGTH -1;
      END; "WITH"
    REPEAT
      SVC1(SVC1_PARM);
      UNTIL (SLOT.DEST_ID = 'UNASSIGN')
         OR (SVC1_PARM.SVC1_STAT <> SVC1_OK);
    IF SLOT.DEST_ID ='UNASSIGN' THEN BEGIN
      SVC1_PARM.SVC1_FUNC := SVC1_COMMAND + SVC1_BSR;
      SVC1 (SVC1_PARM);           "BACK UP TO UNUSED RECORD"
      WITH SVC1_PARM DO BEGIN
        SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
        SVC1_BUFSTART := ADDRESS (SLOT);
        SVC1_BUFEND := SVC1_BUFSTART + SYS_DIR_ENTRY_LENGTH -1;
        END; "WITH"
      SLOT.DEST_ID := ID;
      SLOT.USER_DIR := FILE_N;
      SVC1 (SVC1_PARM);
      IF SVC1_PARM.SVC1_STAT <> SVC1_OK THEN STATUS := NOK
                                ELSE STATUS := OK;
      END "UNASSIGNED END"
    ELSE BEGIN
       WRITETEXT('CREATE DIR ERROR - DIRECTORY FULL (:10:)');
       STATUS := NOK;
     END;
    CLOSE (1);  "LU - 3";
  END "STATUS2 OK"
  ELSE BEGIN
       WRITETEXT('ASSIGN ERROR - MAILDIR.SYS (:10:)');
       STATUS := NOK;
     END;
END "STATUS OK"
ELSE BEGIN
       WRITETEXT('CREATE USER DIR FILE ERROR (:10:)');
       STATUS := NOK;
```

```
      END;
END; "CREATE_DIR"


PROCEDURE DELETE_DIR ( ID : CHAR8; VAR STATUS : BOOLEAN );
TYPE DIR_ENTRY = RECORD
                   DEST_ID : CHAR8;
                   USER_DIR : FD_TYPE;
                  END; "RECORD"
VAR SYS_DIR : FD_TYPE;
    SVC1_PARM : SVC1_BLOCK;
    STATUS1 : BOOLEAN;
    SLOT : DIR_ENTRY;
BEGIN
  WITH SYS_DIR DO BEGIN
    VOLN := 'SYS3';
    FN := 'MAILDIR ';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH"
  ASSIGN_FILE (SYS_DIR,3,SVC7_AP_ERW+SVC7_FTYPE_INDEX,STATUS1);
  IF STATUS1 = OK THEN BEGIN
    WITH SVC1_PARM DO BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
      SVC1_LU := 3;
      SVC1_BUFSTART := ADDRESS (SLOT);
      SVC1_BUFEND := SVC1_BUFSTART + SYS_DIR_ENTRY_LENGTH -1;
      REPEAT
        SVC1 (SVC1_PARM);
        UNTIL (SLOT.DEST_ID=ID) OR (SVC1_STAT<>SVC1_OK);
      STATUS := (SVC1_STAT=SVC1_OK);
      IF STATUS = OK THEN BEGIN
        SVC1_FUNC := SVC1_COMMAND + SVC1_BSR;
        SVC1 (SVC1_PARM);              "BACK UP TO THE CORRECT RECORD"
        SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
        SLOT.DEST_ID := 'UNASSIGN';
        SVC1 (SVC1_PARM);              "RESET SLOT TO UNASSIGNED STATE"
        STATUS := (SVC1_STAT = SVC1_OK);
        END; "FOUND ID SLOT OK"
      END; "WITH"
    CLOSE (1);
  END "ASSIGN OK" ELSE STATUS := NOK;
END; "DELETE_DIR"


PROCEDURE FETCH_USR_DIR ( ID : CHAR8; VAR DIR_FD : FD_TYPE);
TYPE DIR_ENTRY = RECORD
                   DEST_ID : CHAR8;
                   USER_DIR : FD_TYPE;
                  END; "RECORD"
```

```
VAR SVC1_IN : SVC1_BLOCK;
    SLOT : DIR_ENTRY;
    OK_STATUS : BOOLEAN;
    SYS_DIR : FD_TYPE;
BEGIN
  WITH SYS_DIR DO BEGIN
    FN := 'MAILDIR ';
    VOLN := 'SYS3';
    EXTN := 'SYS';
    ACCT := 'P';
  END; "WITH"
  ASSIGN_FILE (SYS_DIR,3,SVC7_AP_SRO+SVC7_FTYPE_INDEX,OK_STATUS);
  IF OK_STATUS THEN  BEGIN
    WITH SVC1_IN DO BEGIN
      SVC1_FUNC := SVC1_COMMAND + SVC1_FSR; "SKIP PAST DIR & TRAFFIC CT"
      SVC1_LU := 3;
      SVC1 ( SVC1_IN );
      SVC1 (SVC1_IN);
      SVC1_FUNC := SVC1_READ + SVC1_WAIT + SVC1_IMAGE;
      SVC1_BUFSTART := ADDRESS (SLOT);
      SVC1_BUFEND := SVC1_BUFSTART + SYS_DIR_ENTRY_LENGTH -1;
      REPEAT
        SVC1 (SVC1_IN);
        UNTIL (SLOT.DEST_ID=ID) OR (SVC1_IN.SVC1_STAT <> SVC1_OK);
      IF SLOT.DEST_ID=ID THEN   DIR_FD := SLOT.USER_DIR;
      CLOSE (1);   "CLOSES LU -3"
    END; "WITH"
  END "OK_STATUS" ELSE BEGIN
    WRITETEXT('ASSIGN ERROR - MAILDIR.SYS (:10:)');
    ABORT := TRUE;
  END; "NOT OK_STATUS"
END; "FETCH_USR_DIR"


PROCEDURE LOAD_DIR ( DIR : FD_TYPE ; VAR QUIT : BOOLEAN );
VAR SVC1_IN : SVC1_BLOCK;
    MAIL : ENVELOPE;
    OK_STATUS : BOOLEAN;
BEGIN
  ASSIGN_FILE (DIR,3,SVC7_AP_SRO+SVC7_FTYPE_INDEX,OK_STATUS);
  IF OK_STATUS THEN BEGIN
    WITH SVC1_IN DO BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
      SVC1_BUFSTART := ADDRESS (MAIL);
      SVC1_BUFEND := SVC1_BUFSTART + ENVELOPE_SIZE -1;
      SVC1_LU := 3;
      SVC1 (SVC1_IN);
      GET_NEW_PTR (MAIL_PTR);
      MAIL_PTR^ := MAIL;
      MAIL_HEAD := MAIL_PTR;
```

```
      MAIL_TAIL := MAIL_PTR;
      WHILE SVC1_IN.SVC1_STAT = SVC1_OK DO BEGIN
         SVC1 (SVC1_IN);
         IF SVC1_IN.SVC1_STAT = SVC1_OK THEN BEGIN
           GET_NEW_PTR (MAIL_PTR);
           MAIL_PTR^ := MAIL;
           MAIL_TAIL^.NEXT_LETTER := MAIL_PTR;
           MAIL_TAIL := MAIL_PTR;
           END; "SVC1_OK"
         END; "WHILE"
       MAIL_PTR^.NEXT_LETTER := NIL;
      END; "WITH"
      CLOSE (1);       "LU - 3"
     END  "OK_STATUS"
    ELSE  WRITETEXT('USER DIR LOAD ERROR (:10:)');
  QUIT := NOT (OK_STATUS);
END; "LOAD DIR"


PROCEDURE DOWN_LOAD_DIR (DIR_NAME : FD_TYPE);
VAR MAIL : ENVELOPE;
    SVC1_OUT : SVC1_BLOCK;
    STATUS1 : BOOLEAN;
BEGIN
  DELETE_FILE (DIR_NAME,STATUS1);
  IF STATUS1 = OK THEN CREATE_FILE (DIR_NAME,ENVELOPE_SIZE, STATUS1);
  IF STATUS1 = OK THEN ASSIGN_FILE (DIR_NAME,3,
    SVC7_AP_SREW+SVC7_FTYPE_INDEX, STATUS1);
  IF STATUS1 = OK THEN BEGIN
    WITH SVC1_OUT DO BEGIN
      SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
      SVC1_LU := 3;
      SVC1_BUFSTART := ADDRESS (MAIL);
      SVC1_BUFEND := SVC1_BUFSTART + ENVELOPE_SIZE;
      IF MAIL_HEAD <> NIL THEN BEGIN
        MAIL_PTR := MAIL_HEAD;
        WHILE MAIL_PTR <> NIL DO BEGIN
          MAIL := MAIL_PTR^;
          SVC1 ( SVC1_OUT );
          MAIL_PTR := MAIL_PTR^.NEXT_LETTER;
          END; "WHILE"
        SVC1_FUNC := SVC1_COMMAND + SVC1_WFM;
        SVC1 (SVC1_OUT);          "WRITE NEW END OF FILE MARK"
        CLOSE (1);
        END "SOMETHING LEFT IN USER DIRECTORY"
      ELSE BEGIN
       CLOSE (1);
        DELETE_FILE (DIR_NAME, STATUS1);
        DELETE_DIR ( LOGON_ID, STATUS1);
        END; "DIRECTORY NOW EMPTY"
```

```
      END; "WITH SVC1"
     WRITETEXT('DOWN_LOAD COMPLETE(:07:)(:10:)');
     DISPLAY (NL);
    END "ASSIGN OK"
    ELSE WRITETEXT('ASSIGN ERROR - USER DIR FD (:10:)');
END; "DOWN_LOAD_DIR"


PROCEDURE ENTER_ACTIVE_TABLE ( VAR NOK_STATUS : BOOLEAN );
VAR SVC1_PARM : SVC1_BLOCK;
    TABLE_ID : FD_TYPE;
    USR_TABLE_ENTRY : CHAR8;
    OK_STATUS : BOOLEAN;
BEGIN
  WITH TABLE_ID DO BEGIN
    FN := 'ACTIVUSR';
    VOLN := 'SYS3';
    EXTN := 'SYS';
    ACCT := 'P';
  END; "WITH"
  ASSIGN_FILE ( TABLE_ID,3,SVC7_AP_ERW+SVC7_FTYPE_INDEX, OK_STATUS );
  IF OK_STATUS THEN BEGIN
    WITH SVC1_PARM DO BEGIN
      GET_LOGON_ID (LOGON_ID);
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
      SVC1_BUFSTART := ADDRESS ( USR_TABLE_ENTRY );
      SVC1_BUFEND := SVC1_BUFSTART + 7;
      SVC1_LU := 3;
      REPEAT
        SVC1 (SVC1_PARM);
        UNTIL (USR_TABLE_ENTRY = 'UNASSIGN');
      SVC1_FUNC := SVC1_BSR + SVC1_COMMAND;
      SVC1 (SVC1_PARM);   "BACK SPACE TO EMPTY RECORD"
      SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
      SVC1_BUFSTART := ADDRESS (LOGON_ID);
      SVC1_BUFEND := SVC1_BUFSTART + 7;
      SVC1 (SVC1_PARM);
      END; "WITH"
      CLOSE (1);   "LU - 3"
      NOK_STATUS := NOT OK_STATUS;
   END  "OK_STATUS"
  ELSE BEGIN
         WRITETEXT('ASSIGN ERROR - ACTIVUSR.SYS (:10:)');
         NOK_STATUS := NOT OK_STATUS;
       END; "NOT OK"
END; "ENTER_ACTIVE_TABLE"


