NonStop™
SYSTEMS

AXCESS
DATA COMMUNICATIONS
USERS MANUAL

TANDEM 16

TANDEM 16

AXCESS

DATA COMMUNICATIONS

USERS MANUAL

TANDEM COMPUTERS INCORPORATED
19333 Vallco Parkway
Cupertino, California 95014

Product no. T16/8044

March 15, 1979
Printed in U.S.A.

PREFACE

This manual describes two communications programs: the Communications Utility Program (CUP) and the X.25 Access Method.

CUP performs operations related to data communications lines and devices. It applies both to the EXPAND network and to access methods. The CUP commands provide the ability to trace activity on a line and to add subdevices to a line. Subdevices are required for the access methods.

The X.25 Access Method provides the ability to connect to an X.25 packet switching network. This access method requires little modification to current application programs. It is configured through SYSGEN and CUP.

TABLE OF CONTENTS

CUP

INTRODUCTION TO CUP

The Communications Utility Program (CUP) performs operations related to data communication lines, devices, and communications network environments. These operations include:

* Initiating line traces
* Changing the characteristics of communication lines or devices
* Displaying the status of network devices
* Adding a communication device to an existing line handler
* Listing the attributes of a particular line, device, or group of devices.
* Displaying line and circuit statistics
* Displaying formatted line trace information
* Enabling and disabling communication subdevices
* Displaying system related network information

CUP is designed to operate interactively or non-interactively by reading commands from a command file. Normally, it is run from the Command Interpreter program.

The CUP program resides in a file designated

```
$SYSTEM.SYSTEM.CUP
```

The command to run CUP is:

```
CUP [/ [IN <command file> ] [,OUT <list file> ] /] [<command>]
```

where

IN <command file>

specifies disc file, non-disc device, or process where CUP reads commands. CUP reads 132 byte records from <command file> until it encounters the end-of-file. Only one command is permitted per record.

If this option is omitted, the <command file> of the Command Interpreter is used. If the Command Interpreter is being used interactively, which is the usual case, this file is the home terminal.

OUT <list file>

specifies a non-disc device, process, or existing disc file where CUP directs its listing output (unless directed elsewhere by a command interpreter). If the <list file> is an unstructured disc file, each <list file> record is 132 characters.

If this option is omitted, the <list file> of the Command Interpreter is used.

<command>

is a CUP command. If <command> is included, it is executed, then CUP terminates.

Example:

```
CUP DUMP TRACE02
```

AN EXAMPLE OF INTERACTIVE USAGE

The CUP program is run through the Command Interpreter as follows:

```
:CUP
CUP - T9008D00 - (01APR79)

>
```

where ">" is the CUP prompt character.

In a normal CUP command execution cycle, CUP prompts the user for a command, the user enters a command, CUP executes the command, and the cycle repeats.

For example, the LINE command sets the default communication line name for other CUP commands:

```
>LINE $NET02
```

The EXIT command terminates CUP. The Command Interpreter then resumes:

```
>EXIT
:
```

When CUP receives a command in the <parameter string> part of the [RUN] command, it executes the specified command, then terminates. For example:

```
:CUP DOWN LINE $NET02 #TERM7
```

is equivalent to

```
:CUP
>DOWN LINE $NET2 #TERM7
>EXIT
```

AN EXAMPLE OF NON-INTERACTIVE USAGE

The CUP program is run through the Command Interpreter as follows:

```
:CUP /IN cmdfile, OUT $LP/
```

CUP commands are read from "cmdfile"; each record contains one command. All listing output is directed to the list device specified as "$LP". When an end-of-file or an EXIT command is encountered in the "cmdfile", CUP terminates and the Command Interpreter is activated.

EXIT Command

The EXIT command terminates CUP. Then, the Command Interpreter prompt appears. Control Y also terminates CUP.

FC (Fix Command)

The FC command provides the ability to edit or to repeat a command line. When this command executes, it displays the previous command line and prompts with a period. FC acepts three subcommands:

I	insert one or more characters
R	replace one or more characters
D	delete one or more characters

FC again displays the command line after the line is edited and prompts for another subcommand. FC terminates when it receives only a carriage return. The "COMINT" section of the Tandem 16 Operating Manual describes the FC command in detail.

Example:

```
:CUP
>LINE net02
 ***** ILLEGAL LINE NAME *****
>fc
 LINE net02
.     i$                (The I command inserts the dollar sign
LINE $net02              before the "n" in net2)
.        d              (The D command deletes the zero)
LINE $net2
.                       (A carriage return terminates the FC
                         command)
DEFAULT LINE IS $NET2, TYPE IS NET, LDEV 20, SYSTEM 1 \CHICAGO
>
```

Break

When the break key is pressed, the currently executing command aborts. If no command is executing, the Command Interpreter resumes control. The command interpreter PAUSE command lets CUP resume.

Control Y

Control Y terminates CUP. Then, the Command Interpreter prompt appears. The EXIT command also terminates CUP.

Exclamation Point !

The exclamation point is a comment character when it is the first character on an input line. CUP ignores the entire comment line.

CUP COMMANDS

The CUP commands can be divided into three groups: general commands, utilities, and configuration commands. This section discusses the commands in this manner and provides examples of their usage.

General Commands

Three commands provide convenient features within CUP. These commands are HELP, LISTLH, and LINE. HELP and LISTLH can be entered at any time. The former provides either a list of CUP commands or the syntax of a specified command. The latter provides a list of lines that were configured during SYSGEN. The LINE command establishes a default line name that is used in other CUP commands. The default can be changed at any time by issuing another LINE command. This command is useful when using the configuration commands to add subdevices to a line.

Utilities

DUMP, STATS, and TRACE, the utilities, provide detailed information about a line handler or a protocol.

Configuration Commands

The configuration commands provide the ability to add subdevices to a logical device. Moreover, these commands may modify certain characteristics of a line or a subdevice.

SHOW and ADD may be entered at any time. These commands display the attributes of a line or add a subdevice to a line.

ALTER, DELETE, and CLEAR only operate on lines that are first made unavailable with the DOWN command. The UP command makes a device available again.

The CONNECT command currently is used only for the NET protocol within the X.25 Access Method.

Sample Session

This example shows the output after running CUP with a command file. This file:

* sets the default line

* alters the line to add the network address

* adds three subdevices in the ITI protocol that accept the charges for an incoming call

* adds one subdevice in the NET protocol

* shows the characteristics of the line and the subdevices

* exits CUP

```
CUP - T9008D61 - (05FEB79)

>LINE $X25
DEFAULT LINE IS $X25, TYPE IS X25, LDEV  20, SYSTEM   0 \SAA

>ALTER NETADDR 311041500031
$X25   3 SYNCS,DTE,EBCDIC,NET ADDR:3110-415-00031-

>ADD #TERM1,PROTO ITI,TYPE(6,0),RECSIZE 80,ACCEPT
#TERM1 WAS ADDED

>ADD #TERM2 LIKE #TERM1
#TERM2 WAS ADDED

>ADD #TERM3 LIKE #TERM1
#TERM3 WAS ADDED

>ADD #SAB,PROTO NET,TYPE(63,3),LHLDEV 21,NEXTSYS 1,PORT 87,ADDR 31104150003234
#SAB WAS ADDED

>SHOW
$X25   3 SYNCS,DTE,EBCDIC,NET ADDR:3110-415-00031-

                            SUB-DEVICE INFORMATION
                                SUB  REC   LH  SYS
NAME       CKT PORT PROTO TYPE TYPE SIZE LDEV NUM OPTIONS  ADDRESS
#TERM1     ---    0  ITI     6    0   80  --- --- AC
#TERM2     ---    0  ITI     6    0   80  --- --- AC
#TERM3     ---    0  ITI     6    0   80  --- --- AC
#SAB       ---   87  NET    63    3  256   21   1 ---       3110-415-00032-34

>EXIT
```

The CUP commands are:

<break key>	aborts the currently executing command or returns control to the Command Interpreter.
<control y>	terminates CUP.
!	indicates that the line is to be treated as a comment.
ADD	adds a subdevice to a line handler. Options include: setting device type, record size, and protocol; assigning network addresses; and setting request or accept reverse charges for certain network protocols.
ALTER	alters the established attributes of a line or subdevice.
CLEAR	causes a connection to be severed from a subdevice.
CONNECT	causes a connection to be made to a NET protocol subdevice.
DELETE	deletes a subdevice from a line handler.
DOWN	disables a subdevice.
DUMP	formats an existing TRACE output file. Options include setting the display mode, displaying a single record at a time, and decoding control characters.
EXIT	terminates CUP.
FC	(fix command) edits and reexecutes a command line.
HELP	provides command syntax for all CUP commands.
LINE	sets the default line name for use by other commands.
LISTLH	lists the communication line handlers configured through SYSGEN.
SHOW	lists the requested information for lines or subdevices. Optional parameters include subdevice name masks.
STATS	displays the accumulated statistics for a line handler and, optionally, resets them. Optional parameters

-->

	reset the statistics to zero and specify line or circuit statistics.
TRACE	invokes a protocol trace for a specified communication line. Options include the ability to start and stop the trace, to establish the types of information to trace, and to set the trace buffer parameters
UP	enables a subdevice.

The ADD command adds a subdevice to a line. The line name may be defined previously through the LINE command or may be specified through the LINE modifier of the ADD command. The form of the ADD command is:

```
ADD <subdevice>, <attributes> [ LINE <line name> ]

where

   <subdevice>
     is the name of the subdevice to be added to the default
     line. The subdevice name must be in the form of #name.

   <attributes> are the following:

     TYPE ( <device type>,<device subtype> )
     RECSIZE <record size in bytes>
     PROTOCOL <protocol name>
     LIKE <subdevice name>
     ADDR <BCD digit string>  (15 digits maximum)
     PORT <port number>
     [NO] ACCEPT   (reverse charging)
     [NO] REQUEST  (reverse charging)
     LHLDEV <ldev> (X.25 subdevices with protocol type of NET)
     NEXTSYS <sys number> (X.25 subdevices with protocol type of
                           NET)
     [NO] PRICALL (for the DATAPAC network only)

   LINE <line name>

     determines the line on which the command operates.


Example:

   ADD #TERM05, TYPE (6,0), RECSIZE 80, PROTOCOL ITI,
                ADDR 4085551212, ACCEPT

   ADD #TERM06, LIKE #TERM05, ADDR 4089966000
```

The descriptions of the access methods in the sections describing configuration needs include suggested values for attributes.

TYPE,RECSIZE and PROTOCOL are required options. LIKE may be used to set the attributes of a new subdevice to be the same as those of an existing subdevice.

ADDR and the reverse charging attributes, ACCEPT and REQUEST, are protocol dependent and are ignored for lines that do not reconize these attributes. For lines that have these attributes, ADDR is required; however, ACCEPT and REQUEST are optional. These options default to NO ACCEPT and NO REQUEST.

Possible errors: General errors and the following specific errors.

```
INVALID PROTOCOL
INVALID TYPE/SUBTYPE
INVALID ADDRESS
<subdevice> DOES NOT EXIST
INVALID RECORD SIZE
--- REQUIRED ITEMS NOT ENTERED --- COMMAND IGNORED
INVALID SYSTEM NAME OR NUMBER
INVALID LDEV NUMBER
MISSING 'NEXTSYS' OR 'LHLDEV'
INVALID CHARACTER MODE
SYNCS MUST BE > 0 AND < 8
MUST BE DTE OR DCE
INVALID PORT NUMBER
```

The ALTER command modifies the attributes associated with a line or subdevice. The line may be defined previously through the LINE command or may be specified through the LINE modifier of the ALTER command. The form of the ALTER command is:

```
ALTER <attributes> [ <subdevice>, LINE <line name> ]

where

   <subdevice>

     is the name of the subdevice whose attributes are to be
     changed. The form of the name is #name. If a subdevice name
     is not included, CUP assumes line attributes.

   <attributes>

     All attributes that may be set with the ADD command and

     SYNCS                   number of sync characters
     NEXTSYS                 system number of the system on the
                             opposite end of the line
     COMPRESS                set the information compression bit
     MODE {EBCDIC|ASCII}     sets the character mode of an X25 type
                             line
     LEVEL2 {DTE|DCE}        establishes the X25 level 2 frames to
                             be DTE or DCE
     NETADDR <nnn>           sets the X25 line handler address

   LINE <line name>

     determines the line on which the command operates.

Example:

   ALTER #TERM34, NO ACCEPT

   ALTER LINE $LHTS, NEXTSYS 7
```

Possible errors: General errors and the following specific errors.

```
INVALID PROTOCOL
INVALID TYPE/SUBTYPE
INVALID ADDRESS
<subdevice> DOES NOT EXIST
INVALID RECORD SIZE
--- NO ITEMS WERE MODIFIED --- COMMAND IGNORED
--- REQUIRED ITEMS NOT ENTERED --- COMMAND IGNORED
```

```
INVALID SYSTEM NAME OR NUMBER
INVALID LDEV NUMBER
MISSING 'NEXTSYS' OR 'LHLDEV'
INVALID CHARACTER MODE
SYNCS MUST BE > 0 AND < 8
MUST BE DTE OR DCE
INVALID PORT NUMBER
```

The CLEAR command severs the connection to a device. The form of the CLEAR command is:

```
--------------------------------------------------------------------
|                                                                  |
|                                                                  |
|  CLEAR <subdevice> [ LINE <line name> ]                          |
|                                                                  |
|  where                                                           |
|                                                                  |
|     <subdevice>                                                  |
|                                                                  |
|       is the name of the subdevice to clear.  The subdevice name |
|       must be in the form of #name.  The subdevice must be down to |
|       be cleared.                                                |
|                                                                  |
|     LINE <line name>                                             |
|                                                                  |
|       determines the line on which the command operates.         |
|                                                                  |
|  Example:                                                        |
|                                                                  |
|     CLEAR #TERM01                                                |
|                                                                  |
--------------------------------------------------------------------
```

Possible errors: General errors only, e.g., <subdevice> NOT FOUND.

The CONNECT command establishes a connection to a device. The form of the CONNECT command is:

```
------------------------------------------------------------------------
|                                                                      |
|  CONNECT <subdevice> [, LINE <line name> ]                           |
|                                                                      |
|                                                                      |
|  where                                                               |
|                                                                      |
|     <subdevice>                                                      |
|                                                                      |
|       is the name of the subdevice to provide a connection.  This    |
|       name must be in the form of #name.                             |
|                                                                      |
|     LINE <line name>                                                 |
|                                                                      |
|       determines the line on which the command operates.             |
|                                                                      |
|  Example:                                                            |
|                                                                      |
|     CONNECT #TERM01                                                  |
|                                                                      |
|                                                                      |
------------------------------------------------------------------------
```

Possible errors: General errors only, e.g., ILLEGAL LINE NAME.

The DELETE command deletes an existing subdevice from a line. The form of the DELETE command is:

```
----------------------------------------------------------------------
|                                                                    |
|                                                                    |
|  DELETE <subdevice> [ LINE <line name> ]                           |
|                                                                    |
|  where                                                             |
|                                                                    |
|                                                                    |
|     <subdevice>                                                    |
|                                                                    |
|       is the name of a subdevice to delete.  The name must be in   |
|       the form of #name.  The subdevice must be down for the DELETE|
|       to take place.                                               |
|                                                                    |
|     LINE <line name>                                               |
|                                                                    |
|       determines the line on which the command operates.           |
|                                                                    |
|  Example:                                                          |
|                                                                    |
|     DELETE #TERM15                                                 |
|                                                                    |
|                                                                    |
----------------------------------------------------------------------
```

Possible errors: General errors and the following specific errors.

```
    <subdevice> DOES NOT EXIST
    LINE/SUB-DEVICE IN USE
```

The DOWN command disables a subdevice on the default line. The form of the DOWN command is:

```
DOWN [ ! ] <subdevice>

where

   <subdevice>

     is the name of the subdevice to disable. The subdevice name
     must be in the form of #name.

   !

     specifies that the subdevice is to be disabled immediately.

   LINE <line name>

     determines the line on which the command operates.

Example:

   DOWN ! #TERM01
```

Possible errors: General errors only, e.g., <subdevice> NOT FOUND.

The DUMP command formats and displays a previously generated TRACE output file. The DUMP command modifiers may be in any order. The form of the DUMP command is:

```
DUMP <trace file name> [, OUT <list file>, STEP, START nnn [/mmm]
                        MASK(<mask items>), PACKET(<packet items>)
                        , <display mode> ]

where

   <trace file name>

     is the partially or fully qualified file name containing the
     trace data.

   OUT <list file>

     specifies a process, unstructured disc file, or device that
     receives the output from the DUMP.  If an unstructured disc
     file is specified, the records are 132 bytes.

   STEP

     specifies that records are to be shown one at a time.  This
     option is invalid with the OUT option.  After CUP displays a
     record, it prompts the user with a question mark.  The
     responses are:

       Q          exit single step mode
       nnn        display record number nnn
       <mode>     display the current record in a different <mode>
       <return>   display next record


   START nnn [/mmm]

     specifies the record sequence number to begin the display.
     An optional ending range [/mmm] may be given.


   MASK ( <mask items> )

     invokes specific frame record selection.  The <mask items>
     are:

       OFRM   O Frames
       IFRM   I Frames
        FRM   Both O and I Frames
         L2   Level 2 Events
```

-->

L3	Level 3 Events
L4	Level 4 Events
LH	Line Handler to Network Control Process messages
ABT	Abort System Messages

PACKET (<packet items>)

invokes specific X.25 packet record selection. The <packet items> are:

CAL	Call Request/Incoming/Accept
CLR	Clear Request/Indicated/Confirmed
RST	Reset Request/Indicated/Confirmed
INT	Interrupt/Confirmed
RXX	RR/RNR/REJ
RES	Restart Request/Indicated/Confirmed
DATA	Data Packets
OTH	Other/unknown Packets

<display mode>

specifies the output format for the raw trace data. Default is OCTAL without ASCII or CONTROL. Only the first letter of the mode is required. The modes are:

BYTE	Split Octal bytes (e.g. xxx xxx)
HEX	Hexadecimal (four digits)
OCTAL	Octal word (e.g. xxxxxx)
EBCDIC	EBCDIC character set of ASCII characters
ASCII	ASCII character set
CONTROL	Decoded ASCII and EBCDIC control characters (e.g. SOH)

When a trace is initiated for a communication line, information pertaining to the line is stored in the trace data file. The DUMP command gathers this information and uses it to set certain parameters.

Possible errors: General errors and the following specific errors.

```
'OUT' NOT PERMITTED IN STEP MODE
WARNING -- OUT OF SEQ RECORD, EXPECTED <nnn> RECEIVED <nnn>
----- TRACE FILE IS EMPTY -----
```

This example shows a trace of a network line handler. The command to format this trace is

```
     DUMP TRACE2 START 0/14
```

This command displays short frames, the first 16 bytes of a record. The DUMP command can display only items that were determined with the TRACE command. The sequence shows a link request (record 0), an acknowledgement (record 9), a link complete (record 10), and an acknowledgement (record 14).

This trace also shows the events that occurred on the line. A listing of the application program is necessary to determine the meaning of these events. The octal number for each event is an address within the program.

```
$SYSTEM.SYSTEM.TRACE2  NET LINE HANDLER TRACE       02/15/79  16:45:21

TRACE OF \SAA.$LHSB  TAKEN AT 15:40:45 ON 02/15/79
TRACE PARAMETERS: FRAMES,SHORT,L2,L3,L4       0-BLOCKS        0-CYCLES

    0: 58406   I FRAME   C[4-0-4-0]  LRQ   TO: 000-000-019  FROM: 001-002-008
        000026  000401   162046  010002   000610   001400   000023   000402
        004016  120761   100003

    1: 58406  L2 EVENT   000002   STATE  %003432
        000012  001002   162046   003432   000002

    2: 58406  L2 EVENT   000003   STATE  %003444
        000012  001402   162046   003444   000003

    3: 58407  L4 EVENT   000003   STATE  %007455
        000012  002004   162047   007455   000003

    4: 58407  L4 EVENT   000002   STATE  %007467
        000012  002404   162047   007467   000002

    5: 58407  L4 EVENT   000001   STATE  %007473
        000012  003004   162047   007473   000001

    6: 58407  L4 EVENT   000001   STATE  %007515
        000012  003404   162047   007515   000001

    7: 58407  L4 EVENT   000001   STATE  %007521
        000012  004004   162047   007521   000001

    8: 58407  L4 EVENT   000001   STATE  %007525
        000012  004404   162047   007525   000001

    9: 58407   O FRAME   C[5-0-4-0]  ACK   TO: 001-002-008  FROM: 000-000-019
        000022  005000   162047  010002   000250   003001   001010   000000
        011417

   10: 58409   O FRAME   C[5-0-5-0]  LCMP  TO: 001-002-008  FROM: 000-000-019
        000026  005400   162051  010002   000252   002001   001010   000000
        011416  107063   100002

   11: 58410   I FRAME   C[5-0-0-1]
        000012  006001   162052  010002   000241

   12: 58410  L2 EVENT   000002   STATE  %003266
        000012  006402   162052   003266   000002

   13: 58410  L2 EVENT   000001   STATE  %003300
        000012  007002   162052   003300   000001

   14: 58416   I FRAME   C[6-0-5-0]  ACK   TO: 000-000-019  FROM: 001-002-008
        000022  007401   162060  010002   000712   003000   000023   000402
        004017
```

This example is a trace of an X.25 line handler with an incoming call. The command to view this trace is

```
     DUMP X25 PACKET MASK(FRM) ASCII,C START 135
```

This display shows only the frames, not events, involved with the incoming call. Outgoing frames are O FRAMES; incoming frames are I FRAMES.

```
$SYSTEM.SYSTEM.X25  X25 LINE HANDLER TRACE      02/15/79  16:02:54

TRACE OF \SAA.$X25  TAKEN AT 01:30:59 ON 01/00/75
TRACE PARAMETERS: FRAMES       0-BLOCKS       0-CYCLES

  135: 36032    I FRAME    C[4-0-3-0] 00/001 DCE INCOMING CALL 3110-613-01097-01
        000055   104001    106300   010002   001606   010001   005756   030420
        ... ...  ... ...   ... ...  DLE STX  ETX ...  DLE SOH  VT  ...    1  DLE
        040520   000061    000061   010141   030020   113401   006001   000400
         A   P   NUL  1    NUL  1   DLE  a    0  DLE  ... SOH  FF  SOH  SOH NUL
        020405   001006    004034   000035   003011   020400   000003
         !  ENQ  STX ACK   BS   FS  NUL GS   ACK HT    !  NUL  NUL ETX

  136: 36033    O FRAME    C[4-0-4-0] 00/001 DTE CALL ACCEPTED
        000015   104400    106301   010002   000610   010001   007403
        ... ...  ... ...   ... ...  DLE STX  SOH ...  DLE SOH  SI  ETX

  137: 36044    O FRAME    C[4-0-5-0] 00/001 DTE DATA  PR-0 PS-0 Q
        000030   105000    106314   010002   000612   110001   000006   001576
        ... ...  ... ...   ... ...  DLE STX  SOH ...  ... SOH  NUL ACK  ETX  ~
        002000   003425    000041   000404
        EOT NUL  BEL NAK   NUL  !   SOH EOT

  138: 36048    O FRAME    C[4-0-6-0] 00/001 DTE DATA  PR-0 PS-1
        000016   105400    106320   010002   000614   010001   001072
        ... ...  ... ...   ... ...  DLE STX  SOH ...  DLE SOH  STX  :

  143: 36192    I FRAME    C[7-0-5-0] 00/001 DCE DATA  PR-1 PS-0 Q
        000030   110001    106540   010002   001752   110001   020000   001576
        ... ...  ... ...   ... ...  DLE STX  ETX ...  ... SOH      NUL  ETX  ~
        002000   003425    000041   000404
        EOT NUL  BEL NAK   NUL  !   SOH EOT

  152: 36880    I FRAME    C[0-0-7-0] 00/001 DCE DATA  PR-2 PS-1
        000037   114401    110020   010002   001416   010001   041114   047507
        ... ...  ... ...   ... ...  DLE STX  ETX SO   DLE SOH   B   L    O   G
        047516   020123    052520   042522   027123   052520   042522   006403
         O   N        S     U   P    E   R    .   S    U   P    E   R   CR  ETX

  155: 37029    O FRAME    C[0-0-1-0] 00/001 DTE DATA  PR-2 PS-2
        000017   116000    110245   010002   000402   010001   042015   005003
        ... ...  ... ...   ... ...  DLE STX  SOH STX  DLE SOH   D  CR   LF  ETX

  156: 37040    O FRAME    C[0-0-2-0] 00/001 DTE DATA  PR-2 PS-3
        000131   116400    110260   010002   000404   010001   043107   052501
        ... ...  ... ...   ... ...  DLE STX  SOH EOT  DLE SOH   F   G    U   A
        051104   044501    047040   041517   046515   040516   042040   044516
         R   D    I   A    N         C   O    M   M    A   N   D         I   N
        052105   051120    051105   052105   051040   026440   052071   030060
         T   E    R   P     R   E    T   E    R        -        T   9    0   0
        030104   033064    020055   020050   030462   043105   041067   034451
         0   D    6   4        -        (    1   2    F   E    B   7    9   )
        026040   047520    042522   040524   044516   043440   051531   051524
         ,        O   P     E   R    A   T    I   N    G        S   Y    S   T
        042515   020040    042066   031415   005003
         E   M              D   6    3  CR   LF  ETX

  160: 37093    O FRAME    C[1-0-3-0] 00/001 DTE DATA  PR-2 PS-4
        000054   120400    110345   010002   000446   010001   044103   050125
        ... ...  ... ...   ... ...  DLE STX  SOH  &   DLE SOH   H   C    P   U
```

This example is the same trace with the HEX option included for easier decoding of the PAD messages to the TIP of the network. The command to view this trace is

```
DUMP X25 PACKET MASK(FRM) ASCII,C,HEX START 135
```

```
$SYSTEM.SYSTEM.X25  X25 LINE HANDLER TRACE       02/15/79  16:08:11

TRACE OF \SAA.$X25  TAKEN AT 01:30:59 ON 01/00/75
TRACE PARAMETERS: FRAMES      0-BLOCKS        0-CYCLES

  135: 36032   I FRAME     C[4-0-3-0] 00/001 DCE INCOMING CALL 3110-613-01097-01
        002D      8801      8CC0      1002      0386      1001      0BEE      3110
       ... ...   ... ...   ... ...   DLE STX   ETX ...   DLE SOH   VT  ...   1  DLE
        4150      0031      0031      1061      3010      9701      0C01      0100
        A   P    NUL  1    NUL  1    DLE  a     0  DLE   ... SOH   FF  SOH   SOH NUL
        2105      0206      081C      001D      0609      2100      0003
        !  ENQ   STX ACK   BS   FS   NUL GS    ACK HT     !  NUL   NUL ETX

  136: 36033   O FRAME     C[4-0-4-0] 00/001 DTE CALL ACCEPTED
        000D      8900      8CC1      1002      0188      1001      0F03
       ... ...   ... ...   ... ...   DLE STX   SOH ...   DLE SOH   SI  ETX

  137: 36044   O FRAME     C[4-0-5-0] 00/001 DTE DATA  PR-0 PS-0 Q
        0018      8A00      8CCC      1002      018A      9001      0006      037E
       ... ...   ... ...   ... ...   DLE STX   SOH ...   ... SOH   NUL ACK   ETX  _
        0400      0715      0021      0104
       EOT NUL   BEL NAK   NUL  !    SOH EOT

  138: 36048   O FRAME     C[4-0-6-0] 00/001 DTE DATA  PR-0 PS-1
        000E      8B00      8CD0      1002      018C      1001      023A
       ... ...   ... ...   ... ...   DLE STX   SOH ...   DLE SOH   STX  :

  143: 36192   I FRAME     C[7-0-5-0] 00/001 DCE DATA  PR-1 PS-0 Q
        0018      9001      8D60      1002      03EA      9001      2000      037E
       ... ...   ... ...   ... ...   DLE STX   ETX ...   ... SOH       NUL   ETX  _
        0400      0715      0021      0104
       EOT NUL   BEL NAK   NUL  !    SOH EOT

  152: 36880   I FRAME     C[0-0-7-0] 00/001 DCE DATA  PR-2 PS-1
        001F      9901      9010      1002      030E      1001      424C      4F47
       ... ...   ... ...   ... ...   DLE STX   ETX SO    DLE SOH   B   L     O   G
        4F4E      2053      5550      4552      2E53      5550      4552      0D03
        O   N         S     U   P     E   R     .   S     U   P     E   R    CR  ETX

  155: 37029   O FRAME     C[0-0-1-0] 00/001 DTE DATA  PR-2 PS-2
        000F      9C00      90A5      1002      0102      1001      440D      0A03
       ... ...   ... ...   ... ...   DLE STX   SOH STX   DLE SOH   D  CR     LF  ETX

  156: 37040   O FRAME     C[0-0-2-0] 00/001 DTE DATA  PR-2 PS-3
        0059      9D00      90B0      1002      0104      1001      4647      5541
       ... ...   ... ...   ... ...   DLE STX   SOH EOT   DLE SOH   F   G     U   A
        5244      4941      4E20      434F      4D4D      414E      4420      494E
        R   D     I   A     N         C   O     M   M     A   N     D         I   N
        5445      5250      5245      5445      5220      2D20      5439      3030
        T   E     R   P     R   E     T   E     R         -         T   9     0   0
        3044      3634      202D      2028      3132      4645      4237      3929
        0   D     6   4         -         (     1   2     F   E     B   7     9   )
        2C20      4F50      4552      4154      494E      4720      5359      5354
        ,         O   P     E   R     A   T     I   N     G         S   Y     S   T
        454D      2020      4436      330D      0A03
        E   M               D   6     3  CR    LF  ETX

  160: 37093   O FRAME     C[1-0-3-0] 00/001 DTE DATA  PR-2 PS-4
        002C      A100      90E5      1002      0126      1001      4843      5055
       ... ...   ... ...   ... ...   DLE STX   SOH  &    DLE SOH   H   C     P   U
```

The LINE command sets the internal default line name for other CUP commands.

The form of the LINE command is:

```
--------------------------------------------------------------------
|                                                                  |
|                                                                  |
|  LINE [ <line name> ]                                            |
|                                                                  |
|  where                                                           |
|                                                                  |
|     <line name>                                                  |
|                                                                  |
|        is a fully or partially qualified line name.  System name may|
|        be included.  If <line name> is not specified, the current |
|        default line name is displayed.                            |
|                                                                  |
|  Example:                                                        |
|                                                                  |
|     LINE $LHNET                                                  |
|                                                                  |
|                                                                  |
--------------------------------------------------------------------
```

Possible errors: General errors only, e.g., ILLEGAL LINE NAME.

The LISTLH command lists the communication line handlers configured through SYSGEN. After issuing this command, a default line is chosen and established with the LINE command.

```
------------------------------------------------------------
|                                                          |
|                                                          |
|    LISTLH                                                |
|                                                          |
|                                                          |
------------------------------------------------------------
```

Possible errors: General errors only.

The SHOW command formats and displays the pertinent information for a requested subdevice. The information includes the subdevice type, subtype, record size, protocol, and DTE address if appropriate.

The form of the SHOW command is:

```
SHOW [ LINE <line name>, <subdevice>, OUT <list file>, DETAIL,
       LHINFO ]

where

   LINE <line name>

     determines the line on which the command operates.

   <subdevice>

     may be a subdevice name or a subdevice name mask.  A
     subdevice name must be in the form of #name.

   OUT <list file>

     specifies a device, process, or unstructured disc file
     where the subdevice information is displayed.

   DETAIL

     provides a more detailed display for subdevices.

   LHINFO

     provides only a display of the line handler information.

 Example:

    SHOW #TERM12, DETAIL

    SHOW LINE $X25, #NET*
```

A subdevice mask is a subdevice name with an asterisk appended to the end of the name. The subdevice mask provides the facility to list subdevices with names that are the same except for one or more trailing characters. For example, the command SHOW #TERM1* displays only information for #TERM10 through #TERM19 for a line with subdevices from #TERM10 to #TERM29.

Possible errors: General errors and the following specific errors:

```
----- NO SUB-DEVICES DEFINED FOR THIS LINE -----
```

This example shows the characteristics of an X.25 line and its subdevices. The command to display this information is:

```
SHOW LINE $X25, DETAIL
```

```
$X25  3 SYNCS,DTE,EBCDIC,NET ADDR:3110-415-00031-

                             SUB-DEVICE INFORMATION
                                  SUB  REC   LH  SYS
NAME      CKT PORT PROTO TYPE TYPE SIZE LDEV NUM OPTIONS   ADDRESS
#TERM1    ---   0   ITI    6    0    80  --- --- AC
#TERM2    ---   0   ITI .  6    0    80  --- --- AC
#TERM3    ---   0   ITI    6    0    80  --- --- AC
#SAB      ---  87   NET   63    3   256   21   1 ---     3110-415-00032-34
```

The STATS command displays the accumulated statistics for a line and, optionally, resets them.

The form of the STATS command is:

```
STATS [ LINE <line name>, OUT <list file>, RESET, REPEAT, SYSTEM,
      SUBDEV ]

where

   LINE <line name>

     determines the line on which the command operates.

   OUT <list file>

     specifies a device, process, or unstructured disc file where
     the statistics are displayed.

   RESET

     resets the <line name> statistics to zero after displaying
     the current statistics.are displayed.

   REPEAT

     provides the ability to reexecute the STATS command as first
     entered.  CUP prompts with a question mark.  The responses to
     the prompt are:

       <carriage return>  - reexecute command
                       Q  - exit STATS command

   SYSTEM

     displays only network system statistics.

   SUBDEV

     displays only subdevice statistcs.

Example:

   STATS $NET01, RESET

   STATS LINE $X25, SUBDEV
```

Possible errors: General errors only, e.g., ILLEGAL LINE NAME.

This example shows one display from an X.25 line and another from a network line. The first two lines of the display are the same for both lines. The level 3 messages apply only to X.25 lines. Level 4 messages apply only to network lines.

LEVEL 1

BCC	is the Block Check Character Error.
NO BUFFER	is the number of times a frame cannot be read because buffer space is insufficient. If this situation occurs frequently, the dedicated buffer space determined at SYSGEN should be increased.
LINE QUALITY	should be between 90 and 100. If this number falls below 90, transmission is difficult.
BUFFER USAGE	reflects the IOPOOL usage.

LEVEL 2

I FRAMES	Information frames (data, control)
S FRAMES	Sequenced frames (Receive Ready, Reject)
U FRAMES	Unsequenced frames (Disconnect, Command Reject)

```
$X25  X25 LINE STATISTICS AT \SAA   01/00/75  00:43:13

------------  LEVEL 1  ---------------
  BCC       NO      LINE      BUFFER USAGE
ERROR   BUFFER  QUALITY  CURRENT      MAX
    0        0      100        0        0

------------  LEVEL 2  -----------
        I FRAMES  S FRAMES  U FRAMES
SENT        33       600        5
RCVD        31       580        4

------  LEVEL 3  PACKETS  ------
  SENT  RCVD  INVALID  DISCARDED
    33    31        0          0

--------------------------------- LEVEL 3 ---------------------------------
          CALLS          CLEAR          RESET        INTERRUPT
        R/I   A/C     R/I  CONF     R/I  CONF     INT   CONF  DATA    REJ    RR   RNR
SENT      0     2       2     0       0     0       0      0    20      0     6     0
RCVD      3     0       0     2       0     0       0      0     6      0    18     0
```

LEVEL 3 PACKETS

SENT RCVD	number of packets sent and received on the line.
INVALID DISCARDED	number of bad incoming packets. If these numbers are significant, the problem can be traced.

LEVEL 3

This segment shows the number and the type of packets being transmitted on the line.

CALLS	R/I	Request/Indication	DATA	Data
	A/C	Accept/Connected	REJ	Reject
CLEAR	R/I	Request/Indication		
	CONF	Confirm	RR	Receive Ready
RESET	R/I	Request/Indication	RNR	Receive Not Ready
	CONF	Confirm		
INTERRUPT	INT	Interrupt		
	CONF	Confirm		

SUBDEVICE

This information shows the activity on each subdevice. It gives the circuit (CKT) number if the subdevice is connected. N/C indicates a subdevice that is not connected.

```
$X25  X25 LINE STATISTICS AT \SAA   01/00/75   01:01:17

------------  LEVEL 1  ----------------
 BCC      NO      LINE     BUFFER USAGE
ERROR   BUFFER  QUALITY   CURRENT    MAX
  0        0      100        0        0

------------  LEVEL 2  -----------
        I FRAMES  S FRAMES  U FRAMES
SENT       187       932        5
RCVD       177       992        4

------  LEVEL 3  PACKETS  ------
  SENT  RCVD  INVALID  DISCARDED
   176   174     0         0

---------------------------------- LEVEL 3 -------------------------------
          CALLS       CLEAR       RESET     INTERRUPT
        R/I   A/C   R/I  CONF   R/I  CONF   INT    CONF  DATA   REJ   RR   RNR
SENT     0     2     2    0      0    0      0      0    136     0    33    0
RCVD     3     0     0    2      0    0      0      0     33     0   134    0

SUB DEVICE      CKT     SENT    RCVD
#TERM1           1      124      29
#TERM2          N/C      -        -
#TERM3          N/C      -        -
```

LEVEL 4

MESSAGE SIZE IN BYTES

indicates the number of application messages sent. These numbers are independent of the packet size.

PACKETS	indicates the number of packets transmitted.
FORWARDS	indicates forwarding a packet from one system to another within the network.
LINKS	indicates the number of message system links.

LEVEL 4

This information shows the number and type of messages occurring on the line.

CONN	Connect	CAN	Cancel
TRACE		ACK	Acknowledged
NCPM	Network Control Process Messages	NAK	Not Acknowledged
LRQ	Link Request	ENQ	Enquiry
LCMP	Link Complete		

SYSTEM

This information shows the communication occurring with each system in the network.

```
$LHSB  NET LINE STATISTICS AT \SAA   02/15/79  15:50:27

------------ LEVEL 1 ----------------
 BCC     NO     LINE    BUFFER USAGE
ERROR  BUFFER  QUALITY  CURRENT   MAX
  0       0     100        0      100

------------ LEVEL 2 -----------
      I FRAMES  S FRAMES  U FRAMES
SENT     71        42        97
RCVD     72        41         1

-------------------------- LEVEL 4 ------------------------------------
      -- MESSAGE SIZE IN BYTES --                      PACKETS  FORWARDS  LINKS
  <64  <128  <256  <512 <1024 <2048 <4096 >4096   SENT     71         3      0
   54     6     0     0     0     0     0     0   RCVD     72         4     30

-------------------------- LEVEL 4 --------------------------
      CONN  TRACE    NCPM    LRQ   LCMP    CAN    ACK    NAK    ENQ
SENT     2      0       5      0     30      0     30      0      0
RCVD     3      0       6     30      0      0     30      0      0


                   CONTROL       DATA         LINKS
SYSTEM     NUM    SENT  RCVD   SENT  RCVD   SENT  RCVD
\SAB         1      30    30     30    30      0    30
```

The TRACE command initiates a protocol trace for a specified communication line. Once the trace has started, it may be stopped or suspended.

The form of the TRACE command is:

```
--------------------------------------------------------------------
|                                                                  |
|                                                                  |
|  TRACE TO <trace data file> [, LINE <line name>, STOP,           |
|                          MASK ( <mask items> ), CYCLES <# of cycles>, |
|                          BLOCKS <number of trace blocks> ]       |
|                                                                  |
|  where                                                           |
|                                                                  |
|     LINE <line name>                                             |
|                                                                  |
|       determines the line on which the command operates.         |
|                                                                  |
|     TO <trace data file>                                         |
|                                                                  |
|       specifies the name of the data file where trace information |
|       is stored.  If the file exists, it is replaced.            |
|                                                                  |
|     STOP                                                         |
|                                                                  |
|       terminates the current trace operation for <line name>.    |
|       The trace data file is closed.                             |
|                                                                  |
|     MASK ( <mask items> )                                        |
|                                                                  |
|       provides a selective information trace.  If MASK is not    |
|       specified, FRM, L2, L3, and L4 are traced.  The <mask items> |
|       are:                                                       |
|             FRM    O Frames/I Frames                             |
|             L2     Level 2 Events                                |
|             L3     Level 3 Events                                |
|             L4     Level 4 Events                                |
|             SF     Short Trace (first 16 bytes of trace data)    |
|                                                                  |
|                                                                  |
|     BLOCKS <number of trace blocks>                              |
|                                                                  |
|       specifies the number of blocks to be written to the trace  |
|       data file.  When <number of trace blocks> is written to the |
|       trace file, the next trace begins at the beginning of the  |
|       trace file.  The default is zero.                          |
|                                                                  |
|                                                                  |
|                                                                  |
|                                                                  |
|                                                                  |
|                                                              --> |
--------------------------------------------------------------------
```

CYCLES <# of cycles>

specifies the maximum number of times <number of trace blocks> is written to the trace data file. This feature provides a quick snapshot capability for tracing. The default is zero.

