and the one with the error. Using some editor, find the last message read. The first line of that message contains a date-and-time followed by a byte count indicating how many characters are in the message body starting on the following line. Skip that many characters of the message body. You should be at the date-and-time line of the next message. If there is junk there, delete it. Otherwise, try to fix the count so it is pointing at the date-and-time of the next message.

The beginning of the file does not conform to the message file format. It could be: (1) the file is not a message file -- sorry, we can't help you there. (2) It is a message file with a bad first line -- probably a blank line. Read the file with a text editor. If the second line begins with a time and date then delete the first line and reuse MSG on the new file. (3) It is a message file with a hole at the beginning. Read it with a text editor to get rid of the hole, write it out and reuse MSG.

MINCOP

MINCOP asks the user to respond to the request:

"TYPE C TO COPY, V TO VERIFY, L TO LIST ONLY."

COPY makes a copy of a MINIDUMPER format tape. VERIFY compares two MINIDUMPER tapes and complains if there are any differences. LIST ONLY simply lists the files on a MINIDUMPER format tape (which requires a read pass on the entire tape).

The user is also asked to type the tape unit numbers and the densities of the tapes involved (L for Low(200), M for Medium (556), H for High (800)).

The MINIDUMPER Format tape being read is checked for both tape errors and format errors. MINCOP will not copy arbitrary format tapes!

MTACPY

MTACPY is a program which reads 7-track magtapes in any format and writes tapes in DECsystem10 format.

MTACPY initializes with:

```
MAGTAPE UNIT NO.=
```

The user should respond 0 or 1 depending on where his tape is mounted. After moving the tape to its load point, MTACPY asks:

```
USE 556 BPI?(Y OR N):
```

An N response here causes an additional query:

```
DESIRED DENSITY (200 OR 800):
```

After the density response, MTACPY asks:

```
NORMAL ODD PARITY?(Y OR N):
```

Most tapes are odd parity, with the exception of BCD card image tapes, which are usually even parity.

The next question is:

```
TO OR FROM MAGTAPE?  (T OR F):
```

The response to this specifies which way information is to be moved. If the answer was "T", it would ask:

```
SOURCE FILE(S):
```

At this point, it accepts a list of file names separated by commas and terminated by carriage return. File names may contain "*" in the name or extension field to specify automatic iteration over all files which match the other fields.

For each file transferred, MTACPY types out the number of six-bit bytes written onto the tape. When the list of source files is exhausted, (all have been transferred), it again asks:

```
SOURCE FILE(S):
```

in order to collect another list of source files. If the user supplies an empty list by typing carriage return, the program asks:

```
DONE?(Y OR N):
```

An N response returns to ask for another list of source files. A Y response finishes up the magtape by writing ten successive end-of-file marks, then rewinding the tape.

If data were to be read from the tape, the answer to the question:

TO OR FROM MAGTAPE(T OR F):

would have been "F". Then the program asks:

TARGET FILE: and the user should respond with a writable file name. If the user responds with, for example, file name "XYZ", the program will move an image of the tape data into file XYZ. In addition, to preserve formatting information, MTACPY creates an auxiliary file with name XYZ.RECSIZ, which is an ASCII file which reports the number of six-bit bytes read from each successive record of the magtape. Programs which do later code conversion of each record make use of this auxiliary file to define record boundaries.

It is possible to skip over uninteresting files on the tape by responding with null target file names, i.e. carriage returns. Answering the question:

TARGET FILE:

with three carriage returns will cause the first three files on the tape to be skipped over, then the question is repeated.

For each file transferred, MTACPY reports the number of six-bit bytes read from the tape.

The logical end of tape is denoted by two successive end-of-file marks. On encountering this, the program asks:

LOGICAL END OF TAPE.KEEP READING?(Y OR N)

A "Y" response here causes the program to continue. An "N" causes the tape to be rewound, and the program exits.

EXAMPLE:

```
@MTACPY

MAGTAPE UNIT NO.=Ø
USE 556 BPI?(Y OR N):Y
NORMAL ODD PARITY?(Y OR N):Y
TO OR FROM MAGTAPE(T OR F):F
TARGET FILE:ABC[NEW FILE]
1600 (DECIMAL) SIX-BIT BYTES.
```

```
TARGET FILE:
LOGICAL END OF TAPE.KEEP READING?(Y OR N)N
EXIT.
^C
@
```

At this point, the file ABC.RECSIZ might contain, for example:

```
DECIMAL LENGTH OF EACH RECORD IN SIX-BIT BYTES

80      80      80      80      80      80      80      80      80      80
80      80      80      80      80      80      80      80      80      80
```

NOTIFY

Notify is a program which allows for "canned" messages. Thus often used messages can be prepared in advance of the time of need. Notify allows the composition of messages with all the editing capabilities of SNDMSG. A description of commands follows.