PROCEDURE LEAVE_ACTIVE_TABLE;
VAR SVC1_PARM : SVC1_BLOCK;
```

```
    TABLE_ID : FD_TYPE;
    USR_TABLE_ENTRY : CHAR8;
    OK_STATUS : BOOLEAN;
BEGIN
  WITH TABLE_ID DO BEGIN
    VOLN := 'SYS3';
    FN := 'ACTIVUSR';
    EXTN := 'SYS';
    ACCT := 'P';
  END; "WITH"
  ASSIGN_FILE (TABLE_ID, 3,SVC7_AP_ERW+SVC7_FTYPE_INDEX,OK_STATUS);
  IF OK_STATUS THEN BEGIN
    WITH SVC1_PARM DO BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_WAIT + SVC1_IMAGE;
      SVC1_BUFSTART := ADDRESS (USR_TABLE_ENTRY);
      SVC1_BUFEND := SVC1_BUFSTART + 7;
      SVC1_LU := 3;
      REPEAT
        SVC1 (SVC1_PARM)          "FIND USER'S ENTRY"
        UNTIL (USR_TABLE_ENTRY=LOGON_ID);
      SVC1_FUNC := SVC1_COMMAND + SVC1_BSR;
      SVC1 (SVC1_PARM);      "BACK SPACE TO USERS RECORD"
      SVC1_FUNC := SVC1_WRITE + SVC1_WAIT + SVC1_IMAGE;
      USR_TABLE_ENTRY := 'UNASSIGN';
      SVC1_BUFSTART := ADDRESS (USR_TABLE_ENTRY);
      SVC1_BUFEND := SVC1_BUFSTART + 7;
      SVC1 (SVC1_PARM);
    END; "WITH"
    CLOSE (1);   "LU - 3"
  END "OK_STATUS"
  ELSE  WRITETEXT('ASSIGN ERROR - ACTIVUSR.SYS (:10:)');
END; "LEAVE_ACTIVE_TABLE"


PROCEDURE ENTER_TEMP ( TEMP_MAIL : ENVELOPE; VAR STATUS : BOOLEAN);
VAR TEMP_SLOT : ENVELOPE;
    TEMP_FD : FD_TYPE;
    STATUS1 : BOOLEAN;
    SVC1_PARM : SVC1_BLOCK;
BEGIN
  WITH SVC1_PARM DO BEGIN
    WITH TEMP_FD DO BEGIN
      VOLN := 'SYS3';
      FN := 'TEMPMAIL';
      EXTN := 'SYS';
      ACCT := 'P';
      END; "WITH TEMP_FD"
    ASSIGN_FILE (TEMP_FD,3,SVC7_AP_SREW+SVC7_FTYPE_INDEX,STATUS1);
    IF STATUS1=OK THEN BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_WAIT + SVC1_IMAGE;
```

```
      SVC1_LU := 3;
      SVC1_BUFSTART := ADDRESS (TEMP_SLOT);
      SVC1_BUFEND := SVC1_BUFSTART + ENVELOPE_SIZE - 1;
      REPEAT
        SVC1 (SVC1_PARM);
        UNTIL (TEMP_SLOT.DEST_NAME='UNASSIGN') OR (SVC1_STAT <> SVC1_OK);
      IF TEMP_SLOT.DEST_NAME = 'UNASSIGN' THEN BEGIN
        SVC1_FUNC := SVC1_COMMAND + SVC1_BSR;
        SVC1 (SVC1_PARM);
        SVC1_FUNC := SVC1_WRITE + SVC1_WAIT + SVC1_IMAGE;
        TEMP_SLOT := TEMP_MAIL;
        SVC1 (SVC1_PARM);
        IF SVC1_STAT = SVC1_OK THEN BEGIN
           STATUS := OK;
           END
         ELSE BEGIN
          STATUS := NOK;
          WRITETEXT('SVC1 ERROR IN TEMP_MAIL UPDATE(:10:)');
          END; "QUIT BY SVC1 ERROR"
         END "UNASSIGN SLOT"
       ELSE BEGIN
        WRITETEXT('ERROR IN TEMPMAIL.SYS - FILE BUFFER FULL(:10:)');
        STATUS := NOK;
        END;
      CLOSE (1); "LU - 3"
     END "STATUS1 OK "
    ELSE BEGIN
     WRITETEXT('ASSIGN ERROR - TEMPMAIL.SYS (:10:)');
     STATUS := NOK;
     END; "STATUS1 NOK"
  END; "WITH SVC1_PARM"
END; "ENTER_TEMP"


PROCEDURE TEMP_EXTRACT ( ID: CHAR8 );
VAR TEMP_DIR : FD_TYPE;
    STATUS1 : BOOLEAN;
    SVC1_PARM : SVC1_BLOCK;
    TEMP_SLOT : ENVELOPE;
BEGIN      "ASSERT: ENTRY HAS BEEN VERIFIED TO EXIST PRIOR TO THIS PROC"
  WITH TEMP_DIR DO BEGIN
    VOLN := 'SYS3';
    FN := 'TEMPMAIL';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH"
  ASSIGN_FILE (TEMP_DIR,3,SVC7_AP_SREW+SVC7_FTYPE_INDEX,STATUS1);
  IF STATUS1 = OK THEN BEGIN
    WITH SVC1_PARM DO BEGIN
      REPEAT
```

```
        SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
        SVC1_LU := 3;
        SVC1_BUFSTART := ADDRESS (TEMP_SLOT);
        SVC1_BUFEND := SVC1_BUFSTART + ENVELOPE_SIZE -1;
        SVC1_STAT := SVC1_OK;
        WHILE (TEMP_SLOT.DEST_NAME<>ID) AND (SVC1_STAT=SVC1_OK)
          DO SVC1 (SVC1_PARM);
        IF SVC1_STAT=SVC1_OK THEN BEGIN
          GET_NEW_PTR (MAIL_PTR);
          MAIL_PTR^ := TEMP_SLOT;           "ADD NEW ENVEL TO USERS"
          MAIL_PTR^.NEXT_LETTER := NIL;     "LINK ENVE LIST      "
          IF MAIL_TAIL <> NIL THEN BEGIN
                              MAIL_TAIL^.NEXT_LETTER := MAIL_PTR;
                              MAIL_TAIL := MAIL_PTR;
                              END "LINKS ALREADY EXISTS"
                          ELSE BEGIN
                              MAIL_HEAD := MAIL_PTR;
                              MAIL_TAIL := MAIL_PTR;
                              END; "USER LINKS EMPTY"
          DIRTY := TRUE;

          SVC1_FUNC := SVC1_COMMAND + SVC1_BSR;
          SVC1 (SVC1_PARM);                 "RESET SLOT IN TEMPMAIL"
          SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
          TEMP_SLOT.DEST_NAME := 'UNASSIGN';
          SVC1 (SVC1_PARM);
          END; "FOUND ACTIVE SLOT"
        UNTIL (SVC1_STAT = SVC1_ERROR + SVC1_EOF);
        CLOSE (1);
      END; "WITH"
    END "STATUS1 OK"
  ELSE WRITETEXT('ASSIGN ERROR - TEMPFILE(:10:)');
END; "TEMP_EXTRACT"


"  ************************************************************************
   *                                                                      *
   *                        MAIL UTILITY PROCEDURES                       *
   *                                                                      *
   ************************************************************************"


PROCEDURE ADD_MAIL ( NEW_MAIL : ENVELOPE; DEST_DIR_NAME : FD_TYPE;
                     VAR STATUS : BOOLEAN );
VAR U_DIR_ENTRY : ENVELOPE;
    SVC1_PARM : SVC1_BLOCK;
    STATUS1 : BOOLEAN;
BEGIN
  WITH SVC1_PARM DO BEGIN
    ASSIGN_FILE ( DEST_DIR_NAME,3,SVC7_AP_SREW+SVC7_FTYPE_INDEX,STATUS1 );
```

```
    IF STATUS1=OK THEN BEGIN
      SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
      SVC1_BUFSTART := ADDRESS (U_DIR_ENTRY);
      SVC1_BUFEND := SVC1_BUFSTART + ENVELOPE_SIZE - 1;
      SVC1_LU := 3;
      REPEAT
        SVC1 (SVC1_PARM);
        UNTIL (U_DIR_ENTRY.DEST_NAME = 'UNASSIGN')
           OR (SVC1_STAT <> SVC1_OK);
      IF U_DIR_ENTRY.DEST_NAME = 'UNASSIGN' THEN BEGIN
        SVC1_FUNC := SVC1_COMMAND + SVC1_BSR;
        SVC1 (SVC1_PARM);          "BACK UP TO UNASSIGNED RECORD"
        END; "UNASSIGNED"
      IF (U_DIR_ENTRY.DEST_NAME='UNASSIGN')
        OR (SVC1_STAT = SVC1_ERROR + SVC1_EOF)
      THEN BEGIN
        SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
        U_DIR_ENTRY := NEW_MAIL;
        SVC1 (SVC1_PARM);
        IF SVC1_STAT = SVC1_OK THEN BEGIN
          STATUS := OK;
          END "SVC1_OK"
         ELSE BEGIN
           STATUS := NOK;
          WRITETEXT('SVC1 ERROR ON ADD FILE(:10:)');
          END; "SVC1 ERROR"
        END " OK CONDITIONS"
       ELSE BEGIN
        WRITETEXT('ERROR IN USER DIR TABLE (:10:)');
        STATUS := NOK;
        END;
      CLOSE (1);
    END "STATUS1 OK"
   ELSE BEGIN
    WRITETEXT('ASSIGN ERROR  DEST DIR NAME (:10:)');
    STATUS := NOK;
    END; "STATUS1 NOK"
  END; "WITH SVC1"
END; "ADD_MAIL"


PROCEDURE REMOVE_MAIL (DISCARD_LETTER :ENVELOPE; VAR STATUS :BOOLEAN);
"ASSUMES THAT THE FILE HAS BEEN VERIFIED TO EXIST WITH FIND_FILE PROC"
BEGIN
  DIRTY := TRUE;
  MAIL_PTR := MAIL_HEAD;
  MAIL_TRAIL := NIL;
  WHILE (MAIL_PTR^.BOX_ADDR <> DISCARD_LETTER.BOX_ADDR) DO BEGIN
    MAIL_TRAIL := MAIL_PTR;
    MAIL_PTR := MAIL_PTR^.NEXT_LETTER;
```

```
    END; "WHILE"
  "ASSERT: MAIL_PTR PTS TO DISCARD LETTER & MAIL_TRAIL PTS TO "
  "        PRIOR LETTER IN LINKED LIST"
  IF MAIL_TRAIL <> NIL
    THEN MAIL_TRAIL^.NEXT_LETTER := MAIL_PTR^.NEXT_LETTER
    ELSE MAIL_HEAD := MAIL_PTR^.NEXT_LETTER;
  IF MAIL_TRAIL <> NIL THEN IF MAIL_TRAIL^.NEXT_LETTER = NIL
    THEN MAIL_TAIL := MAIL_TRAIL;
  RECYCLE_PTR (MAIL_PTR);
  IF MAIL_HEAD = NIL THEN MAIL_TAIL := NIL;
END; "REMOVE_MAIL"