Example:

```
TRACE TO TRACE01, MASK(L2,FRM)

TRACE LINE $NET01, STOP
```

Possible errors: General errors and the following specific errors.

```
CANNOT FIND LOGICAL DEVICE IN THIS SYSTEM
-NEWPROCESS- UNDEFINED EXTERNALS
-NEWPROCESS- NO PCB AVAILABLE
-NEWPROCESS- FILE ERROR ### OCCURRED
-NEWPROCESS- UNABLE TO ALLOCATE MAP
-NEWPROCESS- NO VIRTUAL SPACE AVAILABLE
-NEWPROCESS- ILLEGAL FILE FORMAT
-NEWPROCESS- UNLICENSED PRIVILEGED PROGRAM
-NEWPROCESS- UNABLE TO COMMUNICATE WITH SYSTEM MONITOR
-NEWPROCESS- CREATION ERROR <n>, (<n>)
FILE ERROR <n> OCCURRED, ** TRACE ABORTED **
<process name> DOES NOT EXIST
ERROR <n> OCCURRED DURING TRACE DIRECTIVE
```

The UP command enables a subdevice.

The form of the UP command is:

```
UP <subdevice> [ LINE <line name> ]

where

   <subdevice>

     is the name of the subdevice to make ready.  The subdevice
     name must be in the form of #name.

   LINE <line name>

     determines the line on which the command operates.

Example:

   UP #TERM16
```

Possible errors: General errors only, e.g., ILLEGAL LINE NAME.

ERROR MESSAGES

<subdevice> DOES NOT EXIST
specified subdevice not on default or specified line.

ALL PATHS TO SYSTEM ARE DOWN
fatal error

CANNOT FIND LOGICAL DEVICE IN THIS SYSTEM
attempted to perform a remote trace, i.e., in another system in the network.

CANNOT OPEN FILE <file name>, ERR <n>
file error <n> occurred, e.g., 12 (file in use).

CAN'T OPEN 'OUT' FILE, ERR: <n>
file error <n> prohibits opening the specified file.

EDITREADINIT ERROR -n ON COMMAND FILE
unrecoverable error occurred when reading commands from a command file. Error -2, read error; error -3, text file format error; error -4, sequence error.

FILE ERROR <n> OCCURRED

FILE ERROR <n> OCCURRED, ** TRACE ABORTED **
file error <n> prohibits the trace from continuing.

***** ILLEGAL LINE NAME *****
probably no default line specified.

ILLEGAL LINE TYPE -- <line type>
<line type> is n,(%n) and n is the type specified in SYSGEN. CUP cannot communicate with this type. (%n) is the same number in octal.

-->

IMPROPER FORMAT FOR A SUB-DEVICE NAME
number sign is missing, name is too long, or name has special characters, e.g., ampersand.

IN AND OUT MUST BE THE SAME TERMINAL
the IN <file> specified a different terminal than the OUT <file>.

INVALID ADDRESS
incorrect number of digits, alphabetic characters found, or special characters found.

INVALID CHARACTER MODE
must be either ASCII or EBCDIC.

INVALID LDEV NUMBER
self-explanatory

INVALID OPERATION ATTEMPTED
self-explanatory

INVALID PORT NUMBER
must be a number from 0 to 99.

INVALID PROTOCOL
self-explanatory

INVALID RECORD SIZE
specified number out of range, maximum system size is 4072.

INVALID SYSTEM NAME OR NUMBER
system name mistyped or system number not in network.

INVALID TYPE/SUBTYPE
specified incorrect values for the protocol.

KEYWORD NOT VALID FOR THIS COMMAND
a known keyword is specified for the improper command.

LINE/DEVICE IS DOWN
self-explanatory

LINE HANDLER ABORT
fatal error

LINE HANDLER ERROR
fatal error

-->

LINE/SUB-DEVICE DOES NOT EXIST
self-explanatory

LINE/SUB-DEVICE IN USE
self-explanatory

LINE/SUB-DEVICE NOT FOUND
self-explanatory

MISSING 'NEXTSYS' OR 'LHLDEV'
required parameters for the NET protocol in the X.25 Access Method.

MUST BE DTE OR DCE
only options available for X.25 level 2 frames.

NETWORK ABORT
fatal error

NETWORK ERROR
fatal error

NETWORK PROTOCOL ERROR
fatal error

-NEWPROCESS- CREATION ERROR <n>, (<%n>)
-NEWPROCESS- FILE ERROR <nnn> OCCURRED
-NEWPROCESS- ILLEGAL FILE FORMAT
-NEWPROCESS- NO PCB AVAILABLE
-NEWPROCESS- NO VIRTUAL SPACE AVAILABLE
-NEWPROCESS- UNABLE TO ALLOCATE MAP
-NEWPROCESS- UNABLE TO COMMUNICATE WITH SYSTEM MONITOR
-NEWPROCESS- UNDEFINED EXTERNALS
-NEWPROCESS- UNLICENSED PRIVILEGED PROGRAM
all the -NEWPROCESS- errors result from a TRACE command. These errors are explained in the _Guardian Operating System Programming Manual_, pages 3.2-18 to 3.2-20.

--- NO ITEMS WERE MODIFIED --- COMMAND IGNORED
the ALTER command specified identical modifiers.

NO MORE DEVICES
number of devices requested exceeds the number available.

----- NO SUB-DEVICES DEFINED FOR THIS LINE -----
only occurs with a SHOW command.

NO SUCH SYSTEM
specified a system not in the network.

-->

```
'OUT' NAME IS INVALID
   either the name is incorrect or it is not a process,
   unstructured disc file or device.

'OUT' NOT PERMITTED IN STEP MODE
   STEP is used interactively only.

--- REQUIRED ITEMS NOT ENTERED --- COMMAND IGNORED
   PROTOCOL, RECSIZE, or TYPE not specified in the ADD command.

SUB-DEVICE ALREADY EXISTS
   attempt to add an existing subdevice.

SYNCS MUST BE > 0 AND < 8
   number of syncs out of range.

----- TRACE FILE IS EMPTY -----
   the line to be traced was down when the trace started or the
   trace was stopped before any activity occurred.

UNKNOWN KEYWORD
   keyword is not recognized as a known modifier.

WARNING -- OUT OF SEQ RECORD, EXPECTED <nnn>  RECEIVED <nnn>
   events happened so quickly, the trace process could not fetch
   records from the buffer fast enough.
```

COMMAND SYNTAX SUMMARY

```
ADD <subdevice>, <attributes> [ LINE <line name> ]

ALTER <attributes> [ <subdevice>, LINE <line name> ]

CLEAR <subdevice> [ LINE <line name> ]

CONNECT <subdevice> [, LINE <line name> ]

DELETE <subdevice> [ LINE <line name> ]

DOWN [ ! ] <subdevice>

DUMP <trace file name> [, OUT <list file>, STEP, START nnn [/mmm]
                       MASK(<mask items>), PACKET(<packet items>),
                       <display mode> ]

EXIT

HELP

LINE [ <line name> ]

LISTLH

SHOW [ LINE <line name>, <subdevice>, OUT <list file>, DETAIL,
       LHINFO]

STATS [ LINE <line name>, OUT <list file>, RESET, REPEAT, SYSTEM,
       SUBDEV ]

TRACE TO <trace data file> [, LINE <line name>, STOP,
                     MASK ( <mask items> ), CYCLES <# of cycles>,
                     BLOCKS <number of trace blocks> ]

UP <subdevice> [ LINE <line name> ]
```

X.25 AXCESS
METHOD

X.25 ACCESS METHOD

The X.25 Access Method available on the Tandem system provides the ability to connect to an X.25 packet switching network. This network is accessed through the X.25 Access Method which handles the required procedures without special considerations in the application program. With the X.25 Access Method, a Tandem system may accept calls from anywhere in the network or may initiate calls to anywhere in the network.

X.25 PACKET SWITCHING NETWORKS

X.25 is a standard for public packet switching networks. In order to implement a universal packet switching network, a common protocol is necessary among network carriers and network users. X.25 defines the conventions necessary for a host computer to establish, maintain, and clear calls with a node of a public packet switching network. It also defines the protocols required to transmit data between a host computer and a network node. These protocols require user data to be broken into small segments of data called packets.

A packet switching network never physically links both ends of a communications path as does a leased line or a circuit switched network. Data transfer is performed through virtual circuits, logical associations made within an X.25 network between sending and receiving stations. Virtual circuits permit multiplexing of many applications across a single interface when different messages have different destinations.

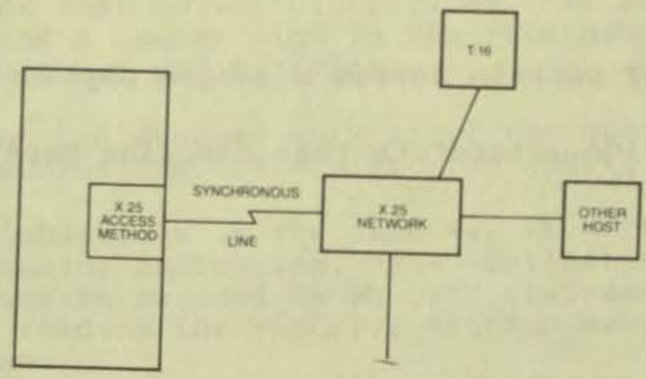


Figure 1 Sample X.25 interface

Packets have a specific format for data and control information. Each packet has a packet header that contains control information and the address to which the packet is to be delivered.

A logical channel identifier (LCI) locally designates each virtual circuit. This LCI is assigned locally when a call is established or when the network subscription is established. The LCI is a combination of two numbers: the logical channel group number and the logical channel number. This information, carried in every packet header, lets the network accept streams of packets from many sources and dynamically interleave these packets.

Figure 2 illustrates the interleaving of packets from three subdevices across a single line. The subdevices are part of the X.25 Access Method. These subdevices are associated through a virtual circuit with a terminal or other device connected to the X.25 network. A number within the packet header is assigned to this temporary association. Thus, the appropriate subdevice receives the proper packet.

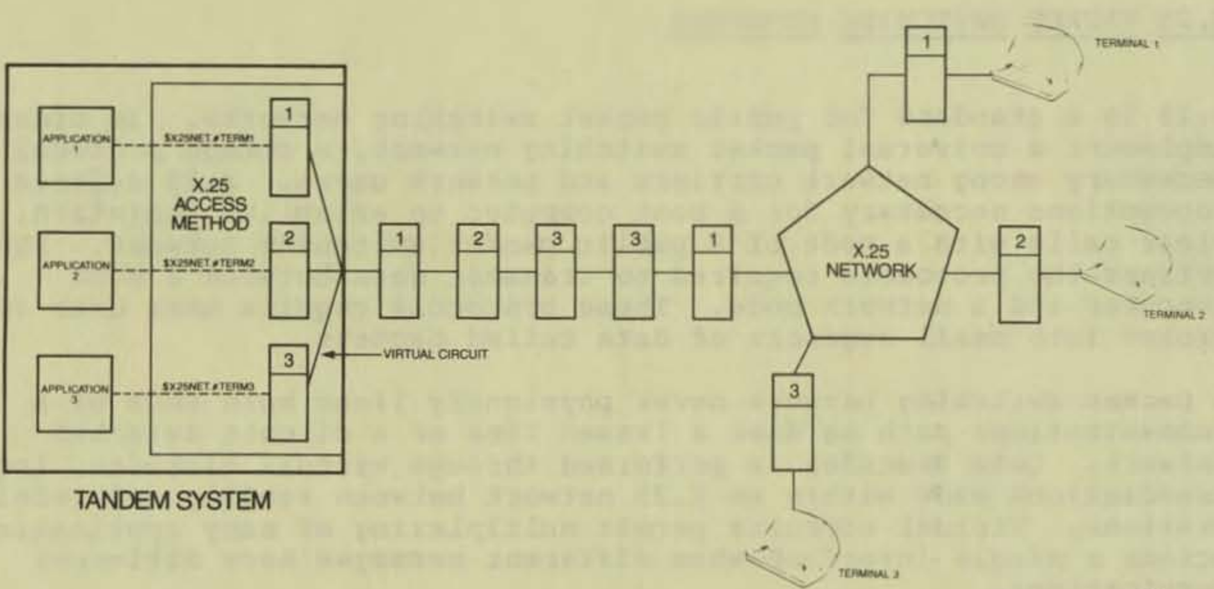


Figure 2 Interleaving packets across a single physical line.

Network carriers provide literature that describe packet switching networks in more detail.

INTRODUCTION OF SUBDEVICE CONCEPT AND FILE SYSTEM INTERFACE

For each X.25 Access Method configured, only one physical connection exists between the system and an X.25 network TIP. This physical connection is a synchronous line that is configured as a logical device, for example, $X25NET. This device is split into separate logical subdevices.

Subdevices are part of the concept that allows multiple logical devices on a single physical line. Subdevices also have attributes like any logical device, such as type, subtype, and record size. Moreover, subdevices are not determined through SYSGEN. Subdevices can be added to a line or altered while the system is up and running.

The subdevice also is an integral part of the file name. The twelve word file name consists of the device name of the physical line and the subdevice name. The subdevice name is a number sign followed by up to seven alphanumeric characters, for example, #TERM06.

```
    Filename [0:3] = "$X25NET "
             [4:7] = "#TERM06 "
             [8:11]= "dontcare"
```

When starting this subdevice by running COMINT, the command is:

```
    COMINT /IN $X25NET.#TERM06, OUT $X25NET.#TERM06/
```

It is the subdevice that the application opens and closes. Since subdevices are part of filenames, file management procedures function for subdevices as they do for other files. If an application program does not check for a number sign in the file name, the program should be able to run without modification.

An OPEN statement can succeed only after the subdevice is defined through the Communication Utility Program (CUP).

OPEN works for subdevices in the same way it works for terminals except for the naming convention. The application program can determine the name to be used in an OPEN statement in three ways: calling MYTERM, reading the $RECEIVE startup message, or asking the user for the name.

If the program calls MYTERM, it can use the name of the home terminal in the OPEN statement.

The application program could read the $RECEIVE startup message it receives from COMINT. Within this message is the name of the subdevice to open. The $RECEIVE message contains the names of the IN and OUT files. These names default to the names used with the COMINT program which may be the home terminal.

The program also may request the user to enter a file name and call FNAMEEXPAND to convert the name into the internal format. For example,

```
     ENTER DEVICE:  $X25NET.#PTP2
```

FNAMECOLLAPSE reverses this process so that the file name may be displayed in the external form. Both of these statements properly handle subdevices in a way similar to the handling of process file subnames.

The DEVICEINFO and FILEINFO statements also properly handle subdevices. These statements return the type, subtype, and record size that are configured for the subdevice.

DESCRIPTION OF CALL SETUP PARAMETERS

Protocols within the X.25 Access Method may initiate or receive calls. However, since charges are associated with calls, either the initiator or the recipient must accept the charges. Several parameters, available through CUP and SETMODE, determine who pays.

Figure 2 depicts the X.25 Access Method, the network, and possible users. The terminal user calls to use the Tandem system. A subdevice, configured through CUP, accepts the charges for the call. Another subdevice might initiate a call to a host system. This subdevice might request the other system to accept the charges.

The CUP and SETMODE 32 parameters that control the charges are:

ACCEPT	accept the charges for an incoming call
NO ACCEPT	do not accept charges for the collect call
REQUEST	ask the recipient of the call to accept the charges
NO REQUEST	accept the charges for the outgoing call

To connect a circuit to the appropriate subdevice, X.25 Access Method requires a port number for identification. The default port number for an incoming call packet is zero. If the port number for a series of subdevices is zero, the circuit is connected to the next available subdevice. However, if the port number is greater than zero, the port number of the subdevice and the port number in the address of the incoming packet must match exactly. The port number may be set through CUP or through SETMODE 32.

Another parameter for priority calls is available for users of the DATAPAC packet switching networks. The parameter is set through CUP or SETMODE 32. DATAPAC requires a priority call for any international calls. Priority calls within the DATAPAC network receive faster service. This parameter also restricts packet size to 128 bytes.

Processing Sequence in Accepting an INCOMING CALL

When the X.25 Access Method receives an incoming call packet, it follows this processing sequence to determine the subdevice to be connected to the circuit.

1) Verification of the INCOMING CALL packet and its call parameters.

2) Search through the set of subdevices in the order displayed by CUP for:

   * a match of the port number contained in the CALLED DTE ADDRESS of the INCOMING CALL packet and the port number currently defined for the subdevice.

   * the subdevice waiting for a call, i.e., an application which has the subdevice open has issued a CONTROL(filenum,11) "wait for modem connect" request.

   The subdevice either accepts reverse charges or the INCOMING CALL does not request reverse charging.

3) If all the above are satisfied, the Access Method gives the INCOMING CALL packet to the level 4 protocol handling the selected subdevice in order to complete the CONTROL request.

   Special note for NET subdevice receiving a call: the CALLING DTE ADDRESS contained in the INCOMING CALL packet is compared to that configured for the subdevice. If they do not match, the call is NOT accepted. This check is done to insure that only the proper DTEs are connected in any given EXPAND network.

4) When the appropriate subdevice is found, the circuit and subdevice are connected until cleared. The clear may be through CONTROL 12 or the CUP commands CLEAR or DOWN.

INTERACTIVE TERMINAL INTERFACE PROTOCOL

Interactive Terminal Interface (ITI) Protocol is a level 4 protocol in the X.25 Access Method. ITI provides the interface to allow terminals on the X.25 network to use a Tandem system interactively. ITI is designed to let a terminal communicate with an application program as if it is connected directly to the computer. It shields the user from most of the elaborate protocol necessary to use the X.25 network.

To use ITI, the remote user simply calls a TIP, a local terminal interface process of the X.25 network, and begins work at a terminal. In the simplest case, the application program (for example, COMINT) is unaware of ITI. However, at times the use of ITI imposes a few limitations.

Most of the features provided by the terminal process of the Guardian operating system also are available through ITI. The available features are:

* Break ownership
* SETMODE interrupt characters (only ASCII control and DEL)
* Forms and space control
* ASCII character set
* Page mode

However, some features cannot be offered because X.25 networks are designed to provide a terminal independent interface to the host system. ITI cannot provide:

* Checksum following ETX
* Programmatic setting of baud rate, parity, and character size
* Pseudo polling
* Multipoint operation (ENVOY)

* Termination of a read on character count

Details on all of the features provided by the terminal process are found in section 2.6 of the Guardian Operating System Programming Manual.

FILE MANAGEMENT SYSTEM INTERFACE

READ, WRITE and WRITEREAD can occur only after the application program establishes a circuit through a CONTROL request.

ITI assembles incoming data packets to form the read buffer to be returned to an application program. It holds these data packets until it receives a read request from the application. Any packets that cannot fit in the allocated buffer space are discarded.

A read always completes when the buffer is full. However, because the X.25 network requires a signal before transmitting packets, termination of a read on character count is not supported. The network has no way of knowing when a user types the required number of characters. For conversational terminals, the read completes when the last character is either a carriage return (CR) or a control Y (End-Of-File). For page mode terminals, the read completes on an end of text (ETX). A read also terminates on other ASCII control characters selected by user SETMODE calls.

When an application program issues a WRITE request, ITI puts the write buffer into one or more data packets. The "more data" bit is set in all but the last packet. For conversational terminals, ITI also inserts the appropriate carriage return and line feed codes according to the SETMODE functions.

ITI handles interrupt packets the same way that a terminal process handles BREAK for interactive terminals. An interrrupt packet is the result sent from a terminal when a user presses the break key.

ITI processes all of the READ, WRITE, and WRITEREAD requests as it receives them. The first request that ITI receives is the first request it completes.

SETMODE

ITI supports eight functions of SETMODE. These functions are described in "Summary of File Management Procedures," section 2.6. The SETMODE functions are:

6	set spacing control
8	select page mode
9	set interrupt characters (ASCII control and DEL)

11	set break ownership
12	set terminal access mode
27	set pre-space
28	initialize to default values
32	set call setup parameters

ITI does not recognize the other SETMODE functions that apply to terminals. It returns an "invalid operation" message if the application program should specify one of these functions. The X.25 network provides functions such as automatic line feed and echo through commands at the terminal interface.

CONTROL

ITI supports three CONTROL requests. The CONTROL requests are described in "Summary of File Management Procedures," section 2.6. These requests are:

1	forms control
11	wait for modem connect, completes when virtual circuit is established
12	disconnect modem, clears the virtual circuit

PROCESS TO PROCESS PROTOCOL

The Process to Process (PTP) protocol is another level 4 protocol in the X.25 Access Method. It provides a half duplex or full duplex communications path from an application to an application in another system connected as a DTE to the X.25 network. PTP provides the ability to initiate and to receive calls throughout the network. It gives the programmer complete control over the contents of a data packet. PTP has two modes of operation: normal and packet.

In the normal mode, PTP performs the packet assembly/disassembly (PAD). It sends write buffers in several packets and sets the "more data" bit in all but the last packet. Moreover, it assembles multiple packet messages into the application read buffer. However, PTP does not insert carriage return/line feeds, nor does it handle interrupt packets on input.

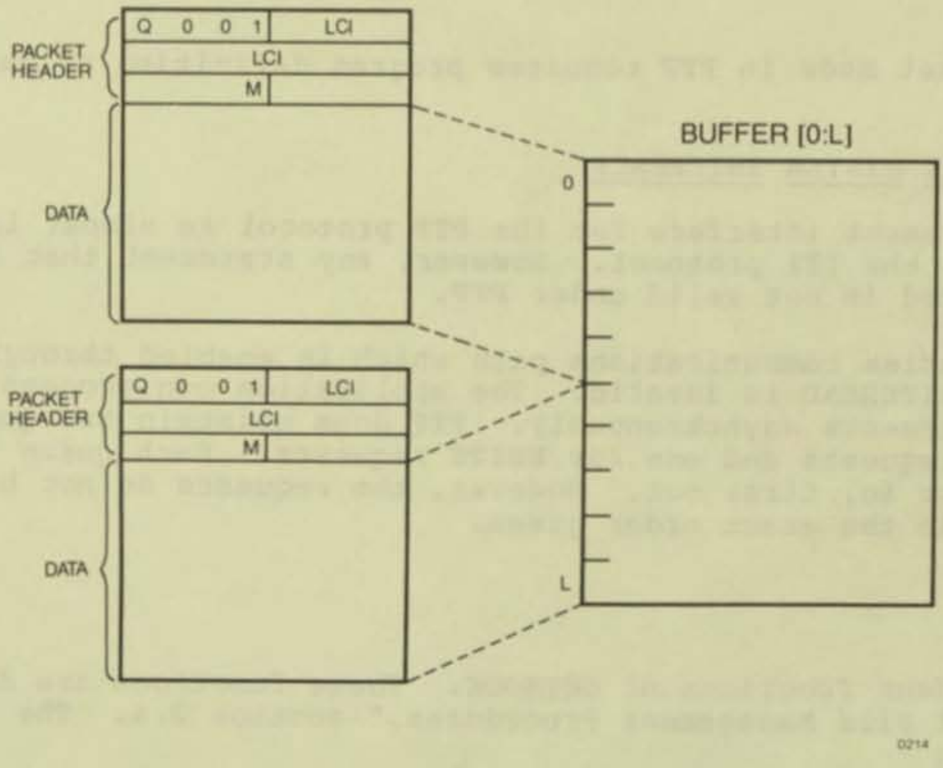


Figure 3 Normal mode in PTP automatically assembles packets.

In the packet mode, PTP and an application program communicate at the packet level by using a message control word (MCW) at the beginning of each buffer. The MCW indicates whether more data follows or whether the Q bit should be set. It also may contain a PAD control message or any other interesting data.

```
MCW.<14> = 1 => set Q bit; PAD control packet
MCW.<15> = 1 => set M bit; more data follows
```

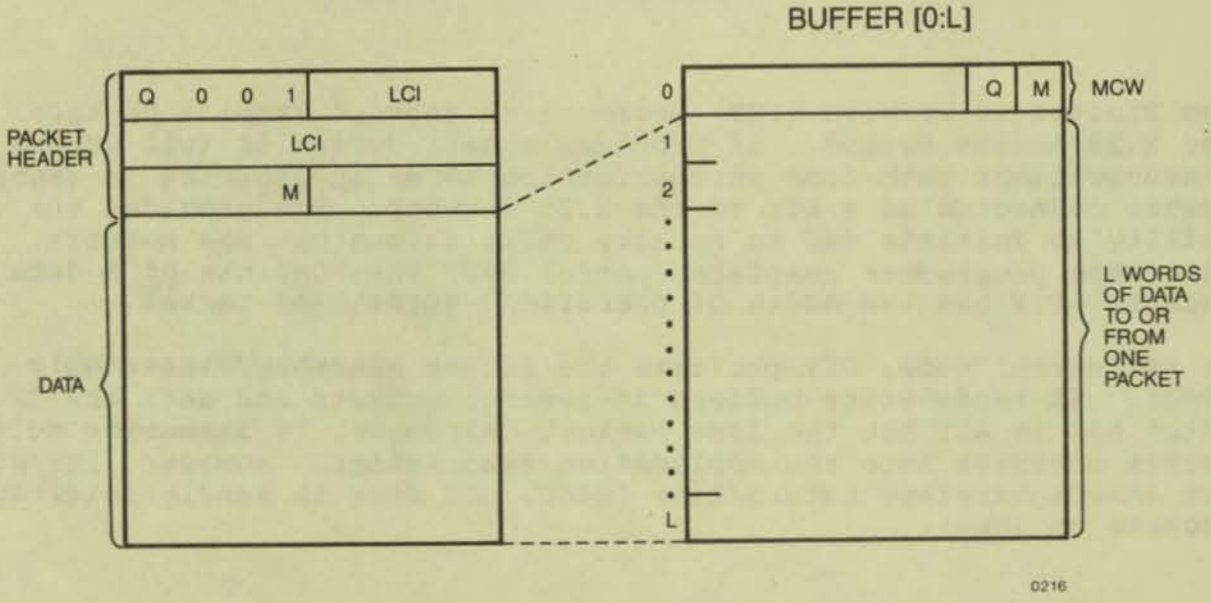


Figure 4 Packet mode in PTP requires program definition of packets.

FILE MANAGEMENT SYSTEM INTERFACE

The file management interface for the PTP protocol is almost identical to the one for the ITI protocol. However, any statement that is terminal related is not valid under PTP.

With a full duplex communications path which is enabled through SETMODE 30, WRITEREAD is invalid. The application can process READ and WRITE statements asynchronously. PTP does maintain two queues: one for READ requests and one for WRITE requests. Each queue is processed first in, first out. However, the requests do not have to be processed in the exact order given.

SETMODE

PTP supports four functions of SETMODE. These functions are described in "Summary of File Management Procedures," section 2.6. The SETMODE functions are:

28 initialize to default values
30 allow nowait I/O's to complete in any order, necessary for full duplex communications
31 packet mode
32 set call setup parameters

PTP returns an "invalid operation" error if the application program specifies other than these four functions.

CONTROL

PTP supports three CONTROL requests:

11 - wait for modem connect, completes when virtual circuit is established

12 - disconnect the modem, clears the virtual circuit

17 - enable connection, initiates a call to a remote DTE in an X.25 network. The subdevice must have a defined ADDRESS configured; otherwise, the application must call SETPARAM to determine the address before this control function.

NETWORK PROTOCOL

The Network (NET) protocol provided by the X.25 Access Method is another level 4 protocol. NET only provides the ability to communicate with another Tandem system that is connected to an X.25 virtual circuit. This protocol is used in conjunction with an EXPAND network line handler.

With NET, two Tandem systems communicate across the X.25 network in the same way they would if connected by a leased line. The EXPAND network handles all file access between the systems. See the EXPAND Users Manual for an overview of the EXPAND facility.

When using the NET protocol, three items must be defined:

1) device type and subtype
2) line handler LDEV of network
3) unique system number

Details are given in "Configuration Needs," section 2.7.

SUMMARY OF FILE MANAGEMENT PROCEDURES

This section briefly describes three file management procedures: CONTROL, SETMODE, and SETPARAM. Each description includes only functions available in the X.25 Access Method. Complete descriptions of each of these procedures are contained in section 2.3 of the Guardian Operating System Programming Manual.

CONTROL Procedure

The CONTROL procedure performs device dependent I/O operations. In the X.25 Access Method, this procedure is used primarily to establish and to clear circuits.

CALL CONTROL (<file number>, <operation>, <parameter>, <tag>)

<file number>, INT:value, is the file number provided by OPEN.

<operation>, INT:value, is defined in the table that follows

<parameter>, INT:value, is defined in the table that follows

<tag>, INT(32):value, for nowait opens only

```
CONTROL Operations

1 = terminal forms control
    <parameter> = 0 => form feed
                = 1 => vertical tab

11 = wait for modem connect (in X.25 Access Method, for receiving
     calls)
     <parameter> = none

12 = disconnect the modem
     <parameter> = none
```

-->

```
17 = enable connection (in X.25 Access Method, for initiating
     calls)
     <parameter> = none
```

The SETMODE procedure sets device dependent functions. In the X.25 Access Method, this procedure is used for terminal control in the ITI protocol and for selecting options within the PTP protocol.

```
CALL SETMODE ( <file number>, <function>,
               <parameter 1>, <parameter 2>, <last params> )
```

<file number>, INT:value, is the file number provided by OPEN.

<function>, INT:value, is defined in the table that follows

<parameter 1>, INT:value, is defined in the table that follows

<parameter 2>, INT:value, is defined in the table that follows

<last params>, INT:ref:2, returns the previous settings of <parameter 1> and <parameter 2> associated with <function>.

```
SETMODE Functions

 6 = set spacing control
     <parameter 1> = 0 => no space
                   = 1 => single space (default)

 8 = select page mode
     <parameter 1> = 0 => conversational mode
                   = 1 => page mode

 9 = set interrupt characters
     <parameter 1>.<0:7>  => character 1
                  .<8:15> => character 2
     <parameter 2>.<0:7>  => character 3
                  .<8:15> => character 4

11 = set break ownership
     <parameter 1> = 0         => disable break  (default)
                   = <cpu,pin> => enable break

12 = set terminal access mode
     <parameter 1> = 0 => normal mode
                   = 1 => break mode

27 = set pre-space
     <parameter 1> = 0 => post-space (default)
                   = 1 => pre-space

28 = initialize to default values
     <parameter 1> = none
```

-->

```
30 = allow nowait I/O's to complete in any order
     <parameter 1> = 0 => half duplex (default)
                   = 1 => full duplex

31 = set packet mode
     <parameter 1> = 0 => normal mode (default)
                   = 1 => packet mode

32 = set call setup parameters
     <parameter 1>.<0> = 0 => no accept charge
                       = 1 => accept charge
                  .<1> = 0 => no request charge
                       = 1 => request charge
                  .<2> = 0 => normal mode
                       = 1 => priority call
                  .<8:15>  => port number
```

The SETPARAM procedure currently is used only to set and to fetch a remote DTE address. When setting the address, <param array> contains an ASCII string of decimal digits representing the network address of a remote DTE that the application calls when a call is issued to the CONTROL procedure to enable the connection. When fetching the remote DTE address, the application may obtain the calling party address after accepting the call via CONTROL ("wait for modem connect").

```
    CALL SETPARAM ( <file number>, <function>,
                    <param array>, <count>,
                    <last param array>, <last count> )
```

<file number>, INT:value, is the file number provided by OPEN.

<function>, INT:value,
1 -- set or fetch a remote DTE address.

<param array>, INT:ref:1, a list or string as required by <function>

<count>, INT:value, number of characters contained in <param array>

<last param array>, INT:ref:1, is returned the previous parameter settings associated with <function>.

<last count>, INT:ref:1, is returned the length of <last param array> in bytes. Maximum 256 bytes.

USES OF SETPARAM

Use of SETPARAM when receiving calls on a subdevice:

* The value of the remote DTE address is set to the calling DTE address whenever a call is received; i.e., any previous setting of this parameter is overwritten when a call is received.

* For both the ITI and PTP subdevices, the application program may fetch the calling DTE address by calling SETPARAM in the following manner:

```
CALL SETPARAM(filenum, 1, , , remote^dte, len);
  ! then user may examine, record, or otherwise process other
  ! end's address
```

  where remote^dte is an INT array eight words long. This array contains len characters in ASCII representing the calling DTE address.

Use of SETPARAM prior to initiating a call into the X.25 network:

* The subdevice may have a predefined remote DTE address (via CUP configuration).

* The application also may call SETPARAM to dynamically set the called DTE address before issuing a call CONTROL(filenum, 17):

```
remote^dte ':=' [ "31104080032155" ];
CALL SETPARAM(filenum, 1, remote^dte, len);
```

SEQUENCE OF APPLICATION CALLS TO FILE SYSTEM PROCEDURES

The application opens a "terminal," accepts calls, interacts with the user, hangs up, and goes away.

```
1)    filename ':=' [ "$X25    #TERM1          " ];
      CALL OPEN(filenum, filename);

2)    CALL CONTROL(filenum, 11);          ! wait for incoming call

2a)   Optional step if application would like to fetch the calling
      DTE network address.

      CALL SETPARAM(filenum, 1, , , calling^dte, calling^len);

3)    CALL WRITE(filenum, greeting, greeting^len);
```

```
4)     do begin
       CALL WRITEREAD(filenum, promptbuf, promplen, readlen);
               ! at beginning of prompt & processing loop
       .
       .
       .
       until done;

5)     say^goodbye:
       CALL WRITE(filenum, goodbye, goodbye^len);

6)     CALL CONTROL(filenum, 12);        ! disconnect

7)     CALL CLOSE(filenum);
```

The application sets up CALL REQUEST parameters and remote DTE address.

```
1)     filename ':=' [ "$X25    #APP2          " ];
       CALL OPEN(filenum, filename);  ! open subdevice #APP2 on $X25

2)     buf ':=' [ "31107030012344" ];     ! set remote DTE address
       CALL SETPARAM(filenum, buf, 14);   ! DNIC 3110, area 703,
                                          ! DTE 123, port #44.

3)     CALL CONTROL(filenum, 17);         ! initiate a call request

4)     CALL WRITE(filenum, initmsg, initmsg^len);

5)     do begin
       CALL WRITEREAD(filenum, msgbuf, msglen, readlen); !for each
                                                         !request
       .
       .
       .
       until done;

6)     send^end^msg:
       CALL WRITE(filenum, endmsg, endmsg^len);

7)     CALL CONTROL(filenum, 12);         ! disconnect

8)     CALL CLOSE(filenum);
```

CONFIGURATION NEEDS

Three steps are required to configure a Tandem system for the X.25 Access Method. First, an account with an X.25 network carrier must be established. Second, the Guardian operating system must be configured through SYSGEN to implement the X.25 Access Method and to establish the number of circuits for incoming and outgoing calls. Only one physical connection exists between the system and an X.25 TIP, a local network node, for each X.25 Access Method configured. This physical connection is a synchronous line that is configured as a logical device. Third, the subdevices are added to this logical device with the Communications Utility Program (CUP).

ESTABLISHING AN ACCOUNT

An X.25 network carrier provides information about X.25 packet switching networks. However, in order to establish an account, the carrier requests certain information:

1) the number of virtual circuits required, i.e., the number of ports to be available for incoming and outgoing calls;

2) ASCII or EBCDIC -- the X.25 Access Method assumes ASCII, but can be altered to EBCDIC through CUP and SYSGEN.

The carrier provides a network address. In the United States this address is the area code of the central office and a number that is unique within that area code. CUP requires this address for certain protocols. For example, 311021300092 might be a network address. The first four digits represent the network carrier. The next eight digits are at the carrier's discretion. An example of this address in use is:

```
@C 213 92                   (User connects to the X.25 network)
 213 92 CONNECTED
:LOGON INVENT.JAN
    (normal session)
:LOGOFF
 213 92 DISCONNECTED
```

SYSGEN INFORMATION

The system generation, SYSGEN, for the X.25 Access Method includes:

* Line parameters: buffer size, timeouts, retries, and level 2 protocol.

* Circuit parameters: level 3 window size, packet size, number of circuits

* Level 4 protocols

The I/O system configuration section of the configuration file for SYSGEN should be modified for the X.25 Access Method.

The X.25 Access Method requires dedicated buffer space for level 2 data transfers. The size of the dedicated buffer can be determined as follows:

```
L = (K + R) * N + 5

   L = Buffer length in words
   K = Level 2 window size
   R = Number of read frames desired
   N = Frame size in words

where

   K = number of frames which may be sent before an acknowledgement
       is required.
   R = 2 for 9600 baud or slower; for faster lines, 3 or 4 to keep
       the no-frame buffer count under control.  See the CUP STATS
       command for details.
   N = (PACKETSIZE/2) + 5

Example: for PACKETSIZE=128, 2 read buffers, L2WINDOW=4

       L = (4 + 2) * 69 + 5 => 419
```

The X.25 Access Method line handler is configured as a logical device of type 61 and uses protocol X25LAP^PROTOCOL.

A driver and a special interrupt handler are configured for the type of controller used, Byte Synchronous.

```
Driver              : AXCOM6201FDX
Interrupt Handler   : AXCOM6201INTERRUPT
```

A Communications Control Block is configured by specifying the SYSGEN macro: X25AM. Two modifiers are required parameters to this macro:

CIRCUITS <number> -- where number is the number of circuits requested from an X.25 carrier.

L4PROTOCOLn <name> -- where n is a number representing the type of the protocol and name is the level 4 protocol to be supported. Currently, the available protocols are:

L4PROTOCOL0	X25^ITI^PROTOCOL
L4PROTOCOL1	X25^PTP^PROTOCOL
L4PROTOCOL7	X25^NET^PROTOCOL

The following modifiers provide for optional parameters:

ASCII -- sets the controller for ASCII sync characters. This modifier overrides the default setting of EBCDIC in the X25AM macro. The value of this modifier must match that of the DCE at the other end of the line.

DCE -- sets up level 2 to operate as a DCE rather than a DTE.

DTE -- included to be symmetrical with the DCE modifier. The X25AM macro uses DTE as the default on the line for a Tandem system.

EBCDIC -- included to be symmetrical with the ASCII modifier. Since the X25AM macro defaults to EBCDIC operation, this modifier is useful only as a comment.

L2RETRIES <number> -- where number is the number of times the line handler retries an operation at the link level before proceeding to the next step in recovery. Default is 20 retries.

L2TIMEOUT <number> -- where number is the number of ticks for which the line handler waits for responses at the link level. Default is 300 ticks which is 3 seconds.

L2WINDOW <number> -- where number is the value of K determined under Buffer Requirements. Default is 4 frames.

L3WINDOW <number> -- where number is the number of outbound packets that may be unacknowledged for each circuit. Default value is 2 packets.

PACKETSIZE <number> -- where number is the size of packets to be sent and received by this system. This value determines the size of all send and receive buffers. Default value is 128 bytes.

STARTDOWN -- disables the line immediately after a cold start. This modifier is convenient because the X.25 line is useful only after certain line parameters are set and after subdevices are added to the line through CUP.

SYNCS <number> -- where number is the number of sync characters that precede each frame. Default is 3 syncs.

The X.25 Access Method process name is "AXCESS^X25" and requires a stack of 600 words. The system code for the X.25 Access Method is contained in the PROCESSX file. The software release tape for version D of the GUARDIAN operating system contains command files that build the PROCESSX file automatically.

Example

```
!
! Packet size = 128 bytes
! Read buffers = 2
!
 19   $X25     0   12   1   12          61          0   128
    D      419      AXCOM6201FDX  X25LAP^PROTOCOL  AXCOM6201INTERRUPT
    8  8    0       8  8    1
 X25AM
    L2RETRIES 10
    L2TIMEOUT 50
    PACKETSIZE 128
    CIRCUITS 25
    L4PROTOCOL0  X25^ITI^PROTOCOL
    L4PROTOCOL1  X25^PTP^PROTOCOL
```

Other Buffer Requirements

The X.25 Access Method also requires packet buffers for inbound data buffering. The number of packet buffers required is a function of the number of virtual circuits actually in use, and the line parameters PACKETSIZE and L3WINDOW. Here, the meaning of active circuit is one on which there is a call established.

Packet buffers for the virtual circuits are allocated when the virtual circuit is established and released when the circuit is cleared. Packet buffer space is maintained by both primary and secondary line handlers.

Message buffer space is required for each read or write request. Incoming data is first read into a packet buffer, then assembled in the read message buffer before being returned to the user. Outbound data is moved directly from the write message buffer into the frame buffer.