APPEND TO CURRENT MSG - This allows the user to append to the currently buffer message. This includes "canned" and composed messages.

COMPOSE MSG - This clears the current buffer and allows the user to type up a message.

GET CANNED MSG - Brings a "canned" message into the buffer. It will be appended to any message already there. These canned messages are kept in the file CANNED.MSGS. The file format is discussed later.

LIST CANNED MSG'S - Types a list of the current canned messages. What is listed is a summary of the actual message. The number is an index for use in the GET command.

PRINT CURRENT MSG - Prints the currently buffered message (that which will be sent) on the user's teletype.

SEND MSG - Sends the message to the desired set of terminals. Two questions will be asked here, 1) Emergency (Y or N)? This is asking how important the notice is. If answered no, those users who have REFUSE LINKS on will not receive the message. Answering yes will cause all users to get the message. 2) TTYS: This is as before, either a list of tty's or a -1 for all teletypes on the system.

The format for the CANNED.MSGS file is: MESSAGE SUMMARY<CRLF> MESSAGE TEXT<Z> ..... MESSAGE SUMMARY<CRLF> MESSAGE TEXT<Z> The text of each message can be any number of lines and contain any characters except Z (control z). The program can currently handle up to 30 canned messages.

PA1Ø/5Ø - Compatible Operation for 1Ø/5Ø CUSPS

In order to make immediate use of the large body of existing user programs written for the 1Ø/5Ø system and to accommodate future distributions of CUSP programs from DEC, TENEX provides facilities for operating in a 1Ø/5Ø compatible fashion. Under such operation, most existing user programs (including CUSPS) can be assembled without change, loaded, and run. In fact, it is possible to RUN immediately most SAVED binary DECtape files.

Limitations

There are two types of limitations on the programs to be run under 1Ø/5Ø compatibility: system call (UUO) limitations and file system limitations.

Much of the information maintained by the TENEX monitor is kept in tables completely different from the tables of the 1Ø/5Ø monitor. It is not practical to provide complete translation of the TENEX monitor tables into 1Ø/5Ø format, so the GETTAB, PEEK, SPY, LOGIN, and LOGOUT UUO's are not permitted, and trap to an error routine.

These restrictions mean that the 1Ø/5Ø Support Cusps (LOGIN, LOGOUT, REACT, FAILSAFE, MONEY, CHECKPOINT, and COMPILE) are not supported under 1Ø/5Ø compatibility. In fact, these programs are superseded by the TENEX EXEC, BSYS (file backup system), CHKPNT, and ACCT1Ø (accounting system).

The second major limitation on compatibility operation is due to the fact that the TENEX file system is different from 1Ø/5Ø; in particular, the user file directory is not available as a ordinary file (c.f. UFD in 1Ø/5Ø). Programs which rely on reading sequentially through such a file, such as PIP directory listing, are therefore not supported. PIP is, however, superseded by the TENEX EXEC.

Implementation

Since TENEX itself makes no use of the MONITOR UUO calls (opcodes 4Ø through 77) these are reserved for compatibility operation. When any program executes one of these calls, it is defined as a compatibility program and the TENEX monitor notes this in the PSB
# of that fork. The compatibility support code is then mapped from
the file <SUBSYS> PA1Ø5Ø.SAV into pages 7ØØ through 717 of that
fork's address space.

RECENT expansions of the compatibility package have made use of

two of the Terminal Interrupt channels of the PSI system. Unfortunately, the choice of channels conflicted with existing programs which share the PSI channel table with the compatibility package. To remedy this we took a survey of known users in this category to determine what channels were mutually available. As a result of this survey we changed over to using channels 3Ø and 31 (decimal) in the current compatibility package, and will define that future expansion will use the group from 3Ø through 35.

Note that this should have no effect on most programs, which are written strictly for TENEX or strictly for 1Ø/5Ø operation. The only problem is with those attempting to use the PSI system AND the compatibility package.

The initial UUO is trapped by the monitor into the second location of the entry vector specified in PA1Ø5Ø.SAV. Subsequent
UUO's are trapped into the first location of the entry vector.
During UUO calls, 4Ø is copied to the location pointed to by
entry vector 14. The return PC is stored in the location pointed
to by entry vector 15. (See SCVEC) in the JSYS Manual)

The UUO calls which are not supported give an "ILLEGAL UUO AT LOCATION" message and halt the fork. All other UUO's are simulated correctly.

Since addresses >7ØØØØØ are not accessible in most medium sized 1Ø/5Ø systems, merging compatibility code into the user's fork poses no new limitation. The compatibility code itself, of course, is reentrant and shared.