PROCEDURE DEPOSIT_MAIL (VAR LETTER : ENVELOPE; VAR STATUS : BOOLEAN);
VAR STATUS1,STATUS2 : BOOLEAN;
    DIR_NAME,OLD_FD,RENAME_FD : FD_TYPE;
BEGIN
  DIRTY := TRUE;
  UPDATE_CT ( TRAFFIC );
  WITH RENAME_FD DO BEGIN
    VOLN := 'SYS3';
    PACK_INT (TRAFFIC_CT, FN);
    FN [1] := 'M';
    EXTN := 'SYS';
    ACCT := 'P';
    END; "WITH"
  RENAME_FILE (LETTER.BOX_ADDR, RENAME_FD, STATUS);
  IF STATUS = OK THEN BEGIN
    OLD_FD := LETTER.BOX_ADDR;
    LETTER.BOX_ADDR := RENAME_FD;
    IF ACTIVE_USR ( LETTER.DEST_NAME ) THEN BEGIN
      ENTER_TEMP ( LETTER,STATUS1);
      STATUS := STATUS1;
      END "ACTIVUSR"
     ELSE BEGIN
      IF NOT MAIL_WAITING ( LETTER.DEST_NAME ) THEN
         CREATE_DIR ( LETTER.DEST_NAME, STATUS1 );
      IF STATUS1=OK THEN BEGIN
         FETCH_USR_DIR (LETTER.DEST_NAME, DIR_NAME);
         ADD_MAIL ( LETTER, DIR_NAME, STATUS2);
         STATUS := STATUS1;
         END "STATUS OK "
        ELSE BEGIN
         WRITETEXT('CREATE DIR ERROR (:10:)');
         STATUS := NOK;
         END; "STAATUS1 NOK"
      END; "IN ACTIVER USER ID"
    LETTER.BOX_ADDR := OLD_FD;   "USED FOR REMOVE IN FORWARD"
  END; "RENAME STATUS OK"
END; "DEPOSIT MAIL"
```

```
PROCEDURE FORWARD_MAIL ( VAR STATUS : BOOLEAN );
VAR FORWARD_FD : FD_TYPE;
    FILE_FOUND : BOOLEAN;
    MAIL : ENVELOPE;
BEGIN
  GET_FILE_NAME ( FORWARD_FD );
  FIND_FILE ( FORWARD_FD, MAIL_PTR, FILE_FOUND );
  IF FILE_FOUND THEN BEGIN
    MAIL := MAIL_PTR ^;
    GET_USERNAME (MAIL.DEST_NAME);
    IF (MAIL.DEST_NAME = PRINTER) OR (MAIL.DEST_NAME = SPINWTR)
       THEN BEGIN
        PRINT_MAIL ( FORWARD_FD, MAIL.DEST_NAME );
        STATUS := OK;
        END  "FORWARD TO PRINTERS "
       ELSE BEGIN
        DEPOSIT_MAIL ( MAIL, STATUS );
    "IF FORWARDED OK THEN REMOVE ENTRY FROM THIS USERS DIRECTORY"
        IF STATUS = OK THEN REMOVE_MAIL (MAIL, STATUS);
        END; "ELSE"
     END "FILE_FOUND"
  ELSE BEGIN
    WRITETEXT('FORWARD MAIL FILE DESCRIPTOR NOT FOUND (:10:)');
    STATUS := NOK;
   END; "FILE_NOT_FOUND"
END; "FORWARD_MAIL"


PROCEDURE SET_MODE;
VAR C : CHAR;
    CORRECT_LETTER : BOOLEAN;
BEGIN
  DISPLAY (NL);
  DISPLAY (NL);
  WRITETEXT('Current Run_mode =(:0:)');
  CASE RUN_MODE OF
     USER : WRITETEXT(' USER(:0:)');
     EXEC : WRITETEXT(' EXEC(:0:)');
     DEBUG : WRITETEXT(' DEBUG(:0:)');
     END; "CASE"
  WRITETEXT('   Enter new mode - ( U, E, D )(:10:)');
  CORRECT_LETTER := FALSE;
  REPEAT
    READ (C);
    C := UC (C);
    IF (C='U') OR (C='E') OR (C='D')
      THEN CORRECT_LETTER := TRUE
      ELSE WRITETEXT('Enter U, E, D only (:10:)');
```

```
    UNTIL CORRECT_LETTER; "FOUND"
  WRITETEXT('New Run_mode = (:0:)');
  CASE C OF
     'U' : BEGIN
             RUN_MODE := USER;
             WRITETEXT ('USER(:10:)');
             END; "U"
     'E' : BEGIN
             RUN_MODE := EXEC;
             WRITETEXT ('EXEC(:10:)');
             END; "E"
     'D' : BEGIN
             RUN_MODE := DEBUG;
             WRITETEXT ('DEBUG(:10:)');
             END; "D"
     END; "CASE"
  WHILE C<>NL DO READ (C);     "GETS RID OF EXTRA CHARS"
END; "SET_MODE"


PROCEDURE SET_EDITOR;
VAR C : CHAR;
    VALID_EDITOR : BOOLEAN;
BEGIN
  DISPLAY (NL);
  DISPLAY (NL);
  WRITETEXT('current editor : (:0:)');
  CASE MAIL_EDITOR OF
    MEDIT : WRITETEXT(' MEDIT (PEDIT)(:0:)');
    SEDIT : WRITETEXT(' SEDIT (:0:)');
    OTHEREDIT : WRITETEXT(' USER DEFINED EDITOR(:0:)');
    END; "CASE"
  WRITETEXT('              enter new editor - ( M, S, O )(:10:)');
  VALID_EDITOR := FALSE;
  REPEAT
    READ (C);
    C := UC (C);
    IF (C='M') OR (C='S') OR (C='O') THEN VALID_EDITOR := TRUE
      ELSE WRITETEXT('enter  M,S,O only (:07:)(:10:)');
    UNTIL VALID_EDITOR; "SELECTED"
  WRITETEXT('new editor : (:0:)');
  CASE C OF
    'M'  : BEGIN
             MAIL_EDITOR := MEDIT;
             WRITETEXT('MEDIT(:10:)');
             END; "M"
    'S'  : BEGIN
             MAIL_EDITOR := SEDIT;
             WRITETEXT('SEDIT(:10:)');
             END; "S"
```

```
    'O'  : BEGIN
             MAIL_EDITOR := OTHEREDIT;
             WRITETEXT('OTHER EDITOR(:10:)');
             END; "O"
    END; "CASE"
  WHILE C<>NL DO READ (C);          "GET RID OF JUNK ON LINE END"
END; "SET_EDIT"


PROCEDURE LIST;
VAR PTR : ENVELOPE_PTR;
BEGIN
  DISPLAY_ENVELOPE_HEAD;
  PTR := MAIL_HEAD;
  WHILE PTR <> NIL DO BEGIN
     DISPLAY_ENVELOPE (PTR);
     PTR := PTR^.NEXT_LETTER;
  END; "WHILE"
END; "LIST"


PROCEDURE GET_COMMAND ( VAR CMD: MAIL_COMMANDS );
VAR C : CHAR;
    I : MAIL_COMMANDS;
    FOUND : BOOLEAN;
BEGIN
  REPEAT
    WRITETEXT('(:07:)(:10:)');   "WAKE UP USER"
    READ (C);
    C := UC(C);
    I:= MIN_CMDS;
    FOUND := FALSE;
    WHILE ( I<=MAX_CMDS ) AND NOT FOUND DO BEGIN
      FOUND := (C=COMMAND [I]);
      IF NOT FOUND THEN I:= SUCC(I);
    END; "WHILE"
    IF C<>NL THEN READ (C);   "READS DUMMY TRAILING '0A' FROM INPUT STREAM"
    IF FOUND THEN CMD := I
           ELSE WRITETEXT('Command Error  - re-enter command(:10:)');
  UNTIL FOUND "VALID COMMAND";
END; "GET_COMMAND"


" ***************************************************************
  *                                                             *
  *                          MAIL EDITORS                       *
  *                                                             *
  ***************************************************************"
```

```
PROCEDURE MEDIT_MAIL (VAR LETTER : ENVELOPE;
                      VAR DEPOSIT_STATUS : BOOLEAN);
VAR NUL_FILE : FD_TYPE;
    STATUS1 : BOOLEAN;
BEGIN
  WITH NUL_FILE DO BEGIN
    VOLN := 'SYS ';
    FN := 'NULL    ';
    EXTN := '   ';
    ACCT := 'S';
    END; "WITH"
  CLOSE_FILE (1);
  CLOSE_FILE (2);
  ASSIGN_FILE ( NUL_FILE,1,SVC7_AP_ERW,STATUS1);
  IF STATUS1 = OK THEN
    ASSIGN_FILE ( LETTER.BOX_ADDR,2,SVC7_AP_ERW+SVC7_FTYPE_INDEX,STATUS1 );
  IF STATUS1 = OK THEN BEGIN
    OVERLAY_NAME := 'MAILEDIT.OBJ';
    RUN ( OVERLAY_NAME, RET_LINE, ERROR_LINE, RUN_RESULT );
    ASSIGN_CON ( 1 );
    ASSIGN_CON ( 2 );
    IF RET_LINE[1]='0' THEN DEPOSIT_MAIL (LETTER, DEPOSIT_STATUS)
                       ELSE  DEPOSIT_STATUS := NOK;
    END "STATUS1 OK"
   ELSE  DEPOSIT_STATUS := NOK;
END; "MEDIT_MAIL"


PROCEDURE SEDIT_MAIL ( LETTER : ENVELOPE; VAR RETURN_STATUS : BOOLEAN );
BEGIN
  WRITETEXT('not yet implemented(:10:)');
  RETURN_STATUS := NOK;
END; "SEDIT_MAIL"


PROCEDURE OTHEREDIT_MAIL ( LETTER : ENVELOPE; VAR RETURN_STATUS : BOOLEAN );
BEGIN
  WRITETEXT('not yet implemented(:10:)');
  RETURN_STATUS := NOK;
END; "OTHER_EDIT"



"  ***********************************************************************
   *                                                                     *
   *                    FILE ENCRYPTION UTILITIES                        *
   *                                                                     *
   ***********************************************************************"
```

```
PROCEDURE ENCRYPT_ALG;
BEGIN
        "PUT ENCRYPTION ALG HERE"
END; "ENCRYPT_ALG"


PROCEDURE DECRYPT_ALG;
BEGIN
        "PUT ENCRYPTION ALG INVERSE HERE"
END; "DECRYPT_ALG"