When the Access Method is initialized, LONGPOOL space also is obtained for the circuit control blocks. The size of this LONGPOOL space is:

number of words = (number of circuits * 11) + 1

COMMUNICATIONS UTILITY PROGRAM

The Communications Utility Program (CUP) performs operations related to data communication lines, devices, and network environments. Through CUP, subdevices for the X.25 Access Method are added, deleted, or reconfigured.

The CUP commands that perform these functions required by the X.25 Access Method are: ADD, DELETE, and ALTER. Two other commands, CONNECT and CLEAR, also are necessary for the NET protocol.

Configuring Subdevices

CUP requests several modifiers for each configuration command. This list explains the possible options for each modifier used with X.25 subdevices.

Required Modifiers

PROTOCOL	Defines the type of protocol. May be: ITI - Interactive Terminal Interface PTP - Process to Process NET - Expand network interface
RECSIZE	Defines the record size. ITI - probably 80 PTP - application dependent NET - CUP sets this to the PACKETSIZE from the EXPAND network
TYPE	Use the following, according to PROTOCOL: ITI - (6,0) PTP - (9,0) (Type 9 specifies "foreign process") NET - don't care

Optional Modifiers

[NO]ACCEPT	Installation and application dependent. ITI - ACCEPT makes it easier for the terminal user PTP - user determines the use of subdevices NET - probably NOACCEPT
ADDR	Used only when a particular subdevice needs to be configured to have a predefined remote DTE address for calling OUT to that DTE. For example, NET subdevices must have the proper DTE ADDR configured at both ends of the EXPAND network path.
PORT	Aids in connecting an incoming call to a subdevice. ITI - if 0, the terminal user gets the next available ITI subdevice without specifying anything special PTP - any non-zero number that is mutually agreeable NET - any non-zero number that is mutually agreeable

[NO]REQUEST Installation and application dependent.
ITI - not applicable since it cannot call out
PTP - Application dependent
NET - If default is NOREQUEST, the billing invoice reflects the charges according to the initiator.

Two modifiers apply ONLY to NET subdevices and MUST be provided to establish the EXPAND connection.

LHLDEV	must be the LDEV, logical device number, of the EXPAND Line Handler that intends to use this subdevice to establish a connection to the other system.
NEXTSYS	the system number within the EXPAND network of the other system.

Sample Subdevice Configuration For an X.25 Line

```
ADD #TERM1,PROTOCOL ITI,TYPE(6,0),RECSIZE 80,ACCEPT
ADD #TERM2 LIKE #TERM1
ADD #TERM3 LIKE #TERM1
            .
            .
ADD #TERM10 LIKE #TERM1
ADD #APP1,PROTOCOL PTP,TYPE(9,0),RECSIZE 80,PORT 44
ADD #APP2 LIKE #APP1,PORT 55,ADDR 31102130009955
ADD #SNOOPY,PROTOCOL NET,TYPE(63,0),LHLDEV 23,NEXTSYS 3, &
            ADDR 31104150000199,PORT 99
```

#TERMn For accepting terminal users calling from the packet network. Defined with ITI protocol, type of 6 to look like a terminal to the application, record size of 80 since the actual width of the terminal is unknown, ACCEPT reverse charges, and the default PORT number of zero to simplify the user's logon sequence for general access.

The user may want to specify groups of ITI subdevices with a unique port number assigned to each member of the group. This configuration allows the terminal user to select any available member of the group attached to a specific set of application programs.

#APP1 A process to process port probably used for accepting calls from a foreign, non-Tandem application attached to the X.25 network. The port number of 44 is used to identify this specific subdevice when processing the INCOMING CALL packet. The subdevice refuses reverse charges. The foreign application attaches to this specific port by addressing it with port=44 in its CALL REQUEST packet. The applications agreed on a record size of 80 characters. The specification of RECSIZE 80 is for information purposes only in a terminal definition.

#APP2 This subdevice also is used to communicate with foreign applications. The notable difference is that it has a predefined remote DTE address so that the local application program does not have to call SETPARAM before calling out into the X.25 network. The remote DTE address in this example specifies a hypothetical network DNIC 3110, area 213, DTE 99, port 55. The PORT 55 specified for this subdevice is used only in identifying the local calling application to the remote application.

#SNOOPY This EXPAND network port attaches one Tandem system to another through an X.25 packet network. NEXTSYS 3 is the system number within the EXPAND network of some friendly system located near Santa Rosa. The LHLDEV identifies the specific EXPAND line handler configured through SYSGEN into the system to handle EXPAND connections through this X.25 Access Method line handler.

The remote ADDRess and PORT configured into each end of this X.25 circuit must accurately and completely specify the partner. An INCOMING CALL packet to a NET protocol subdevice is checked to see if it is from the remote DTE specified by ADDR. This check provides a level of insurance to keep unauthorized foreign systems out of a user's EXPAND network.

MESSAGES

ERRORS RETURNED TO FILE SYSTEM CALLS

0	Normal completion
1	End-of-file, returned by ITI following a Control Y
2	Invalid operation, like READ/WRITE when call not established or unsupported SETMODE/CONTROL function attempted
21	Illegal count parameter in READ/WRITE request
33	X.25 Access Method unable to obtain IOPOOL space required to process the request
122	Request aborted due to possible data loss caused by a reset of the circuit.
140	"modem error", whenever a CLEAR INDICATION is received from the network, following a loss of connection to the network, or failure of call attempt.

MESSAGES LOGGED ON THE OPERATOR CONSOLE

The following messages are logged on the operator console to record the making and breaking of the link connecting the T/16 to the X.25 packet network:

```
"nn  hh:mm  ddmmmyy  FROM  sss,cc,ppp  LDEV ll  X25: LINE READY"

"nn  hh:mm  ddmmmyy  FROM  sss,cc,ppp  LDEV ll  X25: LINE NOT READY"

"nn  hh:mm  ddmmmyy  FROM  sss,cc,ppp  LDEV ll  X25: LINE QUALITY nnn"
```

where

nn	indicates the message number.
hh:mm	indicates the time of day.
ddmmmyy	indicates the date.
sss	indicates the system.
cc	indicates the CPU number.
ppp	indicates the pin.
ll	indicates the logical device number.
nnn	indicates the line quality where nnn is the percentage of good messages to total messages received.

INDEX

INDEX

READER'S COMMENTS

Tandem welcomes your feedback on the quality and usefulness of its publications. Please indicate a specific *section* and *page* number when commenting on any manual. Does this manual have the desired completeness and flow of organization? Are the examples clear and useful? Is it easily understood? Does it have obvious errors? Are helpful additions needed?

Title of manual(s) ______________________________

FOLD ➤

FOLD ➤

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

City/State ____________________ Zip ______________

A written response is requested. yes no ?

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA. U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM
COMPUTERS, INC.

Attn: Technical Publications
19333 Vallco Parkway
Cupertino, CA, U.S.A. 95014

STAPLE HERE

10268739

TANDEM 16

XRAY
USERS MANUAL

NonStop™
SYSTEMS

XRAY
USERS MANUAL

T16/8037 A02
82037

TANDEM 16

XRAY

USERS MANUAL

TANDEM COMPUTERS INCORPORATED
19333 Vallco Parkway
Cupertino California 95014

Product No. T16/8037-A02
Part No. 82037

July 1979
Printed in U.S.A.

INTRODUCTION

LIST OF EFFECTIVE PAGES

Pages	Effective	Software Product
Title	Oct. 1978	
List of Effective Pages	Mar. 1979	
Preface	Oct. 1978	
(1)-1	Mar. 1979	
1-1 thru 1-10	Mar. 1979	T9006C03
2-1 thru 2-11	Mar. 1979	T9006C03
(3)-1	Mar. 1979	
3-1 thru 3-20	Mar. 1979	T9006C03
(4)-1	Mar. 1979	
4-1 thru 4-60	Mar. 1979	T9006C03
(5)-1	Oct. 1978	
5-1 thru 5-13	Mar. 1979	T9006C03
A-1 thru A-7	Mar. 1979	
B-1 thru B-2	Oct. 1978	
Index-1 thru Index-3	Mar. 1979	

This manual describes the Tandem XRAY (TM) Performance Monitor. XRAY is a tool intended for a specific purpose: to analyze the performance of a Tandem 16 system.

This manual will show you how to use XRAY to measure the components of your system. It is organized as follows:

INTRODUCTION TO XRAY

XRAY is a software tool for monitoring the performance of a Tandem 16 computer system. Applications of XRAY include:

- Mix Balancing--distributing applications evenly across system components in order to use the system efficiently
- Growth Management--monitoring system components (processors, memories, discs, communication lines, etc.) on a long term basis to permit orderly management of growth
- Application Tuning--identifying ways in which applications can be restructured to increase transaction throughput
- Distributing data base files among system's disc volumes to avoid bottlenecking at a single spindle
- Identifying programs responsible for excessive processor, message, or virtual memory activty
- Restructuring data base files for speedy access, and determining the optimum disc cache size
- Optimizing packet size for the message traffic generated in an EXPAND network of TANDEM 16s
- Determining an improved layout of communications links based on the fraction of forwarded messages
- Tracking response times to provide an objective measure of system performance.

XRAY runs on a minimum Tandem 16 with the Guardian Version D operating system and firmware. XRAY does not require Enscribe or Envoy, but is fully integrated with each, allowing all data base and communication activity to be measured. XRAY does not require any special hardware or display devices; XRAY can be operated from any asynchronous, point-to-point terminal with a line width of at least 80 characters.

How to Use XRAY

The measurement undertaking has the following steps:

1. Make a clear statement of the question to be answered by the measurement. Skipping this step is the most frequent cause of unsuccessful measurement efforts. The formulation of the performance question usually requires:
   a. A good understanding of the application running on the system, and
   b. A good understanding of the physical hardware configuration of the measured system.

2. Select the hardware and software components to be measured. List these in an EDIT format file, as discussed in the section on XRAYCOM. The file is input to the program XRAYCOM in the next step.

3. Start the measurement using the program XRAYCOM. The system is always measured as a whole, so it is wise to centralize the running of measurements. It is a good idea to appoint a "measurement coordinator" who acts as a central point of control for measurement requests; this also centralizes the accumulated experience needed for determining the measurement configuration, output data file size and location, and measurement resolution, all of which affect measurement overhead.

4. Examine the measurement using XRAYSCAN. Reports and time plots of the gathered data are easy to generate with the reporting and plotting commands. The file under measurement can be scanned online, but usually you will defer this activity until later, so that your examination of the data will not disturb the measurement.

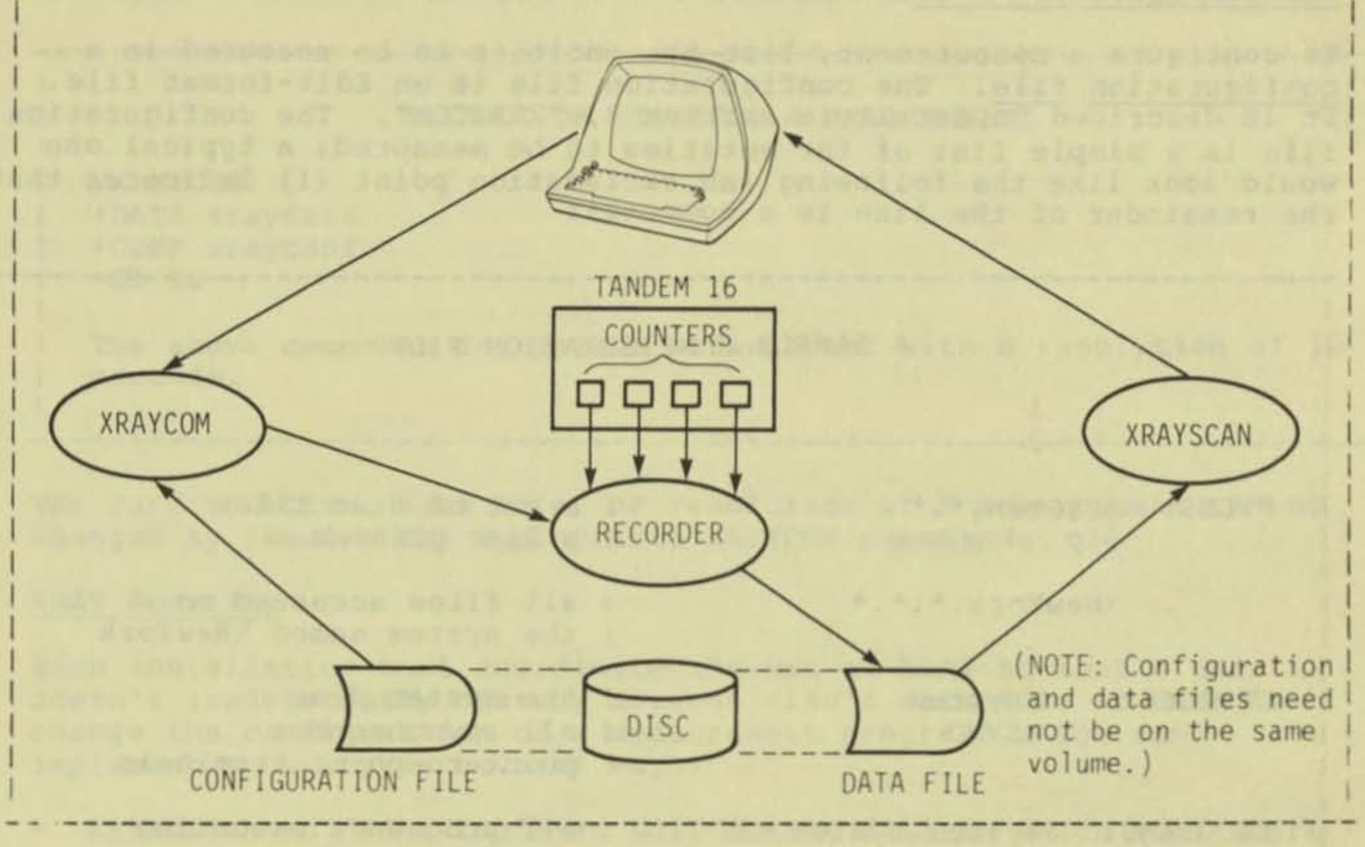
   XRAYSCAN is easy to use if you take it a step at a time. You will learn most quickly by using XRAY on some program you know well, in an isolated environment, i.e., "standalone", if possible. Explore this measurement in some detail, to become familiar with the analysis capabilities. Analysis is simple if you have performed step 1 above. For some guidance in formulating your performance questions, refer to the section on Performance Basics in this manual.

How it Works

The user interface to XRAY consists of two interactive programs, XRAYCOM and XRAYSCAN.

XRAYCOM lets you configure, start, and stop a measurement. A set of counters in the operating system data space keeps track of the events monitored by XRAY. When XRAYCOM is run on a particular system, it creates in each local processor a process known as the Recorder. While a measurement is in progress, the Recorder in each processor periodically writes the status of all of the configured counters to an unstructured disc file called the data file.

To analyze the collected data, XRAYSCAN is run against the data file. XRAYSCAN has a set of commands that allow you to examine measurement data in the form of tables or time plots. If desired, XRAYSCAN can be run while a measurement is in progress, allowing online analysis of the Tandem 16's performance. Section 4 describes XRAYSCAN in detail.



Definitions

XRAY collects statistics relating to various hardware and software components of the Tandem 16. The word "entity" is used throughout this manual to refer to a Tandem 16 component measured by XRAY. Thus, the class of entities includes:

* Physical devices--processors, disc volumes, communication lines, terminals, line printers, tape drives
* Processes--both system and user
* Disc files, and files on other devices
* Systems in the same network with the measured system

A measurement refers to the act of collecting data relating to a given set of entities, over some time interval.

The set of entities on which XRAY collects statistics is known as the configuration of the measurement. Specifying the set of entities XRAY is to measure is referred to as "configuring a measurement."

The Configuration File

To configure a measurement, list the entities to be measured in a configuration file. The configuration file is an Edit-format file. It is described in detail in section 3, "XRAYCOM". The configuration file is a simple list of the entities to be measured; a typical one would look like the following (an exclamation point (!) indicates that the remainder of the line is a comment):

```
                    A SAMPLE CONFIGURATION FILE

             1
             2

 FILES:   $SYSTEM.*.*                  ! a set of disc files
          $lp                          ! a line printer

          \NewYork.*.*.*               ! all files accessed on
                                       ! the system named \NewYork

 DEVICES:    $system                   ! the system disc
             ASYNC                     ! all asynchronous
                                       ! point-to-point terminals

 PROGRAMS:  $system.system.tal         ! all processes executing
                                       ! this object file


            SYSPROCS                   ! all system processes
            $myproc                    ! the named process

 SYSTEMS:   \Zurich                    ! traffic to the system
                                       ! named \Zurich

 EXCLUDE:   $SYSTEM.SYSTEM.COMINT      ! don't measure Command
                                       ! Interpreters
```

XRAYCOM

After the configuration file is created, XRAYCOM is run to:

- Designate the data and configuration files.

- Specify the time interval at which the Recorders are to copy the counters to the data file. This value (which may range from 1 to 3600 seconds) limits the resolution--the ability to discern changes in the counters over short intervals of time--of the measurement.

- Start the measurement.

The following is an example of the commands used to accomplish this:

```
------------------------------------------------------------------------
|                                                                      |
|              EXAMPLE OF STARTING A MEASUREMENT                        |
|                                                                      |
| :XRAYCOM                                                             |
| +DATA xraydata                                                       |
| +CONF xrayconf                                                       |
| +GO 10                                                               |
|                                                                      |
|  The above commands initiate a measurement with a resolution of 10   |
|  seconds.                                                            |
|                                                                      |
------------------------------------------------------------------------
```

The configuration, data file, or resolution of a measurement can be changed by issuing the appropriate XRAYCOM commmands.

XRAY Security

Each installation must coordinate the use of XRAY so that a user doesn't inadvertently destroy someone else's data file or doesn't change the configuration of a measurement progress. You can implement security in several ways:

- Establish an "XRAY log"--an Edit-format file in which XRAY users make a note of what they are doing. Before running XRAYCOM, each user should check the log to see if a measurement is already in progress.

- Using the FUP SECURE command, specify that only a certain user may execute XRAYCOM.

- Using the FUP SECURE command, specify that only a certain user may write to the Recorders' output data file. It is wise to restrict writinng to this file to local users only.

When XRAYCOM is run, it locks a file named $SYSTEM.SYSTEM.XRAYLOCK. If this file is already locked, indicating that another copy of XRAYCOM is already running, XRAYCOM ABENDs. To employ this feature, run XRAYCOM as a named process, start the measurement, and SUSPEND the process. Then issue the ACTIVATE command to terminate or alter the measurement.

Any number of users may simultaneously examine a data file (via XRAYSCAN) without interfering with one another. Only the use of XRAYCOM and access to the configuration file need be controlled.

Sysgen Note

XRAY collects data by monitoring a set of counters in the operating system data space. Throughout this manual, these counters are referred to as the measurement space.

A portion of the measurement space is obtained from LONGPOOL. Space requirements depend on the measurement configuration, but you should allocate an additional 2000 (decimal) words of LONGPOOL at SYSGEN time when you use XRAY. If this is not sufficient, increase the size of LONGPOOL or refine the measurement configuration.

Examining the Collected Data

XRAYSCAN explores and filters the data in an XRAY data file. Any part of the data file can be selected for examination. Sets of the measured entities can then be chosen for analysis with the entity selection commands.

XRAYSCAN ENTITY SELECTION COMMANDS

Command	Reports on:
BUFFER	process buffer activity
CPU	processors
DEVICE	tapes, printers, etc.
DISC	discs
DISCOPEN	disc file opens
FILE	file opens
LINE	data communication lines
NETLINE	network data communication lines
POOL	processor buffer pools
PROCESS	processes
SYSTEM	remote systems
TERMINAL	terminals

For each of the above commands, a set of items is displayed. The items depend on the nature of the component being reported on; for example, the items associated with the CPU report command are:

CPU BUSY	Percent of elapsed time that the processor was in use
SWAP RATE	Virtual memory pages input and output per second
CHIT RATE	Number of disc cache hits per second
SEND RATE	Total number of messages sent per second
CPU QLEN	The average length of the processor queue
DISP RATE	Number of times, per second, that the processor was set up to execute some process
TRAN RATE	Number of terminal interactions per second
RESP TIME	Average wait during a terminal interaction

Section 4 of this manual contains a complete list and description of the items associated with each type of entity.

The XRAYSCAN report can be restricted to entities having values in particular ranges. For example, the command

```
      +PROCESS  1, IF CPU BUSY > 1.5
```

restricts the report to those processes in CPU 1 that have used 1.5% of the time during the portion of the measurement being examined.

The current report can be displayed with the entities sorted on any item. This is done by naming the item in a BY clause:

```
       +FILE  $ORDERS.*.*, BY FILE RATE
```

The files accessed on $ORDERS will be displayed in the order of most active file to least active.

Time Plots

XRAYSCAN also displays collected data in the form of time plots. As the following illustration shows, it is simple to generate a plot. (Illustrative comments are in lower case.)

```
     +CPU
     +PLOT CPU BUSY
     +PLOT
```

```
 time           measured percent busy (% of elapsed time)      legend which
 to the         during interval ended by time at the left       defines the
 second                           |                             plot symbols
   |                              |                                   |
   |                              |                                   |
   |      0::::::::20::::::::40::::::::60::::::::80:::::::100         |
10:30:00 -          +         +         A      B  +          - A: CPU 0
      15 -                           A             B         -   CPU BUSY
      30 -                          A               B        - B: CPU 1
      45 -                        A                    B     -   CPU BUSY
10:31:00 -          +         +    A    +         +     B    -
      15 -                        A                     B    -
      30 -                           A               B       -
      45 -                              A          B         -
10:32:00 -          +         +         +A        B          -
      15 -          |                     A      B           -
      30 -          |                      A    B            -
      45 -      graticules                 A B               -
10:33:00 ----------for--------+         + A       +          -
      15 -       sighting               B A                  -
      30 -          |                   A B                  -
      45 -          |                    A B                 -
10:34:00 -          +         +         + A    B  +          -
      15 -                                   A  B            -
      30 -                                  A                -
      45 -                               A         B         -
10:35:00 -          +         +       A +         + B        -
      15 -                         A                   B     -
          0::::::::20::::::::40::::::::60::::::::80:::::::100
```

The total range and interval size of the time axis, and the range of the horizontal axis, can be adjusted with XRAYSCAN commands.

You can also "build" time plots of any items you like by adding items of interest until you have exactly the items you want plotted on the same set of axes.

Obtaining Copies of Reports and Plots

Use the COPY command to make a line printer copy of the last report or plot generated.

Online Monitoring

Online performance monitoring is achieved by running XRAYSCAN against the currently active disc file. Any item or set of items in the measurement can be plotted on a terminal as they are observed.

Network XRAY

Using the EXPAND network of TANDEM 16 commputer systems, XRAY can measure and analyze entire network from a single system. Additional XRAY features permit observation of network traffic to, from and through each node.

XRAY is used most effectively to collect and scan data at each node. The measurement configuration and other XRAYCOM command sequences be located at a single system in the network. When XRAYCOM is run on the remote system, the Recorders at that system collect data in a file located on that system. Then the XRAYSCAN program is run on that system as well. The reports and plots of XRAYSCAN can be directed to any other system simply by directing XRAYSCAN's Command Interpreter OUT file to the desired system.

The fundamental restriction on the use of XRAY in the network is that the Recorders in separate systems must use separate data files. This restriction prevents ambiguities arising from systems operating in different time zones, and permits XRAYSCAN to deal with one system's physical configuration at a time. If the recommendation suggested in the previous paragraph is followed, this restriction is eliminated, since each data file would be local to the system under measurement.

XRAY features designed specifically for the network environment include:

- Measurement of packet traffic with every remote system and the measured system, with distinction between "local" packets and forwarded packets
- Measurement of the total time to transmit a logical message to each remote system
- Network communication line utilization
- Number of bytes sent and bytes received in Level 2 Protocol communication on the lines
- Number of bytes sent and bytes received in Level 4 Protocol communication, distinguishing between data bytes and control bytes
- Number of messages sent which were smaller than 64, 128, 256, etc. bytes long, respectively
- Measurement of file activity against files opened on a remote system, on a per-file-open basis, tied to the opening process
- Measurement of disc file activity on the measured system which originated on the various remote systems, again on a per-file-open basis.

| These facilities permit the analyst to answer questions such as which
| files account for traffic to a particular system, how much forwarding
| of packets is done as opposed to direct communication, how much local
| disc access is due to processes on remote systems, and what fraction
| of messages is sent by a single packet of a given size.

XRAY EXAMPLE

This section contains a simple example of the use of XRAY, including the design, configuration, and analysis of a measurement.

The subject of the measurement is a TAL compilation using the Spooler as its list file. The source file is named $MKT.TEST.SOURCE. The measurement was carried out on a 4-processor system under a light load (only a few small programs running).

The TAL compiler and the Spooler collection process were run in different processors.

The example consists of four steps:

1. Create the configuration file.
2. Create the data file.
3. Start the measurement, run TAL, and stop the measurement.
4. Run XRAYSCAN to analyze the data.

STEP 1--Create the Configuration File:

```
PROCESSORS:    0 1 2 3

PROGRAMS:      $SYSTEM.SYSTEM.TAL    ! TAL compiler
               $SPOOL                ! Spool Collector
               $SPLS                 ! Spool Supervisor

FILES:         $MKT.TEST.SOURCE      ! TAL source file

DEVICES:       $SYSTEM     $MKT      ! disc volumes containing source
```

This configuration tells XRAY to measure all four processors on the system the TAL compiler, the Spooler collection process $SPOOL, the disc file $MKT.TEST.SOURCE, and the two disc volumes $SYSTEM and $MKT.

STEP 2--Create the Data File:

Estimate the amount of data the measurement is going to generate. The above configuration would be classified as "medium": There are four processors, the measurement will take about one minute, and we intend to write the data to the data file once each second.

A formula in Section 3 of this manual allows us, on the basis of the above information, to estimate the amount of data to be collected:

$4 * 60 = 240$ pages

(four processors, data to be written to the data file a total of 60 times.)

To be on the safe side, a data file containing 400 pages was created. Thus,

```
:CREATE xdata, 25
```

creates an unstructured disc file with a primary and secondary extent size of 25 pages; since a disc file can contain as many as 16 extents, this file should be able to hold all the data.

(Note: the measurement actually wrote about 244 pages to the data file).

STEP 3--Perform the Measurement:

To start the measurement, XRAYCOM was run with the following commands:

```
:XRAYCOM

+DATA xdata
+CONF xconf
+GO 1
+EXIT
```

The GO 1 command tells XRAY to write the measurement data to the data file every second.

When this measurement was performed, processor 3 was down. XRAYCOM initially reported this fact when it was run, and again when processing the "CONF" command.

Next, TAL was run.

```
:TAL/IN source, OUT $spool, CPU 1/
```

CPU 1 was specified because $SPOOL was running in CPU 0 and we wanted to cause a lot of inter-processor message traffic.

After TAL finished, XRAYCOM was run to stop the measurement.

```
:XRAYCOM

+EXIT!
```

STEP 4--Analyze the Data:

Once the measurement has been performed, this step can be done at any time. The following pages take you through a short tour of the data file.

Now that the measurement has been performed and the data stored in $MKT.TEST.XDATA, the XRAYSCAN program can be run to analyze the results. XRAYSCAN allows you to examine the data file and build time plots of those items that interest you. In the following example, we will content ourselves with verifying two facts that we already know to be true:

- There is a very strong correlation between the messages initiated by TAL and the messages obtained by $SPOOL via its $RECEIVE file.
- There is a correlation between the rate at which a particular disc file is read and the rate at which the entire disc is read.

In the remainder of this section, the following conventions apply:

* Text explaining the example is surrounded by less-than and greater-than signs.
* Lines beginning with a plus sign (+) are XRAYSCAN commands. To further separate these from XRAYSCAN's output, all commands are in lower case.

  The first line of the example is the command to run XRAYSCAN, which is preceded by a Command Interpreter prompt (:).
* Everything else is actual output from XRAYSCAN.

```
>  The first thing to do is log on at an interactive terminal and    <
>  run XRAYSCAN against the data file.  Assuming the appropriate     <
>  default volume and subvolume names, this command is               <

:xrayscan xdata

>  XRAYSCAN prints a time plot of processor utilitization for all    <
>  processors under measurement.  This is often time-consuming, but  <
>  XRAYSCAN requires the time to initialize its internal tables.     <
```

```
XRAYSCAN VERSIONS C00
AUG 12 16:16

         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
16:16:33 -   B  C  +         +        A+         +         -  A: CPU 0
      46 - B  C                                   A        -     CPU BUSY
      58 -          C      B         A                     -  B: CPU 1
   17:11 -             C                 AB                -     CPU BUSY
      24 -         C                     A   B             -  C: CPU 2
16:17:36 -         C         +         +       AB+         -     CPU BUSY
      49 -                  C            A    B            -  D: CPU 3
   18:01 -           C                  A        B         -     CPU BUSY
      14 -          C                    A       B         -
      26 -              C            A   B                 -
16:18:39 -         +         C         +A      B +         -
      52 -                  C           A     B            -
   19:04 -                     C     B A                   -
      17 -                           B CA                  -
      29 -                     C    B    A                 -
16:19:42 -         +       C +       B +  A      +         -
      54 -           C              B      A               -
         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
```

```
>  Note that the legend at the right lists CPU 3, to agree with the  <
>  system configuration.  However, since CPU 3 was down at the time  <
>  of the measurement, no data was collected.                        <
>                                                                    <
>  Our first command obtains a report of the process $SPOOL.        <
```

```
+process $spool

PROCESS:   $SPOOL     CPU *    PIN *            AUG 12  16:16:23 FOR  184

               $SPOOL     $SPOOL

CPU BUSY         .880       34.4

MSG RATE                    5.22

RECV RATE        2.04       34.2

RECV QLEN        .069       .116

FAULT RATE       .027

PRES PAGES       29.0       72.0

BLKD WAIT        .122       3.13
```

```
>  It seems as though there are two processes with this name.          <
>  Recalling that the Spool Collector runs NonStop, we realize that    <
>  we are looking at the primary and its backup, and that the busier   <
>  one must be the primary.  To disregard the backup, we request a     <
>  report of all processes named $SPOOL, subject to the requirement       <
>  that the CPU BUSY item must exceed 1.                                  <

+process $spool, if cpu busy > 1

$MKT.XRAYDATA.XDATA7        AUG 12  16:16:23 FOR  184

PROCESS:  $SPOOL     CPU *    PIN *       AUG 12  16:16:23 FOR  184

$SYSTEM.SYSTEM.CSPOOL $SPOOL   CPU 1    PIN AUG 12  16:16:23 FOR  184

CPU   MSG   MBYTE RECV  RECV  VBYTE YBYTE CHKPT FAULT PRES  VSEM  BLKD
BUSY  RATE  RATE  RATE  QLEN  RATE  RATE  RATE  RATE  PAGES RATE  WAIT

34.4  5.22  6344  34.2  .116  5014        2.04        72.0        3.13

>  Whenever a report consists of only one entity, all of its items        <
>  are displayed horizontally.  As you can see, the horizontal            <
>  reports omit certain items in order to fit as much as possible on      <
>  the screen.                                                            <
>                                                                         <
>  Numbers that are missing, such as FAULT RATE, are zero.                <
>                                                                         <
>  The RECV RATE indicates the number of messages per second that         <
>  $SPOOL obtained via its $RECEIVE file.  We will now plot this          <
>  item against time.  First, we add it to the current plot.              <

+plot recv rate

>   The current plot now includes the RECV RATE of $SPOOL.  Let's         <
>   take a look at it.                                                    <

+plot

$MKT.XRAYDATA.XDATA7        AUG 12  16:16:23 FOR  184

         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
16:16:34 -A                                                - A: $SPOOL
16:16:44 A         +         +         +         +         -    RECV RATE
      54 -      A                                          - 14.7
   17:04 -           A                                     - 24.6
      14 -               A                                 - 32.7
      24 -                     A                           - 44.8
16:17:34 -         +         A         +         +         - 39.6
      44 -                 A                               - 36.5
      54 -                   A                             - 40.9
   18:04 -                       A                         - 48.3
      14 -                       A                         - 47.4
16:18:24 -         +         A         +         +         - 39.5
      34 -                    A                            - 41.9
      44 -                    A                            - 42.4
      54 -                      A                          - 45.7
   19:04 -                  A                              - 38.0
16:19:14 -         +       A +         +         +         - 36.3
      24 -                  A                              - 38.6
         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
```

```
> The plot shows, for example, that between 16:17:34 and 16:17:44   <
> $SPOOL averaged 36.5 messages per second on its $RECEIVE file.    <
> Now we want to look at the TAL compilation.  TAL was run as an    <
> unnamed process, so we specify its program file name.             <

+process $system.system.tal

$SYSTEM.SYSTEM.TAL             CPU 0   PIN AUG 12  16:16:25 FOR  182

CPU   MSG   MBYTE RECV  RECV  VBYTE YBYTE CHKPT FAULT PRES  VSEM BLKD
BUSY  RATE  RATE  RATE  QLEN  RATE  RATE  RATE  RATE  PAGES RATE WAIT

51.2  39.1  6220                                .355  62.5       3.62

> The MSG RATE is the number of messages per second initiated by    <
> this process.  We add TAL's MSG RATE to the current plot.         <
>                                                                   <
> It is worth noting here that the IF condition in PROCESS CPU      <
> BUSY, specified earlier, remains in effect.  Thus if TAL had      <
> been less than 1% busy, it would not have appeared in response    <
> to the previous command.                                          <

+plot msg rate

> Note that we don't need to specify the process for which the      <
> message rate is to be plotted.  Specifiying TAL in the previous   <
> report also defines TAL as the "current entity", to which any     <
> subsequent PLOT command refers.                                   <
>                                                                   <
>  Let's take a look at the current plot.                           <

+plot
```

```
$MKT.XRAYDATA.XDATA7          AUG 12  16:16:23 FOR  184

         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
16:16:34 -A      B                                         -  A: $SSPOOL
16:16:44 A       B +         +         +         +         -     RECV RATE
      54 -      A   B                                      -  B: TAL
   17:04 -           A   B                                 -     MSG RATE
      14 -               A    B                            -
      24 -                     AB                          -
16:17:34 -         +         AB        +         +         -
      44 -                 AB                              -
      54 -                   A B                           -
   18:04 -                       AB                        -
      14 -                       AB                        -
16:18:24 -         +         AB        +         +         -
      34 -                    AB                           -
      44 -                    AB                           -
      54 -                     AB                          -
   19:04 -                  AB                             -
16:19:14 -         +       AB+         +         +         -
      24 -                  A B                            -
         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
```

```
>  Now let's manipulate the plot to bring out a few additional      <
>  features of XRAYSCAN.  First, note that the horizontal axis runs <
>  from 0 to 100, but the activity of interest, for the most part,  <
>  takes on values between 20 and 60.  We can obtain more detail by <
>  adjusting the horizontal axis with a SCALE command.              <

+scale 20,60

>  Let's look at the plot now.                                      <

+plot
```

```
      20.0::::+:28.0::::+:36.0::::+:44.0::::+:52.0::::+:60.0
16:16:34 A                                                 -  A: $SSPOOL
16:16:44 A         +         +         +         +         -     RECV RATE
      54 A  B                                              -  B: TAL
   17:04 -     A         B                                 -     MSG RATE
      14 -               A           B                     -
      24 -                              A B                -
16:17:34 -         +         +    A B  +         +         -
      44 -                   A B                           -
      54 -                        A  B                     -
   18:04 -                                A  B             -
      14 -                                 A   B           -
16:18:24 -         +         +   A   B +         +         -
      34 -                          A  B                   -
      44 -                           A  B                  -
      54 -                               A  B              -
   19:04 -                      A  B                       -
16:19:14 -         +         A  B      +         +         -
      24 -                      A   B                      -
      20.0::::+:28.0::::+:36.0::::+:44.0::::+:52.0::::+:60.0
```

```
> The MSG RATE of TAL and the RECV RATE of $SPOOL are indeed very   <
> closely correlated.  Note that TAL always has a little more       <
> message activity than $SPOOL, because the MSG RATE item includes  <
> TAL's read operations from disc.                                  <
>                                                                   <
>  We can restrict XRAYSCAN's attention to a subset of the data     <
>  file using a WINDOW command.  In particular, we will limit       <
>  XRAYSCAN to the period of time between 16:16:54 and 16:18:17.    <

+window 16:16:54, 16:18:17
+plot

$MKT.XRAYDATA.XDATA7        AUG 12  16:16:54 FOR 83.0

      20.0::::+:28.0::::+:36.0::::+:44.0::::+:52.0::::+:60.0
16:17:04 -      A         B                                  -  A: $SPOOL
16:17:14 -          +     A   +       B +         +          -    RECV RATE
      24 -                               A B                 -  B: TAL
      34 -                         A B                       -    MSG RATE
      44 -                     A B                           -
      54 -                          A  B                     -
16:18:04 -          +         +         +   A  B +           -
      14 -                                 A   B             -
      20.0::::+:28.0::::+:36.0::::+:44.0::::+:52.0::::+:60.0

> Now let's take a look at another aspect of the situation under    <
> measurement, namely the activity on TAL's source file, which is   <
> named $MKT.TEST.SOURCE, as opposed to the activity on the disc    <
> that contains the source file.                                    <

+file $mkt.test.source

$MKT.TEST.SOURCE        PID 0, 57     FNUM 6    AUG 12  16:16:54 FOR 83.0

                     FILE  READ  WRITE UPDT  DOWR INFO
                     RATE  RATE  RATE  RATE  RATE RATE

                     1.38  1.38

> XRAY considers each individual opening of a file to be a separate <
> entity.  Thus, the first line of the report shows that we are     <
> looking at statistics for the opening of this file by the process <
> identified by PID 0,57, which is TAL.  It's not surprising that   <
> the FILE RATE is the same as the READ RATE; for an input file,    <
> these quantities are the same.                                    <
>                                                                   <
> Let's start a brand new plot, and then plot the READ RATE.        <

+newplot
+plot read rate
+plot
```

```
$MKT.xraydata.xdata7        AUG 12  16:16:54 FOR 83.0

          20.0::::+:28.0::::+:36.0::::+:44.0::::+:52.0::::+:60.0
16:17:04 A                                                       -  A: SOURCE
16:17:14 A             +         +         +         +           -     READ RATE
      24 A                                                       -  1.70
      34 A                                                       -  1.70
      44 A                                                       -  1.60
      54 A                                                       -  1.80
16:18:04 A             +         +         +         +           -  2.00
      14 A                                                       -  1.50
          20.0::::+:28.0::::+:36.0::::+:44.0::::+:52.0::::+:60.0
```

```
>  The NEWPLOT command starts a new plot, but the SCALE remains as   <
>  it was.  In this case, the SCALE is entirely inappropriate to the <
>  data being plotted, so we redefine it.                            <

+scale 1,5
+plot
```

```
          1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00
16:17:04 A                                                       -  A: SOURCE
16:17:14 A             +         +         +         +           -     READ RATE
      24 -            A                                          -  1.70
      34 -            A                                          -  1.70
      44 -           A                                           -  1.60
      54 -             A                                         -  1.80
16:18:04 -             +  A      +         +         +           -  2.00
      14 -         A                                             -  1.50
          1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00
```

```
>  Much better.  Now we want to plot the READ RATE of the disc $MKT. <

+disc $mkt

>  Normally this would generate a report, which we have omitted for  <
>  the sake of brevity.                                              <

+plot read rate
+plot
```

```
 $MKT.xraydata.xdata7        AUG 12  16:16:54 FOR 83.0

      1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00
16:17:04 A                                            B     -  A: SOURCE
16:17:14 A          +         +         +         +         B     READ RATE
      24 -         A                                        B  B: $MKT
      34 -         A                                        B     READ RATE
      44 -        A                                         B
      54 -          A        B                              -
16:18:04 -          +  A      +    B    +         +         -
      14 -      A           B                               -
      1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00

> As expected, the READ RATE of the disc is larger than, but fairly <
> closely correlated with, the READ RATE of a particular file on    <
> the disc.                                                         <
>                                                                   <
> Using a WINDOW command, we can take a better look at the period   <
> of time starting at 16:17:54.                                     <

+window 16:17:54

> By specifying only the beginning of the time interval that is of  <
> interest, the end remains unchanged.                              <

+plot


 $MKT.xraydata.xdata7        AUG 12  16:17:44 FOR 33.0

      1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00
16:17:54 -          A        B                              -  A: SOURCE
16:18:04 -          +  A      +    B    +         +         -     READ RATE
      14 -      A           B                               -  B: $MKT
      1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00


> We can obtain more detail with a DELTA command, which adjusts the <
> scale of the time axis.                                           <

+delta 3

> This command specifies 3-second intervals.                        <
```

```
$MKT.xraydata.xdata7        AUG 12  16:17:44 FOR 33.0

        1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00
16:17:47 -             A       B                             -  A: SOURCE
16:17:50 -    A      + B       +         +         +         -     READ RATE
      53 -        A            B                             -  B: $MKT
      56 -             A       B                             -     READ RATE
      59 -             A           B                         -
   18:02 -                 A       B                         -
16:18:05 -           + A       +       B+          +         -
      08 -        A        B                                 -
      11 -        A                B                         -
      14 A        B                                          -
      17 -                 A           B                     -
        1.00::::+:1.80::::+:2.60::::+:3.40::::+:4.20::::+:5.00
```

It should be noted that the data file, XRAYDATA, can be saved indefinitely. Thus, you can perform a measurement that lasts an hour, and examine the results over a period of weeks. As you become more familiar with the results of a particular measurement, your questions will become more meaningful, and XRAY's usefulness will increase.