TECHNICAL INFORMATION - INSTALLATION INSTRUCTIONS

THE SOURCE FOR PA1Ø5Ø.SAV is <SOURCES>PAT.MAC. This code has a
'PHASE 7ØØØØØ' around it so it will load into the low segment
using LOADER. After completion of loading, enter DDT and start
at LINIT. This first BLT's the symbols down on top of the
program, PMAPS the previous compatibility code (pages PATORG to
777) out of existence, then BLT's the new program (plus symbols)
up to PATORG. The entry vector is declared and an updated symbol
table pointer is copied into DDT, and control returns to the
EXEC.

At this point, it is convenient to SSAV pages 7ØØ to 777 on PAT.SAV, which provides a copy of the program with symbols and DDT for debugging purposes.

To put up this new compatibility package as the current operating version, enter DDT and start at MAKEPF. This SSAVEs the code with read and execute permission only as a new version of PA1Ø5Ø.SAV. This can then be placed on <SUBSYS> as PA1Ø5Ø.SAV.

PAL1Ø

PAL1Ø offers several advantages over the antiquated PALX assembler. Some of these improvements are:

A. PA11Ø runs in 4K as compared to 7K for PALX.

B. Has a binary search on symbol table which significantly reduces processor time for any given assembly.

C. Stacks symbol table listing in four vertical columns per page, reducing listing time and size.

D. Has conditional assembly and literals.

E. Two flavors of text handling.

F. Assembler generated literals or links have been modified to flag the generated statement with a hash mark next to the object code and be counted as a link and not as an error.

PAL1Ø CODES

PSEUDO OP	CODES
*EXPUNGE	DELETE ALL SYMBOLS EXCEPT PSEUDO OPS
*FIXMRI	ARG=X000 DEFINE MEMORY REFERENCE INSTRUCTION HAVING VALUE "X000"
*FIXTAB	ESTABLISH ALL SYMBOLS CURRENTLY DEFINED AS ASSEMBLER SYMBOLS
*NOTE, PAL1Ø	INITIALIZES ITS SYMBOL TABLE BEFORE EACH ASSEMBLY I.E. FIXTAB, FIXMRI, AND EXPUNGE FUNCTIONS ARE VALID ONLY FOR THE ASSEMBLY IN WHICH THEY OCCUR
ZBLOCK ARG	GENERATE BLOCK OF ZEROES
DECIMAL	CHANGE TO RADIX 1Ø
OCTAL	CHANGE TO RADIX 8
DEFINE	DEFINE MACRO
*DTORG	GENERATE DECTAPE ORIGIN
DUBL ARG	ACCEPT DOUBLE PRECISION
ENPUNCH	ENABLE BINARY OUTPUT
NOPUNCH	DISABLE BINARY OUTPUT
FIELD ARG	GENERATE FIELD ORIGIN

CONDITIONAL ASSEMBLY PSEUDO OPS

IFDEF TAB <ARG> ASSEMBLE "ARG" IF "TAG" IS DEFINED.

```
# IFNDEF    TAG <ARG>  ASSEMBLE "ARG" IF "TAG" IS NOT DEFINED.
# IFZERO    TAG <ARG>  ASSEMBLE "ARG" IF "TAG" IS DEFINED AS ZERO.
# IFNZRO    TAG <ARG>  ASSEMBLE "ARG" IF "TAG" IS DEFINED AS NON-ZERO.
#
# PAGE       Automatic origin at beginning of next core page
# PAUSE      NO-OPERATION
# TEXT      /ARG? Generate text table.  "/" is the next delimiter.
#            If number of characters is odd, table will be
#            terminated with a 6 bit zero, if even table will
#            be terminated by a 12 bit zero.
# XLIST      Toggle listing flip flop.
#
# Literal Facility
#
# [ARG]      Defines a location (determined by assembler) on page Ø
#            which contains "ARG".
# (ARG)      Defines a location (determined by assembler) on current
#            page which contains "ARG".
#
# Operators
# ---------
# !          Logical Shift 6-bits left
# &          Logical And
# *          Integer Multiply
# <>         Special, Alpha only, text mode when not used in Macro
#            definitions or conditional assembly delimiters.
# "          Generate a word which contains the eight bit ASCII
#            equivalent of the next character.
#
# Error Indicators
# ----- ----------
# C          Illegal Character Error (this includes comment field)
# D          Illegal Redefinition
# G          Link Generated
# I          Illegal Indirect
# L          Literal error (overlap)
# M          Macro Error
# N          Number Error
# O          OP code error
# P          Phase or Multiply defined error
# Q          Questionable format?
# T          Illegal Text Character
# U          Undefined Character
# Z          Page Zero Literal or Exceeded Error
#
# COMMAND STRING SWITCHES
#            A     Advanced Magtape One File
#            B     Back Space Magtape One File
#            C     Enable "CREF" Format (Switch must be set
#                  before listing device)
#            D     Enable DECtape Origins i.e. "DTORG"
```

```
#           M     Suppress Symbol Table Output
#           N     Suppress Errors on TTY
#           T     Skip to Logical End of Tape
#           W     Rewind Magtape
#           Z     Zero DECtape Directory
#
#           In addition to the command string switches mentioned
# above, there are a group of special switches which are designed
# to tailor the assemblers symbol table to match up with other PAL
# assemblers.  These switches modify only the symbol table
# including Pseudo-op codes. Operators including and ! remain as
# their current function.
#
#                /P3 PAL III
#                /P8 MACRO 8
#                /PD PALD
#                /PS Super PAL or PAL 8
#
```