PROCEDURE CRYPTER ( CODING_DIR : CYPHER_DIR );
VAR CRYPT_FD : FD_TYPE;
    CODER_STATUS : BOOLEAN;
BEGIN
  WRITETEXT(' - CRYPTER -(:10:)');
  GET_FILE_NAME (CRYPT_FD);
  CRYPT_FD.ACCT := 'P';
  FIND_FILE (CRYPT_FD, MAIL_PTR, CODER_STATUS);
  IF CODER_STATUS = OK THEN BEGIN
    MAIL_PTR^.ENCRYP := NOT MAIL_PTR^.ENCRYP;
    DIRTY := TRUE;
    ASSIGN_FILE (CRYPT_FD,5,SVC7_AP_ERW+SVC7_FTYPE_INDEX,CODER_STATUS);
    IF CODER_STATUS = OK THEN CASE CODING_DIR OF
       ENCODE  : ENCRYPT_ALG;
       DECODE  : DECRYPT_ALG;
      END; "CASE CODING DIRECTION"
    ASSIGN_CON ( 5 );
   END; "CODER_STATU = OK"
  IF CODER_STATUS = OK THEN WRITETEXT(' - CRYPTER COMPLETE -(:10:)')
                       ELSE WRITETEXT(' - CRYPTER ERRORS - (:10:)');
END; "CRYPTER"


 " ***************************************************************
   *                                                             *
   *                          FILE XFER PROCS                    *
   *                                                             *
   ***************************************************************"



PROCEDURE INIT_XFER_SVC;
BEGIN
  WITH XFER_SVC1_IN DO BEGIN
    SVC1_FUNC := SVC1_READ + SVC1_IMAGE + SVC1_WAIT;
    SVC1_LU := XFER_IN_LU;
```

```
    SVC1_BUFSTART := ADDRESS ( XFER_FRAME );
    SVC1_BUFEND := SVC1_BUFSTART + FRAME_LENGTH -1;
   END; "WITH XFER_SVC1_IN"
  WITH XFER_SVC1_OUT DO BEGIN
    SVC1_FUNC := SVC1_WRITE + SVC1_IMAGE + SVC1_WAIT;
    SVC1_LU := XFER_OUT_LU;
    SVC1_BUFSTART := ADDRESS ( XFER_FRAME );
    SVC1_BUFEND := SVC1_BUFSTART + FRAME_LENGTH -1;
   END; "WITH XFER_SVC1_OUT"
END; "INIT_XFER_SVC"

PROCEDURE CALC_FRAME_CRC (CRC_DATA: FRAME; VAR CRC_CH : CHAR);
CONST BIAS1 = 25;     "USED TO SHIFT ABOVE CONTROL CODES"
      BIAS2 = 65;     "FOR MICROENGINE PROTECTION"
VAR C_INDX : INTEGER;
BEGIN
  WITH CRC_DATA DO BEGIN
    CRC_CH := CHR (0);
    FOR C_INDX := 1 TO FRAME_DATA_LENGTH DO
      CRC_CH := CHR (((ORD (CRC_CH) + ORD (DATA [C_INDX]) )
                MOD BIAS1 ) + BIAS2);
  END; "WITH CRC_DATA"
END; "CALC_FRAME_CRC"


PROCEDURE CALC_BLOCK_CRC (CRC_DATA: PAGE; VAR CRC_CH : CHAR);
CONST BIAS1 = 25;     "USED TO SHIFT ABOVE CONTROL CODES"
      BIAS2 = 65;     "FOR MICROENGINE PROTECTION"
VAR C_INDX : INTEGER;
BEGIN
  CRC_CH := CHR (0);
  FOR C_INDX := 1 TO PAGELENGTH DO
    CRC_CH := CHR (((ORD (CRC_CH) + ORD (CRC_DATA [C_INDX]) )
              MOD BIAS1 ) + BIAS2);
END; "CALC_BLOCK_CRC"


PROCEDURE ESTAB_XFER_REC_FILE (FILE_DATA: FRAME_DATA;
          VAR XFER_ENVELOP: ENVELOPE; VAR ESTAB_STATUS: BOOLEAN );
VAR I : INTEGER;
BEGIN
  WITH XFER_ENVELOP DO BEGIN
    UPDATE_CT ( TRAFFIC );
    SVC2FDAT ( POST_MARK );
    SOURCE_NAME := LOGON_ID;
    NEXT_LETTER := NIL;
    WITH BOX_ADDR DO BEGIN
      VOLN := 'SYS3';
      PACK_INT ( TRAFFIC_CT, FN );
      FN [1] := 'M';
```

```
      EXTN := 'SYS';
      ACCT := 'P';
     END; "WITH BOX_ADDR"
    FOR I := 1 TO 8 DO
         DEST_NAME [I] := UC (FILE_DATA [ I + XFER_DEST_OFF ]);
    FOR I := 1 TO 16 DO
         SUBJECT [I] := UC (FILE_DATA [ I + XFER_SUBJECT_OFF ]);
    IF FILE_DATA [ XFER_ENCRYPT_OFF + 1 ] = 'T' THEN ENCRYP := TRUE
                                                ELSE ENCRYP := FALSE;
    CREATE_FILE ( BOX_ADDR, PAGELENGTH, ESTAB_STATUS );
    IF ESTAB_STATUS = OK THEN  ASSIGN_FILE ( BOX_ADDR, XFER_OUT_LU,
         SVC7_AP_ERW+SVC7_FTYPE_INDEX, ESTAB_STATUS );
   END; "WITH XFER_ENVELOP"
END; "ESTAB_XFER_REC_FILE"


PROCEDURE ESTAB_XFER_SEND_FILE ( FILE_DATA: FRAME_DATA;
                                 VAR ESTAB_STATUS : BOOLEAN );
VAR SEND_FD : FD_TYPE;
    I : INTEGER;
BEGIN
  WITH SEND_FD DO BEGIN
    FOR I := 1 TO 4 DO VOLN [I] := UC (FILE_DATA [XFER_VOLN_OFF + I]);
    FOR I := 1 TO 8 DO FN [I] := UC (FILE_DATA [XFER_FN_OFF + I]);
    FOR I := 1 TO 3 DO EXTN [I] := UC (FILE_DATA [XFER_EXTN_OFF + I]);
    ACCT := UC (FILE_DATA [XFER_ACCT_OFF + 1]);
   END; "WITH SEND_FD"
  ASSIGN_FILE ( SEND_FD, XFER_IN_LU, SVC7_AP_ERW+SVC7_FTYPE_INDEX,
                ESTAB_STATUS );
END; "ESTAB_XFER_SEND_FILE"


PROCEDURE START_XFER;
VAR START_CHAR : CHAR;
BEGIN
  WITH XFER_SVC1_OUT DO BEGIN
    SVC1_BUFSTART := ADDRESS (START_CHAR);
    SVC1_BUFEND := SVC1_BUFSTART;
    START_CHAR := STX;       "WAKE UP CHAR FOR REMOTE TERM/COMPUTER"
    SVC1 (XFER_SVC1_OUT);
  END; "WITH"
END; "START_XFER"


PROCEDURE SEND_RESPONSE ( RESPONSE : CHAR );
VAR RESPONSE_FRAME : FRAME;
    SAVE_LU : BYTE;
    I : INTEGER;
BEGIN
  WITH XFER_SVC1_OUT DO BEGIN
```

```
    SVC1_BUFSTART := ADDRESS ( RESPONSE_FRAME );
    SVC1_BUFEND := SVC1_BUFSTART + FRAME_LENGTH -1;
    SAVE_LU := SVC1_LU;
    SVC1_LU := 0;          "RESPONSE FRAMES GO TO CON:"
    WITH RESPONSE_FRAME DO BEGIN
      XFER_COMD := RESPONSE;
      FOR I := 1 TO FRAME_DATA_LENGTH DO DATA [ I ] := ' ';
      BLOCK_NO := '1';
      CALC_FRAME_CRC ( RESPONSE_FRAME,CRC );
     END; "WITH RESPONSE_FRAME"
    SVC1 (XFER_SVC1_OUT);        "SHIP OUT CON: PORT"
    SVC1_LU := SAVE_LU;
   END; "WITH XFER_SVC1_OUT"
END; "SEND_RESPONSE"

PROCEDURE READ_FRAME ( VAR R_FRAME : FRAME );
VAR I : INTEGER;
    SAVE_LU : BYTE;
BEGIN
  WITH XFER_SVC1_IN DO BEGIN
    SVC1_BUFSTART := ADDRESS ( R_FRAME );
    SVC1_BUFEND := SVC1_BUFSTART + FRAME_LENGTH - 1;
    SAVE_LU := SVC1_LU;
    SVC1_LU := 0;       "RESPONSE FRAMES GOTO CON:"
    SVC1 ( XFER_SVC1_IN );
                              " ADJST CHARS FROM CRT DRIVER - "
                              " DRIVER TURN HIGH BIT ON  WHY? "
    WITH R_FRAME DO BEGIN
      XFER_COMD := CHR ( ORD (XFER_COMD) MOD 128 );
      BLOCK_NO := CHR ( ORD (BLOCK_NO) MOD 128 );
      CRC := CHR ( ORD (CRC) MOD 128 );
      FOR I := 1 TO FRAME_DATA_LENGTH DO
        DATA [I] := CHR ( ORD (DATA [I]) MOD 128 );
    END; "WITH R_FRAME"
    SVC1_LU := SAVE_LU;
   END; "WITH"
END; "READ_FRAME"


PROCEDURE WRITE_FRAME ( VAR W_FRAME : FRAME );
BEGIN
  WITH XFER_SVC1_OUT DO BEGIN
    SVC1_BUFSTART := ADDRESS ( W_FRAME );
    SVC1_BUFEND := SVC1_BUFSTART + FRAME_LENGTH - 1;
    SVC1 ( XFER_SVC1_OUT );
   END; "WITH"
END; "WRITE_FRAME"


PROCEDURE RECEIVE_FILE ( REQ_FRAME : FRAME; VAR XFER_STATUS : BOOLEAN );
```

```
VAR XFER_BLOCK : PAGE;
    CONT_STATUS,LAST_BLK : BOOLEAN;
    C_INDX,FRAME_CT,NAK_CT : INTEGER;
    BLOCK_CRC : CHAR;
    XFER_MAIL : ENVELOPE;
BEGIN
  ESTAB_XFER_REC_FILE ( REQ_FRAME.DATA, XFER_MAIL, CONT_STATUS );
  IF CONT_STATUS = OK THEN BEGIN
    SEND_RESPONSE ( ACK );
    LAST_BLK := FALSE;
    NAK_CT := 0;
    REPEAT
      FOR FRAME_CT := 1 TO (PAGELENGTH DIV FRAME_DATA_LENGTH) DO BEGIN
        READ_FRAME ( XFER_FRAME );
        FOR C_INDX := 1 TO FRAME_DATA_LENGTH DO
          XFER_BLOCK [ (FRAME_CT-1) * FRAME_DATA_LENGTH + C_INDX ] :=
           XFER_FRAME.DATA [ C_INDX ];
        END; "FRAME_CT LOOP  - BUILD 1 PAGE BLOCK"
      CALC_BLOCK_CRC ( XFER_BLOCK, BLOCK_CRC );
      IF BLOCK_CRC = XFER_FRAME.CRC THEN BEGIN
         IF XFER_FRAME.XFER_COMD = EOM_FLAG THEN LAST_BLK := TRUE
                                            ELSE LAST_BLK := FALSE;
         IF XFER_FRAME.XFER_COMD = ABORT_XFER THEN BEGIN
            LAST_BLK := TRUE;
            NAK_CT := NAK_LIMIT + 1;     "PUT OVER LIMIT SO NO DEPOSIT"
          END; "ABORT_XFER"
         WRITEPAGE (XFER_BLOCK, FALSE);  "ALWAYS RIGHT LAST DATA OUT"
         IF LAST_BLK THEN BEGIN
                           SEND_RESPONSE (EOM_FLAG);
                           WRITEPAGE (XFER_BLOCK,LAST_BLK);
                         END  "WAS THE LAST BLOCK"
                    ELSE SEND_RESPONSE (ACK);
       END "CRC="
       ELSE BEGIN
         NAK_CT := SUCC (NAK_CT);
         IF NAK_CT > NAK_LIMIT THEN LAST_BLK := TRUE;
         IF LAST_BLK THEN SEND_RESPONSE ( ABORT_XFER )
                     ELSE SEND_RESPONSE ( NAK );
       END; "ELSE CRC <>"
    UNTIL LAST_BLK;
   IF NAK_CT < NAK_LIMIT THEN XFER_STATUS := OK ELSE XFER_STATUS := NOK;
   CLOSE_FILE (XFER_OUT_LU);
   IF XFER_STATUS = OK THEN DEPOSIT_MAIL ( XFER_MAIL, XFER_STATUS );
  END; "CONT_STATUS = OK"
  XFER_STATUS := XFER_STATUS AND CONT_STATUS;
END; "RECEIVE_FILE"