XRAYCOM

XRAYCOM

Starting a measurement is a 4-step process:

1. Decide what entities you want to measure.

2. Create a configuration file. The configuration file is an Edit-format file containing a list of the entities to be measured.

3. Create a data file. This is an unstructured disc file; its size depends on how big your measurement will be.

4. Run XRAYCOM to specify the data and configuration files and start the measurment.

XRAYCOM allows you to change the measurement configuration at any time by simply specifying a new configuration file. You can also change the data file (the effect of this, however, is to actually define a new measurement).

Let's consider the above four steps in detail.

What do You Want to Measure?

You have to decide this for yourself, based on your experience with XRAY and the problem at hand. If you're trying to balance your day-to-day workload over several processors, it may be sufficient to look at only the processors. If you want to structure a transaction-processing application to run efficiently, you'll have to measure the application and all of the system components it uses.

It can be hard, at first, to know what to measure. The "shotgun" approach of selecting everything in the system for measurement is equivalent to exploratory surgery in medicine. When you're really in the dark, a more successful approach--and almost as informative--is to select all devices, processors, and processes for measurement, along with certain terminals. Then sets of disc files can be measured on an as-needed basis. (The configuration can be expanded or decreased at any time during the measurement.)

The Configuration File

Once you've decided what to measure, creating a configuration file involves nothing more than listing the entities to be measured in an Edit file. The format of the configuration file is described later in this section, immediately after the the CONF command description.

The Subentry Point File

The subentry point file is required only if procedure subentry points are to be measured. Like the configuration file, this is an Edit file. The format of the subentry point file is described later in this section, after the configuration file description.

The Data File

This is an unstructured disc file into which the Recorders put the measurement data. It is created with FUP or with the Command Interpreter CREATE command. The only consideration is: How big should it be?

How Big Should a Data File Be?

No exact formula can be given to determine ahead of time how much data will be collected by a given measurement configuration. This quantity depends on factors (such as the number of file opens or the number of processes to be run) that are outside the control of the individual doing the measurement,

However, we can make a reasonable estimate based on

- the number of processors under measurement
- the general size (null, medium or large) of the measurement configuration
- the GO interval (this is the interval, in seconds, between successive entries in the data file)
- the total length of time of the measurement

Specifically, if

n = the number of processors under measurement,

t = the total time of the measurement, and

g = the GO interval,

the total number of bytes you should allocate for your data file is

```
n * t
----- * multiplier
  g
                              { 512 bytes for a null configuration
where:   multiplier =         { 2048 bytes for a medium configuration
                              { 4096 bytes for a large configuration
```

This formula is derived from the fact that each Recorder (one in each processor) writes <multiplier> bytes to the data file each GO interval.

An example of a medium configuration would be the measurement of all system processes and all devices. A large configuration would be the measurement of all processes, all devices, and several disc file sets.

Remember that this is a guideline, not an exact formula. Once your measurement is in process, it is a good idea to do a periodic "FUP INFO" on the data file to see if you're running out of room.

It should be noted that XRAY is capable of generating very large amounts of data in a short time. A medium-sized measurement configuration with a GO interval of 1 second, for example, will generate one megabyte of data in a little over eight minutes.

RUNNING XRAYCOM

XRAYCOM (for "XRAY COMmunication") specifies the data and configuration files, and starts or stops a measurement.

XRAYCOM resides in a file named $SYSTEM.SYSTEM.XRAYCOM. The command to run XRAYCOM is:

```
XRAYCOM [ / [ IN <command file> ] [ , OUT <list file> ] / ]

where:

   <command file>

      designates the file from which XRAYCOM reads its commands.
      If omitted, XRAYCOM accepts commands from the Command
      Interpreter's home terminal.

   <list file>

      designates a file to receive XRAYCOM's output.  If omitted,
      XRAYCOM writes to the Command Interpreter's home terminal.
```

XRAYCOM immediately establishes communication with each processor's Recorder, and creates a Recorder in any processor lacking one. As soon as this task is completed, XRAYCOM signals the user with an OK message, listing each OKAY processor (an "OKAY" processor is one in which a Recorder successfully executed the command; failures if any are reported in a FAILED list of processors.)

XRAYCOM then prompts the user with a plus sign (+). XRAYCOM has five commands:

CONF	specifies a configuration file.
DATA	specifies a data file. The form DATA! stops the measurement and closes the data file.
EXIT	exits from XRAY. The form EXIT! terminates the current measurement, returns measurement space to the system, and stops the Recorders.
GO	starts a measurement, and specifies a GO interval.
LIGHTS	causes each processor's switch register display to indicate, each second, the percentage of time that it was busy during the previous second.

The CONF command designates a measurement configuration file.

The form of the CONF command is:

```
--------------------------------------------------------------------
|                                                                  |
|  CONF  <configuration file>                                      |
|  ----  --------------------                                      |
|                                                                  |
|  where:                                                          |
|                                                                  |
|     <configuration file>                                         |
|                                                                  |
|        is an Edit format file, unstructured disc file, device,   |
|        or process that specifies the entities to be measured     |
|                                                                  |
|  example:                                                        |
|                                                                  |
|     CONF xrayconf                                                |
|                                                                  |
--------------------------------------------------------------------
```

<u>CONSIDERATIONS</u>

* You can issue a CONF command at any time, even during a measurement. The new configuration takes effect immediately, without the need to issue any other commands. For example:

```
   +DATA xraydata
   +CONF conf1
   +GO 10          ! Starts the measurement that has its
                   ! configuration specified in the file CONF1.
   +CONF conf2     ! Immediately following this command, the
                   ! measurement specified in CONF2 takes effect.
```

```
PROCESSORS:

   <cpunum>

      ! The Recorder in the cpus listed will emit data to
      ! the output file.  (If this section is omitted, all
      ! cpus will collect data.)

BUFFERS:

   $<volume>.<subvol>.<filename>
     ! Measure buffer usage by all executions of this program.

   $<volume>.<subvol>.*
     ! Measure buffer usage by all programs executing on this
     ! subvolume.

   $<volume>.*.*
     ! Measure buffer usage by all programs executing on this
     ! volume.

   $<process name>
     ! Measure buffer usage by all processes with this name.

   <cpu>,<pin>
     ! Measure buffer usage by this process.

   SYSPROCS
     ! Measure buffer usage by all system processes.

PROCEDURES:

   <program file name> <processes>
     ! Where <program file name>  specifies the program that is to
     ! be measured by procedure (entry point).  A counter is
     ! reserved for each entry point and during execution, the P
     ! register value is used to determine which counter to bump.
     ! <program file name> is in the form
     ! $<volume>.<subvol>.<filename>.
     !
     ! <processes> specifies the executions of the procedure to
     ! be measured:
     !
     !     <process name> specifies all executions of the pair.
     !     <cpu>,<pin> specifies this particular process.

EXCLUDE:

                                                          -->
```

```
----------------------------------------------------------------------
|                                                                    |
|                                                                    |
|  <any form permitted in the PROGRAMS or FILES section>             |
|                                                                    |
|  ! entities listed in this section will NOT be measured.           |
|                                                                    |
|                                                                    |
----------------------------------------------------------------------
```

The following considerations apply to the configuration file:

* Disc files listed in the FILES section are measured on two levels:
  - user calls to file management procedures
  - calls by local disc I/O processes to the driver
* Only the name supplied in the user's call to OPEN must be specified in the FILES section. In particular, when measuring a file with alternate keys, only the primary key file must be listed. When measuring a partitioned file, only the first partition must be listed.
* File openings by the System Monitor process, or by processes running at the same priority as the memory manager (or higher), will not be measured.

The configuration file is a free-format list of keywords and entity descriptions; the entire file is considered by XRAY to be one long line (except that words can't be split across records). Thus, the configuration file consisting of the text lines

```
PROCESSORS:  0  1
DEVICES:  ASYNC
```

and the file consisting of the single line

```
PROCESSORS:  0  1  DEVICES:  ASYNC
```

are equivalent.

An exclamation point (!) occurring anywhere in a record causes the remainder of the record to be ignored by XRAY. Thus, lines in a configuration file can contain explanatory information; for example:

```
PROGRAMS:  $system.system.tal  ! Measure all TAL compilations.
```

(A record, for an Edit file, consists of one line of text).

Measurement of buffer pool activity (BUFFERS) takes place at the CPU level and the process level. At the CPU level the buffer pools are measured. At the process level the buffer utilization by the process is measured.

The buffer pools that are measured are the shortpool, longpool, control block space, iopool, discpool, and link control blocks.

When measuring procedures, some selectivity in monitoring must be exercised. The measurement uses 2 to 3 words of longpool space per entry point per measured process. The pool can be saturated if care is not taken. The Recorder always leaves at least 256 words of longpool space for use by other processes.

Measuring at the Procedure level may not be sufficient in some cases. In particular, COBOL entry points identify the program units and it may be necessary to sample at the paragraph level. Additionally, there are other cases where the PEP is an inadequate mesh for sampling. For example, sampling of processes composed of subprocedures. In these circumstances, a Subentry Point List is used. A Subentry Point List is a list of entry points and addresses along with the subentry point names and addresses associated with the entry point. When this list is used, the <program file name> in the PROCEDURES section of the Configuration File points to an edit file containing the entry and subentry points. During execution, the <program file name> is replaced with the Subentry Point List.

The format of the Subentry Point List follows:

```
                       SUBENTRY LIST FORMAT

 <entry point name>  <entry point address>:
       <subentry point name> <subentry point address>
       <subentry point name> <subentry point address>
       <subentry point name> <subentry point address>
                 .                        .
                 .                        .
                 .                        .
 <entry point name>  <entry point address>:
       <subentry point name> <subentry point address>
                 .                        .
                 .                        .
                 .                        .
   .
   .
   .

    ! Address are all in octal. The <subentry point address>s
    ! are relative to the preceding <entry point address>
    ! starting at zero.
```

Null Configuration Files

A null configuration file is a file containing no entity descriptions (such as an Edit file consisting solely of blank lines). Omitting the CONF command when configuring a measurement, or supplying a null configuration file, results in the "null configuration." The null configuration measures nothing but processor statistics, except response time and transaction rate, in all processors.

If a non-null configuration is currently installed, a subsequent null configuration releases all measurement space acquired from LONGPOOL.

Non-Edit Configuration Files

The configuration file is almost always an Edit-format disc file, but this is not mandatory. If the configuration file is an unstructured file or a non-disc device, XRAY reads 132-byte records until EOF is detected. If the configuration file is a process, XRAY prompts the process and reads 132-byte records, using WRITEREAD; the process sending the configuration information should READUPDATE and REPLY to its $RECEIVE file, in accordance with the usual conventions for interprocess communication.

The Home Terminal as Configuration File

You can name the same terminal from which you're running XRAYCOM as the configuration file. If you do, XRAYCOM will issue a double prompt (++), at which time you can enter the configuration information one line at a time. Signal the end of the configuration by entering control-Y.

DATA Command

The DATA command specifies the file in which measurement data is to be stored; it also closes any previous data file.

The form of the DATA command is:

```
---------------------------------------------------------------------
|                                                                   |
|  DATA  { <data file name> | ! }                                   |
|  ----                                                             |
|                                                                   |
|  where:                                                           |
|                                                                   |
|     <data file name>                                              |
|                                                                   |
|        is the name of the file in which measurement data is to be |
|        stored.                                                    |
|                                                                   |
|     !  Closes the current data file and deallocates the           |
|        measurement space, but does not open a new data file.      |
|        The Recorders continue to run, and if LIGHTS had been      |
|        previously specified, the lights continue to run.          |
|                                                                   |
|  example:                                                         |
|                                                                   |
|     DATA xraydata                                                 |
|                                                                   |
---------------------------------------------------------------------
```

CONSIDERATIONS

* A GO command must always follow a DATA command to start a measurement.

* The DATA command performs the following tasks, in order:

  - Stops the current measurement.

  - Installs a null configuration.

  - Closes the current data file.

  - Opens the new data file (if one was specified).

  - Installs the latest configuration file, if one exists. The "latest configuration file" is defined as the file named in the most recent CONF command.

  BE CAREFUL WHEN CHANGING THE DATA FILE IN THE MIDDLE OF A MEASUREMENT.

  In view of the above sequence of events, running XRAYCOM and issuing DATA and GO commands will result in a NULL configuration; to keep the same configuration file, a CONF command must be issued

A02

whenever XRAYCOM is run for the purpose of changing the data file. Consider the following examples:

example 1

```
:XRAYCOM
+DATA X
+CONF XRAYCONF
+GO 10
+DATA Y
+GO 10
```

According to the sequence of events that takes place when a DATA command is issued, the command DATA Y installs the latest configuration file, which is defined as the file named in the most recent CONF command; therefore, the configuration in XRAYCONF is installed. Compare this example with the following one:

example 2

```
:XRAYCOM
+DATA X
+CONF XRAYCONF
+GO 10
+EXIT

:XRAYCOM
+DATA Y
+GO 10
```

Example 2 differs from example 1 in that, after starting the measurement, XRAYCOM was EXITed and run a second time to change the data file. The command DATA Y installed a null configuration. Since <u>no other configuration file was defined</u> during this run of <u>XRAYCOM, the subsequent GO command started</u> a null measurement.

The proper way to change the data file from X to Y without losing the configuration file is:

```
:XRAYCOM
+CONF XRAYCONF
+DATA Y
+GO 10
```

* When the file can be opened exclusively by Xraycom, the DATA command causes measurement results to be written starting at the beginning of the file. Therefore, supplying the DATA command with the name of an already existing data file causes the previously collected data to be overwritten if the file is not in use by the Recorders or Xraycom.

* You may issue a DATA command at any time during a measurement to switch to a new data file. Don't forget to re-specify the

configuration file (if necessary).

* Following a DATA command, measurement does not begin until a GO command is issued. For example:

```
+DATA datafile       ! Specifies DATAFILE as the data file.
+GO 10               ! Starts measurement.
 .
 .
+DATA xraydata       ! Stops measurement and specifies XRAYDATA
                     ! as the data file.

+GO 20               ! Starts measurement.
```

* The DATA! command allows you to stop a measurement without also stopping the panel lights.

* The data file may be a magnetic tape unit. Keep in mind the following:

  - Measurement stops when the end-of-tape mark is encountered.

  - Because of the I/O processes involved, using a data file on tape involves less overhead than using a disc file. This is important for high-resolution measurements.

  - XRAYSCAN requires that the data file be on disc; therefore, you must use FUP to copy the tape file to disc before running XRAYSCAN on that file.

  - There is no opportunity to write end-of-file marks after data is collected on tape. Tape runaway or other errors may result when trying to transfer the data to a disc file. (This problem can be avoided by making sure the disc file will not hold all the data collected on a tape.

  - Maximum tape block size is 2048 bytes.

  - A processor that halts during measurement can be re-included in the measurement with the following steps:

    1) Stop Xraycom with an EXIT command.

    2) Reload the processor.

    3) Run Xraycom. A Recorder will be created inn the restarted processor.

    4) Reissue the DATA, CONF, and GO commands for the measurement.

The GO command starts data collection.

The form of the GO command is:

```
----------------------------------------------------------------------
|                                                                    |
|  GO <interval>                                                     |
|  -- ----------                                                     |
|                                                                    |
|  where:                                                            |
|                                                                    |
|     <interval>,                                                    |
|                                                                    |
|        {1:3600}, specifies the interval, in seconds, at which      |
|        measurement data is recorded.                               |
|                                                                    |
|  example:                                                          |
|                                                                    |
|     GO 1     ! Take a measurement every second.                    |
|                                                                    |
----------------------------------------------------------------------
```

<u>CONSIDERATIONS</u>

* A small interval, causing frequent measurements, leads to high resolution at the cost of relatively high overhead from XRAY itself, but is appropriate when a detailed analysis is desired. A small interval could be used to answer the question, "How does running process X impact the system?"

* A large interval, causing infrequent measurements, leads to low resolution and low overhead, and is appropriate when a summary analysis is desired, or when a long-term study is in progress. A large interval could be used to answer the question, "At what time of day is CPU 5 busiest?"

EXIT Command

The EXIT command exits from XRAYCOM, and optionally terminates the measurement in progress.

The form of the EXIT command is:

```
-----------------------------------------------------------------------
|                                                                     |
|  EXIT [ ! ]                                                         |
|  ----                                                               |
|                                                                     |
|                                                                     |
|     Supplying "!" exits from XRAYCOM, terminates the current        |
|     measurement, and stops the Recorders.                           |
|                                                                     |
|     Omitting "!" simply exits from XRAYCOM.                         |
|                                                                     |
|  example:                                                           |
|                                                                     |
|     EXIT                                                            |
|                                                                     |
-----------------------------------------------------------------------
```

<u>CONSIDERATIONS</u>

* The EXIT! command returns all dynamically-acquired measurement space to LONGPOOL, closes the data file, and stops all Recorders. (About one sector of measurement space is permanently allocated).

  Note that stopping the Recorders also stops the CPU BUSY display on the front panel lights.

A02

The LIGHTS command causes the panel lights (also known as the switch register display) to indicate the processor usage during the previous second.

The form of the LIGHTS command is

```
---------------------------------------------------------------------
|                                                                   |
|  LIGHTS                                                           |
|  ------                                                           |
|                                                                   |
---------------------------------------------------------------------
```

CONSIDERATIONS

* Once the lights are turned on, you can't turn them off unless you also stop the measurement with an EXIT! command.

* The DATA! command stops the current measurement without turning off the lights.

THIS MEASUREMENT CANNOT BE CONFIGURED BECAUSE THE DATA FILE IS NOT OPEN

The DATA file must be open to configure PROCEDURE measurements. Issue DATA against the current DATA file and proceed.

```
-----------------------------------------------------------------------
|                                                                     |
|  XRAYCOM [ / [ IN <command file> ] [ , OUT <list file> ] / ]        |
|                                                                     |
|  CONF  <configuration file>                                         |
|  ----  --------------------                                         |
|                                                                     |
|     <configuration file> is                                         |
|                                                                     |
|        PROGRAMS:  [ <program specifier>  ... ]                      |
|                                                                     |
|           <program specifier> is                                    |
|                                                                     |
|              { $<volume>.<subvol>.<filename> }                      |
|              { $<volume>.<subvol>.*          }                      |
|              { $<volume>.*.*                 }                      |
|              { $<process name>               }                      |
|              { <cpu, pin>                    }                      |
|              { SYSPROCS                      }                      |
|                                                                     |
|        FILES:  [ <file specifier>   ... ]                           |
|                                                                     |
|           <file specifier> is                                       |
|                                                                     |
|              { $<volume>.<subvol>.filename            }             |
|              { $<volume>.<subvol>.*                   }             |
|              { $<volume>.*.*                          }             |
|              { $<volume>.#tempnum                     }             |
|              { $<device name>                         }             |
|              { \<sysname>.$<volume>.<subvol>.filename }             |
|              { \<sysname>.$<volume>.<subvol>.*        }             |
|              { \<sysname>.$<volume>.*.*               }             |
|              { \<sysname>.$<volume>.#tempnum          }             |
|              { \<sysname>.$<device name>              }             |
|              { \<sysname>.*.*.*                       }             |
|              { $<linename>.*                          }             |
|              { $<linename>.#<subname>                 }             |
|                                                                     |
|        DEVICES:  [ <device name>   ... ]                            |
|                                                                     |
|           <device name> is  { $<device name>                }       |
|                             { $<logical device number> }            |
|                             { ASYNC                         }       |
|                             { $<linename>.*                 }       |
|                             { $<linename>.#<subname>        }       |
|                             { $<linename>                   }       |
|                                                                     |
|        SYSTEMS:  [<system name>   ... ]                             |
|                                                                     |
|        PROCESSORS:  <cpunum> ...                                    |
|                                                                     |
|                                                                 --> |
-----------------------------------------------------------------------
```

```
        BUFFERS: <program set>

           <program set> is
              { $<volume>.<subvol>.<filename> }
              { $<volume>.<subvol>.*          }
              { $<volume>.*.*                 }
              { $<process name>               }
              { <cpu>,<pin>                   }
              { SYSPROCS                      }

        PROCEDURES: <program  file name> <processes>

           <processes> is a set of
              { <process name>   }
              { <cpu>,<pin>      }

        EXCLUDE:  <entities to be exluded> ...

  DATA  { <data file name> | ! }
  ----

  GO <interval>
  -- ----------

  EXIT [ ! ]
  ----

  LIGHTS
  ------
```

XRAYSCAN

XRAYSCAN

The XRAYSCAN program displays the data that has been collected during a measurement. This data consists of statistics on various items relating to a set of measured entities. For example, the data file contains statistics on the items CPU BUSY and SWAP RATE for the entity processor, and so on.

XRAYSCAN has three ways of displaying the collected statistics: reports, plots, and histograms. Reports are simply lists of items relating to various entities. Plots show the items plotted against time. Histograms show the relative distribution between items. Here are typical examples of each.

A Typical Report

```
CPU: *                                      MAY 22 02:51:39 FOR 95.8

               CPU  0    CPU  1

CPU BUSY        28.5      42.7

SWAP RATE                 1.96

DISC RATE       11.5

CHIT RATE       1.25

TRAN RATE       .020

RESP TIME
```

The above report was generated by the XRAY command

```
   +CPU         ! XRAY prompts with a plus sign (+).
```

This command displays entities for all processors under measurement.

The first line tells which entities are being displayed--in this case CPU *, meaning "all processors."

On the same line are the date and time the statistics were collected, and the number of seconds (95.8) the measurement lasted. Thus, this report displays statistics collected over a period of 95.8 seconds, starting at 2:51:39 a.m. on May 22.

The measured entities (processors 0 and 1) are listed at the top of the report. On the left side are several items of information about each processor. For example, the report tells us that during the time of the measurement, processor 0 was busy 28.5% of the time, and that processor 1 had an average of 1.96 virtual memory swaps per second.

A Typical Plot

```
         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
02:51:45 -   B   A                                         - A: CPU 0
      52 -  B  A                                           -    CPU BUSY
      58 -            B     A                              - B: CPU 1
02:52:05 -         +  A      B         +         +         -    CPU BUSY
      11 -                  A   B                          -
      18 -                  AB                             -
      25 -           A               B                     -
      31 -               A         B                       -
02:52:38 -         A         +         +    B    +         -
      44 -        A                      B                 -
      51 -                A    B                           -
      57 -             A                   B               -
   53:03 -               A             B                   -
02:53:10 -         +     BA  +         +         +         -
      16 -            B      A                             -
      23 - B A                                             -
      30 -  BA                                             -
         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
```

Above is a plot of the CPU BUSY item for the same two processors over the same time interval.

The time axis is on the left, with time increasing as you read down the page. The horizontal axis is the percentage of time the processor was busy.

The text at the right is the legend, which defines the correspondence between the plotted letters and the measured entities. Each legend item consists of two lines: The first line is the measured entity and the second line is the item displayed. In the above plot, "A" represents the CPU BUSY item for the entity CPU 0. For example, it shows that between the times 2:53:23 and 2:53:30, CPU 0 was 8% busy.

The plus signs in the display are aligned on the last digits of the numbers in the top and bottom margins (the actual locations of plotted values 20, 40, 60, and 80) and are an aid to sighting. The plus signs in the top and bottom margins represent the values 10, 30, 50, etc.

XRAY allows you to build plots of various items. For example, you may be interested in seeing CPU BUSY for processor 1 in the same plot with LINE RATE for a data communication line. The remainder of this section describes how to use XRAY to examine the data file, displaying reports and building plots.

A02

A Typical Histogram

```
$SYSTEM.SUBV.XYZ   $XYZ   CPU 2   PIN 65  JUL 15  10:45:35 FOR 129

                0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100

       EDITWRT *************** 28.8                                - 28.8

          PARS *********** 17.7                                    - 46.6

           APP ******** 13.3        +          +          +        - 59.9

         SCANR **** 6.66                                           - 66.6

       RD^FILE **** 6.66                                           - 73.3

      WRT^FILE **** 6.66                                           - 79.9

     STR^VALUE *** 4.44  +          +          +          +        - 84.4

  SRCE^COMMENT *** 4.44                                            - 88.8

     DO^CMMNT *** 4.44                                             - 93.3

PROCESS^COMND ** 2.22                                              - 95.5

   RESERVED^WD ** 2.22   +          +          +          +        - 97.7

   DO^TAL^OPUT ** 2.22                                             - 99.9

MAP BUSY 9.24  0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
```

The preceding is a histogram of the entry points of a process $XYZ.

The horizonal axis is marked off in percentages (0 to 100). The percentages for each entry point indicate a portion of the total cpu time used by the measured process. The horizonal axis can be adjusted with the SCALE command.

The figures down the right side of the histogram are the cumulative percentage of cpu used for each entry point. The cumulative may not equal a full 100 percent because of truncation (for example, the sample only reaches 99.9 percent.

Those entry points (if any) that were not sampled during the measurement period are not included in the histogram.

The fraction of the total cpu time used by the subject process is available from the PROCESS report of the process. The figure in the lower left hand corner to the left of the scale and labeled MAP BUSY gives the fraction of the process's cpu time spent in the sampled map. The MAP BUSY figure is a percent in the range of 0 to 100.

XRAY resides in a file designated "$SYSTEM.SYSTEM.XRAYSCAN", and is run with the command:

```
---------------------------------------------------------------------
|                                                                   |
|  XRAYSCAN                                                         |
|  --------                                                         |
|     [ / [ IN <in file> ] [ , OUT <out file> ] / ] <data file>     |
|                                                   -----------     |
|                                                                   |
|  where:                                                           |
|                                                                   |
|     <in file>                                                     |
|                                                                   |
|        is the file from which XRAYSCAN reads its commands.        |
|        Omitting IN causes XRAYSCAN to prompt the Command          |
|        Interpreter's home terminal for commands.                  |
|                                                                   |
|     <out file>                                                    |
|                                                                   |
|        is the file to which XRAYSCAN writes its reports and plots.|
|        Omitting OUT causes XRAYSCAN to write to the Command       |
|        Interpeter's home terminal.                                |
|                                                                   |
|     <data file>                                                   |
|                                                                   |
|        contains XRAY measurement data.  This file may be from a   |
|        previous experiment or from an experiment in progress (see |
|        "Online Measurement," later in this section).              |
|                                                                   |
|  example:                                                         |
|                                                                   |
|     XRAYSCAN xraydata                                             |
|                                                                   |
---------------------------------------------------------------------
```

When XRAYSCAN is initially run, it displays a plot of CPU BUSY for each processor under measurement. After this plot is completed, a prompt (+) appears, and you can begin examining the data by looking at various reports and building plots.

The remainder of this section is organized as follows:

- Generating reports
- Building plots
- XRAYSCAN commands
- XRAYSCAN errors and syntax summary

A report is obtained by simply naming an entity or set of related entities. For example, to get a report on all processes, enter the command

```
    +PROCESS
```

To get a report on the disc volume $SYSTEM, enter

```
    +DISC $SYSTEM.
```

Report Formats

Reports are displayed in either linear or tabular format. Linear reports are used when the selected entity set has only one member. Tabular reports are used to display statistics for two or more entities. Some examples will clarify this point.

The command CPU 1 generates a linear report on processor 1 similar to the following:

```
    CPU  1                                    MAY 22   02:51:39 FOR 95.7

         CPU   SWAP  DISC  CHIT  SEND  SEND  CPU   DISP  TRAN  RESP
         BUSY  RATE  RATE  RATE  BUSY  RATE  QLEN  RATE  RATE  TIME

         42.7  1.96              2.57  3.69  .436  68.2
```

The name of the entity is CPU 1. The date of the measurement is May 22, and the time over which the statistics were collected (known as the "Window") began at 2:51:39 and lasted for 95.7 seconds.

Each item relating to CPU 1 is displayed along with its associated value; thus, CPU BUSY is 42.7%, SWAP RATE is 1.96 swaps per second, and so on. No number associated with an item (as for DISC RATE) implies a value of zero.

The command CPU generates a tabular report on all processors:

```
    CPU: *                                    MAY 22   02:51:39 FOR 95

                        CPU  0     CPU  1

      CPU BUSY          28.5       42.7

      SWAP RATE                    1.96

      DISC RATE         11.5

      CHIT RATE         1.25

      TRAN RATE         .020

      RESP TIME
```

The entities (in this case, processors 0 and 1) are listed across the top, while the items are listed in the left hand column. In this example, processor 0 had a DISC RATE of 11.5 transfers per second (see Section 4 for the meaning of this and other item statistics), while the CHIT RATE for processor 1 was zero.

Note that not all items relating to each entity are displayed by the tabular format. For the report to fit on a terminal's screen (when XRAYSCAN's OUT file is a terminal), tabular reports display the statistics for only certain items relating to each entity. (If the OUT file is not a terminal, all items are displayed).

Those items displayed in tabular format are known as an entity's "compare items"--so named because they are the items most useful to be seen in comparison with other entities. The compare items for any processor are CPU BUSY, SWAP RATE, DISC RATE, CHIT RATE, and TRAN RATE.

The Double Prompt (++)

It may sometimes happen that not all the entities specified in a report fit on one line. For example, the command PROCESS reports, in tabular form, on all the processes under measurement. If 40 processes are being measured, the resulting table won't fit on one screen. When this occurs, a table containing data for only six entities at a time is displayed, and a double prompt (++) is issued. At that time, you have two alternatives:

- enter a carriage return to see the next few entities. If more entities remain, another "++" follows.
- enter any other XRAYSCAN command. This terminates the report and the command is executed normally.

You also get a double prompt when there are exactly six entities being reported on. In this case, entering a carriage return causes a report heading to be displayed.

Conditional Reporting

You can specify reports that contain only those entities that meet a certain condition. For example, you may be interested in those processes havin a CPU BUSY item in excess of 10%. In that case, the command

```
+PROCESS, IF CPU BUSY > 10
```

generates the desired report.

Once specified, the IF condition REMAINS IN EFFECT in all subsequent reporting or plotting of that entity type. In the above example, any following PROCESS command will show only those processes having a CPU BUSY statistic in excess of 10.

Multiple IF clauses can be combined, as in

```
   +DISC,  IF DISC BUSY > 1.5, SEEK RATE < 2
```

The above command produces one report of those disc devices satisfying both the given conditions.

The same effect could have been achieved with the following commands:

```
   +DISC $SYSTEM          ! Produces a report of all discs.
   +IF DISC BUSY > 1.5  ! Reports on discs satisfying the condiiton.
   +IF SEEK RATE < 2     ! Reports on discs satisfying both conditions.
```

The difference between the two methods above is that specifying both IF conditions in one command generates one report, while using separate commands generates three reports (that is, each IF command generates a report).

Note that using an IF command without specifying the entity implies the most recently selected entity type. Thus, IF DISC BUSY > 1.5 in the above example refers to the DISC BUSY item of the DISC entity type.

The form IF? is used to display the current IF conditions. An IF condition is removed by specifiying the item without any value, as in:

```
   IF CPU BUSY
```

which removes any IF condition from the CPU BUSY item.

Reports Sorted by Items

Entities in reports can be sorted in order of a particular item using a BY command. The command CPU, BY CHIT RATE displays the same information as the command CPU, except that the former orders the processors from left to right by CHIT RATE, with the highest rate at the left.

Unlike the IF command, the BY command affects only the current report and not subsequent ones.

Like the IF command, however, a BY command can be used by itself, with the most recently named entity being implied. For example:

```
   +CPU             ! Report on all processors.
   +BY SEND RATE    ! Report on all processors,
                    ! ordering them by SEND RATE.
```

BY commands may be combined with IF commands in a single command:

```
  FILE  $SYSTEM.*.*, BY FILE RATE, IF READ RATE < 1
```

When used together, the BY clause must precede the IF clause.

XRAYSCAN lets you plot individual items against time. Moreover, time plots can be "built"--that is, items can be added to an existing plot one at a time--in order to discover relationships not revealed by reports.

A simple example would be the following plot, which shows that the CPU BUSY statistic for process 2,5 has the same general shape as the DISC BUSY item of the disc volume $MKT.

```
         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
10:58:30 -   A  B  +         +         +         +         -  A:  PID 2, 5
11:00:00 -   A       B                                     -      CPU BUSY
   01:30 -     A        B                                  -  B:  $MKT
   03:00 -       A    B                                    -      DISC BUSY
   04:30 -      A       B                                  -
11:06:00 -      A  +     B   +         +         +         -
   07:30 -A                                                -
   09:00 -    A    B                                       -
   10:30 -   A       B                                     -
   12:00 -   A    B                                        -
11:13:30 -     A   +        B+        +         +          -
   15:00 -     A      B                                    -
   16:30 -              A   B                              -
   18:00 -        A         B                              -
   19:30 -     A            B                              -
11:21:00 -    A    +  B      +         +         +         -
         0::::+:20.0::::+:40.0::::+:60.0::::+:80.0::::+::100
```

The SYSGEN listing for the system being XRAYed in the above example revealed that process 2,5 is, in fact, the disc process controlling $MKT.

Note that B was not printed at 11:07:30. This is because A and B had the same value at that time; in this situation, only the earliest letter is printed.

The above plot was created with the commands:

```
+WINDOW 10:58:30, 11:21:00   ! Define range of time axis.
+DELTA 90                    ! Measure at 90-second intervals.

+PROCESS 2,5                 ! Specify an entity.
+PLOT CPU BUSY               ! Add item to plot.

+DISC $MKT                   ! Specify an entity.
+PLOT DISC BUSY              ! Add item to plot.

+PLOT                        ! Display plot.
```

To build a plot, you use the commands discussed in the previous section, along with four other commands: PLOT, NEWPLOT, SCALE, and DELTA.

The PLOT command has two functions: It adds an item to the current plot, and it displays the current plot.

NEWPLOT "erases" the current plot, letting you start over. SCALE adjusts the upper and lower bounds of the horizontal axis, and DELTA adjusts the resolution of the time axis.

The entity selection commands are used to define the current entity set.

Current Entity Set

The current entity set is defined simply as the entity or entities named in the most recent report. For example, when you generate a report of the processes in CPU 0 with the command

```
+PROCESS 0
```

you are also designating those processes as the current entity set.

The current entity set defines the entity or entities for which items are to be added to the current plot by the PLOT command. If the current entity set is "processes in CPU 0," then the command

```
+PLOT CPU BUSY
```

adds the CPU BUSY item of all those processes to the current plot.

Any IF clauses currently in effect restrict the current entity set. An illustration (exclamation points indicate explanatory text; they are not part of XRAYSCAN's syntax):

```
+CPU, IF CPU BUSY > 10
     ! Reports on those processors meeting the condition.
+PLOT CPU BUSY
     ! Adds to the current plot the CPU BUSY item of those
     ! processors having a CPU BUSY item greater than 10%.
```

Using XRAYSCAN to Generate Reports and Plots

In general, the use of XRAYSCAN involves looking through reports of various sets of entities, occasionally plotting an item, examining the current plot, and formulating questions about the meaning of the output. Especially interesting plots and reports are reproduced on line printers via the COPY command.

If desired, following the interactive session with XRAYSCAN, you can make an Edit-format XRAYSCAN command file that automatically generates reports and plots of particular interest. For this reason, you may find it useful to keep a written record of the commands sequence as you proceed.

The commands that generate reports and specify the current entity include both the Entity Selection commands and the IF and BY clauses.

Entity Selection commands:

BUFFER	Process buffer usage
CPU	Processors
DEVICE	Line printers, tape drives, card readers, paper tape, etc.
DISC	Discs
DISCOPEN	Individual disc file openings
FILE	File openings on any device
LINE	Communication lines
NETLINE	Network communication lines
POOL	Buffer pools at processor level
PROCESS	Processes
SYSTEM	Remote system traffic
TERMINAL	Terminals

IF and BY clauses:

BY	Orders entities in a report by the value of specific items.
IF	Limits reports to those entities satisfying a set of conditions.

Three commands are used in conjunction with the entity selection commands to build plots:

DELTA	Sets the interval on the time axis.
PLOT	Adds items to plots; displays plots.
SCALE	Sets the upper and lower bounds on the horizontal axis.

A single command is provided for displaying entry-point histograms for a process:

PROCEDURE	Display entry-point histogram for the currently viewed process.

There are three additional commands:

WINDOW	Limits reports and plots to a subset of the data file.
COPY	Writes the most recent report or plot to a file.
OUTLEN	Sets the width of XRAYSCAN's output.

Units of Measurement

XRAY always uses seconds as the unit of time. The following conventions are followed regarding the units of measured entities:

XRAY ITEM UNITS

An item expressed as a RATE is in counts per second

An item expressed as BUSY is a percentage (from 0 to 100)

An item expressed as a CNTR is a simple count

An item expressed as a TIME is a count of seconds

Overflow and Underflow

Counts of events are kept by XRAY in 32-bit integer format. When an item in a report overflows, a display of ???? indicates where the count should be. When an item to be plotted overflows, a ? appears at the edge of the plot.

Underflow can occur when a RATE whose number of events, divided by the time, is too small to be displayed. In this case, the actual count is displayed, followed by a number sign (#) to indicate that this is a count and not a rate. Extreme underflow can cause the ???? pattern to occur.

Requesting a report for a time period of greater than one hour can lead to undetected overflow. To be sure that a counter has not

overflowed, it is mandatory that the reporting period be less than one hour. Equivalently, overflow of an item can always be checked
| by plotting the item with a DELTA of less than 3600 seconds. Certain
| buffer pool measurements (CB space pool measurements in particular)
| have been found to underflow in just a few minutes. Plot the value to
| observe the result at a higher resolution.

The BUFFER command reports on the buffer activity or a process or set of processes.

The form of the BUFFER command is:

```
BUFFER <process entity set> [, <BY clause> ][, <IF clause>]
------ --------------------

where:

   <process entity set> is one of the following

        (blank)
        <cpu>
        <cpu>,<pin>
        <discfileset>
        <discfileset>,<cpu>
        <discfileset>,<cpu>,<pin>
        <process name>

examples:

   BUFFER              ! all
   BUFFER 4            ! processor 4
```

The BUFFER command reports on the following items:

SHORT BUSY

   Average percentage utilization of space in the shortpool.

SHORT REQ

   Allocations of shortpool space per second.

SHORT FAIL

   Shortpool allocation failures per second.

SHORT BYTES

   Average number of shortpool bytes in use.

IO BUSY

-->

Average percentage utilization of space in the iopool.

IO REQ

Allocations of iopool space per second.

IO FAIL

Iopool allocation failures per second.

IO BYTES

Average number of iopool bytes in use.

LONG BUSY

Average percentage utilization of space in the longpool.

LONG REQ

Allocations of longpool space per second.

LONG FAIL

Longpool allocation failures per second.

LONG BYTES

Average number of longpool bytes in use.

CB BUSY

Average percentage utilization of space in the control block space.

CB REQ

Allocations of control block space per second.

CB FAIL

Control block space allocation failures per second.

CB WORDS

Average number of control block space words in use.

The BY command specifies the order in which a set of entities in a report are arranged. The BY command can be used by itself, in which case it refers to the current entity, and generates a report. BY can also be used as a clause in another entity selection command.

The form of the BY command is:

```
-----------------------------------------------------------------------
|                                                                     |
|  BY  <item>                                                         |
|  --  ------                                                         |
|                                                                     |
|     where:                                                          |
|                                                                     |
|        <item>                                                       |
|                                                                     |
|           is associated with the current entity.                    |
|                                                                     |
|  example:                                                           |
|                                                                     |
|     BY MSG QLEN                                                     |
|                                                                     |
|       ! Assuming that the current entity is a set of processes,     |
|       ! generate a report of the processes and display them in      |
|       ! order of MSG QLEN, from largest to smallest.                |
|                                                                     |
-----------------------------------------------------------------------
```

<u>CONSIDERATONS</u>

* The use of the BY statement as a clause in an entity selection command is illustrated by

  PROCESS 0, BY PRES PAGES

  This command generates a report consisting of all processes in CPU 0, arranged in decreasing order of the number of pages present in main memory for each process when it was executed.