PCSAMP

A program to measure the operation of other user programs.

PCSAMP is used to take runtime statistics on another user program by sampling its program counter at fixed intervals. The sampled PC values are sorted in ascending order and written to a file. By comparing this file with a LOADER map of the user program, one can get an idea of the relative runtime spent by the user program in each of its subroutines, thus discovering bottlenecks in the program.

Upon starting PCSAMP asks:

PROGRAM TO BE SAMPLED: (answer with any RUNable SAV file)

FILE TO STORE SAMPLE PCS: (any writable ASCII file)

SAMPLING INTERVAL (MSEC): (20 to 50 is reasonable)

Then PCSAMP types

SAMPLING MAY BE FINISHED BY CNTL-Z.

and starts up the requested user program. Type input to the program in the normal fashion. If the program terminates with a HALTF, the sorting and writing out of the PC samples happens automatically. If the program instead hangs waiting for more input, type a control-Z to finish the sorting and outputting of the sample PC values.

The result is an ASCII file which may be LIST'ed or COPY'ed to LPT:.

PPL

PPL is an interactive, extensible programming language. It was designed by T. A. Standish and implemented for the PDP-10 by Standish and E. A. Taft at Harvard.

The version of PPL on TENEX consists of a typeless conversational language similar to Iverson's APL in which the Iversonian mechanics of arrays have been subtracted out, and to which data definition facilities as well as operator definition facilities have been added. PPL's conversational mechanics include the abilities to trace running programs, to set and remove breakpoints, to write programs that converse with users, and to edit the text of programs. Running time is proportional to the demand for computation, and, in particular, trivial requests are satisfied rapidly. Defined operations may be associated with unary and binary operators of the user's own choosing, including redefining the meaning of operators given in the initial state of the language.

The program below is an example of PPL programming using both the operator definition capabilities and the data definition capabilities of PPL to define a small language extension that differentiates formulas. Formulas are tree-like objects that constitute an example of structured data. The example reveals how PPL can be used to write programs that manipulate structured data and how the normal syntax for arithmetic expressions can be shared for new types of arithmetic, such as formula arithmetic. In the following transcript, comments are preceded by three dashes. Abbreviations are as follows:

```
FORM means formula
UF means Unary Formula
BF means Binary Formula
LO means Left Operand
OP means Operator
RO means Right Operand
INT means Integer
CHAR means Character
```

```
                    ---Data Definitions

$FORM = UF!BF!ATOM
$UF = [OP:CHAR,RO:FORM]
$BF = [LO:FORM,OP:CHAR,RO:FORM]
$ATOM = INT!REAL!CHAR

                    ---Operator Definitions

BINARY("+",FADD)
```

```
BINARY("-".FSUB)
UNARY("-",FMINUS)
BINARY("*",FMUL)
BINARY("/",FDIV)
BINARY("\",DERIV)
UNARY("@",SHOW)

       ---The Derivative of F with respect to X

     $DERIV(F,X)
[1]  DERIV_(IF F==ATOM THEN (IF F=X THEN 1 ELSE Ø) ELSE
       IF F==UF   THEN  UF(OP(F),RO(F)\X)     ELSE
       L_LO(F);R_RO(F);O_OP(F);
          IF O='+ THEN (L\X)+(R\X) ELSE
          IF O='- THEN (L\X)-(R\X) ELSE
          IF O='* THEN (L*R\X)+(R*L\X) ELSE
          IF O='/ THEN ((R*L\X)-(L*R\X))/(R*R)  )
          $


           ---Functions that Print Formulas

               $SHOW(F)
          [1]  FPRINT(F);""
               $


               $FPRINT(F)
          [1]  IF F==ATOM THEN PRINT(F); GOTO %Ø
          [2]  PRINT('(); IF F==UF THEN GOTO %4
          [3]  FPRINT(LO(F))
          [4]  PRINT(OP(F));FPRINT(RO(F));PRINT(')



          ---Functions that Construct Formulas

               $FADD(A,B)
          [1]  FADD_BF(A,'+,B)
               $

               $FSUB(A,B)
          [1]  FSUB_BF(A,'-,B)
               $

               $FMUL(A,B)
          [1]  FMUL_BF(A,'*,B)
               $

               $FDIV(A,B)
```

```
          [1]  FDIV_BF(A,'/,B)
               $

               $FMINUS(A)
          [1]  FMINUS_UF('-,A)
               $
```