PROCEDURE SEND_FILE ( REQ_FRAME : FRAME; VAR XFER_STATUS : BOOLEAN );
VAR XFER_BLOCK : PAGE;
```

```
    LAST_BLK, FINISHED, CONT_STATUS : BOOLEAN;
    NAK_CT, C_INDX, FRAME_CT : INTEGER;
    BLOCK_CRC : CHAR;
BEGIN
  ESTAB_XFER_SEND_FILE ( REQ_FRAME.DATA, CONT_STATUS );
  IF CONT_STATUS = OK THEN BEGIN
    SEND_RESPONSE ( ACK );
    DELAY_IT (SHORT);   "LET THE REMOTE COMP PROCESS THIS RESPONSE"
    FINISHED := FALSE;
    READPAGE ( XFER_BLOCK, LAST_BLK );
    CALC_BLOCK_CRC ( XFER_BLOCK, BLOCK_CRC );
    NAK_CT := 0;
    REPEAT
      FOR FRAME_CT := 1 TO (PAGELENGTH DIV FRAME_DATA_LENGTH) DO BEGIN
         FOR C_INDX := 1 TO FRAME_DATA_LENGTH DO
            XFER_FRAME.DATA [C_INDX] := XFER_BLOCK [ (FRAME_CT - 1 ) *
               FRAME_DATA_LENGTH + C_INDX ];
         IF LAST_BLK THEN XFER_FRAME.XFER_COMD := EOM_FLAG
                     ELSE XFER_FRAME.XFER_COMD := TRANSFER;
         XFER_FRAME.CRC := BLOCK_CRC;
         WRITE_FRAME ( XFER_FRAME );
        END; "BLOCK FRAME LOOP"
      READ_FRAME ( XFER_FRAME );     "GET REMOTE RESPONSE"
      IF XFER_FRAME.XFER_COMD = NAK
      THEN BEGIN
         NAK_CT := SUCC ( NAK_CT );
         IF NAK_CT > NAK_LIMIT THEN BEGIN
            XFER_STATUS := NOK;
            FINISHED := TRUE;
          END; "OVER NAK LIMIT"
       END;    "NAK RESPONSE"
      IF LAST_BLK AND (XFER_FRAME.XFER_COMD = EOM_FLAG)
      THEN BEGIN
           FINISHED := TRUE;
           XFER_STATUS := OK;
        END; "ACK & LAST BLOCK"
      IF (XFER_FRAME.XFER_COMD=ACK)
          AND NOT LAST_BLK THEN BEGIN   "ACK BUT NOT LAST BLOCK"
           READPAGE ( XFER_BLOCK, LAST_BLK );
           CALC_BLOCK_CRC ( XFER_BLOCK, BLOCK_CRC );
          END; "ACK BUT NOT LAST BLOCK"
      IF (XFER_FRAME.XFER_COMD = ABORT_XFER) THEN FINISHED := TRUE;
      DELAY_IT (SHORT);
     UNTIL FINISHED;
   END "CONT_STATUS = OK"
  ELSE BEGIN
    XFER_STATUS := NOK;
    SEND_RESPONSE (ABORT_XFER);
    DELAY_IT (MEDIUM);
   END; "NOT VALID FILE"
```

```
END; "SEND_FILE"

 " ********************************************************************
   *                                                                  *
   *                          COMMAND MENU                            *
   *                                                                  *
   ********************************************************************"


PROCEDURE MENU;
VAR OK_STATUS : BOOLEAN;
    I : INTEGER;
BEGIN
  DISPLAY (NL);
  DISPLAY (NL);
  DISPLAY (NL);
  DISPLAY (NL);
  DISPLAY (NL);
  WRITETEXT('Mail Manager R00-03      type "H"elp for command summary(:10:)');
  WRITETEXT('Please report problems to  K.W. Janne(:10:)');
  DISPLAY (NL);
  WRITETEXT('Enter command   H)elp, I)nquire, G)et, C)reate, (:0:)');
  WRITETEXT('R)emove, P)urge, M)ail,(:10:)');
  WRITETEXT('                F)orward, T)ype, X)fer, (:0:)');
  WRITETEXT('E)ncrypt, D)ecrypt, S)et_edit,(:10:)');
  WRITETEXT('                (:0:)');   "SPACING"
  IF (RUN_MODE=EXEC) OR (RUN_MODE=DEBUG)
    THEN WRITETEXT('N)vironment, L)ist_stat, (:0:)');
  IF RUN_MODE = DEBUG THEN WRITETEXT('B)reak, (:0:)');
  WRITETEXT('Q)uit(:10:)');
  DISPLAY (NL);
  GET_COMMAND (USER_CMD);
  DISPLAY (NL);
  ABORT := FALSE;
    CASE USER_CMD OF
        HELP : COMMAND_SUMMARY;
        INQUIRE : LIST;
        GET : BEGIN
                WRITETEXT(' - GET FILE -(:10:)');
                GET_FILE_NAME (FILE_NAME);
                FIND_FILE (FILE_NAME, MAIL_PTR, EXEC_STATUS);
                IF EXEC_STATUS = OK THEN BEGIN
                  FILE_NAME.ACCT := 'P';
                  RENAME_FILE (MAIL_PTR^.BOX_ADDR, FILE_NAME, EXEC_STATUS);
                  IF EXEC_STATUS = OK THEN BEGIN
                    REMOVE_MAIL (MAIL_PTR^, EXEC_STATUS);
                    IF EXEC_STATUS = OK THEN
                      WRITETEXT(' - GET COMPLETED -(:10:)');
```

```
            END;
          END;
        IF EXEC_STATUS = NOK THEN
          WRITETEXT(' - GET ERRORS - FILE NOT RETRIEVED(:10:)');
        END;
CREATE : BEGIN
          WITH MAIL DO BEGIN
            GET_USERNAME ( DEST_NAME );
            GET_SUBJECT ( SUBJECT );
            SOURCE_NAME := LOGON_ID;
            SVC2FDAT ( POST_MARK );
            ENCRYP := FALSE;
            NEXT_LETTER := NIL;
            UPDATE_CT ( TRAFFIC );
            WITH BOX_ADDR DO BEGIN
              VOLN := 'SYS3';
              PACK_INT ( TRAFFIC_CT , FN );
              FN [1] := 'M';
              EXTN := 'SYS';
              ACCT := 'P';
              END; "WITH BOX_ADDR"
           CREATE_FILE (BOX_ADDR, PAGELENGTH, EXEC_STATUS );
            IF EXEC_STATUS = OK THEN BEGIN
              CASE MAIL_EDITOR OF
                MEDIT : MEDIT_MAIL ( MAIL, EXEC_STATUS );
                SEDIT : SEDIT_MAIL ( MAIL, EXEC_STATUS );
                OTHEREDIT : OTHEREDIT_MAIL ( MAIL, EXEC_STATUS );
               END; "CASE"
              IF EXEC_STATUS = OK THEN
                WRITETEXT(' - MAIL DEPOSITED -(:10:)');
            END; "EXEC_STATUS OK "
            IF EXEC_STATUS = NOK THEN BEGIN
              WRITETEXT(' - MAIL CREATE ERRORS -(:10:)');
              DELETE_FILE ( MAIL.BOX_ADDR, EXEC_STATUS );
             END; "CREATE ERRORS"
            END; "WITH"
            END; "CREATE"
REMOVE : BEGIN
            WRITETEXT(' - REMOVE MAIL -(:10:)');
            GET_FILE_NAME (FILE_NAME);
            FIND_FILE (FILE_NAME,MAIL_PTR,EXEC_STATUS);
            IF EXEC_STATUS = OK THEN BEGIN
              DIRTY := TRUE;
              MAIL := MAIL_PTR^;
              REMOVE_MAIL ( MAIL, EXEC_STATUS );
              DELETE_FILE (MAIL.BOX_ADDR, EXEC_STATUS);
              IF EXEC_STATUS = OK THEN
              WRITETEXT(' - MAIL REMOVED -(:10:)');
              END; "EXEC OK"
            IF EXEC_STATUS = NOK THEN BEGIN
```

```
            WRITETEXT(' - REMOVE MAIL ERRORS(:0:)');
            WRITETEXT(' - NOT REMOVED(:10:)');
          END; "NOK"
        END; "REMOVE"
TYPE_M : DISPLAY_FILES;
FORWARD_M : BEGIN
              WRITETEXT(' - FORWARD MAIL -(:10:)');
              FORWARD_MAIL (EXEC_STATUS);
              IF EXEC_STATUS = OK THEN WRITETEXT(' - MAIL FORWARDED -(:10:
                                  ELSE WRITETEXT(' - MAIL NOT FORWARDED -(
            END; "FORWARD_M"
LIST_STAT : BEGIN
        IF (RUN_MODE=EXEC) OR (RUN_MODE=DEBUG) THEN BEGIN
          UPDATE_CT (NEITHER);
          WRITETEXT('- System Statistics -(:10:)');
          DISPLAY (NL);
          WRITETEXT('Total Traffic Ct = (:0:)');
          WRINT (TRAFFIC_CT);
          WRITETEXT('       Current Entry Ct = (:0:)');
          WRINT (DIRECT_CT);
          DISPLAY (NL);
          DISPLAY (NL);
          DISPLAY_USERS;
          END "PRIVILEDGED USERS"
        ELSE WRITETEXT('(:07:)(:10:)');
      END; "STAT"
NVIRONMENT : SET_MODE;
SET_EDIT : SET_EDITOR;
PURGE : BEGIN
          DIRTY := TRUE;
          MAIL_PTR := MAIL_HEAD;
          WHILE MAIL_PTR <> NIL DO BEGIN
            MAIL_TRAIL := MAIL_PTR^.NEXT_LETTER;
            DELETE_FILE (MAIL_PTR^.BOX_ADDR,EXEC_STATUS);
            RECYCLE_PTR ( MAIL_PTR );
            MAIL_PTR := MAIL_TRAIL;
            END; "WHILE"
          MAIL_HEAD := NIL;
          MAIL_TAIL := NIL;
          WRITETEXT(' - MAIL PURGED -(:10:)');
        END; "PURGE"
BREAK : IF (RUN_MODE=DEBUG) THEN BREAKPT (2000)
          ELSE WRITETEXT('(:07:)(:10:)');
MAIL_ : BEGIN
            WRITETEXT(' - DEPOSIT MAIL -(:10:)');
            WITH MAIL DO BEGIN
              GET_USERNAME ( DEST_NAME );
              GET_SUBJECT (SUBJECT);
              GET_FILE_NAME (BOX_ADDR);
              BOX_ADDR.ACCT := 'P';  "ASSUME GET PRIV FILE"
```