The COPY command writes the most recent report or plot to a file. It is generally used to produce a line printer copy of a report or plot of interest.

The form of the COPY command is:

```
------------------------------------------------------------------------
|                                                                      |
|  COPY  [ <file name> | ! ]                                           |
|  ----                                                                |
|                                                                      |
|     where:                                                           |
|                                                                      |
|        <file name>                                                   |
|                                                                      |
|           designates the file to which the report or plot is         |
|           to be written; closes the current copy file, if any.       |
|                                                                      |
|           Omitting <file name> causes the report or plot to          |
|           be written to the current copy file;that is, to the        |
|           file most recently specified in a COPY command, if         |
|           one exists (therefore, the first COPY command must         |
|           specify a file name)                                       |
|                                                                      |
|  example:                                                            |
|                                                                      |
|     COPY $lp   ! Copies previous report or plot to $LP.              |
|                                                                      |
------------------------------------------------------------------------
```

CONSIDERATIONS

* The OUTLEN command can be used prior to a COPY command to adjust the output width of the report. Otherwise, the output width is automatically set to the configured record length of the file.

A02

The CPU command reports on measured processors.

The form of the CPU command is:

```
CPU <cpu entity set> [ , <BY clause> ] [ , <IF clause> ]
---

where:

   <cpu entity set> is either

      (blank)      ! Report on all processors.

         or

      <cpu #>      ! Report on the specified processor.

 examples:

     CPU           ! all
     CPU 2         ! processor 2
```

The CPU command reports on the following items:

CPU BUSY

Percentage of time (during the WINDOW) that the processor was in use, either executing a program or handling a non-high priority interrupt

SWAP RATE

Average number of virtual memory pages swapped in or out per second

Because code pages are never swapped out, each swap (whether in or out) is counted separately. Thus, the exchange of one data page in and one out counts as two swaps.

DISC RATE

Average number of input and output disc transfers on this processor's i/o channel per second

This item counts actual transfers, not EIO instructions. For example, when two disc spindles are connected to the same controller, an I/O transfer to either of the discs involves one EIO for the seek, and another for the actual byte transfer.

CHIT RATE

Average number of disc cache hits per second

Hits are counted on both input and output transfers.

SEND BUSY

Percentage of WINDOW interval spent sending messages to other processors

CPU QLEN

The length of the processor queue, averaged over the WINDOW

This includes all processes on the ready list, except the one running and the ones waiting for virtual memory.

-->

DISP RATE

The average number of times per second that the processor was set up to execute a process

The processor is "set up" for a process by intializing the P, E, L, S, and G registers, and loading the memory maps.

TRAN RATE

Average number of terminal interactions (read completion followed by a write) on the terminals under measurement

RESP TIME

The average time from a read completion to the start of the next write to a terminal under measurement

LINK BUSY

The percentage of the link control blocks (LCBs) that are allocated.

LINK REQ

The number of link control block allocations per second.

LINK FAIL

The number of link control block allocation failures per second.

LINKS INUSE

The average number of link control blocks in use.

The DELTA command sets the interval between points on the time axis of a plot.

The form of the DELTA command is:

```
DELTA [ <interval> ]
-----

where:

   <interval>,

        in the range {1:2,147,482.999}, specifies the number
        of seconds between points on the time axis.

   Omitting <interval> causes the time axis to be adjusted so
   that the plot fills the screen on a CRT, or a line printer
   page (60 lines) on another type of device.  (This is known
   as the default DELTA interval.)

example

   DELTA 600  ! Partition the time axis into 10-minute intervals.
```

<u>CONSIDERATIONS</u>

* Do not enter a DELTA interval of 0. This will cause the time axis of the plot to stay constant.

  Specifying a small WINDOW with the default DELTA interval in effect may cause the same problem. If you observe that the time axis of a plot remains constant, type <break> and adjust either the WINDOW or the DELTA interval.

* A time plot can never have better resolution than the original measurement from which the data was obtained. The time axis of any plot is therefore partitioned into intervals whose length is:

  MAX(DELTA interval, original GO interval)

* Resetting the DELTA interval to the default value does not affect the current WINDOW setting.

A02

The DEVICE command reports on the activity of measured devices.

The form of the DEVICE command is:

```
---------------------------------------------------------------------
|                                                                   |
|  DEVICE <device entity set> [ , <BY clause> ] [ , <IF clause> ]   |
|                                                                   |
|  where:                                                           |
|                                                                   |
|     <device entity set> is one of the following.                  |
|                                                                   |
|        (blank)                                                    |
|                                                                   |
|          all devices under measurement                            |
|                                                                   |
|        <cpu>                                                      |
|                                                                   |
|          devices connected to the I/O channel of the given        |
|          processor                                                |
|                                                                   |
|        <cpu> , <controller>                                       |
|                                                                   |
|          devices connected to the specified controller via the    |
|          I/O channel of the given processor                       |
|                                                                   |
|        <cpu> , <controller> , <unit>                              |
|                                                                   |
|          the device connected to the given unit                   |
|                                                                   |
|        $<device name>                                             |
|                                                                   |
|          the particular device                                    |
|                                                                   |
|        $<device name> , <cpu>                                     |
|                                                                   |
|          the path from the processor to the device                |
|                                                                   |
|        $<device name> , <cpu> , <controller>                      |
|                                                                   |
|          the path from the particular controller to the device    |
|                                                                   |
|        $<device name> , <cpu> , <controller> , <cpu>              |
|                                                                   |
|          the particular unit                                      |
|                                                                   |
---------------------------------------------------------------------
```

The DEVICE command reports on the following items:

DEV BUSY

Percent of time during the WINDOW that the device was busy reading or writing data

REQ RATE

Average number of requests per second to the process managing this device

READ BUSY

Percent of time data was being read from the device into main memory

WRITE BUSY

Percent of time spent writing data to the device

DEV RATE

Average number of read and write operations per second

READ RATE

Average number of read operations per second

WRITE RATE

Average number of write operations per second

BYTE RATE

Number of bytes transferred to and from the device per second

IBYTE RATE

Number of bytes input by the device per second

OBYTE RATE

Number of bytes output to the device per second

The DISC command reports on the activity of measured disc units.

The form of the DISC command is:

```
-----------------------------------------------------------------------
|                                                                     |
|  DISC <disc entity set> [ , <BY clause> ] [ , <IF clause> ]         |
|  ---- -----------------                                             |
|                                                                     |
|  where:                                                             |
|                                                                     |
|     <disc entity set> is one of the following.                      |
|                                                                     |
|        (blank)                                                      |
|                                                                     |
|           all discs                                                 |
|                                                                     |
|        <cpu number>                                                 |
|                                                                     |
|           all discs connected to the specified processor            |
|                                                                     |
|        <cpu number> , <controller number>                           |
|                                                                     |
|           all discs connected to the specified processor via the    |
|           given controller                                          |
|                                                                     |
|        <cpu number> , <controller number> , <unit number>           |
|                                                                     |
|           the disc connected to the specified processor via the     |
|           given controller and unit number                          |
|                                                                     |
|        $<volume>                                                    |
|                                                                     |
|           the specified disc                                        |
|                                                                     |
|        $<volume> , <cpu number>                                     |
|                                                                     |
|           the path from the specified disc to the processor         |
|                                                                     |
|        $<volume> , <cpu number> , <controller number>               |
|                                                                     |
|           the path from the disc to the controller                  |
|                                                                     |
|        $<volume> , <cpu number> , <controller number>               |
|           , <unit number>                                           |
|                                                                     |
|           the path from the disc to the unit                        |
|                                                                     |
|                                                                     |
|                                                                 --> |
-----------------------------------------------------------------------
```

```
examples:

    DISC          ! all discs

    DISC 1        ! disc activity over processor 1's I/O channel

    DISC $SYSTEM  ! activity on $SYSTEM
```

The DISC command reports on the following items:

DISC BUSY

Percent of time the disc was busy either seeking, rotating to position or transferring data

XFER BUSY

Percent of time the disc was busy rotating to position or transferring data

This also includes the time the disc was automatically seeking, if the controller commands only one spindle.

SEEK BUSY

Percent of time the disc was busy seeking

Note that seeks are not issued separately unless the controller commands multiple spindles.

REQ RATE

Average number of requests per second to the process that manages this disc

SEEK RATE

Average number of seeks per second

DISC RATE

Average number of input or output operations on the disc per second

Note that seeks are not included in this figure.

READ RATE

Input operations per second (to main memory)

WRITE RATE

Output operations per second (from main memory)

BYTE RATE

Number of bytes transferred to and from the disc per second

-->

IBYTE RATE

Number of bytes input by the disc per second

OBYTE RATE

Number of bytes output to the disc per second

SWAP RATE

Average number of input and output requests for virtual memory pages residing on this disc, per second

CHIT RATE

Average number of calls to the driver satisfied by finding the data in the disc cache

STALL RATE

Average number of requests for files or records, per second, that were blocked because a prior LOCKFILE or LOCKRECORD was still pending

The DISCOPEN command reports on disc file activity, considering each physical opening of a file by a process to be a separate entity.

"Physical" file openings include file openings on behalf of, but invisible to, application programs (such as opening partitioned files or alternate key files).

The form of the DISCOPEN command is:

```
---------------------------------------------------------------------------
|                                                                         |
|  DISCOPEN <discopen entity set> [ , <BY clause> ] [ , <IF clause> ]    |
|                                                                         |
|  where:                                                                 |
|                                                                         |
|     <discopen entity set> is one of the following.                      |
|                                                                         |
|        (blank)                                                          |
|                                                                         |
|           all file openings under measurement                           |
|                                                                         |
|        <cpu>                                                            |
|                                                                         |
|           files opened by processes in this processor                   |
|                                                                         |
|        <cpu> , <pin>                                                    |
|                                                                         |
|           files opened by any process having this PID                   |
|                                                                         |
|        <cpu> , <pin> , <file number>                                    |
|                                                                         |
|           the file having this file number with respect to the          |
|           given process                                                 |
|                                                                         |
|        <discfileset>                                                    |
|                                                                         |
|           all file openings in the discfile set                         |
|                                                                         |
|        <discfileset> , <cpu>                                            |
|                                                                         |
|           files opened in the discfile set by processes in this         |
|           processor                                                     |
|                                                                         |
|        <discfileset> , <cpu> , <pin>                                    |
|                                                                         |
|           files opened in the set by processes with this PID            |
|                                                                         |
|        <discfileset> , <cpu> , <pin> , <file number>                    |
|                                                                         |
|           files opened in the set that have the given file number       |
|           with respect to processes with the designated PID             |
|                                                                         |
|                                                                     --> |
---------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
|                                                                      |
|                                                                      |
|      \<systemname>                                                   |
|                                                                      |
|         files opened from the named system                           |
|                                                                      |
|    <discfileset> is one of                                           |
|                                                                      |
|      { $<local discfile set>               }                         |
|      { \<systemname>.$<local discfile set> }                         |
|                                                                      |
|    <local discfile set> is one of                                    |
|                                                                      |
|      $<volume>.<subvolume>.<file name>   ! one file                  |
|      $<volume>.<subvolume>.*             ! all files in subvol       |
|      $<volume>.*.*                       ! all file on volume        |
|      $<volume>.#<temporary file>         ! one temporary file        |
|                                                                      |
|      <systemname> is one of                                          |
|                                                                      |
|         <sysname>                                                    |
|         <systemnumber>                                               |
|                                                                      |
|  examples:                                                           |
|                                                                      |
|    DISCOPEN  $SYSTEM.SYSTEM.*  ! all file openings in the subvol     |
|                                                                      |
|    DISCOPEN  $SYSTEM.SYSTEM.TAL, 2,35, 2                             |
|                                                                      |
|      ! opening of the given file, having file number 2,              |
|      ! by process 2,35                                               |
|                                                                      |
------------------------------------------------------------------------
```

The DISCOPEN command reports on the following items:

DRIVE RATE

Average number of calls per second to the disc driver on behalf of this file opening

DRIN RATE

Average number of driver input calls per second on behalf of this file opening

DROUT RATE

Average number of physical driver output calls per second on behalf of this file opening

CHIT RATE

Average number of driver calls per second, on behalf of this file opening, that resulted in cache hits

STALL CNTR

Total number of times during the WINDOW that requests for this file or a record within it were blocked from access because the file or record was locked

Note that this is a total and not a rate.

SPLIT RATE

The count of block splits at the file open level. Used to determine whether the slack built into a file is sufficient to avoid excessive block splitting.

FREE RATE

The count of the number of disc free space table I/Os.

The FILE command reports on the activity of measured files, including devices and disc files. Each logical file opened by a process is considered to be a separate entity.

A "logical" file opening is the opening of a file by an application process, as opposed to file openings performed by the file system on behalf of the application (such as partitioned files and alternate key files). The latter type of file opening is accessed via the DISCOPEN command.

The form of the FILE command is:

```
----------------------------------------------------------------------
|                                                                    |
|  FILE <file entity set> [ , <BY clause> ] [ , <IF clause> ]        |
|                                                                    |
|  where:                                                            |
|                                                                    |
|     <file entity set> is one of the following.                     |
|                                                                    |
|        (blank)                                                     |
|                                                                    |
|           Report on all files.                                     |
|                                                                    |
|        <cpu>                                                       |
|                                                                    |
|           Report on all files opened by processes in the given     |
|           processor.                                               |
|                                                                    |
|        <cpu> , <pin>                                               |
|                                                                    |
|           Report on all files opened by processes with the given   |
|           <cpu>, <pin>.                                            |
|                                                                    |
|        <cpu> , <pin> , <file number>                               |
|                                                                    |
|           Report on this particular file opening.                  |
|                                                                    |
|        <fileset>                                                   |
|                                                                    |
|           Report on files in this set opened by any process.       |
|                                                                    |
|        <fileset> , <cpu>                                           |
|                                                                    |
|           Report on files in the set opened by any process running |
|           in this processor.                                       |
|                                                                    |
|        <fileset> , <cpu> , <pin>                                   |
|                                                                    |
|           Report on files in the set opened by processes having    |
|           the given PID.                                           |
|                                                                    |
|                                                               -->  |
----------------------------------------------------------------------
```

A02

```
\<systemname>

    Report on all files opened on the named system.

<fileset> , <cpu> , <pin> , <file number>

    Report on files in the set having the given file number
    with respect to any process having the designated PID.

<fileset> is:

    { $<device name>                   }
    { $<discfile set>                  }
    { <subdevice>                      }
    { \<systemname>                    }
    { \<systemname>.$<device name>     }
    { \<systemname>.$<discfile set>    }
    { \<systemname>.<subdevice>        }

    <subdevice> is one of:

        { $<linename>.*              }
        { $<linename>.#<subdevice>   }

    <discfile set> is one of:

        $<volume>.<subvolume>.<file name>  ! one file
        $<volume>.<subvolume>.*            ! all files in subvol
        $<volume>.*.*                      ! all files on volume
        $<volume>.#<temporary file>        ! one temporary file

    <systemname> is one of

        <sysname>
        <systemnumber>
```

The FILE command reports on the following items:

FILE RATE

Average number of logical file transfers per second

READ RATE

Average number of calls to READ, READUPDATE, or READUPDATELOCK per second

WRITE RATE

Average number of calls per second to WRITE

UPDT RATE

Average number of calls per second to WRITEUPDATE or WRITEUPDATEUNLOCK

DOWR RATE

Average number of deletions or WRITEREADs per second

Note that every file permits either deletions or WRITEREADs, but not both.

INFO RATE

Number of calls to FILEINFO or FILERECINFO which pass messages to the process managing the device

The IF command generates a report of those entities satisfying a condition or set of conditions. At the same time, the IF command reduces the current entity set to those entities meeting the IF condition.

Used by itself, the IF command generates a report; it can also be used as a clause in an entity selection command.

The form of the IF command is:

```
----------------------------------------------------------------------
|                                                                    |
| [ <entity> , ] IF <condition> [ , <condition > ... ]               |
|                -- -----------                                      |
|                                                                    |
|  where:                                                            |
|                                                                    |
|    <entity>                                                        |
|                                                                    |
|       is a legal entity selection command.  If omitted, the most   |
|       current entity set is implied.                               |
|                                                                    |
|    <condition> is                                                  |
|                                                                    |
|       { <item> [ { "<" | ">" } <num> ] }                           |
|       { ?                              }.                          |
|                                                                    |
|       <item> is an item relating to the given (or implied)         |
|              entity type.                                          |
|                                                                    |
|       <num>  is a legal value for that item.                       |
|                                                                    |
|              If the inequality sign and <num> are absent, the      |
|              IF command removes any previous IF condition on       |
|              the specified item.                                   |
|                                                                    |
|       ?      displays the current entity set and its current       |
|              IF conditions, and suppresses the generation of       |
|              a report.                                             |
|                                                                    |
|  examples:                                                         |
|                                                                    |
|    PROCESS 1, IF CPU BUSY > 10                                     |
|                                                                    |
|       ! Report on all processes in CPU 1 having a CPU BUSY item    |
|       ! greater than 10.                                           |
|                                                                    |
|    IF REQ RATE > 1, READ RATE < 0.4                                |
|                                                                    |
|       ! Report on all processes in CPU 1 (the most recently        |
|       ! named entity is implied) that meet both conditions.        |
|                                                                    |
|                                                               -->  |
----------------------------------------------------------------------
```

```
--------------------------------------------------------------------
|                                                                  |
|                                                                  |
|    CPU 2, IF SWAP RATE                                           |
|                                                                  |
|       ! Remove any existing IF condition on the SWAP RATE item.  |
|                                                                  |
--------------------------------------------------------------------
```

CONSIDERATIONS

* It is important to observe that an IF condition alters the current entity set. The command CPU, IF CPU BUSY > 10 not only generates a report of those processors having a CPU BUSY item greater than 10%, but also limits the current entity set to those processors meeting that condition, unless the IF? form appears on the same line. For example,

  ```
  IF CPU BUSY > 10, ?
  ```

  displays the IF conditions, but does not restrict the current entity set until the next report or plot.

  Note that the IF condition applies to any subsequent reports involving not only the same entity, but also the same entity type. For example,

  ```
  +PROCESS $SYSTEM.SYSTEM.TAL, IF CPU BUSY > 10
  ```

  generates a report of those TAL compilations having a CPU BUSY item greater than 10; however, the IF clause actually refers to the entity type PROCESS. Therefore, the subsequent command

  ```
  +PROCESS
  ```

  generates a report of all processes with a CPU BUSY item greater than 10, because the IF condition applies to all subsequent reports involving entities of the same type (that is, processes) and not just to the particular entity named in the original IF clause.

* An IF condition, once installed, remains in effect until it is specifically removed.

  If something you expected in a report or plot doesn't show up, examine the IF conditions in effect with an IF? command. (If that doesn't help, check your WINDOW with a CPU command.)

* The form IF? displays the current entity set, its associated items, and any IF conditions in effect. The display shows a less-than condition (such as IF CPU BUSY < 10.0) by writing the value (10.0) to the RIGHT of the item name; the display shows a greater-than condition (such as IF MSG QLEN > .8) by writing the value to the

LEFT of the item name. (Always imagine a "<" sign between the value and the item name.)

You can take advantage of the IF? form to suppress a report. For example, suppose you want to plot DISC BUSY for $SYSTEM. Instead of entering DISC $SYSTEM and waiting for the tabular report to finish before you can enter PLOT DISC BUSY, you can use the command

```
+DISC $SYSTEM, IF?
```

to specify $SYSTEM as the current entity and display the DISC items; this is much faster than waiting for a report.

* Plots are affected differently by If conditions, depending on the order in which commands are given. For example:

```
Situation A                   Situation B
-----------                   ------------

+CPU                          +CPU, IF CPU BUSY > 10
+PLOT CPU BUSY                +PLOT CPU BUSY
+IF CPU BUSY > 10             +PLOT
+PLOT
```

In situation A, the CPU command puts all processors into the current entity set; the second command, PLOT CPU BUSY, adds each processor's CPU BUSY item to the current plot. The third command restricts the current entity set, <u>but does not eliminate any items</u> <u>from the current plot (once an item is added to the plot, it</u> continues to be plotted). The command PLOT displays a plot consisting of the CPU BUSY item of each processor. However, the only points plotted are those with a CPU BUSY item greater than 10.

In situation B, the IF condition in the first command restricts the current entity set <u>before</u> items are added to the plot; thus, PLOT CPU BUSY adds the <u>CPU BUSY</u> item to the current plot for each processor having a CPU BUSY item that meets the IF condition. As in situation A, the only points plotted are those above 10.

The LINE command reports on measured communication lines.

The form of the LINE command is:

```
LINE <line entity set> [ , <BY clause> ] [ , <IF clause> ]

where:

  <line entity set> is one of the following.

     (blank)

        all communication lines

     <cpu>

        lines connected to the given processor

     <cpu> , <controller>

        lines connected to the specified controller

     <cpu> , <controller> , <unit>

        one processor, controller, and unit number

     $<line name>

        the designated communication line

     $<line name> , <cpu>

        the path from the line to the processor

     $<line name> , <cpu> , <controller>

        the path from the line to the controller

     $<line name> , <cpu> , <controller> , <unit>

        the given line and unit

examples:

   LINE             ! all lines

   LINE $bsc2, 3, 4  ! line $BSC2, CPU 3, controller 4
```

The LINE command reports on the following items:

LINE BUSY

Percent of time (during the WINDOW) the line is busy

READ BUSY

Percent of time the unit is busy reading (input to memory or hardware polling)

WRITE BUSY

Percent of time the unit is busy writing (output from memory)

REQ RATE

Rate at which requests for the unit arrive at the process controlling the line

BYTE RATE

Average number of input and output bytes transferred by the unit per second

IBYTE RATE

Average number of bytes per second input by the unit

OBYTE RATE

Average number of bytes per second output to the unit

LINE RATE

Average number of input and output transfers per second on the unit

READ RATE

Average number of input transfers (to memory) per second on the unit

WRITE RATE

Average number of output transfers per second on the unit

-->

RETRY CNTR

Total number of retries on the line during the WINDOW

Note that ENVOY's retry counter, which prints on the system console when the line is closed, is cleared when XRAY monitors the line (it still prints, but shows 0).

TRAN RATE

For multidrop lines, the number of read completions followed by write starts, per second, for all terminals on the line

RESP TIME

Average time from read completion until initiation of the subsequent write, averaged over all the terminals on the line

The NETLINE command reports on measured communication lines which connect the system to other TANDEM systems in an EXPAND network.

The form of the NETLINE command is:

```
NETLINE <line entity set> [ , <BY clause> ] [ , <IF clause> ]

where:

  <line entity set> is one of the following.

     (blank)

         all communication lines

     <cpu>

         lines connected to the given processor

     <cpu> , <controller>

         lines connected to the specified controller

     <cpu> , <controller> , <unit>

         one processor, controller, and unit number

     $<line name>

         the designated communication line

     $<line name> , <cpu>

         the path from the line to the processor

     $<line name> , <cpu> , <controller>

         the path from the line to the controller

     $<line name> , <cpu> , <controller> , <unit>

         the given line and unit

examples:

   NETLINE                  ! all lines

   NETLINE $pipe2, 3, 4  ! line $PIPE2, cpu 3, controller 4
```

The NETLINE command reports on the following items:

WRITE BUSY

Percent of time the unit is busy writing (output from memory)

READ BUSY

Percent of time the unit is busy reading (waiting for input to memory; there is usually a "Read Continue" outstanding on active, full-duplex lines)

REQ RATE

Rate at which requests for the unit arrive at the process controlling the lines

DEV RATE

Average number of input and output transfers per second on the lines

READ RATE

Average number of input transfers (to memory) per second on the line

WRITE RATE

Average number of output transfers per second on the line

LEV2 BYTES

Average number of input and output bytes transferred by the line per second

L2IN BYTES

Average number of bytes per second input on the read line

L2OUT BYTES

Average number of bytes per second output on the write line

LEV4 BYTES

Average number of input and output bytes of Level 4 Protocol information transferred by the line per second

-->

DIN4 BYTES

Average number of Level 4 data (user message) bytes per second input on the read line

DOUT4 BYTES

Average number of Level 4 data (user message) bytes per second output on the write line

CIN4 BYTES

Average number of Level 4 Protocol control bytes per second input on the read line

COUT4 BYTES

Average number of Level 4 Protocol control bytes per second output on the write line

U64 RATE

Number of input and output messages per second which, after compression (if any), were under 64 bytes long

U128 RATE

Number of input and output messages per second which, after compression (if any), were less than 128 bytes long

U256 RATE

Number of input and output messages per second which, after compression (if any), were less than 256 bytes long

U512 RATE

Number of input and output messages per second which, after compression (if any), were under 512 bytes long

U1024 RATE

Number of input and output messages per second which, after compression (if any), were under 1024 bytes long

-->

U2048 RATE

Number of input and output messages per second which, after compression (if any), were less than 2048 bytes long

U4096 RATE

Number of input and output messages per second which, after compression (if any), were less than 4096 bytes long

O4095 RATE

Number of input and output messages per second which, after compression (if any), were over 4095 bytes long

The NEWPLOT command erases all items from the current plot, allowing a new plot to be started.

The form of the NEWPLOT command is:

```
--------------------------------------------------------------------
|                                                                  |
|  NEWPLOT                                                         |
|  -------                                                         |
|                                                                  |
--------------------------------------------------------------------
```

CONSIDERATIONS

* Note that NEWPLOT does NOT affect the current values of the WINDOW, DELTA interval, or SCALE.

* To copy a report or plot to a file, issue the COPY command before entering NEWPLOT.

The OUTLEN command sets XRAYSCAN's output width, both to its <out> file and to any copy files (a copy file is a file designated in a COPY command).

The form of the COPY command is:

```
------------------------------------------------------------------------
|                                                                      |
|  OUTLEN  <width>                                                     |
|  ------  -------                                                     |
|                                                                      |
|     where:                                                           |
|                                                                      |
|        <width>, in the range [72:132],                               |
|                                                                      |
|           specifies XRAYSCAN's output width.                         |
|                                                                      |
|  example:                                                            |
|                                                                      |
|     OUTLEN 80                                                        |
|                                                                      |
------------------------------------------------------------------------
```

CONSIDERATIONS

* Note that although you may elect to set the output width to 72, plots are 76 characters wide. Part of the legend will be truncated.

* If the <out> file is a terminal, the default output width is 80; for other devices, the default is 132.

The PLOT command adds an item to, or displays, the current plot.

The form of the PLOT command is:

```
------------------------------------------------------------------------
|                                                                      |
|  PLOT  [ <item> , ... ]                                              |
|  ----                                                                |
|                                                                      |
|     where:                                                           |
|                                                                      |
|        <item>                                                        |
|                                                                      |
|           is an item to be added to the current plot.                |
|                                                                      |
|        Omitting <item> causes the current plot to be written to      |
|        XRAYSCAN's <out file>; usually, this is the terminal from     |
|        which you are running XRAYSCAN.                               |
|                                                                      |
|  example:                                                            |
|                                                                      |
|     PLOT LINE BUSY                                                   |
|                                                                      |
|       ! Add the LINE BUSY item to the current plot (assuming         |
|       ! that the current entity is a communication line).            |
|                                                                      |
------------------------------------------------------------------------
```

<u>CONSIDERATIONS</u>

* It is important to keep track of the current entity; the command PLOT CPU BUSY will plot either PROCESSOR CPU BUSY or PROCESS CPU BUSY, depending on the current entity.

  The command IF? displays the current entity set and its current IF conditions.

* If the current entity set consists of more than one entity, the command PLOT <item> plots each entity's values of that item. For example,

  ```
  +PROCESS
  +PLOT CPU BUSY
  ```

  adds as many items to the current plot as there are processes under measurement.

* Each PLOT command adds an item to the current plot; to start a new plot, issue the command NEWPLOT.

* A maximum of 26 items can be added to one plot.

The POOL command reports on the buffer pool acvitity at the cpu level.

The form of the POOL command is:

```
POOL [ cpu ] [, BY clause ] [, IF clause ]
----

where:

   <cpu> specifies for which processor the report is
   to be generated.   If <cpu> is omitted, the report
   is for all processors.

example:

   POOL 3
```

The POOL command reports on the following items:

SHORT BUSY

Average percentage utilization of space in the shortpool.

SHORT REQ

Allocations of shortpool space per second.

SHORT FAIL

Shortpool allocation failures per second.

SHORT BYTES

Average number of shortpool bytes in use.

IO BUSY

Average percentage utilization of space in the iopool.

IO REQ

Allocations of iopool space per second.

IO FAIL

-->

Iopool allocation failures per second.

IO BYTES

Average number of iopool bytes in use.

LONG BUSY

Average percentage utilization of space in the longpool.

LONG REQ

Allocations of longpool space per second.

LONG FAIL

Longpool allocation failures per second.

LONG BYTES

Average number of longpool bytes in use.

CB BUSY

Average percentage utilization of space in the control block space.

CB REQ

Allocations of control block space per second.

CB FAIL

Control block space allocation failures per second.

CB WORDS

Average number of control block space words in use.

The PROCEDURE command displays a histogram of entry point usage of the most recent process named in the previous PROCESS command. Entry points that are not used during the measurement are not included in the histogram. The histogram is "pseudo-paged". The graphic portion of the histogram is adapted to the current terminal or line printer.

The form of the PROCEDURE command is:

```
PROCEDURE [, BY clause ]

where:

   Including a BY CPU BUSY clause causes the entry points to be
   listed in order from the most active to the least active.

examples:

   PROCESS $XYZ
   PROCEDURE                   ! generate histogram of process XYZ.

   PROCESS $XYZ
   PROCEDURE, BY CPU BUSY      ! generate histogram of process XYZ
                                 in order of activity, from most to
                                 least active.
```

CONSIDERATIONS

* The previous report should have been a PROCESS command, resulting in the display of the statistics for the process in question. This determines which process's histogram is to be displayed.

* Insufficient samples causes noticable striations in the data (many buckets with 1, 2, or 3 observations). Observe the phenomenon for a longer window.

* The SCALE command adjusts the horizonal axis. Hardcopy output can be obtained via the COPY command.

The PROCESS command reports on measured processes.

The form of the PROCESS command is:

```
-------------------------------------------------------------------------
|                                                                       |
|  PROCESS <process entity set> [ , <BY clause> ] [ , <IF clause> ]     |
|  ------- --------------------                                         |
|                                                                       |
|  where:                                                               |
|                                                                       |
|     <process entity set> is one of the following.                     |
|                                                                       |
|        (blank)                                                        |
|                                                                       |
|           Report on all processes.                                    |
|                                                                       |
|        <cpu>                                                          |
|                                                                       |
|           Report on any process executing in this processor.          |
|                                                                       |
|        <cpu> , <pin>                                                  |
|                                                                       |
|           Report on processes having this PID.                        |
|                                                                       |
|        <discfile set>                                                 |
|                                                                       |
|           Report on any process executing a file in <discfileset>.    |
|                                                                       |
|        <discfile set> , <cpu>                                         |
|                                                                       |
|           Report on any process, having its program file in the       |
|           specified discfile set, running in the given processor.     |
|                                                                       |
|        <discfile set> , <cpu> , <pin>                                 |
|                                                                       |
|           Report on processes having this PID executing a program     |
|           file in the discfile set.                                   |
|                                                                       |
|        <process name>                                                 |
|                                                                       |
|           Report on the named process.                                |
|                                                                       |
-------------------------------------------------------------------------
```

The PROCESS command reports on the following items:

CPU BUSY

Percent of time during the WINDOW that this process executed in the processor

MSG RATE

Average number of messages per second initiated by this process

MBYTE RATE

Average number of bytes per second transferred as a result of messages initiated by this process

RECV RATE

Average number of messages per second arriving on the message input queue (for user processes, the messages arriving via $RECEIVE)

RECV QLEN

Average number of messages in the message input queue to the process, measured only over the time that the process was executing in the processor

VBYTE RATE

Average number of bytes per second received by this process (for user processes, limited to bytes read over $RECEIVE)

YBYTE RATE

Average number of bytes per second replied by this process (for user processes, the bytes sent via the REPLY procedure)

CHKPT RATE

Average number of checkpoints per second by this process

FAULT RATE

Average number of page faults per second

-->

PRES PAGES

Average number of pages present in main memory that were made present by this process, averaged over the time the process executed in the processor

Note that PIN 1 in each processor is the memory manager, which owns all memory pages not made present by any other process.

Also note that if there is no memory pressure (indicated by a FAULT RATE under 2 per second) then the PRES PAGES may include pages swapped in by the process and no longer in use, but not swapped out because the space isn't needed.

VSEM RATE

Total number of times this process was restrained in forward progress because of a logical resource conflict

BLKD WAIT

Percent of elapsed time that this process needed the processor and was unable to obtain it (because another process of higher or conflicting priority was using the processor)

LINK BUSY

The percentage of the link control blocks (LCBs) that are allocated.

LINK REQ

The number of link control block allocations per second.

LINK FAIL

The number of link control block allocation failures per second.

LINKS INUSE

The average number of link control blocks in use.

DISP RATE

The dispatch rate of individual processes. This helps understand how a cpu is partitioned on the average (a few long bursts of service vs. lots of short bursts).

A02

The SCALE command sets the upper and lower bounds of the horizontal axis of a plot.

The form of the SCALE command is:

```
--------------------------------------------------------------------------
|                                                                        |
|  SCALE [ [ <lower bound> ] [ , <upper bound> ] ]                       |
|  -----                                                                 |
|                                                                        |
|     where:                                                             |
|                                                                        |
|        <lower bound> and <upper bound>                                 |
|                                                                        |
|           are each in the range { 0:2147482.999 }; the lower bound     |
|           of the current plot is less than the current upper bound.    |
|                                                                        |
|        Supplying only one causes the other to retain its current       |
|        value; omitting both causes the current lower and upper         |
|        bounds to revert to their default values, 0 and 100.            |
|                                                                        |
|  example:                                                              |
|                                                                        |
|     SCALE 25         !  new lower bound = 25                           |
|     SCALE ,10000     !  new uper bound = 10000                         |
|     SCALE 0,1        !  new upper and lower bounds                     |
|                                                                        |
--------------------------------------------------------------------------
```

CONSIDERATIONS

* The horizontal axis of a plot is always partitioned linearly between the current lower and upper bounds; specifically, it is divided into 50 equal parts between the two extremes when the output width is less than 127, and 100 equal parts otherwise.

* The units of the horizontal axis are those of the item being plotted. When PROCESS CPU BUSY is plotted, the horizontal axis is calibrated in percentage; DISC CHIT RATE, however, is measured in cache hits per second.

* The horizontal axis should be appropriate to the item being measured. Thus, to plot PROCESS RECV QLEN, which typically takes values between 0 and 10, you would specify SCALE 0,10; to plot DISC IBYTE RATE, which takes values in the range 1000 to 10,000, you would set your scale accordingly. If these two items were plotted on the same set of axes, the units of the horizontal axis would depend on which item you were most concerned with.

  Note that it is impossible to compare the two items above in any meaningful way; setting the SCALE appropriate to one puts the other greatly out of focus. (You could, however, make a hard copy of each on a separate plot and compare them that way.)

The SYSTEM command reports on traffic to measured systems.

The form of the SYSTEM command is:

```
------------------------------------------------------------------
|                                                                |
| SYSTEM <system entity set> [ , <BY clause> ] [ , <IF clause> ] |
| ------                                                         |
|                                                                |
| where:                                                         |
|                                                                |
|  <system entity set> is either                                 |
|                                                                |
|      (blank)    ! Report on all measured system traffic.       |
|                                                                |
|         or                                                     |
|                                                                |
|      \<sysname> ! Report on the specified system.              |
|                                                                |
|         or                                                     |
|                                                                |
|      \<systemnumber>    ! Report on the specified system.      |
|                                                                |
|   examples:                                                    |
|                                                                |
|      SYSTEM             ! all                                  |
|      SYSTEM \PODUNK     ! the system in Podunk                 |
|                                                                |
------------------------------------------------------------------
```

The SYSTEM command reports on the following items:

LINK TIME

Average time in seconds to send a message from the local system and obtain the acknowledgment from the remote system

PACK RATE

Total number of packets per second sent to or received from the remote system. This is the sum of the next four items

SENT RATE

Number of packets per second originating at the local system and directed to the remote system

RCVD RATE

Number of packets per second originating at the remote system and directed to the local system

SENT FOR

Number of packets forwarded to the remote system by the local system on behalf of some other remote system

RCVD FOR

Number of packets forwarded from the remote system by the local system to some other system in the network

LINK RATE

Number of data messages per second sent to the remote system, not including Level 4 acknowledgments

CONSIDERATION

* Counts of packets include both data and Level 4 acknowledgments.

A02

The form of the TERMINAL command is:

```
TERMINAL <terminal entity set> [ , <BY clause> ] [ , <IF clause> ]
-------- ---------------------

where:

   <terminal entity set> is one of the following.

      (blank)

         all terminals under measurement

      $<terminal name>

         this particular terminal

      $<terminal name> , <cpu>

         this terminal, as accessed via the given processor

      <subdevice>

         this particular subdevice

      <subdevice>.<cpu>

         this subdevice, as accessed via the given processor
```

The TERMINAL command reports on the following items:

RESP TIME	Average time from the completion of a read from the terminal to the start of a write on the terminal
TERM RATE	Average number of input and output transfers from/to this terminal, per second.
READ RATE	Average number of input transfers per second by the terminal
WRITE RATE	Average number of output transfers per second to the terminal
BYTE RATE	Average flow of characters from/to this terminal, per second (Hardware polling characters are not included in any of these statistics.)
IBYTE RATE	Average number of characters read from the terminal per second
OBYTE RATE	Average number of characters sent from the main memory to the terminal per second
TRAN RATE	Average number of write operations, preceded immediately by read operations, to this terminal per second

The WINDOW command restricts XRAYSCAN's attention to a portion of the data file by specifying a particular time interval. The subset of the data file so selected is known as the WINDOW interval.

The form of the WINDOW command is:

```
-------------------------------------------------------------------------
|                                                                       |
|           [ [ + | - ] <number of seconds> ]                           |
|  WINDOW   [ [ <time> ] [ , <time> ]       ]                           |
|  ------                                                               |
|                                                                       |
|  where:                                                               |
|                                                                       |
|    <number of seconds>  is in the range { .001:2147482.999 }          |
|                                                                       |
|    <time>  is { <date and time> }                                     |
|               { <number of seconds> }                                 |
|                                                                       |
|       <date and time>  is [ <date> , ] <time of day>                  |
|                                                                       |
|                          { <month> <day> }                            |
|          <date>  is  { <day> <month> } [ <year> ]                     |
|                                                                       |
|          <time of day>  is <hour>:<minute> [ :<second> ]              |
|                                                                       |
|             <day>  is {1:31}                                          |
|                                                                       |
|             <month>  is  {JAN | FEB | MAR | APR | MAY | JUN |         |
|                            JUL | AUG | SEP | OCT | NOV | DEC }        |
|                                                                       |
|             <year>  is  {0:2047}                                      |
|                                                                       |
|             <hour>  is  {0:23}                                        |
|                                                                       |
|             <minute>  is  {0:59}                                      |
|                                                                       |
|             <second>  is  {0:59}                                      |
|                                                                       |
|  The meanings of the various forms of the WINDOW command are as       |
|  follows (note that you can specify the beginning of the time         |
|  interval of interest, or the end, or both):                          |
|                                                                       |
|    WINDOW                                                             |
|                                                                       |
|       selects the entire data file.                                   |
|                                                                       |
|    WINDOW  [ + | - ] <number of seconds>                              |
|                                                                       |
|       adjusts the entire WINDOW (front and rear edges) forward        |
|       or back by <number of seconds>.                                 |
|                                                                       |
|                                                                   --> |
-------------------------------------------------------------------------
```

```
WINDOW  <date and time> , <date and time>

   selects the interval between the two times.

   examples:  WINDOW 3 AUG 1978, 15:32:40, 5 AUG 1978, 17:00
              WINDOW 12:00, 1:00

WINDOW  <date and time> , <number of seconds>

   selects the interval starting at the given date and
   lasting for <number of seconds>.

   example:  WINDOW 1:52:34, 3600

WINDOW  <number of seconds> , <date and time>

   selects the interval starting <number of seconds> from the
   beginning of the data file and ending at the given time.

   example:  WINDOW 7200, 12 SEPT 1979, 5:12:02

WINDOW  <number of seconds> , <duration>

   selects the interval starting <number of seconds> from the
   beginning of the data file and lasting for <duration>
   seconds.

   example:  WINDOW 60, 10000

WINDOW  <date and time>
  WINDOW  , <date and time>
    WINDOW  <number of seconds>
      WINDOW  , <number of seconds>

   In each case, the omitted time retains its previous value.
   Thus,

      WINDOW 1:00

         changes the beginning of the current interval to 1:00
         and leaves the end unchanged.

      WINDOW , 1:00

         leaves the beginning unchanged, but alters the end.
```

-->

```
WINDOW 60

    sets the beginning of the interval to 1 minute from
    the start of the data file.

WINDOW , 60

    sets the end of the interval to 1 minute from the
    beginning.
```

<u>CONSIDERATIONS</u>

* Online monitoring is achieved by:

  - running XRAYSCAN against an open data file (that is, the data file of a measurement currently in progress).