---Examples

```
     X_'X          ---Assign character 'x to be value of variable x
     F_(X*(X-3))   ---Assign formula (X*(X-3)) to be value of F
     @F            ---Print formula F using defined formula print
                   operator @
(X*(X-3))
     F             ---Print F without using defined print operator @
[LO:X,OP:*,RO:[LO:X,OP:-,RO:3]]
     @F\X          ---Print Derivative of F with respect to X
((X*(1-Ø))+((X-3)*1))
```

(FOR USE ONLY AT BBN CAMBRIDGE SITE)

PTIP User's Information

Getting the PTIP's attention Once you have connected your terminal to the PTIP, by turning it on and switching it on-line (or by dialing a dataset and pressing "DATA"), you must get the PTIP's attention and tell it your terminal's speed. This is done by typing the Hunt Character.

The Hunt character for the PTIP is "Control Q".

After you connect your terminal and type a control Q, you should receive the immediate response "BBN RCC n.m" where n and m are the PTIP software version number. If you do not receive this response, or if you receive a garbled response, press the "BREAK" key, wait a second, and type the control Q again.

Connecting to TENEX After receiving the PTIP version number, type a Control C. This will cause the PTIP to attempt a connection to a TENEX. The PTIP message "Trying" means that a connection attempt is being started. When the connection opens, you will receive a standard TENEX greeting from one of the BBN systems. Check the message to make sure you are on the right TENEX.

If you are not on the TENEX you want, use the the TENEX "MOVE" command to move your connection to the right system. Type "MOVE ?" to see the list of systems to which you can move. It is just the list of BBN TENEXes. Example: Q BBN RCC I.7 C Trying BBN-TENEX 1.99.99, BBN-SYSTEM-B EXEC 1.98.98 @MOVE$ (TERMINAL TO HOST) C<return>
BBN-TENEX 1.97.97, BBN-SYSTEM-C EXEC 1.96.96 @

Note that the PTIP will not connect you to any site except a BBN TENEX site.

Disconnecting from TENEX There is nothing special to do to disconnect from the PTIP. After logging out of TENEX, and after a minute's delay to allow you to log back in, the connection from PTIP to TENEX will be closed. The PTIP message "Closed" will be printed, but there is no need to wait for it. After the "Closed" is printed, you are back to the beginning and should type "Control Q" to use the PTIP again.

Error messages and problems After a "MOVE" command, the PTIP will wait indefinitely for a response from the requested host. This removes the need to try connecting every few minutes after a host has crashed. If you want to give up and quit waiting, type the "BREAK" key. This will produce the message "Aborting", and a moment later the message "Connection failed, trying other hosts", followed by a connection to some other TENEX.

After a "MOVE" command, you might get a "TENEX RESTARTING, WAIT" message or a "TENEX NOT AVAILABLE" message from the TENEX system. You can break the connection in this situation, too, and in any situation before logging in, by typing "BREAK".

If you are connected to a TENEX, you may receive the message "Host reset, connection closed" if the TENEX crashes. This will usually not come immediately after the crash, but will be delayed until the restart begins or until the operator forces the PTIP to discard connections to the host.

If the PTIP crashes, you may receive the "BBN RCC n.m" message, or some garble, as if you had typed "Control Q". In this case, the PTIP has reinitialized and forgotten your connection. You should reconnect to the TENEX you were using and you will probably find your job there detached (unless, of course, the crash caused a complete TENEX restart).

Other features or their lack The PTIP has no command language, per se. The only characters of interest to the PTIP are Control Q, Control C and BREAK. Even those are of no interest to the PTIP once you have logged in to a TENEX. There is no need to double "at-sign"s as on the TIP. At the same time, there is no way to find out whether the PTIP is still there but the TENEX is not there when your terminal stops responding, as you could do on the TIP by typing "@X".

The PTIP does not do padding for terminals. Rather than using @D C E and the like on the TIP, you use TERMINAL (TYPE IS) on the TENEX to get your padding. The PTIP does not support 2741's or other non-ASCII terminals. It does support a wide range of baud rates, and can hunt to rates up through 2400 baud. This is why the new hunt character, control Q, was chosen rather than the "E" of the TIP or some other more familiar character.

Default hosts and rates When the installation of lines to the PTIP has been completed, we will set default baud rates and default hosts for users who request them. This will remove the need for typing Control Q to set a baud rate, and will cause the Control C to request a connection from the TENEX you normally use, if that TENEX is up.