```
                    SOURCE_NAME := LOGON_ID;
                    ENCRYP := FALSE;
                    SVC2FDAT (POST_MARK);
                    NEXT_LETTER := NIL;
                    END; "WITH"
                  DEPOSIT_MAIL (MAIL,EXEC_STATUS);
                  IF EXEC_STATUS = OK THEN WRITETEXT('- MAIL DEPOSITED -(:10:)')
                                      ELSE WRITETEXT('- NOT DEPOSITED -(:10:)');
                END; "DEPOSIT MAIL"
        ENCRYPT : CRYPTER ( ENCODE );
        DECRYPT : CRYPTER ( DECODE );
        XFER : BEGIN
                START_XFER; "WAKE UP & LET KNOW ALIVE"
                READ_FRAME ( XFER_FRAME );  "GET MICRO COMMAND"
                CASE XFER_FRAME.XFER_COMD OF
                 REQ_SEND: RECEIVE_FILE ( XFER_FRAME, EXEC_STATUS );
                 REQ_REC : SEND_FILE ( XFER_FRAME, EXEC_STATUS );
                 ABORT_XFER   : EXEC_STATUS := NOK;
                 END; "CASE"
                ASSIGN_CON ( XFER_IN_LU );
                ASSIGN_CON ( XFER_OUT_LU );
               END; "XFER"
        QUIT_CMD  : ABORT := TRUE;
        END; "CASE"
END; "MENU"


PROCEDURE INIT_GLOBAL_VARS;
BEGIN
  RUN_MODE := USER;
  FREE_PTR := NIL;
  DIRTY := FALSE;
  USRS_DIR_NAME.FN := 'UNUSED  ';
  MAIL_EDITOR := MEDIT;
  INIT_XFER_SVC;    "INITIALIZES SVC1 BLOCK FOR FILE XFERS"
END; "INIT_GLOBAL_VARS"


PROCEDURE INIT_COMMANDS;
BEGIN
  COMMAND [ HELP ]          := 'H';
  COMMAND [ INQUIRE ]       := 'I';
  COMMAND [ GET ]           := 'G';
  COMMAND [ CREATE ]        := 'C';
  COMMAND [ REMOVE ]        := 'R';
  COMMAND [ MAIL_ ]         := 'M';
  COMMAND [ FORWARD_M ]     := 'F';
  COMMAND [ LIST_STAT ]     := 'L';
  COMMAND [ NVIRONMENT ]    := 'N';
  COMMAND [ PURGE ]         := 'P';
```

```
  COMMAND [ BREAK ]          := 'B';
  COMMAND [ TYPE_M ]         := 'T';
  COMMAND [ ENCRYPT ]        := 'E';
  COMMAND [ DECRYPT ]        := 'D';
  COMMAND [ SET_EDIT ]       := 'S';
  COMMAND [ XFER ]           := 'X';
  COMMAND [ QUIT_CMD ]       := 'Q';
END; "COMMAND INIT"


BEGIN "MAIN"
  ENTER_ACTIVE_TABLE (ABORT);
  INIT_GLOBAL_VARS;
  INIT_COMMANDS;
  IF MAIL_WAITING (LOGON_ID) THEN BEGIN
     FETCH_USR_DIR (LOGON_ID, USRS_DIR_NAME);
     LOAD_DIR (USRS_DIR_NAME, ABORT);
     END "MAIL_WAITING"
    ELSE MAIL_HEAD := NIL;
  WHILE (ABORT=FALSE) DO MENU;
  IF TEMP_MAIL (LOGON_ID) THEN TEMP_EXTRACT (LOGON_ID);
  IF (USRS_DIR_NAME.FN='UNUSED  ') AND DIRTY THEN BEGIN
    CREATE_DIR (LOGON_ID,EXEC_STATUS);
    FETCH_USR_DIR (LOGON_ID, USRS_DIR_NAME);
    END; "DIRECT DID NOT EXIST BEFORE"
  IF DIRTY THEN DOWN_LOAD_DIR ( USRS_DIR_NAME );
  LEAVE_ACTIVE_TABLE;
END.
```

```
(*$i-*)                 (* enables use of ioresult checking              *)
(* lprinter: *)         (* sends compiler genreated listing to printer   *)
(* lremote:  *)         (* sends compiler generated listing to remote    *)
(* q+        *)         (* suppresses output of compiler to console      *)

program maillink;

  (*********************************************************************)
  (*                                                                   *)
  (*   This program establishes the western digital as a smart term-   *)
  (*  inal interface with the Interdata 8/32 operating the MAILMAN3    *)
  (*  remote computer communication interface program.  The connection*)
  (*  is via the USCD convention remote port                           *)
  (*                                                                   *)
  (*  Author: K.W. Janne                                               *)
  (*  Revision: 7/01/82                                                *)
  (*                                                                   *)
  (*  Microengine: G1 operating system                                 *)
  (*               REMOTE port at 1200 baud                            *)
  (*  8/32: MTM R00-04 OS                                              *)
  (*        direct connect with PA2E at 1800/1200  "A"/"B"             *)
  (*        dial in communicaation at 1200                             *)
  (*                                                                   *)
  (*********************************************************************)


  CONST
    console  = 1;          (* UCSD device labels  *)
    systerm  = 2;          (* non echoing console *)
    printer  = 6;          (* parallel port       *)
    remote   = 8;          (* second serial port  *)

    ack      = '1';        (* file xfer command consts *)
    nak      = '2';
    abort    = '3';
    req_send = '4';        (* microengine -> 8/32      *)
    req_rec  = '5';        (* 8/32 -> microengine      *)
    eom_flag = '6';
    transfer = '7';        (* valid xfer data          *)

    ok = true;
    nok = false;

    no_error = 0;          (* ioresult return code *)

    frame_length = 67;     (* length of transfer frame. this length is *)
                           (* restricted by 8/32 to 80 chars           *)

    frame_d_length = 64;   (* number of valid data chars in 1 frame *)
```

```
  xfer_comd      = 1;      (* frame array offsets                    *)
  xfer_block_no = 2;
  xfer_data      = 3;
  xfer_crc       = 67;

  xfer_dest_off     = 2;
  xfer_subject_off = 10;     (* data offsets within command frame     *)
  xfer_encrypt_off = 26;
  xfer_voln_off     = 2;
  xfer_fn_off       = 6;
  xfer_extn_off     = 14;
  xfer_acct_off     = 17;

  nak_limit = 3;         (* limit before xfer abort *)

  pagelength = 512;


TYPE
  unit_char = 0..255;          (* used for chars in unitread - aligns correc *)

  frame_data = packed array [ 1..frame_d_length ] of unit_char;

  frame = packed array [ 1..frame_length ] of unit_char;

  page = packed array [ 1..pagelength ] of unitchar;

  me_fd = string [20];

  delimiter = set of unit_char;



VAR
  c,lf, cr, request_char : char;
  unit_ch : packed array [ 1..1 ] of unit_char;
  block : page;
  xfer_frame : frame;
  xfer_file : me_fd;
  frame_ct, frame_offset, block_ct, nak_ct, blocks_xfereo  : integer;
  cont_status, quiting, eof_flag ,abend : boolean;
  mode_delimiters : delimiter;
  abort_char, stx, etx : unit_char;
  bit_bucket : packed array [1..99] of unit_char;  (* used for pad char discard *)
  diskin,diskout : file;


PROCEDURE clear_screen;
begin
  writeln( chr(26) );      (* clear screen code for adm 3a *)
```

```
end;  (* clear screen *)


PROCEDURE dum_term (var aborting : boolean);
var change_mode : boolean;
begin
  change_mode := false;
  repeat
    if unitbusy (systerm) then begin
       unitread (systerm,unit_ch [1],1,,13);
       if unit_ch [1] in mode_delimiters then change_mode := true
         else unitwrite (remote,unit_ch [1],1,,13);
      end; (* unit busy systerm *)
    if unitbusy (remote) then begin
       unitread (remote,unit_ch [1],1,,13);
       unit_ch [1] := unit_ch [1] mod 128;   (* 8/32 turns high bit on? *)
       if unit_ch [1] in mode_delimiters then change_mode := true
         else unitwrite (systerm,unit_ch [1],1,,13);
      end; (* unit busy remote *)
   until change_mode;
  if unit_ch [1] = abort_char then aborting := true else aborting := false;
end; (* dum_term *)


PROCEDURE calc_frame_crc ( crc_frame : frame; var crc_ch : unit_char );
const bias1 = 25;    (* used to shift above control codes *)
      bias2 = 65;    (* for microengine protection        *)
var c_indx : integer;
begin
  crc_ch := ord(0);
  for c_indx := xfer_data to xfer_data + frame_d_length do
    crc_ch := ((crc_ch + crc_frame [c_indx]) mod bias1 ) + bias2;
end;


PROCEDURE calc_block_crc ( crc_data : page; var crc_ch : unit_char );
const bias1 = 25;    (* used to shift above control codes *)
      bias2 = 65;    (* for microengine protection        *)
var c_indx : integer;
begin
  crc_ch := ord(0);
  for c_indx := 1 to pagelength do
    crc_ch := ((crc_ch + crc_data [c_indx]) mod bias1 ) + bias2;
end;  (* calc_block_crc *)


PROCEDURE get_request ( var comd_char : char );
var mode : packed array [1..1] of char;
begin
  clear_screen;
```

```
  writeln (' ');
  writeln (' ');
  writeln (' ');
  writeln ('                                UCSD Remote <=> 8/32');
  writeln ('                             File Transfer Control  R00-01');
  writeln (' ');
  writeln (' ');
  writeln ('Enter Xfer Mode          1    UCSD -> 8/32');
  writeln ('                         2    8/32 -> UCSD');
  writeln ('                         3    ABORT');
  writeln (' ');
  write   ('Mode = ');
  unitread(console,mode[1],1,,13);
  writeln (' ');
  writeln (' ');
  if mode[1] = '1' then comd_char := req_send else
    if mode[1] = '2' then comd_char := req_rec else
      comd_char := abort;
end; (* get_request *)