  - setting the end of the WINDOW into the future.

  The data file should have at least 30 data points in it before online monitoring is started. If necessary, temporarily set XRAYCOM's GO interval to 1 for thirty seconds to get the online monitoring started.

* The default DELTA interval causes any plot to fill the screen or printer page. When the WINDOW is set into the future, the default DELTA interval becomes 5 seconds.

  If the GO interval is greater than 5 seconds, it may cause lines to be duplicated in the plot. Press the break key to stop the plot, set the DELTA interval equal to the GO interval, and issue another PLOT command.

* Another cause of faulty online plots is monitoring a file that is not actually being extended (for example, it might be full). Press <break> and examine the file with FUP.

* Make sure that the WINDOW specifies part of the data file. You can set the WINDOW entirely into the past or future, but your plots and reports won't contain any information.

* Note that when you specify a time of day, the hour is required.

* When a report is requested, its endpoint will never be beyond the next-to-last point plotted. To get reports on events close to the end of the data file, set the DELTA interval smaller and plot to the end.

CPU	DISC	DISCOPEN	LINE
CPU BUSY	DISC BUSY	DRIV RATE	LINE BUSY
SWAP RATE	XFER BUSY	DRIN RATE	READ BUSY
DISC RATE	SEEK BUSY	DROUT RATE	WRITE BUSY
CHIT RATE	REQ RATE	CHIT RATE	REQ RATE
SEND BUSY	SEEK RATE	STALL CNTR	BYTE RATE
SEND RATE	DISC RATE	SPLIT RATE	IBYTE RATE
CPU QLEN	READ RATE		OBYTE RATE
DISP RATE	WRITE RATE		LINE RATE
TRAN RATE	SWAP RATE		READ RATE
RESP TIME	CHIT RATE		WRITE RATE
STALL RATE	FREE RATE		RETRY CNTR
LINK BUSY			
LINK REQ			
LINK FAIL			
LINKS INUSE			

DEVICE	TERMINAL	PROCESS	FILE
DEV BUSY	RESP TIME	CPU BUSY	FILE RATE
REQ RATE	TERM RATE	RECV QLEN	READ RATE
READ BUSY	READ RATE	RECV RATE	WRITE RATE
WRITE BUSY	WRITE RATE	MSG RATE	UPDT RATE
DEV RATE	BYTE RATE	CHKPT RATE	DOWR RATE
READ RATE	IBYTE RATE	MBYTE RATE	INFO RATE
WRITE RATE	OBYTE RATE	FAULT RATE	
	TRAN RATE	PRES PAGES	
		VSEM CNTR	
		BLKD WAIT	
		LINK BUSY	
		LINK REQ	
		LINK FAIL	
		LINKS INUSE	

-->

NETLINE		SYSTEM
WRITE BUSY	DOUT4 RATE	LINK TIME
READ BUSY	CIN4 RATE	PACK RATE
REQ RATE	COUT4 RATE	SENT RATE
DEV RATE	U64 RATE	RCVD RATE
READ RATE	U128 RATE	SENT FOR
WRITE RATE	U256 RATE	RCVD FOR
LEV2 RATE	U512 RATE	LINK RATE
L2IN RATE	U1024 RATE	
L2OUT RATE	U2048 RATE	
LEV4 RATE	U4096 RATE	
DIN4 RATE	O4095 RATE	

The error messages output by XRAYSCAN are listed below. Where the meaning is obvious, no explanation is given.

ILLEGAL COPY FILE NAME SUPPLIED

ILLEGAL SYNTAX: COMMAND NOT EXECUTED

ILLEGAL XRAYDATA FILE NAME SUPPLIED; MEASUREMENT NOT PROCESSED

The specified data file has the wrong format.

INPUT ERROR ON XRAY DATA FILE; MEASUREMENT NOT PROCESSED

A file management error occurred while attempting to read the data file.

REPORT OR PLOT NOT COPIED

The COPY command was not executed.

REPORT OVERFLOW ON n ENTITIES; PLEASE REQUEST A SMALLER ENTITY SET

There are too many individual entities in the current entity set for XRAYSCAN to generate a report. Restrict the entity set with one or more IF commands.

SORRY, BUT ONLY THE FIRST TWENTY-SIX ENTITIES WILL FIT ON THE PLOT

The maximum number of items allowed in any one plot is 26.

SUBSYSTEM ERROR: PLEASE REPORT THIS TO TANDEM COMPUTERS, INC TRAP CAUSE: ## AT LOCATION %%%%%%% WITH E = %%%%%%%, L = %%%%%%%, S = %%%%%%%

THIS CHARACTER SEEMS OUT OF PLACE; COMMAND NOT EXECUTED

THIS IS NOT A LEGAL FILENAME

THIS REPORT HAS NO SUCH ITEM; COMMAND NOT EXECUTED

THIS WORD SEEMS OUT OF PLACE; COMMAND NOT EXECUTED

UNABLE TO CREATE THE TEMPORARY FILE #nnnn DUE TO FILE SYSTEM ERROR nn

UNABLE TO OPEN THE TEMPORARY FILE #nnnn DUE TO FILE SYSTEM ERROR nn

-->

UNABLE TO OPEN THE XRAYDATA FILE; MEASUREMENT NOT PROCESSED

UNABLE TO UNDERSTAND MY OWN INITIAL COMMAND

Report this error to Tandem Computers Incorporated.

UNEXPECTED DATA (OR FILE END) IN THE XRAY DATA FILE; ANALYSIS ABORTED

WARNING: REMOTE REPORT REQUESTED ON LOCAL-ONLY SYSTEM

A report of remote activity was requested on a system incapable of being connected to a network.

XRAY DATA FILE MUST BE ON DISC; MEASUREMENT NOT PROCESSED

Although a tape file may be used as the XRAY data file, it must be copied to disc before XRAYSCAN can process it.

THE CURRENT REPORT HAS NO SUCH PROCESS

The PROCEDURE command failed because the previous command was not a PROCESS command, or no displayed PROCESS had a PROCEDURES under measurement.

```
XRAYSCAN  <data file>
--------  -----------

BUFFER <process entity set>
------ --------------------

   <process entity set> is one of:

       { (blank)                     }
       { <cpu>                       }
       { <cpu>,<pin>                 }
       { <discfileset>               }
       { <discfileset>,<cpu>         }
       { <discfileset>,<cpu>,<pin>   }
       { <process name>              }

[ <entity selection command> , ] BY <item>
                                 --
COPY  [ <file name> | ! ]
----

CPU <cpu entity set> [ , <BY clause> ] [ , <IF clause> ]
--- ----------------

   <cpu entity set> is

       { (blank) }
       { <cpu>   }

DELTA [ <interval> ]        ! <interval> = { .001:2147482.999 }
-----

DEVICE <device entity set> [ , <BY clause> ] [ , <IF clause> ]
------ -------------------

   <device entity set> is one of:

      { (blank)                                               }
      { <cpu>                                                 }
      { <cpu> , <controller>                                  }
      { <cpu> , <controller> , <unit>                         }
      { $<device name>                                        }
      { $<device name> , <cpu>                                }
      { $<device name> , <cpu> , <controller>                 }
      { $<device name> , <cpu> , <controller> , <unit>        }

                                                            -->
```

```
--------------------------------------------------------------------------
|                                                                        |
| DISC <disc entity set> [ , <BY clause> ] [ , <IF clause> ]             |
| ---- -----------------                                                 |
|                                                                        |
|    <disc entity set> is one of:                                        |
|                                                                        |
|       { <blank)                                                     }  |
|       { <cpu number>                                                }  |
|       { <cpu number> , <controller number>                          }  |
|       { <cpu number> , <controller number> , <unit number>          }  |
|       { $<volume>                                                   }  |
|       { $<volume> , <cpu number>                                    }  |
|       { $<volume> , <cpu number> , <controller number>              }  |
|       { $<volume> , <cpu number> , <controller number>                 |
|            , <unit number>                                          }  |
|                                                                        |
| DISCOPEN <discopen entity set>                                         |
| -------- ---------------------                                         |
|    [ , <BY clause> ] [ , <IF clause> ]                                 |
|                                                                        |
|    <discopen entity set> is one of:                                    |
|                                                                        |
|       { (blank)                                              }         |
|       { <cpu>                                                }         |
|       { <cpu> , <pin>                                        }         |
|       { <cpu> , <pin> , <file number>                        }         |
|       { <discfile set>                                       }         |
|       { <discfile set> , <cpu>                               }         |
|       { <discfile set> , <cpu> , <pin>                       }         |
|       { <discfile set> , <cpu> , <pin> , <file number> }               |
|                                                                        |
|          <discfile set> is                                             |
|                                                                        |
|             { <local discfile set>                 }                   |
|             { \<systemname>                        }                   |
|             { \<systemname>.<local discfile set> }                     |
|                                                                        |
|             <local discfile set> is one of                             |
|                                                                        |
|                                                                        |
|                { $<volume>.<subvolume>.<file name> }                   |
|                { $<volume>.<subvolume>.*           }                   |
|                { $<volume>.*.*                     }                   |
|                { $<volume>.#<temporary file>       }                   |
|                                                                        |
|             <systemname> is one of                                     |
|                                                                        |
|                { <sysname>        }                                    |
|                { <systemnumber>   }                                    |
|                                                                        |
|                                                                        |
|                                                                    --> |
--------------------------------------------------------------------------
```

```
FILE <file entity set> [ , <BY clause> ] [ , <IF clause> ]
---- -----------------

  <file entity set> is one of:

     { (blank)                                  }
     { <cpu>                                    }
     { <cpu> , <pin>                            }
     { <cpu> , <pin> , <file number>            }
     { <fileset>                                }
     { <fileset> , <cpu>                        }
     { <fileset> , <cpu> , <pin>                }
     { <fileset> , <cpu> , <pin> , <file number> }

     <fileset> is one of

        { <local discfile set>                }
        { $<device name>                      }
        { <subdevice>                         }
        { \<systemname>                       }
        { \<systemname>.<local discfile set>  }
        { \<systemname>.<device name>         }
        { \<systemname>.<subdevice>           }

           where <subdevice> is one of

              { $<linename>.*              }
              { $<linename>.#<subdevice>   }

[ <entity selection command> , ] IF <condition>
                                  -- -----------
                                     [ , <condition > ... ]

  <condition> is

    { <item> [ { "<" | ">" } <num> ] }
    { ?                              }
```

-->

```
--------------------------------------------------------------------------
|                                                                        |
| LINE <line entity set> [ , <BY clause> ] [ , <IF clause> ]             |
| ---- -----------------                                                 |
|                                                                        |
|    <line entity set> is one of:                                        |
|                                                                        |
|       { (blank)                                             }          |
|       { <cpu>                                               }          |
|       { <cpu> , <controller>                                }          |
|       { <cpu> , <controller> , <unit>                       }          |
|       { $<line name>                                        }          |
|       { $<line name> , <cpu>                                }          |
|       { $<line name> , <cpu> , <controller>                 }          |
|       { $<line name> , <cpu> , <controller> , <unit>        }          |
|                                                                        |
|                                                                        |
| NETLINE <line entity set> [ , <BY clause> ] [ , <IF clause> ]          |
| ------- -----------------                                              |
|                                                                        |
|    <line entity set> is one of:                                        |
|                                                                        |
|       { (blank)                                             }          |
|       { <cpu>                                               }          |
|       { <cpu> , <controller>                                }          |
|       { <cpu> , <controller> , <unit>                       }          |
|       { $<line name>                                        }          |
|       { $<line name> , <cpu>                                }          |
|       { $<line name> , <cpu> , <controller>                 }          |
|       { $<line name> , <cpu> , <controller> , <unit>        }          |
| NEWPLOT                                                                |
| -------                                                                |
|                                                                        |
| OUTLEN  <width>                       ! <width> = { 72:132 }           |
| ------  -------                                                        |
|                                                                        |
| PLOT  [ <item> ]                                                       |
| ----                                                                   |
|                                                                        |
|                                                                        |
| POOL [<cpu>]                                                           |
|                                                                        |
| ----                                                                   |
|                                                                        |
| PROCEDURE                                                              |
|                                                                        |
| ---------                                                              |
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                        |
|                                                                    --> |
--------------------------------------------------------------------------
```

```
PROCESS <process entity set> [ , <BY clause> ] [ , <IF clause> ]
------- --------------------

   <process entity set> is one of:

      { (blank)                        }
      { <cpu>                          }
      { <cpu> , <pin>                  }
      { <discfile set>                 }
      { <discfile set> , <cpu>         }
      { <discfile set> , <cpu> , <pin> }
      { <process name>                 }

SCALE [ [ <lower bound> ] [ , <upper bound> ] ]
-----
         ! range = { 0:22147482.999 }


SYSTEM <system entity set> [ , <BY clause> ] [ , <IF clause> ]
------ -------------------

    <system entity set> is

       { (blank)          }
       { \<sysname>        }
       { \<systemnumber>   }

TERMINAL <terminal entity set>
-------- ---------------------
   [ , <BY clause> ] [ , <IF clause> ]

   <terminal entity set> is one of:

      { (blank)                    }
      { $<terminal name>           }
      { $<terminal name> , <cpu>   }
      { <subdevice>                }
      { <subdevice>.<cpu>          }
```

```
---------------------------------------------------------------------
|                                                                   |
|          [ [ + | - ] <number of seconds> ]                        |
| WINDOW   [ [ <time> ] [ , <time> ]       ]                        |
| ------                                                            |
|                                                                   |
|    <number of seconds>  is  { 0:2147482.999 }                     |
|                                                                   |
|    <time>  is { <date and time> }    }                            |
|               { <number of seconds> }                             |
|                                                                   |
|       <date and time>  is [ <date> , ] <time of day>              |
|                                                                   |
|                       { <month> <day> }                           |
|          <date>  is   { <day> <month> } [ <year> ]                |
|                                                                   |
|          <time of day>  is <hour>:<minute>[:<second>]             |
|                                                                   |
|             <day>  is { 1:31 }                                    |
|                                                                   |
|             <month>  is  { JAN | FEB | MAR | APR | MAY | JUN |    |
|                            JUL | AUG | SEP | OCT | NOV | DEC }    |
|                                                                   |
|             <year>  is  { 0:2047 }                                |
|                                                                   |
|             <hour>  is  { 0:23 }                                  |
|                                                                   |
|             <minute>  is  { 0:59 }                                |
|                                                                   |
|             <second>  is  { 0:59 }                                |
|                                                                   |
---------------------------------------------------------------------
```

ADVANCED
CONSIDERATIONS

ADVANCED CONSIDERATIONS

This section contains material on two topics of importance for those seriously interested in measuring computer systems:

- Performance basics

  What sort of concepts are used in the effort to make your computer run faster? Some of the theoretical ideas involved are presented here.

- Sampling errors

  When XRAY says that CPU 0 is 43% busy, how busy is CPU 0? If you said 43%, you should read this.

These topics have in common the fact that, in each case, we have so far presented only the most basic information. The Performance Basics section examines the subject of modeling of computer systems, while the Sampling Errors section discusses the theory of statistics. Those becoming expert in the use of XRAY will eventually find themselves delving into one or both of these subjects.

This section contains a brief introduction to the subject of modeling systems to make them run better. This material is derived from Buzen, J. P., "Fundamental Operational Laws of Computer System Performance," in Acta Informatica 7, 167-182 (1976), Springer Verlag. The reader interested in pursuing this subject is urged to read that paper, as well as the references in its bibliography.

A system is considered to be made up of a number of devices (including processors), numbered consecutively from 1 through "n." The system is measured over a time interval, I, and the following definitions are made:

trans = total number of transactions occurring during I

> The meaning of "a transaction" is specified by the application under measurement. A transaction could be a data base update, a program compilation, or perhaps the compilation of a single source line. It doesn't matter as long as you're consistent throughout the discussion.

throughput = trans / I

> This is the number of transactions per second. The goal of computer system analysis is to maximize throughput.

busy(j) = amount of time during I that device j is busy

util(j) = utilization of device j = busy(j) / I

> Note that 0 <= util(j) <= 1.

demand(j) = busy(j) / trans

> This is the amount of time device j is in service, divided by the number of transactions; thus, it is the average amount of time device j spends on each transaction.

An easily-determined, but important, result of these definitions is the Throughput Law.

Throughput Law: throughput = util(j) / demand(j)

That is, the system throughput is equal to the utilization of any device, divided by the demand for that device. As proof:

$$\frac{\text{util(j)}}{\text{demand(j)}} = \frac{\text{busy(j) / I}}{\text{busy(j) / trans}} = \text{trans / I} = \text{throughput}$$

It is important to observe that the throughput law is independent of "device j"; specifically, this implies that

$$\frac{util(j)}{demand(j)} = \frac{util(k)}{demand(k)} \quad \text{for any two devices j and k.}$$

In light of the throughput law, it can be seen that balancing the system does not imply attempting to achieve equal usage of all components. Equal demand for all devices is not a practical state of affairs in the solution of most real problems.

Note that util(j) <= 1; this puts a physical ceiling on system throughput, unless demand(j) can be reduced.

There are two important, independent considerations that could prevent util(j) from ever reaching 1.

- Serial dependence: Another single device is required in series with each request for this device, so this device can never be busy all the time. The new physical ceiling is:

  $$ceiling(j) = \frac{service(j)}{prep(j) + service(j)}$$

  where service(j) is the amount of time device j spends working, and prep(j) is the amount of time it waits for the other device.

- When the system is handling multiple concurrent transactions, it is not desirable to operate a device too close to its ceiling, or queueing delays may inflate response times. The optimal operating point is dependent on many factors, but adopting a practical ceiling equal to six-tenths of the physical ceiling is generally considered to be conservative. Thus, we have a practical upper bound on util(j):

  $$maxutil(j) = .6 * \frac{service(j)}{prep(j) + service(j)}$$

  When the system is handling a relatively small number of concurrent transactions, queueing delays (at equal priorities) cannot become severe, and maxutil can be taken as the physical ceiling.

Finding the Bottleneck

Now we can identify the bottleneck in the system as the device with the smallest ratio

maxutil(j) / demand(j)

since this is the maximum throughput value that device can attain. (The "floodgate" is the device with the greatest such ratio.) The bottleneck can be removed by reducing demand(j) until the throughput is sufficiently large to expose the next bottleneck.

The throughput law applies to any elapsed time greater than zero, so when answering the question "why won't it go faster?" we will look for intervals in which there is a drop in util(j) for some device involved in the activity. The device of choice is often the processor, since it is clear that the application needs to execute a finite set of processor instructions before completing the transaction. A drop in usage of the processor by the transaction indicates a reduction in forward progress of the algorithm. Drops in usage are associated (via the throughput law) with drops in throughput. Then we determine where the time was spent instead, by looking for increases in the usage of other devices.

The idea is to manipulate demand(j) to drive util(j) toward our conservative maxutil(j). The manipulation of demand(j) generally involves configuration changes, such as moving or partitioning files, or moving processes to other processors.

We may also discover that util(j) is less than maxutil(j) for all devices, showing hidden sequencing constraints between devices. Sometimes these can be eliminated (using no-wait I/O, for example). Thus, without changing demand(j), util(j) is forced closer to our conservative maxutil(j), and throughput consequently increases.

Sequencing

Suppose there is no device operating at, or even near, the

maxutil(j) / demand(j)

throughput limit. This indicates a problem in which delays are being introduced because some device requires the activity of another device in series with its own activity.

```
    -> |-----------------------|------------------------| - ->
  |       Device A prepares for      Activity:  Device B        |
  |       activity on device B    proceeds to give service      |
  |                                                             |
   <- - - - - - - - - - - - - - <- - - - - - - - - - - - - - - <-
```

You can see that device B cannot be busy all the time because of the preparation time required by device A.

There are a variety of cures to sequencing. The best is to do the same job but do activity B less often. Or, if the preparation done by device A can be performed by a set of multiple devices instead of just one, device B need not wait for device A at the end of each activity. Another possibility is to restructure for a no-wait mechanism to obtain some overlap of A with B.

Sequencing effects are particularly common in the "stand-alone," or single-program, environment. Note that tuning a program in this environment may result in the elimination of sequencing problems that would not arise if the data base or the mix were real.

Notice how a dilution of the usage of device B -- due to sequencing with device A -- will necessarily be reflected by an increase in the usage of device A.

If activity A involves human intervention, the resultant loss of utilization is called system "slack." When the usage of a device is low due to system slack, this can be detected by the failure to find an "activity A" device; that is, by failure to find a device on which utilization increases as that of device B drops.

The detection of slack can be difficult; the TRAN RATE terminal item is intended to help uncover system slack.

Determining Demand

The application defines what a transaction is. Therefore, finding demand(j) from the XRAY data requires knowledge of the application.

There is nothing mysterious about demand(j). It is simply the number of seconds device j required for each transaction (on the average, over the elapsed time window in use).

$$\text{demand(j)} = \frac{\text{seconds of device use during WINDOW}}{\text{transactions during WINDOW}}$$

Knowing the application, you must determine the number of transactions that occurred during the WINDOW; often XRAY can provide this number in terms of the number of inputs from a particular file or some other application-dependent indicator of that type.

Now the seconds of device use during the WINDOW can be obtained by entering the NORATE command. This causes all the following data to be displayed without being divided by the WINDOW time. SEEK RATE becomes the number of seeks, and CPU BUSY becomes seconds of CPU TIME. (To return to normal operations, enter the RATE command to XRAYSCAN.)

Note that once the throughput law's ratio has been computed for some device, demand(j) can subsequently be obtained from util(j) for all other devices (and vice versa).

Isolating Transaction Activity in a Mix

XRAY takes a system-level view of performance problems. Thus, if your transaction of interest is the only thing occurring on the system at a given point, the computation of demand(j) is performed directly by multiplying the device's BUSY item by the WINDOW.

This "transaction" that is being measured on the system may well be a mix of many different transaction types. Yet the computation of demand(j) will still work, provided the data averaged over the several transaction types will answer your performance question, and provided you have some way to count the number of transactions that occurred during the WINDOW.

But frequently we are concerned with the behavior of a particular transaction in a mix of other transactions. We can still apply the throughput law if we can determine the portion of device usage resulting from the particular type of transaction in question.

XRAY permits the isolation of many data items to the particular file opening and the originating process. (If this is insufficient, it will be necessary to measure the particular transaction type in a stand-alone environment.)

Now we illustrate, briefly, a general technique for isolating demand(j, class) for some particular transaction of type "class."

Demand(j, class) can be found from the amount of time needed to service a request for the device by transactions of type "class," and the number of requests for the device by transactions of type "class." This technique works only if it is true (to a first approximation) that

```
    service(j) = service(j, class)
```

for all transaction classes. If so, then

```
                          service(j) * visits(j, class)
    demand(j, class)  =  -----------------------------
                               transactions(class)
```

The number of visits to the device by transactions of type "class" can usually be found by examining the FILE and DISC fileset reports with XRAYSCAN for those entities used by transactions of type "class". Sometimes the message statistics for a process can help determine the number of transactions of the given class -- represented above as transactions(class) -- that occurred during the elapsed time.

The average service time of the device can be obtained from the XRAYSCAN output by dividing the number of device busy seconds by the total I/O operations on the device. The device busy seconds are computed by multiplying the WINDOW by the DEVICE BUSY percentage.

Assume that util(j, class) is the usage of the device by the class:

```
util(j, class)  =  service(j) * visits(dev, class) / I
```

Then the reduction of the throughput law to a transaction of a specific class is

```
throughput(class)  =  util(dev, class) / demand(dev, class)
```

In this way the requirements by transactions of the type "class" for the device j can be deduced from measurements of a mix containing those transactions.

Sampling

Certain XRAY statistics are obtained by sampling, rather than by counting each individual event. A processor's CPU BUSY statistic, for example, is one such item. A high-priority interrupt occurring an average of 27.6 times per second allows XRAY to estimate the true CPU BUSY, based on the number of times XRAY found the processor busy at the time of the interrupt.

Sampling allows XRAY to reduce its overhead. However, any sampling technique will introduce some amount of sampling error. Though we cannot hope to eliminate this source of inaccuracy, we can precisely quantify it. In particular, we can state the likelihood, based on the number of samples taken, that the sampling error is less than a given tolerance.

The following pages present:

- A list of XRAY's sampled items, along with their sampling rates
- A set of tables showing the probability that the sampling error is within a certain tolerance, given the sampling rate and the number of seconds over which the samples were taken
- An explanation of how the tables were obtained

Sampling Rates

The following table indicates the number of samples per second used to calculate each sampled item. Items not on this list are explicitly counted rather than sampled.

SAMPLING RATES OF SAMPLED ITEMS

CPU:		DISC:		LINE:	
CPU BUSY	27.6	DISC BUSY	9.2	LINE BUSY	9.2
SEND BUSY	9.2	XFER BUSY	9.2	READ BUSY	9.2
CPU QLEN	9.2	SEEK BUSY	9.2	WRITE BUSY	9.2
DEVICE:		PROCESS:		NETLINE:	
DEV BUSY	9.2	CPU BUSY	27.6	READ BUSY	9.2
READ BUSY	9.2	RECV QLEN	27.6	WRITE BUSY	9.2
WRITE BUS	9.2	PRES PAGES	27.6		

Sampling Error Tables

To use the tables on the following page, first determine the acceptable amount of sampling error that your purpose can tolerate. Deciding that your acceptable sampling error is .1, for example, means that you will be satisfied if XRAY's value for an item is within 10%, in either direction, of that item's true value.

There are tables for acceptable sampling errors of .01, .05, .1, and .15. After choosing the appropriate table, find the length, in seconds, of the interval during which the item was sampled in the left column. This is the WINDOW interval, if the item is listed in a report, or the DELTA interval, if the item is displayed on a time plot.

Then, look up the number of samples taken per second for the item (the list is on the page previous to this). The 4-digit decimal number found at the intersecting row and column is the probability that the sampling error introduced by XRAY is acceptable.

Example:

A time plot with a DELTA interval of 10 seconds reveals that CPU 1 was 43% busy between 12:00:00 and 12:00:10. The probability is .9990 (almost certain) that CPU 1 was busy between 33% and 53% (that is, within 10% of the stated 43%) during that time interval. There is only a .2586 chance that CPU 1 was between 42% and 44% busy (0.1% sampling error) during the interval.

A discussion of how the probabilities were derived is presented following the tables.

sampling error = .01

	27.6 samples/sec	9.2 samples/sec
1	.0796	.0478
5	.1895	.1114
10	.2586	.1506
15	.3182	.1820
20	.3616	.2128
30	.4380	.2586
60	.5820	.3616
120	.7498	.4908
300	.9312	.7062
600	.9898	.8638
1200	.9998	.9642

sampling error = .05

	27.6 samples/sec	9.2 samples/sec
1	.4038	.2358
5	.7580	.5034
10	.9030	.6630
15	.9576	.7580
20	.9808	.8262
30	.9960	.9030
60	*	.9808
120	*	.9990
300	*	*
600	*	*
1200	*	*

sampling error = .1

	27.6 samples/sec	9.2 samples/sec
1	.7062	.4514
5	.9812	.8262
10	.9990	.9452
15	*	.9812
20	*	.9932
30	*	.9990
60	*	*
120	*	*
300	*	*

sampling error = .15

	27.6 samples/sec	9.2 samples/sec
1	.8836	.6372
5	.9996	.9576
10	*	.9958
15	*	.9996
20	*	*
30	*	*
60	*	*
120	*	*
300	*	*

* The probability is 1.0000 to four decimal places

How the Tables were Derived

For the benefit of those interested in either finding probabilities not listed in the tables, or establishing confidence in the values that are present, this section describes how those values were obtained.

Given a population whose members fall into one of two classes, we want to determine the percentage of members in each class. In addition, we assume that it is impractical to actually count each member. An easily-visualized example would be an enormous bowl containing one million marbles, some black and some white. We plan to pull out a handful and thereby estimate the relative proportions of black and white marbles in the sample. The question is: How big a handful do we need to be reasonably certain that the percentage of black marbles in our handful is reasonably close to the actual percentage of black marbles in the bowl?

In the case of a processor on the Tandem 16, we can visualize a "bowl" containing a collection of instants. During each instant, the processor is either busy or not; our problem is to determine how accurately we can estimate the true number of instants during which the processor was busy, by looking at a relatively small (27.6 per second) number of them.

Note that what we mean by an "instant" in this context is actually a small but positive interval of time; for example, the 100-nanosecond microinstruction cycle time. For a processor, it is meaningless to divide time any finer than that, so we are justified in modeling time as a large (but finite) collection of small, positive intervals, which we choose to call "instants."

If we let

- P = true percentage of black marbles (or true CPU BUSY),
- Q = 1 - P,
- d = magnitude of the sampling error (that is, the amount by which the percentage of black marbles in our handful differs from the true percentage of black marbles in the bowl), and
- t be related to the probability that our sampled statistic is within "d" percent of the true proportion (exactly how it's related is discussed presently),

then

$$d = \frac{t * \mathrm{sqrt}(P * Q)}{\mathrm{sqrt}(n)}$$

(according to Cochran, William G., Sampling Techniques, third edition, John Wiley and Sons, pp 75ff).

Note that this formula requires that we know P and Q, which are the very quantities we're trying to estimate. We can get around this difficulty by observing that the larger the product P * Q, the larger the numerator on the right side of the above equation, hence the larger d will be. Thus we need to put an upper bound on P * Q. This is simple: Since

$P + Q = 1$

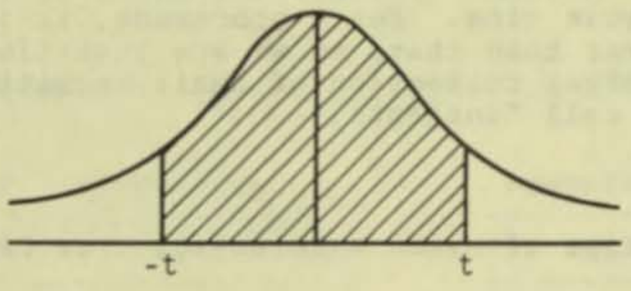
and both P and Q lie between 0 and 1, the maximum value of P * Q occurs when both are 1/2, and their product is 1/4 .

Therefore:

$$d = \frac{t * \mathrm{sqrt}(P * Q)}{\mathrm{sqrt}(n)} <= \frac{t * \mathrm{sqrt}(1/4)}{\mathrm{sqrt}(n)} = \frac{t}{2 * \mathrm{sqrt}(n)}$$

Now all that remains is to define what t is, and then we can calculate tables of d and n at will.

You may be familiar with the "bell-shaped curve," which represents the "normal distribution." A picture of this curve is shown below. Its key property is that the total area under the curve is 1.



Recall that we said previously that t is related to the probability that our sampled statistic is within "d" percent of the true proportion. The exact manner in which t is related to that probability is that t is the point on the X axis with the property that the shaded area (the area bounded above by the curve, below by the X axis, and on the sides by the lines X = t and X = -t) is equal to the probability that the proportion of black marbles in the sample is within "d" percent of the proportion of black marbles in the bowl, or, in the actual case of interest, that the CPU BUSY statistic measured by XRAY is within "d" percent of the amount of time the processor was actually busy.

A specific example:

Suppose that we sample CPU BUSY for some processor over an interval of five seconds. Therefore, n, the total number of samples, is 5 * 27.6, or 138.

Suppose further that it is important to us that the number XRAY gives us should be within 5% of the true CPU BUSY statistic for those five seconds.

Setting d = .05, n = 138, and solving for t, using the formula given previously, gives t = 1.28 (approximately). With t = 1.28, the shaded area under the bell curve is .7580. Therefore, we are 76% certain that XRAY's CPU BUSY statistic is within 5% of the true proportion of time that the processor was busy.

At this point you may well ask how we knew that the area under the curve corresponding to t = 1.28 was .76. The answer is simply that this number, as well as all the other numbers in the tables, were looked up in

Handbook of Tables for Probability and Statistics, 2nd edition, edited by William Beyer, published by the Chemical Rubber Company of Cleveland, pages 125ff.

Note that statistics such as MSG QLEN are not covered by the preceding discussion, since members of the underlying population can take on more than two values. That is, the length of a queue, at any instant, can be 0, 1, 2, ...; this is a much more complicated situation than the case of a processor, which at any instant is either busy or not.

APPENDIX

FILE MANAGEMENT ERROR LIST

<error>		description	<device type>
CCE			
0		operation successful	any
CCG			
1		end-of-file	3,4,6,8
2		operation not allowed on this type file	any
3		failure to open or purge a partition	3E
4		failure to open an alternate key file	3E
5		failure to provide sequential buffering	3E
6		system message received	2
7		process not accepting CONTROL or SETMODE	0
CCL			
10	(%12)	file/record already exists	3
11	(%13)	file not in directory or record not in file	3
12	(%14)	file in use	3 - 8
13	(%15)	illegal filename specification	any
14	(%16)	device does not exist	3 - 8
15	(%17)	volume specification supplied does not match name of volume on which the file actually resides	3
16	(%20)	file number has not been opened	any
17	(%21)	paired-open was specified and the file is not open by the primary process, the parameters supplied do not match the parameters supplied when the file was opened by the primary, or the primary process is not alive	any

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
18 (%22)	the referenced system does not exist	any
20 (%24)	attempted network access by a process with a five character name, or a seven character home terminal name	any
21 (%25)	illegal <count> specified	any
22 (%26)	application parameter or buffer address out of bounds	any
23 (%27)	illegal disc address	3
25 (%31)	AWAITIO or CANCEL attempted on "wait" file	any
26 (%32)	AWAITIO or CANCEL attempted on a file with no outstanding operations	any
27 (%33)	"wait" operation attempted when outstanding requests pending	any
28 (%34)	number of outstanding no-wait operations would exceed that specified at OPEN, or attempt to open a disc file or $RECEIVE with maximum no. of concurrent operations greater than 1	any
29 (%35)	missing parameter	any
30 (%36)	unable to obtain main memory space for a link control block	0,1,3 - 8
31 (%37)	unable to obtain SHORTPOOL space for a file system buffer area	any
32 (%40)	unable to obtain main memory space for a control block	any
33 (%41)	I/O process is unable to obtain IOPOOL space for an i/o buffer or <count> too large for dedicated i/o buffer	1,3 - 8

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
40 (%50)	operation timed out. AWAITIO did not complete within the time specified by its <time limit> parameter. If a 0D <time limit> (completion check) or -1 <file number> (any file) was specified, then the operation is considered incomplete. Otherwise, the operation is considered completed	any
42 (%52)	attempt to read from unallocated extent	3
43 (%53)	unable to obtain disc space for extent	3
44 (%54)	directory is full	3
45 (%55)	file is full	3
46 (%56)	invalid key specified	3E
47 (%57)	key not consistent with file data	3E
48 (%60)	security violation	3
49 (%61)	access violation	any
50 (%62)	directory error	3
51 (%63)	directory is bad	3
52 (%64)	error in disc free space table	3
53 (%65)	file system internal error	3
54 (%66)	i/o error in disc free space table	3
55 (%67)	i/o error in directory	3
56 (%70)	i/o error on volume label	3
57 (%71)	i/o error in file label	3
58 (%72)	disc free space table is bad	3
59 (%73)	file is bad	3

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
60 (%74)	volume on which this file resides has been removed or device has been downed since the file was opened	3 - 8
61 (%75)	no file opens are permitted	3
62 (%76)	volume has been mounted, but mount order has not been given, file open not permitted	3
63 & 64 (%77 & %100)	volume has been mounted and mount is in progress, file open not permitted	3
65 (%101)	only special requests permitted	3
66 (%102)	device has been downed by operator	1,3 - 8
71 (%107)	duplicate record	3E
72 (%110)	attempt to access unmounted partition	3
73 (%111)	file/record locked	3
74 (%112)	READUPDATE called for $RECEIVE and number of messages queued exceeds <receive depth> - or REPLY called with an invalid <message tag> - or REPLY called and no message is outstanding	2
99 (%143)	Enscribe option not installed	3
100 (%144)	device not ready	3,4,5,6,8
101 (%145)	no write ring	4
102 (%146)	paper out	5
103 (%147)	disc not ready due to power fail	3
110 (%156)	only break access permitted	6
111 (%157)	operation aborted because of break	6
112 (%160)	READ or WRITEREAD preempted by operator message	6

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
120 (%170)	data parity error	1,3 - 7
121 (%171)	data overrun error	1,3 - 8
130 (%202)	illegal address to disc	3
131 (%203)	write check error from disc	3
132 (%204)	seek incomplete from disc	3
133 (%205)	access not ready on disc	3
134 (%206)	address compare error on disc	3
135 (%207)	write protect violation with disc	3
136 (%210)	unit ownership error (dual-port disc)	3
137 (%211)	controller buffer parity error	3
140 (%214)	modem error (communication link not yet established, modem failure, momentary loss of carrier, or disconnect)	6,7
	unexpected interrupt from auto-call unit	7.56
145 (%221)	card reader motion check error	8
146 (%222)	card reader read check error	8
147 (%223)	card reader invalid Hollerith code read	8
150 (%226)	end-of-tape marker detected	4
151 (%227)	runaway tape detected	4
152 (%230)	unusual end - tape unit went off line	4
153 (%231)	tape drive power on	4
154 (%232)	BOT detected during backspace files or backspace records	4
160 (%240)	request is invalid for line state	7

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
161 (%241)	impossible event occurred for line state	7
162 (%242)	operation timed out	7
163 (%243)	EOT received	7
	power at auto-call unit is off	7.56
164 (%244)	disconnect received	7
	data line is occupied (busy)	7.56
165 (%245)	RVI received	7
	data line is not occupied after setting call request	7.56
166 (%246)	ENQ received	7
	auto-call unit failed to set "present next digit"	7.56
167 (%247)	EOT received on line bid	7
	"data set status" is not set after dialing all digits	7.56
168 (%250)	NAK received on line bid	7
	auto-call unit failed to clear "present next digit" after "digit present" was set	7.56
169 (%251)	WACK received on line bid	7
	auto-call unit set "abandon call and retry"	7.56
170 (%252)	no id sequence received during circuit assurance mode	7
171 (%253)	no response received	7
172 (%254)	reply not proper for protocol	7

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
173 (%255)	maximum allowable NAKs received	7
174 (%256)	WACK received	7
175 (%257)	incorrect alternating ACK received	7
176 (%260)	poll sequence ended with no responder	7
177 (%261)	text overrun (insufficient buffer space for data transfer)	7
178 (%262)	no address list specified	7
190 (%276)	invalid status received from device	1,3 - 8
200 (%310)	device is owned by alternate port	1,3 - 8
201 (%311)	the current path to the device is down	1,3 - 8
	an attempt was made to write to a non-existent process	0
210 (%322)	device ownership changed during operation	1,3 - 8
211 (%323)	failure of CPU performing this operation	any
212 (%324)	EIO instruction failure	1,3 - 8
213 (%325)	channel data parity error	1,3 - 8
214 (%326)	channel timeout	1,3 - 8
215 (%327)	i/o attempted to absent memory page	1,3 - 8
216 (%330)	map parity error during this i/o	1,3 - 8
217 (%331)	memory parity error during this i/o	1,3 - 8
218 (%332)	interrupt timeout	1,3 - 8
219 (%333)	illegal device reconnect	1,3 - 8
220 (%334)	protect violation	1,3 - 8

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
222 (%336)	bad channel status from EIO instruction	1,3 - 8
223 (%337)	bad channel status from IIO instruction	1,3 - 8
230 (%346)	CPU power failed then restored	1,3 - 8
231 (%347)	controller power failed then restored	1,3 - 8
248 (%370)	a line handler process failed while this request was outstanding. The file system recovers from this error for files opened with non-zero sync depth	any
249 (%371)	a network failure occurred while this request was outstanding. The file system recovers from this error for files opened with non-zero sync depth	any
250 (%372)	the referenced system is down	any
251 (%373)	a network protocol error occurred	any

n	2**n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1 024
11	2 048
12	4 096
13	8 192
14	16 384
15	32 768
16	65 536
17	131 072
18	262 144
19	524 288
20	1 048 576
21	2 097 152
22	4 194 304
23	8 388 608
24	16 777 216
25	33 554 432
26	67 108 864
27	134 217 728
28	268 435 456
29	536 870 912
30	1 073 741 824
31	2 147 483 648
32	4 294 967 296
33	8 589 934 592
34	17 179 869 184
35	34 359 738 368
36	68 719 476 736
37	137 438 953 472
38	274 877 906 944
39	549 755 813 888
40	1 099 511 627 776
41	2 199 023 255 552
42	4 398 046 511 104
43	8 796 093 022 208
44	17 592 186 044 416
45	35 184 372 088 832
46	70 368 744 177 664
47	140 737 488 355 328
48	281 474 976 710 656
49	562 949 953 421 312
50	1 125 899 906 842 624

51	2 251 799 813 685 248
52	4 503 599 627 370 496
53	9 007 199 254 740 992
54	18 014 398 509 481 984
55	36 028 797 018 963 968
56	72 057 594 037 927 936
57	144 115 188 075 855 872
58	288 230 376 151 711 744
59	576 460 752 303 423 488
60	1 152 921 504 606 846 976
61	2 305 843 009 213 693 952
62	4 611 686 018 427 387 904
63	9 223 372 036 854 775 808

INDEX

A02

READER'S COMMENTS

Tandem welcomes your feedback on the quality and usefulness of its publications. Please indicate a specific *section* and *page* number when commenting on any manual. Does this manual have the desired completeness and flow of organization? Are the examples clear and useful? Is it easily understood? Does it have obvious errors? Are helpful additions needed?