RD

The RD subsystem is used to look at and manipulate the mail entries in your MESSAGE.TXT files (see also READMAIL and SNDMSG).

It is actually a version of TECO with a set of TECO macros preloaded. All commands are thus calls of these macros, or other normal TECO commands. Some knowledge of TECO is desirable if you plan to use RD.

When started it reads your MESSAGE.TXT file and, unless there are no entries, types a summary of the heading information (date, author, subject) for each entry. You may then go through them, listing and/or copying them to other files for later use.

The intention is that you will dispose of all entries in your MESSAGE.TXT file by reading and/or copying them to other files, so that you can use the MK command to clear your MESSAGE.TXT file at the end of each RD session.

Below is a typescript of a session with RD showing what is typed at startup, and the help text string (given by MH$) which describes all the commands. Underlined characters were typed by the user.

```
@RD

1941 CHARS
1  29-MAY-73 HGM at CCA-TENEX - -
2  29-MAY-73 ROBERTS   RD
3  29-MAY-73 HGM at CCA-TENEX   RSSER - 'UP' TIMING
4  29-MAY-73 PLUMMER   RD
5  29-MAY-73 PLUMMER   NETLOAD
6  29-MAY-73 PLUMMER   RSYSTAT
TYPE MH$ FOR HELP
MH$
ALL COMMANDS ARE TECO MACROS, TERMINATE WITH ESCAPE KEY
TO ABORT PRINTOUTS HIT DELETE KEY TWICE
DISPLAY COMMANDS

 MN    TYPE NEXT MESSAGE
#MJ    JUMP TO MESSAGE NUMBERED #
#MT    TYPE MESSAGE NUMBERED #
 ML    LISTS ALL MESSAGES IN BUFFER
```

```
# FILE READ COMMANDS
#
#  M@    READS IN MESSAGE.TXT AND LISTS IT (AUTOMATIC ON START)
#  MY    ASKS FOR FILE NAME-READS IN AND LISTS
# #MR    READS IN SPECIAL MSG FILES #=Ø :MESSAGE.TXT
#                                   #=1 :ACTION.MSG
#                                   #=2 :FILE.MSG
#
# FILE WRITE COMMANDS
#
# #MW    WRITES MESSAGE NUMBERED # TO NEW FILE T.MSG;T
#          EACH USE CREATES A NEW FILE (:1,;2, ..)
#
#   TO APPEND MESSAGES TO SPECIAL FILES
#
# #MS    SAVE- APPEND CURRENT MSG (THE ONE JUST READ VIA MT OR MN)
#           TO SPECIAL FILE #=1 or 2.  (OPENS, APPENDS, AND CLOSES)
#
# #MV    OPEN NEW VERSION OF SPECIAL FILES.
#                 #=1 :ACTION.MSG
#                 #=2 :FILE.MSG
#
# #MO    OPEN OLD VERSION OF FILE - #=1 : ACTION.MSG
#                                   #=2 : FILE.MSG
#        (TO OPEN OTHER FILES FOR APPENDING, USE :
#          EAFILE.NAME$    FOLLOWED BY #MA'S.)
# #MA    APPEND MESSAGE NUMBERED # TO OPEN FILE
#  EF    CLOSE FILE
#
# (NOTE: ACTION.MSG & FILE.MSG MUST EXIST FOR MO TO WORK.
#   TO START THEM, USE THE #MV COMMAND.)
#
#
# EXIT COMMAND
#
#  MK    CHECKS TO SEE IF CURRENT BUFFER IS SAME LENGTH AS
#          MESSAGE.TXT AND IF SO CLEARS (KILLS) MESSAGE.TXT AND EXITS
#          TO EXEC.  IF NOT IT LEAVES NEW MESSAGE.TXT IN BUFFER AND LISTS.
#        WATCH OUT!  THIS COMMAND IS INTENDED TO BE
#        USED AFTER WRITING OUT USEFUL MSGS SINCE IT WIPES OUT
#        MESSAGE.TXT (BUT DOES NOT RESET SIZE)
#
# *;H$
#
# @
```

READMAIL

READMAIL is a subsystem which does exactly what is implied in the name. When a user logs in and finds he has a message he then calls for READMAIL. Below is a sample of the dialogue.

```
# @readmail
# READMAIL 2C(13)
# TYPE ? FOR HELP
# *?
# Type just a CR or EOL to get your MESSAGE.TXT printed from point of
# last reading on your terminal. Use "D" to specify another date. Use
# "F" to specify another input file. Use "O" to specify another output
# file. Use "R" to specify reverse order (ie. oldest last). While
# printing, typing rubout will skip to the next message in the file if
# the output file specification is not used.
#
#
# Commands to READMAIL are a single character.
#
# Reading does not actually begin until you give the
# carriage-return command, at which point READMAIL prints on the
# output file in the specified order all messages from the input
# file received after the date.
#
# Default values (i.e. for those items you do not give commands
# for) are:
#
#      Input file: <connected-directory>MESSAGE.TXT;1
#      date: read date of file
#      output file: TTY: (your terminal)
#      forward order (oldest first)
#
# Dates may be entered in almost any reasonable form, but must
# include month, date, and year. A time may also be specified
# after the date, separated from it by a space.
```

Normal TENEX editing characters apply when typing in file names, but no editing is possible while typing dates.