PROCEDURE estab_send_file ( var assign_file : integer );
var ucsd_file : string [20];
begin
  clear_screen;
  writeln(' ');
  writeln(' ');
  writeln(' ');
  writeln('                                UCSD => 8/32');
  writeln('                             File Transfer  R00-01');
  writeln(' ');
  writeln(' ');
  writeln(' ');
  writeln(' ');
  write('Enter UCSD source file name (must use full name) ... ');
  readln (ucsd_file);
  reset (diskin, ucsd_file);
  assign_file := ioresult;
end; (* estab_send_file *)


PROCEDURE estab_rec_file ( var assign_file : integer );
var ucsd_file : string [20];
begin
  clear_screen;
  writeln(' ');
  writeln(' ');
  writeln(' ');
  writeln('                                8/32 => UCSD');
  writeln('                             File Transfer  R00-01');
```

```
  writeln(' ');
  writeln(' ');
  writeln(' ');
  writeln(' ');
  write('Enter UCSD dest file name (must use full name) ... ');
  readln (ucsd_file);
  rewrite (diskout, ucsd_file);
  assign_file := ioresult;
end; (* estab_rec_file *)


PROCEDURE get_dest_info;
var dest_id : string [8];
    subject : string [16];
    encrypt_flag : packed array [1..1] of char;
    i : integer;
begin
  writeln (' ');
  dest_id := '        ';    (* init strings to blanks *)
  subject := '                ';
  write  ('Enter destination user ID ..... ');
  readln (dest_id);
  write  ('Enter file/mail subject  (16 chrs).. ');
  readln (subject);
  write  ('Is file encrypted .. (y/n) ..... ');
  unitread (console,encrypt_flag[1],1,,13);
              (* init xfer_frame fields with spaces *)
  for i := 1 to 8 do xfer_frame [ xfer_dest_off + i ] := 32;
  for i := 1 to 16 do xfer_frame [ xfer_subject_off + i ] := 32;
  for i := 1 to length (dest_id ) do
      xfer_frame [ xfer_dest_off + i ] := ord ( dest_id [i] );
  for i := 1 to length (subject ) do
      xfer_frame [ xfer_subject_off + i ] := ord ( subject [i] );
  if (encrypt_flag [1] = 'y') or (encrypt_flag [1] = 'Y')
    then xfer_frame [ xfer_encrypt_off + 1 ] := ord ('T')
    else xfer_frame [ xfer_encrypt_off + 1 ] := ord ('F');
  end;   (* get_dest_info *)


PROCEDURE get_source_info;
var s_voln : string [4];
    s_fn : string [8];
    s_extn : string [3];
    s_acnt : string [1];
    i : integer;
begin
  writeln (' ');
  s_voln := '    ';          (* init fields to blanks *)
  s_fn := '        ';
  s_extn := '   ';
```

```
  s_acnt := ' ';
  write ('Enter source voln name ..... ');
  readln (s_voln);
  write ('Enter source file .. (8 chrs) ..... ');
  readln (s_fn);
  write ('Enter source extn ..... ');
  readln (s_extn);
  write ('Enter source account  ( P/G ) ..... ');
  readln (s_acnt);
                        (* now clear out and init xfer frame *)
  for i := 1 to 4 do xfer_frame [xfer_voln_off + i ] := 32;
  for i := 1 to 8 do xfer_frame [xfer_fn_off + i ] := 32;
  for i := 1 to 3 do xfer_frame [xfer_extn_off + i ] := 32;
  xfer_frame [ xfer_acct_off + 1 ] := 32;
                        (* now fill xfer frame *)
  for i := 1 to length (s_voln) do
     xfer_frame [xfer_voln_off + i ] := ord ( s_voln [i] );
  for i := 1 to length (s_fn) do
     xfer_frame [xfer_fn_off + i ] := ord ( s_fn [i] );
  for i := 1 to length (s_extn) do
     xfer_frame [xfer_extn_off + i ] := ord ( s_extn [i] );
  xfer_frame [xfer_acct_off + i ] := ord ( s_acnt [1] );
end;  (* get_source_info *)


PROCEDURE send_frame (output_frame : frame);
begin
  unitwrite (remote, output_frame[1], frame_length,,13 );
end;   (* send_frame *)


PROCEDURE read_frame ( var input_frame : frame );
begin
  unitread (remote, bit_bucket[1],4,,13);  (* read pads *)
  unitread (remote, input_frame[1], frame_length,, 13 );
  unitread (remote, bit_bucket[1],8,,13);  (* read trailing pads *)
end;  (* read_frame    from remote port *)


PROCEDURE send_blocks ( var ack_frame : frame; rel_block : integer );
var crc : unit_char;
    offset, i : integer;
begin
  blocks_xfered := blocks_xfered + blockread (diskin,block,1,rel_block );
  calc_block_crc ( block, crc );
  if eof (diskin) then xfer_frame [ xfer_cond ] := ord (eom_flag)
                  else xfer_frame [ xfer_cond ] := ord (transfer);
  xfer_frame [ xfer_block_no ] :=  255; (* lower control values give me errors *)
  xfer_frame [ xfer_crc ] := crc;
  for frame_ct := 0 to 7 do begin
```

```
    for offset := 0 to frame_d_length-1 do
       xfer_frame [ xfer_data + offset  ] :=
        block [ offset + frame_ct * frame_d_length + 1];
    write ('.');
    unitwrite (remote, xfer_frame [1],frame_length,,13);
    if frame_ct < 7 then unitread (remote,bit_bucket[1],8,,13);
    end; (* block frame loop *)
  unitread (remote, bit_bucket[1],12+frame_length,,13);
  for i := 1 to frame_length do
    ack_frame [i] := bit_bucket [i+4];
  write (ack_frame[1]);
end;   (* send_blocks *)


PROCEDURE read_blocks ( var ack_frame : frame; var rel_block : integer );
var crc : unit_char;
    i, offset, length : integer;
    block_buf : packed array [1..600] of unit_char;
begin
  if rel_block >= 0 then
     (* don't do this the first time through *)
      unitwrite (remote, ack_frame[1],frame_length,,13);
  unitread (remote,block_buf[1],576,,13);  (* read everthing incl pads *)
  write('*');
  for frame_ct := 0 to 7 do begin
    for offset := 1 to frame_d_length do
      block [ frame_ct * frame_d_length + offset ] :=
        block_buf [ (frame_ct * (frame_length + 4)) + 6 + offset ];
    write ('.');
   end; (* frame_loop *)
  if block_buf [ xfer_comd + 4 ] = ord (eom_flag)
    then begin
           close (diskout, lock);
           ack_frame [xfer_comd] := ord (eom_flag);
           for i := 1 to frame_length -1 do
               ack_frame [ xfer_comd + i ] := ord (ack);
           unitwrite (remote, ack_frame [1],frame_length,,13);
         end  (* eom *)
    else begin
           calc_block_crc ( block, crc );
           if rel_block < 0 then rel_block := 0;  (* reset after 1 pass *)
           if crc = block_buf [ xfer_crc + 4 ]
            then begin
                 blocks_xfered := blockwrite (diskout,block,1,rel_block);
                 ack_frame [xfer_comd] := ord (ack);
                end  (* crc = *)
            else ack_frame [xfer_comd] := ord (nak);
         end;  (* not eom *)
  for i := 1 to frame_length -1 do
      ack_frame [ xfer_comd + i ] := ord (ack);
```

```
end; (* read_blocks *)


PROCEDURE send_file ( var aborting : boolean );
var return_code, i : integer;
    response_frame : frame;
begin
  aborting := false;
  block_ct := 0;
  nak_ct := 0;
  estab_send_file ( return_code );
  if return_code = no_error
  then begin
         get_dest_info;
         writeln(' ');
         writeln(' ');
         writeln('TRANSFERING FILE');
         unitwrite (remote, xfer_frame[1], frame_length,,13);
         unitread (remote, bit_bucket[1],12+frame_length,,13);
         for i := 1 to frame_length do
           response_frame [i] := bit_bucket [i+4];
         repeat
           send_blocks ( response_frame , block_ct );
           if response_frame [xfer_comd] = ord (ack)     (* shift for 8/32  *)
              then block_ct := succ (block_ct)
              else nak_ct := succ (nak_ct);
         until (nak_ct > nak_limit) or (response_frame [1] = ord (eom_flag));
         end   (* estab_send_file ok *)
  else begin
         writeln (' ');
         writeln ('UCSD file establish error = ',return_code);
         writeln (' ');
         xfer_frame [ xfer_comd ] := ord (abort);
         send_frame (xfer_frame);
         aborting := true;
       end;
  close (diskin, lock);
  if nak_ct > nak_limit then aborting := true;
end; (* send_file *)


PROCEDURE receive_file ( var aborting : boolean );
var return_code, i : integer;
    response_frame : frame;
begin
  aborting := false;
  block_ct := -1;
  nak_ct := 0;
  estab_rec_file ( return_code );
```

```
  if return_code = no_error
  then begin
         get_source_info;
         writeln(' ');
         writeln(' ');
         writeln('TRANSFERING FILE');
         unitwrite (remote,xfer_frame [1],frame_length,,13);
         unitread (remote,bit_bucket[1],frame_length+4,,13);
         if bit_bucket[xfer_comd+4] = ord (ack) then begin
           repeat
             read_blocks ( response_frame, block_ct );
             if response_frame [xfer_comd] = ord (ack) then
                write ('+') else write ('-');
             if response_frame [xfer_comd] = ord (ack)
               then block_ct := succ (block_ct)
               else nak_ct := succ (nak_ct);
            until (nak_ct > nak_limit) or
                  (response_frame [xfer_comd] = ord (eom_flag));
           if nak_ct > nak_limit
             then begin
                aborting := true;
                for i := 1 to frame_length -1 do
                    response_frame [ xfer_comd + i ] := ord (ack);
                unitwrite (remote, response_frame [1],frame_length,,13);
               end; (* > then nak limit *)
          end;  (* established ok *)
       end;
    if (return_code <> no_error) or (bit_bucket[5] = ord (abort))
    then begin
         writeln (' ');
         writeln ('UCSD file establish error = ',return_code);
         writeln (' ');
         xfer_frame [ xfer_comd ] := ord (abort);
         if bit_bucket[5] <> ord (abort) then send_frame (xfer_frame);
         aborting := true;
         close (diskout);
       end; (* establish errors *)
end; (* receive_file *)


PROCEDURE initilize;
const
  heading1 = '            Western Digital <-> 8/32 Communication Interface';
  heading2 = '                  R00-01   June 1982  K.W. Janne';
  heading3 = '                       type cntrl_q to abort';
begin
  lf := chr(10);
  cr := chr(13);
  stx := 1; (*2*)     (* make compatible with 8/32 - offset 1 *)
  etx := 2;
```

```
    abort_char := 17;       (* cntrl_q *)
    mode_delimiters := [ abort_char, stx ];
    nak_ct := 0;
    clear_screen;
    writeln (cr);
    writeln (heading1);
    writeln (heading2);
    writeln (cr);
    writeln (heading3);
    writeln (cr);
    writeln (cr);
    block_ct := 0;
  end; (* initilize *)