Title of manual(s): ______________________________

FOLD ➤

FOLD ➤

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

City/State ______________________ Zip __________

A written response is requested, yes no ?

NonStop™
SYSTEMS

ENTRY™ SCREEN FORMATTER OPERATING & PROGRAMMING MANUAL

T16/8020

TANDEM 16

TANDEM 16

ENTRY (TM)

SCREEN FORMATTER

OPERATING

AND

PROGRAMMING MANUAL

TANDEM COMPUTERS INCORPORATED
20605 Valley Green Drive
Cupertino, California 95014

Printed July 1, 1977
Printed in U.S.A.

Product no. T16/8020

Pages	Effective Date	Software Product and Version
Title	July 1977	
List of Effective Pages	July 1977	
Preface	July 1977	
Table of Contents	July 1977	
1-1 thru 1-2	July 1977	T9608B01
2-1 thru 2-11	July 1977	T9608B01
3-1 thru 3-2	July 1977	T9608B01
4-1 thru 4-10	July 1977	T9608B01
A-1 thru A-14	July 1977	T9608B01
B-1 thru B-6	July 1977	T9608B01

MANUAL VERSION A00. DATE: JULY 1977

This manual is organized into six sections:

Section 1.	Introduction
Section 2.	Form Creation
Section 3.	Field Access
Section 4.	ENTRY Procedures
Appendix A	Sample Program
Appendix B	File Management Error Codes

For more information regarding the Tandem 16 Computer System, refer to the these manuals:

* Tandem 16 System Description [Product no. T16/8000]

- Overview of the Tandem 16 system, hardware and software
- Details of the hardware from a programming standpoint
- Tandem 16 approach to NonStop programming
- Tandem 16 machine instructions
- Description of the operating system and its responsibilities

* Tandem 16 Standard Programming Manual [Product no. T16/8012]

- Introduction to writing programs for the Tandem 16 Computer System
- Description of the T/TAL Programming Language
- File Management System calls
- Input/output devices
- System calls for running, suspending and stopping programs
- System Utility Procedures
- Checkpointing Facility
- Security Facility
- Interactive Debugging Facility
- NonStop programming example
- General Purpose Procedures

* Tandem 16 Operating Manual [Product no. T16/8019]

- Interactive Command Interpreter (COMINT)
- Interactive Text Editor (EDIT)
- File Utility Program (FUP)
- Backing up/restoring disc files (BACKUP/RESTORE)
- Peripheral Utility Program (PUP)
- Program File Editor (UPDATE)
- System Generation (SYSGEN)
- Console error messages
- System Load Procedures
- Peripheral User's Guide

FORMS CREATION AND ACCESS UTILITY

INTRODUCTION

The Forms Creation and Access Utility offers an easy way to make application-defined forms displayed on a Tandem T16/6511 or T16/6512 page mode terminal, and a simple method of accessing the fields within the forms.

The use of the utility is separated into three parts:

* FORM CREATION

  Designing a form becomes a simple task with Tandem's SCREEN program and a page mode terminal. You arrange the fields on the screen and delimit the places where data entries are made. Once you are satisfied with the form, SCREEN asks for a field name, data attribute, and optionally, a default value for each field. Once the form is written in a file, you may test it with SCRNTEST, which means you can test the efficiency and completeness of your form before the application program is ready.

* FORMS DISPLAY

  Calls to the ENTRY procedures display a form and read input data from it. The name of the form to be displayed is passed to the procedure, EXPAND^SCREEN, which fills the application program's I/O buffer with control characters and screen fields. Then a call to a file management procedure writes the buffer to the terminal.

  After entries are made at the terminal, READ^SCREEN is called to fill the application program's I/O buffer with the terminal control sequence, then WRITEREAD, a file management procedure, transfers the screen data to the program. CHECK^SCREEN is called to check the data entries. Incorrect entries may be re-displayed, one at a time, with the blinking feature to make them stand out.

  In all cases, the application program is responsible for doing the actual I/O though Tandem's file system, while the ENTRY procedures create buffers and process input buffers. This means you have full use of the file system features, plus ease of communication with the page mode terminal.

* FIELD ACCESS

  Each field of a form has a name, length and data attribute associated with it. Using the naming conventions of SCREEN, a program has access to all three. For example, if your form "CARD" has a field "NAME", then your application program would reference it as "CARD^NAME", its length as "CARD^NAME^LEN" and its data attribute as "CARD^NAME^CHK".

To examine a field's content, you might code a T/TAL statement like:

```
IF CARD^NAME = "..." THEN .....
```

and move the field's contents by:

```
S ':=' CARD^NAME FOR CARD^NAME^LEN;
```

CONVENTIONS IN THIS MANUAL

Reserved Symbols and Syntactic Elements

Program names, command names, keywords, and procedure names are shown in upper case throughout this manual. All other syntax is shown in lower case, surrounded by less-than and greater-than symbols <...>. You may enter the commands and calls with their parameters in either upper or lower case - it makes no difference.

Required/Optional Elements

Required elements in a command are underlined, while optional ones are surrounded by brackets [...].

CREATING A FORM

The SCREEN program creates a form image once you have designed a form and supplied essential information. You first arrange the fields on the terminal screen, name each field and select its data attribute, and optionally define a default value. SCREEN uses all this information to build a form image in a disc file. Usually, one form image is created per disc file, but several forms may be combined in one file using EDIT.

In addition to creating new forms, SCREEN will make a new form by modifying an old one.

```
              --------------------                      ------------
              |page mode terminal|                      | file "X" |
              --------------------                      ------------
   you        | form             |        SCREEN        |          |
   enter -->  | field names      | <---> program  <-->  |   form   |
              | field attributes |                      |   image  |
              | any defaults     |                      |          |
              --------------------                      ------------
```

Later, when the application program is ready, the form image file and SCREEN library SCRNLIB are compiled with the program into an object file.

```
   ----------------------------------------------
   |                                            |
   |      global data declarations              |
   |                                            |
   |      ?SOURCE x                             |
   |         .                                  |
   |         .                                  |
   |      external procedure declarations       |
   |                                            |
   |      ?SOURCE $SYSTEM.SYSTEM.SCRNLIB        |
   |                                            |
   |      application program code              |
   |                                            |
   ----------------------------------------------
```

RUNNING THE SCREEN PROGRAM

SCREEN runs from Tandem's Command Interpreter. Designing a form is done on the "home" terminal - the same terminal used to run SCREEN.

The command to run SCREEN is:

```
---------------------------------------------------------------------
|                                                                   |
|  SCREEN / [ IN <old file> , ] OUT <new file> /  [ <sf> ]          |
|  ------ -                     --- ---------- -                    |
|                                                                   |
|  where:                                                           |
|                                                                   |
|     <old file>  is a file which contains a form definition.  If   |
|                 named, a copy of the old form is used as the base  |
|                 for a new form.                                    |
|                                                                   |
|     <new file>  is the name of the file where the form image is   |
|                 written.  If this file currently exists, it is    |
|                 purged.  <new file> cannot be the same file as    |
|                 <old file>.  Note:  The <disc file name> portion  |
|                 of <new file> is used as the name of the new form. |
|                                                                   |
|     <sf>        is a two character sequence which defines the     |
|                 characters delimiting a form's entry fields.  <s> |
|                 starts the field, and <f> terminates the field.   |
|                 When the form is displayed, the field delimiters  |
|                 become protected spaces.  Delimiters must be      |
|                 different from one another and neither may be a   |
|                 space.                                            |
|                 DEFAULTS:  <s> = "["                              |
|                            <f> = "]"                              |
|                                                                   |
|  example:                                                         |
|                                                                   |
|     SCREEN /OUT myforms/                                          |
|                                                                   |
---------------------------------------------------------------------
```

USING THE SCREEN PROGRAM

When no <old file> name is supplied, SCREEN assumes you want to make a form from scratch. SCREEN clears the terminal's screen and positions the cursor in the upper left-hand corner. At this point, you design the new form on the screen.

DESCRIPTOR FIELDS: A form consists of descriptor fields and entry fields. A descriptor field is simply one or more words that tells the form user what type of data is expected as input, or several lines of text explaining the form as a whole. Descriptor fields are protected from modification when the form is displayed.

ENTRY FIELDS: An entry field is a contiguous string of bytes where input data is typed by the form user. The start of a field is marked by the <s> delimiter (default "["), the end by the <f> delimiter (default "]"). Delimiters become protected spaces when the form is displayed.

STEP 1 - ENTER DESCRIPTOR AND ENTRY FIELDS

```
:SCREEN / OUT x /

   where x is the name of the form and the file to write it in.

SCREEN then clears the screen and these fields are entered:


NAME    [                    ]
ADDRESS [                    ]
        [                    ]
        [                    ]
ACCOUNT NUMBER [          ] SECURITY LEVEL [  ]
SOCIAL SECURITY NUMBER [          ]
```

In this example, "NAME", "ADDRESS", "ACCOUNT NUMBER", "SECURITY LEVEL", and "SOCIAL SECURITY NUMBER" are descriptor fields. The first four entry fields are each 20 characters long, "ACCOUNT NUMBER" is ten characters, "SECURITY LEVEL" is two characters and "SOCIAL SECURITY NUMBER" is ten characters.

DEFAULT VALUES: Values used if none are entered (default entries) may be defined after the form is designed.

```
---------------------------------------------------------------------
|                                                                   |
|                  STEP 2 - ENTER DEFAULT VALUES                    |
|                                                                   |
|  NAME    [                    ]                                   |
|  ADDRESS [                    ]                                   |
|          [                    ]                                   |
|          [                    ]                                   |
|          [                    ]                                   |
|  ACCOUNT NUMBER [        ]  SECURITY LEVEL [00]                   |
|  SOCIAL SECURITY NUMBER [         ]                               |
|                                                                   |
---------------------------------------------------------------------
```

A default entry of "00" is specifed for the "SECURITY LEVEL" field.

TRAILING BLANK LINES: If the form image is less than 24 lines, it's a good idea to make the trailing blank lines a protected field. To do this, place the cursor on the last position of the screen - the lower, right-hand corner - before pressing a function key. HINT: A fast way to position the cursor to that corner is to press the "home" key and then the left-arrow cursor key "<--".

By making the trailing blank lines a protected field, the form user cannot type data into this area. (Extraneous data will cause trouble.) Any attempt to do so will place the cursor at the first entry field on the screen.

ENTERING FORM: After all the default entries are specified, position the cursor on the last character of the form, or the last character of the screen (see above), and press any function key to transmit the form and any default entries to the computer.

FIELD NAME: A field name must be given for each entry field of your form. After entering the form, SCREEN clears the screen so you may enter the name of each field followed by its data attribute. Fields are assigned names from left to right, top to bottom of the form. Field names and attributes may be entered in a free-form manner. The field names must be alphanumeric, beginning with an alpha, and may be up to 8 characters in length.

DATA ATTRIBUTE: The data attribute is a number identifying the type of checking the CHECK^SCREEN procedure is to perform when the form is read. Negative numbers mean a field is optional, while positive numbers mean a field is required. Numbers <= -256 and => 256 are left for you to define in the application program.

DATA ATTRIBUTES

256 and above are application-defined	required entry
9 through 255 are reserved for future use	" "
8 = numeric financial without commas	" "
7 = numeric financial with commas	" "
6 = numeric preceded by optional plus or minus	" "
5 = numeric and special only	" "
4 = alpha and/or numeric only	" "
3 = numeric only	" "
2 = alpha only	" "
1 = any characters	" "
0 = no checking, not required	
-1 = any characters	optional entry
-2 = alpha only	" "
-3 = numeric only	" "
-4 = alpha and/or numeric only	" "
-5 = numeric and special only	" "
-6 = numeric preceded by optional plus or minus	" "
-7 = numeric financial with commas	" "
-8 = numeric financial without commas	" "
-9 thru -255 reserved for future use.	" "
-256 and below are application-defined.	" "

Note: Embedded blanks are not allowed except for attributes 1, -1, or an attribute you define.

In the example form, the entry fields are assigned names in this order:

```
NAME    [          1        ]
ADDRESS [          2        ]
        [          3        ]
        [          4        ]
ACCOUNT NUMBER [      5      ]  SECURITY LEVEL [ 6]
SOCIAL SECURITY NUMBER [     7    ]
```

```
-------------------------------------------------------------------
|                                                                 |
|        STEP 3 - ENTER FIELD NAMES AND DATA ATTRIBUTES           |
|                                                                 |
|  Fields are named and data attributes assigned:                 |
|                                                                 |
|  NAME        1                                                  |
|  ADDRESS1    1                                                  |
|  ADDRESS2    1                                                  |
|  ADDRESS3    1                                                  |
|  ACCOUNT     5           SECLEVEL   3                           |
|  SOCSEC     -5                                                  |
|                                                                 |
-------------------------------------------------------------------
```

Notice that the field names were placed in positions similar to the actual fields for easy reference.

ENTERING ATTRIBUTES: When the attributes of all entry field are defined, you position the cursor over the last character on the screen and press any function key to transmit this information to the computer.

FORM IMAGE FILE: The output file will then be written. This file contains the definition of the screen in a form suitable for use by the ENTRY procedures, as well as a "picture" of the screen which can be used for documentation purposes or as a handy reference when designing a new screen from an old one.

MODIFYING AN EXISTING FORM

If <old file> is named, then you want to make a new form by changing a copy of an old form. SCREEN displays the old form on the screen (including any default values). At this point, changes can be made to the form and the default entries. When the form is changed to your satisfaction, place the cursor over the last character in the form, or the bottom right-hand corner of the screen, and press any function key to send it to the computer.

Next, SCREEN permits the field names and data attributes to be changed. SCREEN outputs the old field names and attributes for you to edit. When finished, place the cursor over the last character and press any function key.

This process is identical to making a screen from scratch, except rather than having a blank screen to start with, you have a copy of the form in <old file>.

For example, using the previous form, you want to add a zip code field. The operations would be:

```
:SCREEN / IN x, OUT y /
```

The previous screen is then displayed:

```
NAME    [                    ]
ADDRESS [                    ]
        [                    ]
        [                    ]
ACCOUNT NUMBER [            ]  SECURITY LEVEL [00]
SOCIAL SECURITY NUMBER [            ]
```

```
-------------------------------------------------------------------
|                                                                 |
|               STEP 1 - ADD THE NEW FIELD                        |
|                                                                 |
|  Enter the new descriptor and entry field and send the form     |
|  back to the computer:                                          |
|                                                                 |
|  NAME    [                    ]                                 |
|  ADDRESS [                    ]                                 |
|          [                    ]                                 |
|          [                    ]   ZIP CODE [      ]             |
|  ACCOUNT NUMBER [            ]   SECURITY LEVEL [00]           |
|  SOCIAL SECURITY NUMBER [            ]                          |
|                                                                 |
-------------------------------------------------------------------
```

The old field names and attributes are then displayed:

```
NAME        1
ADDRESS1    1
ADDRESS2    1
ADDRESS3    1
ACCOUNT     5            SECLEVEL   3
SOCSEC     -5
```

```
-------------------------------------------------------------------------
|                                                                       |
|                STEP 2 - ADD FIELD NAME AND ATTRIBUTE                  |
|                                                                       |
|  You add the field name "ZIP" and its data attribute and press        |
|  a function key:                                                      |
|                                                                       |
|  NAME        1                                                        |
|  ADDRESS1    1                                                        |
|  ADDRESS2    1                                                        |
|  ADDRESS3    1            ZIP        3                                |
|  ACCOUNT     5            SECLEVEL   3                                |
|  SOCSEC     -5                                                        |
|                                                                       |
-------------------------------------------------------------------------
```

The new form, created from a copy of "x", is written to file "y".

ERROR MESSAGES

You may make an error as you design your form. If so, after SCREEN receives the form or the names and attributes, the cursor is placed over the field in error and a message is written to the bottom of the screen. Correct the field, erase the SCREEN message (otherwise it will become part of your form), position the cursor over the last character of the screen and press any function key.

The possible errors are:

* FIELD NOT TERMINATED

  An entry field does not have a terminating delimiter or is greater than 255 characters in length.

* ILLEGAL FIELD NAME

  The field name was not an alphanumeric string of 8 or fewer characters, or did not start with an alphabetic character.

* ILLEGAL ATTRIBUTE

  The data attribute is either missing or not a legal integer value

* NOT ENOUGH FIELDS DEFINED

  You did not supply enough field names.

* TOO MANY FIELDS DEFINED

  You supplied too many field names or attempted to define more than 255 fields.

I/O errors will result in a file system error message (FILEERROR) and cause SCREEN to ABEND. Refer to Appendix B for FILEERROR codes.

Pressing the BREAK key during any part of the form building process will cause an immediate termination of SCREEN. You should then press one of the function keys to restore the terminal to conversation mode.

USING THE SCRNTEST PROGRAM

The SCRNTEST program lets you test the appearance and data checking of a form without your application program being available. To use SCRNTEST, do the following:

TESTING YOUR FORM

1. Create the form with SCREEN and place it into the file "TESTSCRN":

   ```
   :SCREEN / OUT TESTSCRN /
   ```

2. Compile the program SCRNTEST:

   ```
   :TAL / IN $SYSTEM.SYSTEM.SCRNTEST /
   ```

3. Run the resulting object program:

   ```
   :RUN OBJECT
   ```

The screen is displayed on the home terminal with protected fields "half bright" and initial data values "full bright". Enter data and press any function key to send the screen data to the computer.

DATA ENTRY/CORRECTION: Data entries are checked according to the data attributes you set. An error will cause that field on the screen to blink. Correct it and press any function key to send the retyped data to the computer.

When all entries are correct, the data entry fields are reset to their initial values and the above process may be repeated.

STOPPING SCRNTEST: Stop SCRNTEST by pressing the BREAK key. Any terminal I/O errors will ABEND SCRNTEST.

FIXING THE FORM: Any errors in the form may be corrected by modifying the form using SCREEN. Remember, you cannot use the same name for the IN and OUT files. Before running SCREEN, rename TESTSCRN:

```
        :RENAME TESTSCRN,junk
        :SCREEN /IN junk, OUT TESTSCRN/
                      .
                      .
               fix errors in form
                      .
        :PURGE junk
        :TAL..... compile and retest
```

When you are satisfied with the form, use SCREEN to put it in the correct file:

```
        :SCREEN /IN TESTSCRN,OUT myfile/
                      .
                      .
              Form image displayed
                Position cursor ("home key" and "<--")
                and press any function key
                      .
                      .
              Field names and attributes displayed
                Position cursor and press any
                function key.
```

ACCESSING FIELDS IN YOUR FORM

Required Buffers

Your application program must have two buffers available to the ENTRY procedures:

1. A string array, which must be named SCREEN, that is big enough to hold all the entry fields of the largest form;

2. An I/O buffer to hold terminal control characters and all the data to be displayed.

SCREEN ARRAY: This array is reserved for entry field data. The application program may move values into it (see IDENTIFIERS section) to initialize some of the form's fields, rather than use any default values. Primarily, the ENTRY procedures use this array to return checked field entries from the screen so the program has ready access to each value.

Form length is conveniently calculated by SCREEN and placed in the identifier <form name>^SBUF of the form image file. If your application program uses multiple forms, choose the SBUF with the largest number when declaring this array.

The string array for "x" is declared as:

```
    STRING .SCREEN [ 0:X^SBUF ]
```

I/O BUFFER: Terminal I/O is done with this buffer. Terminal control characters, a form image, initialized entry field data or default entry data are moved to the I/O buffer before it is written to the terminal. SCREEN calculates the length of this buffer and stores the value in the identifier <form name>^IOBUF of the form image file. This value is also used as the read count for returning entry field data from the terminal.

If your application program uses multiple forms, choose the IOBUF with the largest number for your declaration. NOTE: The form with the largest value doesn't necessarily have the largest IOBUF value.

The declaration for the I/O buffer of "x" is:

```
    STRING .IOBUFFER [ 0:X^IOBUF-1 ]
```

Identifiers

A number of identifiers are produced by the SCREEN Program that are used to access fields within a form. These identifiers fall into three classes: field name, field length, and data attribute.

FIELD NAME: Access the contents of a field of a given form by referencing an identifier:

```
    <form name>^<field name>
```

For example, access to the fields in the sample form "x" is done using the these identifiers:

```
    X^NAME
    X^ADDRESS1
    X^ADDRESS2
    X^ADDRESS3
    X^ACCOUNT
    X^SECLEVEL
    X^SOCSEC
```

FIELD LENGTH: The length, in bytes, of fields of a given form is referenced with identifiers:

```
  <form name>^<field name>^LEN
```

For example, the field lengths of the fields in the sample form "x" are identified by

```
    X^NAME^LEN
    X^ADDRESS1^LEN
    X^ADDRESS2^LEN
    X^ADDRESS3^LEN
    X^ACCOUNT^LEN
    X^SECLEVEL^LEN
    X^SOCSEC^LEN
```

DATA ATTRIBUTE: The data attributes of form fields are referenced as:

```
  <form name>^<field name>^CHK
```

For example, the data attributes of the fields in the sample form "x" are

```
    X^NAME^CHK
    X^ADDRESS1^CHK
    X^ADDRESS2^CHK
    X^ADDRESS3^CHK
    X^ACCOUNT^CHK
    X^SECLEVEL^CHK
    X^SOCSEC^CHK
```

USING THE ENTRY PROCEDURES

Once you have designed and tested a form, or several forms, you are faced with integrating them with the application program. Six procedures are available to do this: EXPAND^SCREEN, READ^SCREEN, CHECK^SCREEN, BLINK^SCREEN, POSITION^SCREEN and FL^SCREEN.

Normally, the procedures are used in this order:

```
                                                   Form User at
Application program                                  Terminal

1.  EXPAND^SCREEN is called
    to fill the I/O buffer

2.  WRITE is called to output
    the I/O buffer (your form)
    to the terminal--------------------> 3. The form is displayed
                                            at the terminal; form
                                            user types in data, and

5.  READ function key <------------------4. Presses any function key
    (three characters) from
    terminal

6.  WRITEREAD done to the
    terminal to get the data
    entries.  The write portion
    of the data is supplied by
    READ^SCREEN------------------------> 7. Terminal sends entry
                                            field data to the
8.  CHECK^SCREEN is called to <-------------computer
    check the entry data

9.  If there is an error,
    BLINK^SCREEN is called to fill
    the I/O buffer with the control
    sequence to position the cursor
    and blink the field in error,
    then the buffer is sent to the
    terminal by WRITE----------------> 10. Form user corrects the
                                           blinking field and presses
11. Data is received by the program <------any function key
    as described above (steps 5 and
    6)

12. BLINK^SCREEN is called to fill
    the I/O buffer with the control
    sequence to turn off the blinking
    field and then the buffer is sent
    to the terminal by WRITE
```

13. The program checks the retyped data and continues processing if the data is correct, or repeats the correction procedure until it is correct. (Steps 8 to 12)

In the above method, the terminal's Send Keys are not used. If you want to move data with those keys, then large amounts of resident buffer space are required. The economy of the outlined method above is that a large buffer is only needed when a data transfer between the terminal and the computer is actually taking place. While the form user is filling the screen, only a small buffer is needed to read the function key.

If a polling terminal is used, then a different method of terminal I/O may be done. See the Data Communications (ENVOY) Manual, T16/8018 for details.

ENTRY Procedures Library

The ENTRY procedures are kept in a library named

```
$SYSTEM.SYSTEM.SCRNLIB
```

Code a "?SOURCE" command in the application program for this library file.

Terminal Mode

The procedures assume the terminal you are using is in "page mode". This may be set when the system is generated during SYSGEN, or by using the file system SETMODE procedure in the application program. (See Appendix A).

Advanced Features

You are encouraged to have a good understanding of the T16/6511-12 terminal itself by reading the manual that comes with it. By taking advantage of certain terminal features it's possible to do things in combination with the ENTRY procedures, such as write out a protected field. Refer to Appendix A for an example.

The ENTRY procedures and syntax conventions are:

```
CALL <procedure name> ( <parameters> ) ;

where

   required parts of the calling sequence are underlined

   CALL is a T/TAL CALL statement

   <procedure name> is

      BLINK^SCREEN       fills the buffer with code to blink an
                         entry field or turn the blinking off.

      CHECK^SCREEN       moves the entry data from the I/O buffer
                         to the SCREEN array and does data checking
                         based on the data attributes.

      EXPAND^SCREEN      expands a form completely, or just the
                         entry fields, into the I/O buffer.

      FL^SCREEN          computes the actual length of an entry.

      POSITION^SCREEN    fills the buffer with code to position
                         the cursor over a specific entry field.

      READ^SCREEN        fills the buffer with the code to read
                         the data enries from the screen.

   <parameters>          are required if underlined, optional if
                         not.  Placeholder commas must be present
                         for parameters omitted.  Parameters are
                         described like this:

                         <parameter>, <type> : { ref   }
                                               { value }

                         where

                         <type>    is INT or STRING.

                         ref       means the address of a value
                                   (reference parameter) is passed.

                         value     means the value itself is passed.
```

BLINK^SCREEN

BLINK^SCREEN is called to place control characters into the application program's I/O buffer to blink, or clear the blink on a data entry field. When output, this control sequence will leave the cursor over the first character of the field that is being blinked or unblinked.

```
---------------------------------------------------------------------
|                                                                   |
| ! INT:function ! BLINK^SCREEN ( @<screen name> , SCREEN ,         |
|                  ------------ - ------------- - ------ -          |
|                                 <buffer> , <field name> ,         |
|                                 -------- - ------------ -         |
|                                 <blink> )                         |
|                                 ------- -                         |
|  where:                                                           |
|                                                                   |
|    BLINK^SCREEN, INT:function, returns the number of control      |
|    characters that were placed into the application program's    |
|    I/O buffer.                                                    |
|                                                                   |
|    <screen name>  INT:value, is the address of the read-only      |
|                   array which has the form definition.            |
|                                                                   |
|    SCREEN         STRING:ref, is the required array named SCREEN. |
|                                                                   |
|    <buffer>       STRING:ref, is the application program's I/O    |
|                   buffer where the control sequence is placed.    |
|                   The sequence is 20 characters long to blink a   |
|                   field and 18 characters long to unblink.        |
|                                                                   |
|    <field name>   STRING:ref, is the name of the entry field you  |
|                   want to blink or unblink.                       |
|                                                                   |
|    <blink>        INT:value, is non-zero to blink an entry field, |
|                   or zero to remove the blink.                    |
|                                                                   |
|  example:                                                         |
|                                                                   |
|    CALL  WRITE(term, buf,                                         |
|               BLINK^SCREEN (@x, SCREEN, buf, x^name, 1), cnt);    |
|                                 ! blink the name field            |
|    IF  <  THEN  ...             ! error occurred                  |
|                                                                   |
---------------------------------------------------------------------
```

CHECK^SCREEN is called after the entry field data has been input from the terminal by the program. The entry fields are moved out of the application program's I/O buffer and into the array SCREEN. Each field is checked, one at a time, according to its data attribute and then your checking procedure is called to do further checking.

```
-----------------------------------------------------------------------
|                                                                     |
|  ! INT:function ! CHECK^SCREEN ( @<screen name> , SCREEN ,          |
|                   ------------ - ------------- - ------ -           |
|                                 <buffer> , <check procedure> )      |
|                                 -------- - ----------------- -      |
|                                                                     |
|  where:                                                             |
|                                                                     |
|     CHECK^SCREEN, INT:function, returns the last value returned     |
|     by your procedure <check procedure>.  A "1" means no errors     |
|     were detected, while a "0" means there was at least one error.  |
|                                                                     |
|     <screen name>   INT:value, is the address of the read-only      |
|                     array that has the form definition.             |
|                                                                     |
|     SCREEN          STRING:ref, is the required array named SCREEN  |
|                     where the entry data is placed.  All fields are |
|                     null terminated so they may be accessed using   |
|                     the T/TAL SCAN statement.  Each field has lead- |
|                     ing and trailing blanks deleted, in other words,|
|                     left justified.                                 |
|                                                                     |
|     <buffer>        STRING:ref, is the I/O buffer that has the      |
|                     data entries as they were transmitted to your   |
|                     program from the terminal.                      |
|                                                                     |
|     <check          INT PROC, is your data checking procedure.      |
|      procedure>     It will be called for each field to do          |
|                     furthur checking. If an error is detected,      |
|                     then this procedure may abort the checking      |
|                     and provide an application dependent diag-      |
|                     nostic.  Declare the procedure like this:       |
|                                                                     |
|     INT PROC <check procedure>( <fieldnum> , <field> , <check>,     |
|                                 <error> );                          |
|                                                                     |
|     INT fieldnum;   ! number of the field; fields are numbered      |
|                     ! starting at 1                                 |
|     STRING .field;  ! pointer to the entry field in SCREEN array    |
|     INT check;      ! data attribute defined when the form was      |
|                     ! created                                       |
|     INT error;      ! "1" if CHECK^SCREEN has detected an error     |
|                     ! in this field, "0" if it has not              |
|                                                                 --> |
-----------------------------------------------------------------------
```

```
-----------------------------------------------------------------
|                                                               |
|     Your check procedure should return a "1" if the checking is |
|     to continue, or a "0" if it should not.                   |
|                                                               |
|  example:                                                     |
|                                                               |
|     IF  CHECK^SCREEN(@x, SCREEN, buf, checkx)  THEN ...       |
|                                                               |
-----------------------------------------------------------------
```

The general control flow of CHECK^SCREEN is:

```
     For each field from the screen,
         move the field entry data into the SCREEN array,
         do type checking depending on a field's data attribute,
         call your procedure; if your procedure finds an error,
            a zero is returned.
     Return a 1.
```

In the case of a zero, the program could re-display the bad field with the blinking feature, which also places the cursor at the first character of the bad data.

EXPAND^SCREEN is called to place the control sequences and variable data you may want displayed, or default values, and optionally the entire form, into the application program's I/O buffer. The actual I/O is done with the file system I/O procedures.

```
 ! INT:function ! EXPAND^SCREEN ( @<screen name> , SCREEN ,
                  ------------- - -------------- -          -

                                   <buffer> , <rewrite form> )
                                   -------- - -------------- -

where:

   EXPAND^SCREEN, INT:function, returns the number of bytes that
   were placed into the program's I/O buffer.

   <screen name>  INT:value, is the address of the read-only
                  array which has the screen definition.

   SCREEN         STRING:ref, is the required array named SCREEN.
                  If this parameter is omitted, the screen is
                  displayed with the default values.  If you do
                  give this parameter, the screen is displayed
                  with only the data entries moved into the
                  SCREEN array.  Data entries shorter than a de-
                  fined field should be null terminated.

   <buffer>       STRING:ref, is the program's I/O buffer.  Form
                  control and data characters are moved here
                  prior to terminal display

   <rewrite form> INT:value, is non-zero if both the protected
                  form and the data entry fields are to be
                  written to the terminal.  A zero means only
                  write the entry fields.  For multiple inputs or
                  outputs using the same form, it's faster to
                  just rewrite the data entry fields (<rewrite
                  form> = 0).
example:

   X^NAME ':=' [ "JAMES CARTER", 0 ];  ! null terminated by ",0"
   X^ADDRESS1 ':=' [ "1600 PENNSYLVANIA AV" ];  ! data fills
         .                                      ! entire field
         .
         .
   CALL  WRITE(term, buf, EXPAND^SCREEN(@x, SCREEN, buf, 1), I);
   IF  <  THEN ...              ! error occurred
```

Forms with no data entry fields are displayed at full brightness. Normally, the form is protected and is displayed "half-bright", while the entry fields are displayed at full intensity.

FL^SCREEN is called to find the actual length of the data entered.

```
-------------------------------------------------------------------------
|                                                                       |
|  ! INT:function ! FL^SCREEN ( <field name> )                          |
|                   --------- - ------------ -                          |
|                                                                       |
|  where:                                                               |
|                                                                       |
|     FL^SCREEN, INT:function, is the length of the input data          |
|     the form user actually typed in the entry field.                  |
|                                                                       |
|     <field name>  STRING:ref, is the name of an entry field.          |
|                                                                       |
|  example:                                                             |
|                                                                       |
|     form user entered "JAMES CARTER" in the field x^name              |
|                                                                       |
|     a := FL^SCREEN( x^name );        ! sets a to 12                   |
|     b := x^name^LEN;                 ! sets b to 20                   |
|                                                                       |
-------------------------------------------------------------------------
```

Be aware of the difference between the length of the entry field (x^name^LEN), and the length of the data actually entered (FL^SCREEN(x^name)).

Where the form user is required to fill in the entire field, your program should check that <field>^LEN equals FL^SCREEN(<field>).

POSITION^SCREEN is called to place control characters into the application program's I/O buffer to position the cursor over the first character of a data entry field.

```
--------------------------------------------------------------------
|                                                                  |
| ! INT:function ! POSITION^SCREEN ( @<screen name> , SCREEN ,     |
|                  --------------- - -------------- - ------ -     |
|                                    <buffer> , <field name> )     |
|                                    -------- - ------------ -     |
|                                                                  |
| where:                                                           |
|                                                                  |
|    POSITION^SCREEN, INT:function, returns the number of control  |
|    characters placed into the application program's I/O buffer.  |
|                                                                  |
|    <screen name>  INT:value, is the address of the read-only     |
|                   array which has the form definition.           |
|                                                                  |
|    SCREEN         STRING:ref, is the required array named SCREEN.|
|                                                                  |
|    <buffer>       STRING:ref, is the program's I/O buffer where  |
|                   the control sequence is placed.  The sequence  |
|                   is 4 characters long.                          |
|                                                                  |
|    <field name>   STRING:ref, is the name of the field you want  |
|                   the cursor positioned at.                      |
|                                                                  |
| example:                                                         |
|                                                                  |
|    CALL  WRITE(term, buf,                                        |
|                POSITION^SCREEN(@x, SCREEN, buf, x^name), cnt);   |
|                                    ! put cursor at name field    |
|    IF  <  THEN  ...                ! error occurred              |
|                                                                  |
--------------------------------------------------------------------
```

READ^SCREEN is used to place the control sequence required to read the screen from the terminal into the application program's I/O buffer. Normally, READ^SCREEN is used to input data entries when the form user has signaled with a function key that the screen data is ready for input.

```
-------------------------------------------------------------------------
|                                                                       |
|  ! INT:function ! READ^SCREEN ( @<screen name> , <buffer> )           |
|                   ----------- - ------------- - -------- -            |
|                                                                       |
|  where:                                                               |
|                                                                       |
|     READ^SCREEN, INT:function, returns the number of control          |
|     characters entered in the application program's I/O buffer.       |
|                                                                       |
|     <screen name>  INT:value, is the address of the read-only         |
|                    array that has the form definition.                |
|                                                                       |
|     <buffer>       STRING:ref, is the program's I/O buffer where      |
|                    the control sequence is placed. The read           |
|                    control sequence is 6 characters long.             |
|  example:                                                             |
|                                                                       |
|     CALL  WRITEREAD(term, buf, READ^SCREEN(@x, buf), x^iobuf, cnt);   |
|     IF  <  THEN  ...                ! error occurred                  |
|                                                                       |
-------------------------------------------------------------------------
```

This program displays a form that closely resembles a California Driver's License. A picture of what is displayed on the terminal can be found at the beginning of the listing, which is the form image file "LICENSE".

Besides showing most of the procedures in action, the program does two other interesting things:

- entry data error messages are displayed in the blank, but protected, upper right hand area of the form. Instead of getting only a blinking entry for incorrect data, the form user is told specifically what the problem is with a blinking message.
- when the program is stopped, the terminal is returned to conversational mode by doing a SETMODE and sending a special control sequence to the terminal.