RELRIM

Converts a .REL file created by MACRO, FAIL, or FORTRAN into a RIM10B self-loading paper tape containing standard check-summed blocks.

When started, RELRIM asks:

INPUT FILE:

to which the user responds appropriately. After the paper tape is punched, the program EXITS to the EXEC.

RENBR, the FORTRAN Renumbering Program

RENBR is a program written in hardware independent FORTRAN which sequentially statement numbers and/or forms cross-reference listings of FORTRAN programs read as data.

Instructions for Use

When started, RENBR will ask the user to supply the input, output, lister and scratch unit numbers and file names as well as other necessary information. If the only response to a request is a carriage return (blank line), a default answer is assumed. A negative response to any request for numeric information (except statement number increment) will cause the default values to be assumed for the present and all remaining requests the asking of which will then be suppressed. The default names of all files except the scratch file are defined by DATA statements in subroutine REUNIT. The default value of the lowest statement number is the absolute value of the statement number increment. All other default values are defined by executable statements prior to the call of subroutine REUNIT in the main program. Possible requests and their default values are as follow:

Request	Default
IS RENUMBERING DESIRED (Y OR N)	YES
IS OUTPUT TO BE IN CARD FORMAT (Y OR N)	YES
IS LISTING DESIRED (Y OR N)	YES
TITLE FOR LISTING	
INPUT UNIT NUMBER	1
INPUT FILE NAME	INPUT
LISTER UNIT NUMBER	20
LISTER FILE NAME	LIST
OUTPUT UNIT NUMBER	21
OUTPUT FILE NAME	OUTPU
SCRATCH UNIT NUMBER	22
SCRATCH FILE NAME	SPARE
INITIAL PAGE NUMBER	1
LOWEST STATEMENT NUMBER	1
STATEMENT NUMBER INCREMENT (NEGATIVE IS LEGAL)	1

The tab character will separate statement number fields from statement text in the output if the question concerning output format is answered with an "N".