BEGIN                          (* main *)
  initilize;
  repeat
    dum_term (quiting);
    if not quiting then begin  (* must be a file transfer *)
      get_request ( request_char );
      case request_char of
        req_send : begin
                     xfer_frame [xfer_comd] := ord (req_send);
                     send_file ( abend );
                     writeln(' ');
                     if abend then writeln(' FILE TRANSFER ERRORS ')
                              else writeln(' FILE TRANSFER COMPLETE ');
                     end; (* req_send *)
        req_rec  : begin
                     xfer_frame [xfer_comd] := ord (req_rec);
                     receive_file ( abend );
                     writeln(' ');
                     if abend then writeln(' FILE TRANSFER ERRORS ')
                              else writeln(' FILE TRANSFER COMPLETE ');
                     end; (* rec_rec *)
        abort    : begin
                     xfer_frame [xfer_comd] := ord (abort);
                     send_frame (xfer_frame);
                     end; (* abort transfer *)
        end; (* case *)
    end;   (* file transfer *)
  until quiting;
end.  (* maillink *)
```

```
                    (*                  compiler options                    *)
(*$i-        *)      (* enables use of ioresult checking                    *)
(* lprinter:*)      (* sends compiler generated listing to printer         *)
(* lremote: *)      (* sends compiler generated listing to spinwriter      *)
(* q+        *)     (* suppresses output of compiler to console            *)

program incnvtr;        (*  Microengine  ->  8/32  *)

(*************************************************************************)
(*                                                                       *)
(*   This program transforms a file created under the Microengine        *)
(*   editor to a format usable under the 8/32 text line editor.          *)
(*   The converted version should not be labeled with an extension       *)
(*   so that the file transfer mechansims will correctly read the        *)
(*   file as ascii data.                                                 *)
(*                                                                       *)
(*   Author:   K.W. Janne                                                *)
(*   Revision:   5/23/82         G1: Operating System                    *)
(*                                                                       *)
(*************************************************************************)


const  pgsize = 1024;

var    ind_ct : integer;       (* how far line is indented *)
       ch : char;
       cr,lf,nul,soh,dle,eom : char;  (* ascii control characters *)
       pg_byte_ct : integer;  (* current page byte count  1<=x<=1024 *)
       end_mark_changed,finished,end_of_prog : boolean;
       i : integer;
       disktemp,diskin,diskout : file of char;
       destination_file,source_file : string [20];


procedure clear_screen;

  begin
    writeln( chr(26) );   (* clears the screen on an ADM 3A *)
  end; (* clear screen *)


procedure ioerror ( error_no : integer );

  begin
    clear_screen;
    writeln(' ');
    writeln('***** ERROR ***** routine aborted   i/o error ',error_no);
    writeln('                  see micro ref manual pg 124');
    writeln(' ');
    writeln('type <CR> to cont.   ');
```

```
    readln(ch);
  end; (* ioerror *)


procedure init_files ( var aborting : boolean );

  begin
    clear_screen;
    writeln(' ');
    writeln(' ');
    writeln(' ');
    writeln('                                Microengine  ->  8/32 ');
    writeln('                              Editor Text File Converter');
    writeln(' ');
    writeln(' ');
    write('enter name of file to convert..(must be ".TEXT").....');
    readln( source_file );
    reset(diskin,source_file);
    if ioresult = 0
      then aborting := false
      else begin
             ioerror (ioresult);  (* aborts the conversion due to an abnormal *)
                                  (* return code of the disk operation       *)
             aborting := true;
             end;
    if not aborting
    then begin
           write('enter name of dest. file  ( do NOT use extension)....');
           readln( destination_file );
           rewrite(diskout,destination_file);
           if ioresult = 0
             then aborting := false
             else begin
                    ioerror ( ioresult );  (* same as above *)
                    aborting := true;
                    end;
           end;
    clear_screen;
    end; (* init_files *)


procedure byte_ct;

  begin
    pg_byte_ct := succ ( pg_byte_ct );
    if pg_byte_ct = pgsize
    then begin
           pg_byte_ct := 0;
           writeln('*');
```

```
          end
    end; (* byte count *)



begin    (* main procedure *)

  nul := chr(00);
  soh := chr(1);
  lf  := chr(10);
  cr  := chr(13);
  dle := chr(16);
  eom := chr(25);

  pg_byte_ct := 0;
  end_of_prog := false;

  init_files ( finished );
  if not finished then
    begin
      writeln(' ');
      writeln(' ');
      writeln(' ');
      writeln('                                  Microengine  ->  8/32 ');
      writeln('                                Editor Text File Converter');
      writeln(' ');
      writeln(' ');
      writeln('"converting"');
      writeln(' ');
      ch := diskin ^;
      repeat
        while not eoln ( diskin )  and  not eof ( diskin )  do begin
           diskout ^ := ch;
           put ( diskout );
           byte_ct;
           get ( diskin );
           ch := diskin  ^;
         end; (* while *)

    (* ASSERT: ch = cr  or eof ( diskin )   *)

        diskout ^ := lf;
        put ( diskout );
        byte_ct;
        write('.');
        if eof ( diskin ) then finished := true
                          else begin
                                 get ( diskin );
                                 ch := diskin ^;
```

```
                                    end;
      until finished;   (* eof diskin *)
    diskout ^ := eom;
    put (diskout);
    byte_ct;
    for i := pg_byte_ct to pgsize - 1  (* fill remaining page with nuls *)
    do begin
         diskout^ := nul;
         put (diskout);
         end;
    close(diskin,lock);
    close(diskout,lock);
    writeln(' ');
    writeln(' ');
    write('normal termination                    ');
  end; (* not finsihed *)
  write('press <CR> to return to O.S.');
  readln(ch);
  clear_screen;
end.
```

```
                    (*              compiler options                    *)
(*$i-        *)      (* enables use of ioresult checking                *)
(* lprinter:*)      (* sends compiler generated listing to printer     *)
(* lremote: *)      (* sends compiler generated listing to spinwriter  *)
(* q+        *)     (* suppresses output of compiler to console        *)

program mecnvtr;        (*  8/32  ->  Microengine  *)

(*************************************************************************)
(*                                                                       *)
(*   This program transforms a file created under the 8/32 editor        *)
(*   to a format usable under the microengine full scree editor.         *)
(*   The original  version should not be labeled with an extension       *)
(*   so that the converted version is unique as '.TEXT'                  *)
(*                                                                       *)
(*   Author:   K.W. Janne                                                *)
(*   Revision:   5/23/82         G1: Operating System                    *)
(*                                                                       *)
(*************************************************************************)


const  pgsize = 1024;

var    ind_ct : integer;       (* how far line is indented *)
       ch : char;
       cr,lf,nul,soh,die,eom : char;  (* ascii control characters *)
       pg_byte_ct : integer;  (* current page byte count  1<=x<=1024 *)
       end_mark_changed,finished,end_of_prog : boolean;
       i : integer;
       disktemp,diskin,diskout : file of char;
       destination_file,source_file : string [20];


procedure clear_screen;

  begin
    writeln( chr(26) );    (* clears the screen on an ADM 3A *)
  end; (* clear screen *)


procedure ioerror ( error_no : integer );

  begin
    clear_screen;
    writeln(' ');
    writeln('***** ERROR ***** routine aborted   i/o error ',error_no);
    writeln('                  see micro ref manual pg 124');
    writeln(' ');
    writeln('type <CR> to cont.   ');
    readln(ch);
```

```
  end; (* ioerror *)


procedure init_files ( var aborting : boolean );

  begin
    clear_screen;
    writeln(' ');
    writeln(' ');
    writeln(' ');
    writeln('                              8/32  ->  Microengine ');
    writeln('                             Editor Text File Converter');
    writeln(' ');
    writeln(' ');
    write('enter name of file to convert.....................');
    readln( source_file );
    reset(diskin,source_file);
    if ioresult = 0
      then aborting := false
      else begin
             ioerror (ioresult);  (* aborts the conversion due to an abnormal *)
                                  (* return code of the disk operation       *)
             aborting := true;
             end;
    if not aborting
    then begin
           write('enter name of dest. file  ( .TEXT is assumed )      ');
           readln( destination_file );
           destination_file := concat(destination_file,'.text');
           rewrite(diskout,destination_file); (* This will create a 1024
                                                 byte header by the .TEXT
                                                 extension default       *)
           if ioresult = 0
             then aborting := false
             else begin
                    ioerror ( ioresult );  (* same as above *)
                    aborting := true;
                    end;
           end;
    clear_screen;
    end; (* init_files *)


procedure byte_ct;

  begin
    pg_byte_ct := succ ( pg_byte_ct );
    if pg_byte_ct = pgsize
    then begin
```

```
            pg_byte_ct := 0;
            writeln('*');
            end
    end; (* byte count *)



begin    (* main procedure *)

  nul := chr(00);
  soh := chr(1);
  lf  := chr(10);
  cr  := chr(13);
  dle := chr(16);
  eom := chr(25);

  pg_byte_ct := 0;
  end_of_prog := false;

  init_files ( finished );
  if not finished then
    begin
      writeln(' ');
      writeln(' ');
      writeln(' ');
      writeln('                                  8/32  ->  Microengine ');
      writeln('                                Editor Text File Converter');
      writeln(' ');
      writeln(' ');
      writeln('"converting"');
      writeln(' ');
      ch := diskin ^;
      get (diskin);
      while not finished
      do begin
        diskout ^ := dle;
        put ( diskout );
        byte_ct;
        ind_ct := 32;
        if ch = ' '
        then repeat
               ind_ct := succ ( ind_ct );
               ch := diskin ^;
               get ( diskin );
               until (ch<>' ') or eof ( diskin );
   (* assert : ch is non blank character          *)
        diskout^ := chr(ind_ct);
        put ( diskout );              (* indent code is 20h + space count *)
        byte_ct;
```

```
        while ch <> lf
        do begin
             diskout^ := ch;
             put ( diskout );
             byte_ct;
             ch := diskin ^;
             get ( diskin );
             end;
     (* assert : ch is lf ascii character          *)
        diskout ^ := cr;
        put ( diskout );
        write('.');
        byte_ct;
        ch := diskin ^;
        if ch = eom
        then begin
               ch := nul;
               diskout^ := ch;
               put (diskout);
               byte_ct;
               finished := true;
               end
        else get (diskin);
      end;      (* end not eof  *)
   end_of_prog := true;
   diskout ^ := nul;
   put (diskout);
   byte_ct;
   for i := pg_byte_ct to 1023  (* fill remaining page with nuls *)
   do begin
        diskout^ := nul;
        put (diskout);
        end;
   close(diskin,lock);
   close(diskout,lock);
  end; (* not finsihed *)
  writeln(' ');
  writeln(' ');
  write('normal termination                  ');
  write('press <CR> to return to O.S.');
  readln(ch);
  clear_screen;
end.
```

A COMPUTER COMMUNICATIONS SYSTEM

by

KIM W. JANNE

B.S., Kansas State University, 1980

AN ABSTRACT OF A MASTER'S PROJECT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

ABSTRACT

A computer communication support system has been constructed. The system facilitates the communication of information to disjoint users whose concurrent connection to a common host system is not required. Furthermore, the communication mechanisms also provide for intercomputer communication across heterogeneous media and architectures remotely operating under UCSD Pascal conventions.