Input File Limitations

The input file must have a name of 5 or fewer characters and the extension DAT, must be line blocked and must not contain internal

```
# control characters such as form feeds.  A final  form  feed  will
# cause no difficulty.
#
# The input file can contain tag characters if the FORTRAN compiler
# and  operating  system  allow  these.  If  the  compiler  and/or
# operating system do not allow tabs but the input program contains
# tabs, such tabs should first be converted by the text editor to 6
# or more blanks or else be converted to sufficient blanks to  fill
# to  the  normal  tab  stops if the left tab stop is in, or to the
# right of, column  7.  The  tab  character  is  used  in  FORTRAN
# programs  on  some  non-card  systems  to  separate the statement
# number field from the statement.
#
# RENBR detects the end of the input file by end-of-file  tests  in
# its  READ  statements.  If  the  end-of-file test feature is not
# available, the input file should be terminated by  an  extra  END
# statement which will not appear in the generated output.
#
# Restriction
# Continuation lines following  a  comment  line  are  taken  as  a
# continuation  of  the  comment  and  are  written into the output
# unchanged.  For  this  reason,  comment  lines  can  separate
# statements, but cannot appear within a single statement.
#
# Restriction
# A line with a non-blank non-tab non-digit character other than  C
# (which  would  indicate a comment) in column 1 followed by a tab,
# or by a number (formed of no more than 4 digits) and a tab, or by
# 4  characters  formed of any combination of blanks (also known as
# the space character) and/or digits  will  be  taken  as  a  legal
# FORTRAN  statement.  If the line begins a new statement, then the
# initial character will appear in the output at the start of  each
# line  of  the statement including all continuation lines (whether
# or not the character originally appeared at the  start  of  these
# continuation  lines).  Some compilers require B, D or I in column
# 1 to specify variable type.  Also, DEC PDP-10 FORTRAN allows D in
# column 1 to indicate a debugging line the compilation of which is
# optional.  In DEC FORTRAN, the use of the D at the start of lines
# which continue a debugging line is acceptable but not necessary.
#
# Restriction
# Blanks are trimmed from the right end of lines  prior  to  output
# regardless of syntax.  If alphameric strings are specified as the
# number of characters followed by the letter H and the  characters
# of  the  string,  then  lines  which  end  in  alphameric strings
# containing  terminal  blanks  will  have  these  blanks  removed.
# Therefore,  unless  the output is written onto cards, a statement
# such as
#      A=1H
#
# should instead be written as
```

```
#       A=(1H )
# or
#       A=' '
#
# Restriction
# A single statement can be continued on no more than 19 lines.
# Unless the dimensions of the storage arrays are increased, a
# single main program or routine being renumbered can contain at
# most 1000 numbered statements (or 25 less than this if a listing
# is also being made). There is no restriction on the number of
# statement number references within a single main program or
# routine.
#
# Restriction
# The END statement at the end of a program must appear on a single
# line (although the letters of the word END can be preceded by or
# be separated by blanks or tabs). The input can contain any
# number of programs or routines, each with its own END statement.
# If the end-of-file test in a read statement is not available, an
# additional END statement, after the final program or routine in
# the input, can be used to force a normal exit which includes
# printing of the table of contents.
#
# Restriction
# A line beginning a new statement must have one of the following
# formats.
#  A) A line beginning with a non-tab character other than C
#       followed by 4 blanks and/or digits followed in column 6 by a
#       blank or by a zero. It is possible for the character in
#       column 1 to be a blank or a digit of the statement number.
#
#  B) A line beginning with a tab followed by the first character
#       of the statement which cannot be a digit.
#
#  C) A line beginning with a non-tab non-blank non-digit
#       character other than C followed by a tab followed by the
#       first character of the statement.
#
#  D) A line beginning with a digit or digits of the statement
#       number followed by a tab followed by the first character of
#       the statement.
#
#  E) A line beginning with a non-tab non-blank non-digit
#       character other than C followed by the digit or digits of
#       the statement number followed by a tab followed by the first
#       character of the statement.
#
# If (BLANK) represents a blank, (TAB) represents a tab and (TEXT)
# represents the text of the statement, then the following are
# typical lines which start new statements. Of course, the text of
# the statement can itself begin with one or more blanks or tabs.
```

```
#       (BLANK)(BLANK)(BLANK)(BLANK)(BLANK)(BLANK)(TEXT)
#       (BLANK)(BLANK)(BLANK)(BLANK)(BLANK)Ø(TEXT)
#       D(BLANK)(BLANK)(BLANK)(BLANK)(BLANK)(TEXT)
#       D(BLANK)(BLANK)(BLANK)(BLANK)(BLANK)Ø(TEXT)
#       (BLANK)(BLANK)(BLANK)22(BLANK)(TEXT)
#       (BLANK)(BLANK)(BLANK)22Ø(TEXT)
#       D(BLANK)(BLANK)22(BLANK)(TEXT)
#       D(BLANK)(BLANK)22Ø(TEXT)
#       22(BLANK)(BLANK)(BLANK)(BLANK)(TEXT)
#       22(BLANK)(BLANK)(BLANK)Ø(TEXT)
#       D22(BLANK)(BLANK)(BLANK)(TEXT)
#       D22(BLANK)(BLANK)Ø(TEXT)
#       (TAB)(TEXT)
#       D(TAB)(TEXT)
#       22(TAB)(TEXT)
#       D22(TAB)(TEXT)
#
# Restriction
# A continuation line must have one of the following formats.
#  A)  A line beginning with a non-tab non-digit character other
#      than C followed by 4 blanks followed by a non-blank non-tab
#      non-zero character which is ignored. The initial character
#      can, of course, be a blank.
#
#  B)  A line beginning with a tab (or with 5 or more blanks)
#      followed by a non-zero digit which is ignored.
#
#  C)  A line beginning with a non-tab non-blank non-digit
#      character other than C followed by a tab (or by 4 or more
#      blanks) followed by a non-zero digit which is ignored.
#
#  The following are typical continuation lines.
#
#       (BLANK)(BLANK)(BLANK)(BLANK)(BLANK)2(TEXT)
#       (BLANK)(BLANK)(BLANK)(BLANK)(BLANK)A(TEXT)
#       D(BLANK)(BLANK)(BLANK)(BLANK)2(TEXT)
#       D(BLANK)(BLANK)(BLANK)(BLANK)A(TEXT)
#       (TAB)2(TEXT)
#       D(TAB)2(TEXT)
#
#            Description of the Listing Produced by RENBR
#
# RENBR can, at the user's request, produce a listing of the
# programs read as data. (IF renumbering has not been requested,
# RENBR assumes by default that a listing is to be made.) The
# listing of each separate main program or routine is begun on a
# new page. In the upper right corner of each page are printed
# both the current page number and the name of the program or
# routine. To the left of each non-comment statement is printed
# the count or sequence number of the statement within the current
# program or routine.
```