NOTE: Refer to the ADM-2's operation manual for a discussion of the ESC codes.

```
P

TAL - TANDEM COMPUTERS VERSION B02  ( 5/11/77 -  6 PM)            SOURCE LANGUAGE: TAL - TARGET MACHINE TANDEM/16
DATE - TIME :  5/20/77 - 14: 0:58                    OPTIONS: ON (LIST,CODE,MAP,WARN,LMAP) - OFF (ICODE,INNERLIST)

000000 0  0  !
000000 0  0  !  THIS IS A SAMPLE PROGRAM TO SHOW USE OF THE "ENTRY" PROCEDURES. THE
000000 0  0  !  PROGRAM IS RUN:
000000 0  0  !
000000 0  0  !     :RUN  SAMPLE
000000 0  0  !
000000 0  0  !  IT WILL DISPLAY A FORM AND INSTRUCTIONS ON THE TERMINAL. THE PROGRAM
000000 0  0  !  IS TERMINATED BY PRESSING BREAK WHEN IT IS WAITING FOR DATA TO BE INPUT.
000000 0  0
000000 0  0  !
000000 0  0  !  SAMPLE SCREEN
000000 0  0  !
000000 0  0  ?SOURCE  LICENSE
000000 0  0  ?SECTION   LICENSE
000000 0  0  ?IF 15
000000 0  0  ----------------------------------------------------------------------
000000 0  0  |                      CALIFORNIA DRIVER'S LICENSE                    |
000000 0  0  |---------------------------------------------------------------------|
000000 0  0  |                                                                     |
000000 0  0  |  EXPIRES ON BIRTHDAY  19[  ]                                        |
000000 0  0  |  LICENSE NUMBER [ ][        ]                                       |
000000 0  0  |                                                                     |
000000 0  0  |  NAME          [                                                ]   |
000000 0  0  |  ADDRESS       [                                                ]   |
000000 0  0  |                [                          ] STATE [CA]  ZIP [   ]   |
000000 0  0  |                                                                     |
000000 0  0  |  SEX [ ]    HAIR [   ]    EYES [   ]    HEIGHT [5]-[  ]    WEIGHT [   ] |
000000 0  0  |  PREVIOUS LICENSE EXPIRED  19[  ]       DATE OF BIRTH [  ]-[  ]-[   ]  |
000000 0  0  |  CLASS [3]                              CORRECTIVE LENSES [ ]       |
000000 0  0  |  OTHER RESTRICTIONS[                                            ]   |
000000 0  0  |                                                                     |
000000 0  0  ----------------------------------------------------------------------
000000 0  0
000000 0  0  ENTER DATA IN EACH FIELD, WHEN FINISHED, PRESS ANY FUNCTION KEY TO SEND ENTRIES
000000 0  0  TO THE COMPUTER. FIELDS IN ERROR WILL BE FLAGGED VIA A BLINKING MESSAGE. THE
000000 0  0  FIELD SHOULD THEN BE CORRECTED AND SENT BACK TO THE COMPUTER. CORRECT DATA WILL
000000 0  0  CAUSE THE FORM TO BE REFILLED WITH ITS INITIAL VALUES FOR MORE DATA ENTRY.
000000 0  0
000000 0  0                                            PRESS BREAK TO TERMINATE PROGRAM
000000 0  0  EXPIRE 3
000000 0  0  LICLET 2  LICNUM 3
000000 0  0  NAME 1
000000 0  0  ADDRESS1 1
000000 0  0  ADDRESS2 1         STATE 2   ZIP 3
000000 0  0  SEX 2  HAIR 2  EYES 2  FEET 3  INCHES 3  WEIGHT 3
000000 0  0  PREVEXP 3  MONTH 3  DAY 3  YEAR 3
000000 0  0  CLASS 3    LENSES -2
000000 0  0  OTHER -1
000000 0  0
000000 0  0
000000 0  0
000000 0  0
000000 0  0
```

```
PAGE 2    $DATA.M020A00.LICENSE  [2]          SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

   41.    000000 0  0
   42.    000000 0  0
   43.    000000 0  0
   44.    000000 0  0
   45.    000000 0  0
   46.    000000 0  0
   47.    000000 0  0
   48.    000000 0  0
   49.    000000 0  0
   50.    000000 0  0
   51.    000000 0  0  ?ENDIF 15
   52.    000000 0  0  STRING  LICENSE = 'P' := [    %004,%253,%025,%002,%000,%003,%001,%133,%001,
   53.    000000 0  0  %000,%002,%001,%243,%007,%000,%003,%001,%246,%074,%000,%001,%002,%100,%074,
   54.    000000 0  0  %000,%001,%002,%220,%044,%000,%001,%002,%340,%002,%000,%002,%003,%015,%005,
   55.    000000 0  0  %000,%003,%003,%027,%001,%000,%002,%003,%170,%003,%000,%002,%003,%204,%003,
   56.    000000 0  0  %000,%002,%003,%222,%001,%000,%003,%003,%242,%002,%000,%003,%003,%246,%003,
   57.    000000 0  0  %000,%003,%003,%265,%002,%000,%003,%003,%340,%002,%000,%003,%003,%371,%002,
   58.    000000 0  0  %000,%003,%003,%376,%004,%000,%003,%004,%003,%001,%000,%003,%004,%032,%001,
   59.    000000 0  0  %377,%376,%004,%115,%066,%377,%377,%004,%166,%003,%203,%055,%055,%055,%055,
   60.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   61.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   62.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   63.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   64.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   65.    000000 0  0  %055,%174,%226,%103,%101,%114,%111,%106,%117,%122,%116,%111,%101,%040,%104,
   66.    000000 0  0  %122,%111,%126,%105,%122,%047,%123,%040,%114,%111,%103,%105,%116,%123,%105,
   67.    000000 0  0  %235,%174,%174,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   68.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   69.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   70.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   71.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   72.    000000 0  0  %055,%055,%055,%055,%055,%055,%174,%174,%316,%174,%174,%202,%105,%130,%120,
   73.    000000 0  0  %111,%122,%105,%123,%040,%117,%116,%040,%102,%111,%122,%124,%110,%104,%101,
   74.    000000 0  0  %131,%202,%061,%071,%040,%000,%002,%262,%174,%174,%202,%114,%111,%103,%105,
   75.    000000 0  0  %116,%123,%105,%040,%116,%125,%115,%102,%105,%122,%202,%000,%001,%202,%000,
   76.    000000 0  0  %007,%262,%174,%174,%316,%174,%174,%202,%116,%101,%115,%105,%211,%000,%074,
   77.    000000 0  0  %203,%174,%174,%202,%101,%104,%104,%122,%105,%123,%123,%206,%000,%074,%203,
   78.    000000 0  0  %174,%174,%217,%000,%044,%202,%123,%124,%101,%124,%105,%202,%000,%002,%203,
   79.    000000 0  0  %132,%111,%120,%202,%000,%005,%203,%174,%174,%316,%174,%174,%202,%123,%105,
   80.    000000 0  0  %130,%202,%000,%001,%205,%110,%101,%111,%122,%202,%000,%003,%205,%105,%131,
   81.    000000 0  0  %105,%123,%202,%000,%003,%205,%110,%105,%111,%107,%110,%124,%202,%000,%001,
   82.    000000 0  0  %040,%055,%040,%000,%002,%205,%127,%105,%111,%107,%110,%124,%202,%000,%003,
   83.    000000 0  0  %207,%174,%174,%202,%120,%122,%105,%126,%111,%117,%125,%123,%040,%114,%111,
   84.    000000 0  0  %103,%105,%116,%123,%105,%040,%105,%130,%120,%111,%122,%105,%104,%202,%061,
   85.    000000 0  0  %071,%040,%000,%002,%210,%104,%101,%124,%105,%040,%117,%106,%040,%102,%111,
   86.    000000 0  0  %122,%124,%110,%202,%000,%002,%040,%055,%040,%000,%002,%040,%055,%040,%000,
   87.    000000 0  0  %004,%210,%174,%174,%202,%103,%114,%101,%123,%123,%202,%000,%001,%237,%103,
   88.    000000 0  0  %117,%122,%122,%105,%103,%124,%111,%126,%105,%040,%114,%105,%116,%123,%105,
   89.    000000 0  0  %123,%202,%000,%001,%221,%174,%174,%202,%117,%124,%110,%105,%122,%040,%122,
   90.    000000 0  0  %105,%123,%124,%122,%111,%103,%124,%111,%117,%116,%123,%040,%000,%066,%203,
   91.    000000 0  0  %174,%174,%316,%174,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   92.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   93.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   94.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   95.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,%055,
   96.    000000 0  0  %055,%055,%055,%055,%055,%055,%055,%055,%055,%320,%105,%116,%124,%105,%122,
   97.    000000 0  0  %040,%104,%101,%124,%101,%040,%111,%116,%040,%105,%101,%103,%110,%040,%106,
PAGE 3     $DATA.M020A00.LICENSE  [2]               SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

   98.     000000 0  0  %111,%105,%114,%104,%054,%040,%127,%110,%105,%116,%040,%106,%111,%116,%111,
   99.     000000 0  0  %123,%110,%105,%104,%054,%040,%120,%122,%105,%123,%123,%040,%101,%116,%131,
  100.     000000 0  0  %040,%106,%125,%116,%103,%124,%111,%117,%116,%040,%113,%105,%131,%040,%124,
  101.     000000 0  0  %117,%040,%123,%105,%116,%104,%040,%105,%116,%124,%122,%111,%105,%123,%040,
  102.     000000 0  0  %124,%117,%040,%124,%110,%105,%040,%103,%117,%115,%120,%125,%124,%105,%122,
  103.     000000 0  0  %056,%040,%106,%111,%105,%114,%104,%123,%040,%111,%116,%040,%105,%122,%122,
  104.     000000 0  0  %117,%122,%040,%127,%111,%114,%114,%040,%102,%105,%040,%106,%114,%101,%107,
  105.     000000 0  0  %107,%105,%104,%040,%126,%111,%101,%040,%101,%040,%102,%114,%111,%116,%113,
  106.     000000 0  0  %111,%116,%107,%040,%115,%105,%123,%123,%101,%107,%105,%056,%040,%124,%110,
  107.     000000 0  0  %105,%204,%106,%111,%105,%114,%104,%040,%123,%110,%117,%125,%114,%104,%040,
  108.     000000 0  0  %124,%110,%105,%116,%040,%102,%105,%040,%103,%117,%122,%122,%105,%103,%124,
  109.     000000 0  0  %105,%104,%040,%101,%116,%104,%040,%123,%105,%116,%124,%040,%102,%101,%103,
  110.     000000 0  0  %113,%040,%124,%117,%040,%124,%110,%105,%040,%103,%117,%115,%120,%125,%124,
  111.     000000 0  0  %105,%122,%056,%040,%103,%117,%122,%122,%105,%103,%124,%040,%104,%101,%124,
  112.     000000 0  0  %101,%040,%127,%111,%114,%114,%040,%103,%101,%125,%123,%105,%040,%124,%110,
  113.     000000 0  0  %105,%040,%106,%117,%122,%115,%040,%124,%117,%040,%102,%105,%040,%122,%105,
  114.     000000 0  0  %106,%111,%114,%114,%105,%104,%040,%127,%111,%124,%110,%040,%111,%124,%123,
  115.     000000 0  0  %040,%111,%116,%111,%124,%111,%101,%114,%040,%126,%101,%114,%125,%105,%123,
  116.     000000 0  0  %040,%106,%117,%122,%040,%115,%117,%122,%105,%040,%104,%101,%124,%101,%040,
  117.     000000 0  0  %105,%116,%124,%122,%131,%056,%377,%207,%120,%122,%105,%123,%123,%040,%102,
  118.     000000 0  0  %122,%105,%101,%113,%040,%124,%117,%040,%124,%105,%122,%115,%111,%116,%101,
  119.     000000 0  0  %124,%105,%040,%120,%122,%117,%107,%122,%101,%115,%000,%011,%377,%247,%103,
  120.     000000 0  0  %101,%214,%065,%217,%063,%267 ];
  121.     000000 0  0  DEFINE  LICENSE^EXPIRE = SCREEN[ 0000 ]#;
  122.     000000 0  0  LITERAL LICENSE^EXPIRE^LEN = 002, LICENSE^EXPIRE^CHK = %000003;
  123.     000000 0  0  DEFINE  LICENSE^LICLET = SCREEN[ 0003 ]#;
  124.     000000 0  0  LITERAL LICENSE^LICLET^LEN = 001, LICENSE^LICLET^CHK = %000002;
  125.     000000 0  0  DEFINE  LICENSE^LICNUM = SCREEN[ 0005 ]#;
  126.     000000 0  0  LITERAL LICENSE^LICNUM^LEN = 007, LICENSE^LICNUM^CHK = %000003;
  127.     000000 0  0  DEFINE  LICENSE^NAME = SCREEN[ 0013 ]#;
  128.     000000 0  0  LITERAL LICENSE^NAME^LEN = 060, LICENSE^NAME^CHK = %000001;
  129.     000000 0  0  DEFINE  LICENSE^ADDRESS1 = SCREEN[ 0074 ]#;
  130.     000000 0  0  LITERAL LICENSE^ADDRESS1^LEN = 060, LICENSE^ADDRESS1^CHK = %000001;
  131.     000000 0  0  DEFINE  LICENSE^ADDRESS2 = SCREEN[ 0135 ]#;
  132.     000000 0  0  LITERAL LICENSE^ADDRESS2^LEN = 036, LICENSE^ADDRESS2^CHK = %000001;
  133.     000000 0  0  DEFINE  LICENSE^STATE = SCREEN[ 0172 ]#;
  134.     000000 0  0  LITERAL LICENSE^STATE^LEN = 002, LICENSE^STATE^CHK = %000002;
  135.     000000 0  0  DEFINE  LICENSE^ZIP = SCREEN[ 0175 ]#;
  136.     000000 0  0  LITERAL LICENSE^ZIP^LEN = 005, LICENSE^ZIP^CHK = %000003;
  137.     000000 0  0  DEFINE  LICENSE^SEX = SCREEN[ 0181 ]#;
  138.     000000 0  0  LITERAL LICENSE^SEX^LEN = 001, LICENSE^SEX^CHK = %000002;
  139.     000000 0  0  DEFINE  LICENSE^HAIR = SCREEN[ 0183 ]#;
  140.     000000 0  0  LITERAL LICENSE^HAIR^LEN = 003, LICENSE^HAIR^CHK = %000002;
  141.     000000 0  0  DEFINE  LICENSE^EYES = SCREEN[ 0187 ]#;
  142.     000000 0  0  LITERAL LICENSE^EYES^LEN = 003, LICENSE^EYES^CHK = %000002;
  143.     000000 0  0  DEFINE  LICENSE^FEET = SCREEN[ 0191 ]#;
  144.     000000 0  0  LITERAL LICENSE^FEET^LEN = 001, LICENSE^FEET^CHK = %000003;
  145.     000000 0  0  DEFINE  LICENSE^INCHES = SCREEN[ 0193 ]#;
  146.     000000 0  0  LITERAL LICENSE^INCHES^LEN = 002, LICENSE^INCHES^CHK = %000003;
  147.     000000 0  0  DEFINE  LICENSE^WEIGHT = SCREEN[ 0196 ]#;
  148.     000000 0  0  LITERAL LICENSE^WEIGHT^LEN = 003, LICENSE^WEIGHT^CHK = %000003;
  149.     000000 0  0  DEFINE  LICENSE^PREVEXP = SCREEN[ 0200 ]#;
  150.     000000 0  0  LITERAL LICENSE^PREVEXP^LEN = 002, LICENSE^PREVEXP^CHK = %000003;
  151.     000000 0  0  DEFINE  LICENSE^MONTH = SCREEN[ 0203 ]#;
  152.     000000 0  0  LITERAL LICENSE^MONTH^LEN = 002, LICENSE^MONTH^CHK = %000003;
  153.     000000 0  0  DEFINE  LICENSE^DAY = SCREEN[ 0206 ]#;
  154.     000000 0  0  LITERAL LICENSE^DAY^LEN = 002, LICENSE^DAY^CHK = %000003;
```

```
PAGE 4    $DATA.M020A00.LICENSE  [2]            SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

000000 0  0  DEFINE  LICENSE^YEAR = SCREEN[ 0209 ]#;
000000 0  0  LITERAL LICENSE^YEAR^LEN = 004, LICENSE^YEAR^CHK = %000003;
000000 0  0  DEFINE  LICENSE^CLASS = SCREEN[ 0214 ]#;
000000 0  0  LITERAL LICENSE^CLASS^LEN = 001, LICENSE^CLASS^CHK = %000003;
000000 0  0  DEFINE  LICENSE^LENSES = SCREEN[ 0216 ]#;
000000 0  0  LITERAL LICENSE^LENSES^LEN = 001, LICENSE^LENSES^CHK = %177776;
000000 0  0  DEFINE  LICENSE^OTHER = SCREEN[ 0218 ]#;
000000 0  0  LITERAL LICENSE^OTHER^LEN = 054, LICENSE^OTHER^CHK = %177777;
000000 0  0  LITERAL LICENSE^IOBUF = 2044,  LICENSE^SBUF = 0272;
000000 0  0
000000 0  0  !
000000 0  0  !  I/O BUFFER, SCREEN BUFFER, TERMINAL FILE NUMBER, BLINK POINTER
000000 0  0  !
000000 0  0  STRING  .SCREEN[ 0:LICENSE^SBUF ],           ! FOR ACCESSING FIELDS
000211 0  0          .SCREENIOS[ 0:LICENSE^IOBUF-1 ];   ! I/O BUFFER
002207 0  0
002207 0  0  INT  .SCREENIO := @SCREENIOS'>>'1,         ! WORD POINTER TO BUFFER
002207 0  0       ERRORMSG := 0,                         ! TRUE IF THERE IS AN ERROR
002207 0  0                                              ! MESSAGE ON THE SCREEN
002207 0  0       TERM;                                  ! HOME TERMINAL FILE NUMBER
002207 0  0
002207 0  0  LITERAL  ESC = %33;                         ! ESCAPE ASCII CODE
002207 0  0
002207 0  0  !
002207 0  0  !  EXTERNAL PROCEDURES
002207 0  0  !
002207 0  0  ?NOLIST, SOURCE $SYSTEM.SYSTEM.EXTDECS( ABEND,MYTERM,OPEN,SETMODE,FILEINFO )
000000 0  0  ?NOLIST, SOURCE $SYSTEM.SYSTEM.EXTDECS( READ,WRITE,WRITEREAD,STOP,NUMOUT )
000000 0  0  ?NOLIST, SOURCE $SYSTEM.SYSTEM.EXTDECS( NUMIN,MYPID )
000000 0  0
000000 0  0  !
000000 0  0  !  SCREEN LIBRARY
000000 0  0  !
000000 0  0  ?NOLIST, SOURCE  $SYSTEM.SYSTEM.SCRNLIB
000000 0  0
000000 0  0  !
000000 0  0  !  THE FOLLOWING PROCEDURE IS CALLED TO CHECK UP ON THE STATUS
000000 0  0  !  OF TERMINAL I/O OPERATIONS. ANY ERROR OTHER THAN BREAK BEING
000000 0  0  !  DEPRESSED WILL RESULT IN AN ABEND PRECEEDED BY THE ERROR NUMBER
000000 0  0  !  BEING PRINTED OUT. BREAK ERRORS WILL RESULT IN THE PROGRAM
000000 0  0  !  STOPPING.
000000 0  0  !
000000 0  0  PROC  IOCHK;
000000 1  0     BEGIN
000000 1  1     INT  ERROR,             ! RETURN FROM FILEINFO
000000 1  1          CNT;               ! DUMMY COUNT FOR I/O ROUTINES
000000 1  1
000000 1  1     CALL  FILEINFO( TERM, ERROR );
000010 1  1     IF  ERROR  THEN
000012 1  1        BEGIN  ! HAVE AN ERROR, CLEAR THE SCREEN AND SET THE
000012 1  2               ! TERMINAL INTO CONVERSATIONAL MODE
000012 1  2        SCREENIOS ':=' [ ESC, "*", ESC, "C", "TERMINAL I/O ERROR ..." ];
000021 1  2        CALL  NUMOUT( SCREENIOS[23], ERROR, 10, 3 );
000030 1  2        CALL  SETMODE( TERM, 8, 0 );
000037 1  2        IF  ERROR = 111  THEN
000042 1  2           BEGIN  ! BREAK WAS PUSHED, NORMAL TERMINATION
000042 1  3           CALL  WRITE( TERM, SCREENIO, 4, CNT );
```

```
PAGE 5    $DATA.M020A00.SAMPLES  [1]            SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

   67.    000052 1  3            CALL  STOP;
   68.    000055 1  3            END;
   69.    000055 1  2         ! ABNORMAL TERMINATION, PRINT OUT THE FILE SYSTEM ERROR NUMBER
   70.    000055 1  2         CALL  WRITE( TERM, SCREENIO, 26, CNT );
   71.    000065 1  2         CALL  ABEND;
   72.    000070 1  2         END;
   73.    000070 1  1      END;

CNT                       VARIABLE      INT          L+002    DIRECT
ERROR                     VARIABLE      INT          L+001    DIRECT


00000  002002 040004 070401 024711 002013 005030 024700 027000     00010 040401 014456 170001 000025 020054 000200 030001 100032
00020  126047 103027 173001 040401 100012 100003 024733 027000     00030 040004 100010 100000 000002 100034 024755 027000 040401
00040  001157 015013 040004 170002 100004 070402 000002 100036     00050 024766 027000 000002 024711 027000 040004 170002 100032
00060  070402 000002 100036 024766 027000 000002 024711 027000     00070 125003 000056 015452 015503 052105 051115 044516 040514
00100  020111 027517 020105 051122 047522 020056 027056

   74.    000000 0  0
   75.    000000 0  0   !
   76.    000000 0  0   !  THE FOLLOWING PROCEDURE IS CALLED BY CHECK^SCREEN FOR EACH FIELD.
   77.    000000 0  0   !  ON AN ERROR DETECTED BY CHECK^SCREEN, A BLINKING ERROR MESSAGE
   78.    000000 0  0   !  WILL BE OUTPUT IN A PROTECTED AREA OF THE SCREEN AND THE CURSOR
   79.    000000 0  0   !  WILL BE POSITIONED OVER THE FIELD IN ERROR.
   80.    000000 0  0   !
   81.    000000 0  0  INT PROC  FIELDCHECK( FIELDNUM, FIELD, CHECK, ERROR );
   82.    000000 1  0     INT  FIELDNUM,            ! INDEX  OF THE FIELD
   83.    000000 1  0          CHECK,               ! CHECKING TYPE
   84.    000000 1  0          ERROR;               ! ERROR FLAG
   85.    000000 1  0     STRING  .FIELD;           ! FIELD POINTER
   86.    000000 1  0     BEGIN
   87.    000000 1  1     INT  I,                   ! MISC.
   88.    000000 1  1          VAL,                 ! VALUE OF NUMERIC FIELDS
   89.    000000 1  1          CNT;                 ! TERMINAL TRANSFER COUNT
   90.    000000 1  1     STRING  .SP,              ! MISC. STRING POINTER
   91.    000000 1  1             MSG = 'P ' := [ "76 <= EXPIRATION <= 84 HERE   ",
   92.    000017 1  1                             "ALPHABETIC CHARACTER EXPECTED ",
   93.    000036 1  1                             "6 OR 7 DIGIT NUMBER EXPECTED  ",
   94.    000055 1  1                             "NAME EXPECTED IN THIS FIELD   ",
   95.    000074 1  1                             "STREET ADDRESS IN THIS FIELD  ",
   96.    000113 1  1                             "CITY NAME EXPECTED HERE       ",
   97.    000132 1  1                             "STATE NAME ABREVIATION HERE   ",
   98.    000151 1  1                             "5 DIGIT ZIP CODE EXPECTED HERE",
   99.    000170 1  1                             "M OR F EXPECTED HERE          ",
  100.    000207 1  1                             "3 LETTER HAIR COLOR EXPECTED  ",
  101.    000226 1  1                             "3 LETTER EYES COLOR EXPECTED  ",
  102.    000245 1  1                             "1 <= HEIGHT IN FEET <= 9      ",
  103.    000264 1  1                             "0 <= HEIGHT IN INCHES <= 11   ",
  104.    000303 1  1                             "50 <= WEIGHT IN POUNDS <= 600 ",
  105.    000322 1  1                             "PREVIOUS EXPIRATION <= 76     ",
  106.    000341 1  1                             "1 <= DATE OF BIRTH MONTH <= 12",
  107.    000360 1  1                             "1 <= DATE OF BIRTH DAY <= 31  ",
  108.    000377 1  1                             "1850<=DATE OF BIRTH YEAR<=1961",
  109.    000416 1  1                             "1 <= LICENSE CLASS <= 4       ",
  110.    000435 1  1                             "Y OR BLANK CHARACTER EXPECTED " ];
  111.    000454 1  1
  112.    000454 1  1  IF  NOT ERROR  THEN
```

```
PAGE 6    $DATA.M020A00.SAMPLES  [1]          SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

  113.    000457 1  1       BEGIN  ! INITIALLY O.K., NOW FOR A FEW SEMANTIC CHECKS
  114.    000457 1  2       IF  $ABS( CHECK ) = 3  THEN  CALL  NUMIN( FIELD, VAL, 10, I );
  115.    000473 1  2       CASE  FIELDNUM  OF
  116.    000475 1  2          BEGIN
  117.    000475 1  3          ;
  118.    000475 1  3          ! EXPIRATION DATE, 2 DIGITS
  119.    000475 1  3          ERROR := VAL < 76  OR  VAL > 84;
  120.    000510 1  3          ! LICENSE NUMBER, ALPHABETIC PART, NO ADDITIONAL TESTS
  121.    000510 1  3          ;
  122.    000510 1  3          ! LICENSE NUMBER, NUMERIC PART, AT LEAST SIX DIGITS
  123.    000510 1  3          ERROR := FL^SCREEN( LICENSE^LICNUM ) < 6;
  124.    000523 1  3          ;
  125.    000523 1  3          ! NAME FIELD, NO ADDITIONAL TESTS
  126.    000523 1  3          ;
  127.    000523 1  3          ! ADDRESS FIELD, NO ADDITIONAL TESTS
  128.    000523 1  3          ;
  129.    000523 1  3          ! 2 LETTER STATE ABREVIATION EXPECTED HERE
  130.    000523 1  3          ERROR := FL^SCREEN( LICENSE^STATE ) <> LICENSE^STATE^LEN;
  131.    000536 1  3          ! ZIP CODES MUST BE 5 DIGITS
  132.    000536 1  3          ERROR := FL^SCREEN( LICENSE^ZIP ) <> LICENSE^ZIP^LEN;
  133.    000551 1  3          ! SEX MUST BE "M" OR "F"
  134.    000551 1  3          ERROR := LICENSE^SEX <> "M"  AND  LICENSE^SEX <> "F";
  135.    000565 1  3          ! HAIR COLOR, NO ADDITIONAL TESTS
  136.    000565 1  3          ;
  137.    000565 1  3          ! EYE COLOR, NO ADDITIONAL TESTS
  138.    000565 1  3          ;
  139.    000565 1  3          ! HEIGHT MUST BE BETWEEN 1 AND 9 FEET INCLUSIVE
  140.    000565 1  3          ERROR := VAL < 1  OR  VAL > 9;
  141.    000600 1  3          ! INCHES MUST BE LESS THAN 12
  142.    000600 1  3          ERROR  := VAL > 11;
  143.    000610 1  3          ! WEIGHT MUST BE BETWEEN 50 AND 600 POUNDS
  144.    000610 1  3          ERROR := VAL < 50  OR  VAL > 600;
  145.    000625 1  3          ! PREVIOUS LICENSE EXPIRATION MUST BE BEFORE THIS YEAR
  146.    000625 1  3          ERROR := VAL > 76;
  147.    000635 1  3          ! MONTH OF BIRTH MUST BE BETWEEN 1 AND 12 INCLUSIVE
  148.    000635 1  3          ERROR := VAL = 0  OR  VAL > 12;
  149.    000650 1  3          ! DAY OF BIRTH MUST BE NON-ZERO, LESS OR EQUAL 31
  150.    000650 1  3          ERROR := VAL = 0  OR  VAL > 31;
  151.    000666 1  3          ! DATA OF BIRTH MUST BE BETWEEN 1850 AND 1961 INCLUSIVE
  152.    000666 1  3          ERROR := VAL < 1850  OR  VAL > 1961;
  153.    000705 1  3          ! LICENSE CLASS MUST BE BETWEEN 1 AND 4 INCLUSIVE
  154.    000705 1  3          ERROR := VAL = 0  OR  VAL > 4;
  155.    000722 1  3          ! CORRECTIVE LENSE FIELD, MUST BE LEFT BLANK OR A "Y"
  156.    000722 1  3          ERROR := LICENSE^LENSES <> "Y"  AND  FL^SCREEN( LICENSE^LENSES ) = 1;
  157.    000740 1  3          ! OTHER RESTRICTIONS, NO ADDITIONAL TESTS
  158.    000740 1  3          ;
  159.    000740 1  3          END;
  160.    000767 1  2       END;
  161.    000767 1  1
  162.    000767 1  1       IF  ERROR  THEN
  163.    000771 1  1          BEGIN  ! ERROR DETECTED, OUTPUT THE APPROPRIATE ERROR MESSAGE AS
  164.    000771 1  2                 ! A BLINKING PROTECTED FIELD. THE CURSOR IS THEN POSITIONED
  165.    000771 1  2                 ! AT THE START OF THE FIELD IN ERROR.
  166.    000771 1  2          ERRORMSG := 1;
  167.    000773 1  2          SCREENIOS ':=' [ ESC, "'", ESC, ")", ESC, "=%G", ESC, "^" ]  &
  168.    001002 1  2                         MSG[ (FIELDNUM-1)*30 ] FOR 30  &
  169.    001015 1  2                         [ ESC, "^", ESC, "(", ESC, "&" ] -> @SP;
PAGE 7     $DATA.M020A00.SAMPLES  [1]             SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

 170.      001024 1  2          @SP := POSITION^SCREEN( @LICENSE, SCREEN, SP, FIELD )+@SP;
 171.      001040 1  2          CALL  WRITE( TERM, SCREENIO, @SP-@SCREENIOS, CNT );
 172.      001052 1  2          CALL  IOCHK;
 173.      001053 1  2          RETURN 0;
 174.      001061 1  2          END;
 175.      001061 1  1       RETURN 1;
 176.      001063 1  1       END;

CHECK                           VARIABLE      INT          L-004     DIRECT
CNT                             VARIABLE      INT          L+003     DIRECT
ERROR                           VARIABLE      INT          L-003     DIRECT
FIELD                           VARIABLE      STRING       L-005     INDIRECT
FIELDNUM                        VARIABLE      INT          L-006     DIRECT
I                               VARIABLE      INT          L+001     DIRECT
MSG                             VARIABLE      STRING       P+000     DIRECT
SP                              VARIABLE      STRING       L+004     INDIRECT
VAL                             VARIABLE      INT          L+002     DIRECT


00000  033466 020074 036440 042530 050111 051101 052111 047516     00010 020074 036440 034064 020110 042522 042440 020040 040514
00020  050110 040502 042524 044503 020103 044101 051101 041524     00030 042522 020105 054120 042503 052105 042040 033040 047522
00040  020067 020104 044507 044524 020116 052515 041105 051040     00050 042530 050105 041524 042504 020040 047101 046505 020105
00060  054120 042503 052105 042040 044516 020124 044111 051440     00070 043111 042514 042040 020040 051524 051105 042524 020101
00100  042104 051105 051523 020111 047040 052110 044523 020106     00110 044505 046104 020040 041511 052131 020116 040515 042440
00120  042530 050105 041524 042504 020110 042522 042440 020040     00130 020040 020040 051524 040524 042440 047101 046505 020101
00140  041122 042526 044501 052111 047516 020110 042522 042440     00150 020040 032440 042111 043511 052040 055111 050040 041517
00160  042105 020105 054120 042503 052105 042040 044105 051105     00170 046440 047522 020106 020105 054120 042503 052105 042040
00200  044105 051105 020040 020040 020040 020040 020040 031440     00210 046105 052124 042522 020110 040511 051040 041517 046117
00220  051040 042530 050105 041524 042504 020040 031440 046105     00230 052124 042522 020105 054505 051440 041517 046117 051040
00240  042530 050105 041524 042504 020040 030440 036075 020110     00250 042511 043510 052040 044516 020106 042505 052040 036075
00260  020071 020040 020040 020040 030040 036075 020110 042511     00270 043510 052040 044516 020111 047103 044105 051440 036075
00300  020061 030440 020040 032460 020074 036440 053505 044507     00310 044124 020111 047040 050117 052516 042123 020074 036440
00320  033060 030040 050122 042526 044517 052523 020105 054120     00330 044522 040524 044517 047040 036075 020067 033040 020040
00340  020040 030440 036075 020104 040524 042440 047506 020102     00350 044522 052110 020115 047516 052110 020074 036440 030462
00360  030440 036075 020104 040524 042440 047506 020102 044522     00370 052110 020104 040531 020074 036440 031461 020040 030470
00400  032460 036075 042101 052105 020117 043040 041111 051124     00410 044040 054505 040522 036075 030471 033061 030440 036075
00420  020114 044503 042516 051505 020103 046101 051523 020074     00430 036440 032040 020040 020040 020040 054440 047522 020102
00440  046101 047113 020103 044101 051101 041524 042522 020105     00450 054120 042503 052105 042040 002004 040703 115577 040704
00460  013001 000214 001003 015007 170705 070402 100012 070401     00470 024733 027000 000107 040706 110562 040402 001114 014003
00500  040402 001124 016002 100777 010401 100000 044703 110577     00510 103005 173000 024700 027000 001006 013002 100777 010401
00520  100000 044703 110564 103254 173000 024700 027000 001002     00530 012002 100777 010401 100000 044703 110551 103257 173000
00540  024700 027000 001005 012002 100777 010401 100000 044703     00550 110536 103265 153000 001115 012005 153000 001106 012002
00560  100777 010401 100000 044703 110522 040402 001001 014003     00570 040402 001011 016002 100777 010401 100000 044703 010567
00600  040402 001013 016002 100777 010401 100000 044703 010557     00610 040402 001062 014004 040402 020004 000215 016003 100777
00620  010402 001130 100000 044703 010542 040402 001114 016002     00630 100777 010401 100000 044703 010532 040402 001000 012003
00640  040402 001014 016002 100777 010401 100000 044703 010517     00650 040402 001000 012006 040402 001037 010402 000111 000061
00660  016002 100777 010401 100000 044703 010501 040402 020010     00670 000215 014004 040402 020005 000215 016004 100777 010403
00700  003472 003651 100000 044703 010462 040402 010401 000060     00710 001000 012003 040402 001004 016002 100777 010401 100000
00720  044703 010445 103330 153000 001131 012007 173000 024700     00730 027000 001001 015002 100777 010401 100000 044703 010427
00740  000030 000026 177533 000024 177544 000022 000021 000020     00750 177553 177565 177577 000014 000013 177610 177622 177631
00760  177645 177654 177666 177703 177721 177735 000001 040703     00770 014470 100001 044003 170001 000025 020057 000200 030001
01000  100012 126040 040706 104777 100036 000212 000025 030001     01010 020045 000200 000200 100036 126040 000025 020040 000200
01020  030001 100006 126040 044404 000025 030001 020031 000200     01030 170000 170404 170705 024733 027000 170404 000210 044404
01040  040004 170002 170404 170001 000211 070403 000002 100036     01050 024766 027000 027000 100000 125007 000066 175762 000052
01060  170556 100001 125007 015447 015451 015475 022507 015536     01070 015536 015450 015446

 177.      000000 0  0
```

```
PAGE 8    $DATA.M020A00.SAMPLES  [1]          SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

000000 0  0  !
000000 0  0  !  THIS IS THE ACTUAL TEST PROCEDURE. IT OPENS THE HOME TERMINAL,
000000 0  0  !  SETS IT INTO PAGE MODE AND TAKES OVER BREAK, THEN PROCEEDS WITH
000000 0  0  !  SCREEN TESTING. THE PROTECTED FORM AND THE DEFAULT VALUES ARE
000000 0  0  !  OUTPUT. WHEN A FUNCTION KEY IS TYPED, THE SCREEN IS READ IN.
000000 0  0  !  THE FIRST FIELD IN ERROR WILL BE FLAGGED. WHEN A FORM WITH
000000 0  0  !  NO ERRORS IS DETECTED, THE DEFAULT SCREEN VALUES WILL THEN BE
000000 0  0  !  REWRITTEN TO THE TERMINAL. THE PROGRAM IS TERMINATED BY PRESSING
000000 0  0  !  BREAK.
000000 0  0  !
000000 0  0  PROC  SCREENTEST  MAIN;
000000 1  0     BEGIN
000000 1  1     INT  CNT,                    ! TERMINAL TRANSMISSION COUNT
000000 1  1          I,                      ! BYTE COUNTER
000000 1  1          FIRST := 1;             ! FIRST TIME FLAG
000000 1  1
000000 1  1     ! OPEN TERMINAL
000000 1  1     CALL  MYTERM( SCREENIO );
000006 1  1     CALL  OPEN( SCREENIO, TERM, 0 );
000016 1  1     CALL  IOCHK;
000017 1  1
000017 1  1     ! SET THE TERMINAL INTO PAGE MODE
000017 1  1     CALL  SETMODE( TERM, 8, 1 );
000026 1  1     CALL  IOCHK;
000027 1  1
000027 1  1     ! TAKE OVER THE BREAK KEY
000027 1  1     CALL  SETMODE( TERM, 11, MYPID, 0 );
000041 1  1     CALL  IOCHK;
000042 1  1
000042 1  1     ! DO THE SCREEN TESTING
000042 1  1     WHILE  1  DO
000042 1  1        BEGIN
000042 1  2        ! OUTPUT THE FORM AND DEFAULT VALUES TO THE TERMINAL. THE FORM ITSELF
000042 1  2        ! IS ONLY OUTPUT THE FIRST TIME.
000042 1  2        CALL  WRITE( TERM, SCREENIO,
000042 1  2                     EXPAND^SCREEN( @LICENSE, , SCREENIOS, FIRST ), CNT );
000066 1  2        CALL  IOCHK;
000067 1  2        DO
000067 1  2           BEGIN
000067 1  3           ! READ THE 3 CHARACTER FUNCTION KEY SEQUENCE
000067 1  3           CALL  READ( TERM, SCREENIO, 3, CNT );
000077 1  3           CALL  IOCHK;
000100 1  3           IF  ERRORMSG  THEN
000102 1  3              BEGIN  ! OVERWRITE THE PROTECTED BLINKING ERROR MESSAGE WITH
000102 1  4                     ! PROTECTED SPACES.
000102 1  4              SCREENIOS ':=' [ ESC, "'", ESC, ")", ESC, "=%G",
000103 1  4                                 32*[" "], ESC, "(", ESC, "&" ]  -> I;
000112 1  4                           ! OVERWRITES THE ERROR MESSAGE WITH BLANKS
000112 1  4              CALL  WRITE( TERM, SCREENIO, I-@SCREENIOS, CNT );
000124 1  4              CALL  IOCHK;
000125 1  4              ERRORMSG := 0;
000127 1  4              END;
000127 1  3           ! INPUT UNPROTECTED FIELDS FROM THE TERMINAL AND CHECK
000127 1  3           CALL  WRITEREAD( TERM, SCREENIO,
000127 1  3                            READ^SCREEN( @LICENSE, SCREENIOS ),
000127 1  3                            LICENSE^IOBUF, CNT );
000150 1  3           CALL  IOCHK;
```

```
PAGE 9    $DATA.M020A00.SAMPLES  [1]           SAMPLE  --  A SAMPLE DATA ENTRY PROGRAM

 235.     000151 1  3            END
 236.     000151 1  2          UNTIL  CHECK^SCREEN( @LICENSE, SCREEN, SCREENIOS, FIELDCHECK );
 237.     000163 1  2          FIRST := 0;
 238.     000165 1  2          END;
 239.     000174 1  1       END;

CNT                              VARIABLE      INT           L+001     DIRECT
FIRST                            VARIABLE      INT           L+003     DIRECT
I                                VARIABLE      INT           L+002     DIRECT

00000  002002 100001 024700 170002 024700 027000 170002 070004     00010 100000 024722 002005 100340 024700 027000 027000 040004
00020  100010 100001 000002 100034 024755 027000 027000 040004     00030 100013 024711 027000 000112 124211 000002 100036 024755
00040  027000 027000 040004 170002 000025 030001 020117 000200     00050 100000 170001 040403 024755 100013 024700 027000 000112
00060  124211 070401 000002 100036 024766 027000 027000 040004     00070 170002 100003 070401 000002 100036 024766 027000 027000
00100  040003 014425 170001 000025 020062 000200 030001 100054     00110 126040 044402 040004 170002 040402 170001 000211 070401
00120  000002 100036 024766 027000 027000 100000 044003 040004     00130 170002 000025 030001 020034 000200 170001 024733 027000
00140  000112 124211 020026 070401 000002 100076 024777 027000     00150 027000 000025 030001 020016 000200 170000 170001 020013
00160  024733 027000 014704 100000 044403 010654 170330 000073     00170 170156 003774 170116 000000 000002 024711 127000 015447
00200  015451 015475 022507 020040 020040 020040 020040 020040     00210 020040 020040 020040 020040 020040 020040 020040 020040
00220  020040 020040 020040 015450 015446
```

```
PAGE 10   $DATA.M020A00.SAMPLES  [1]           GLOBAL MAP


ABEND                           PROC
BLINK^SCREEN                    PROC       INT
BYTE^^                          PROC       INT
CHECK^SCREEN                    PROC       INT
ERRORMSG                        VARIABLE   INT          G+003     DIRECT
ESC                             LITERAL                           %000033
EXPAND^SCREEN                   PROC       INT
FIELDCHECK                      PROC       INT
FILEINFO                        PROC
FL^SCREEN                       PROC       INT
IOCHK                           PROC
LICENSE                         VARIABLE   STRING       P+000     DIRECT
LICENSE^ADDRESS1                DEFINE                            SCREEN[0074]
LICENSE^ADDRESS1^CHK            LITERAL                           %000001
LICENSE^ADDRESS1^LEN            LITERAL                           %000074
LICENSE^ADDRESS2                DEFINE                            SCREEN[0135]
LICENSE^ADDRESS2^CHK            LITERAL                           %000001
LICENSE^ADDRESS2^LEN            LITERAL                           %000044
LICENSE^CLASS                   DEFINE                            SCREEN[0214]
LICENSE^CLASS^CHK               LITERAL                           %000003
LICENSE^CLASS^LEN               LITERAL                           %000001
LICENSE^DAY                     DEFINE                            SCREEN[0206]
LICENSE^DAY^CHK                 LITERAL                           %000003
LICENSE^DAY^LEN                 LITERAL                           %000002
LICENSE^EXPIRE                  DEFINE                            SCREEN[0000]
LICENSE^EXPIRE^CHK              LITERAL                           %000003
LICENSE^EXPIRE^LEN              LITERAL                           %000002
LICENSE^EYES                    DEFINE                            SCREEN[0187]
LICENSE^EYES^CHK                LITERAL                           %000002
LICENSE^EYES^LEN                LITERAL                           %000003
LICENSE^FEET                    DEFINE                            SCREEN[0191]
LICENSE^FEET^CHK                LITERAL                           %000003
LICENSE^FEET^LEN                LITERAL                           %000001
LICENSE^HAIR                    DEFINE                            SCREEN[0183]
LICENSE^HAIR^CHK                LITERAL                           %000002
LICENSE^HAIR^LEN                LITERAL                           %000003
LICENSE^INCHES                  DEFINE                            SCREEN[0193]
LICENSE^INCHES^CHK              LITERAL                           %000003
LICENSE^INCHES^LEN              LITERAL                           %000002
LICENSE^IOBUF                   LITERAL                           %003774
LICENSE^LENSES                  DEFINE                            SCREEN[0216]
LICENSE^LENSES^CHK              LITERAL                           %177776
LICENSE^LENSES^LEN              LITERAL                           %000001
LICENSE^LICLET                  DEFINE                            SCREEN[0003]
LICENSE^LICLET^CHK              LITERAL                           %000002
LICENSE^LICLET^LEN              LITERAL                           %000001
LICENSE^LICNUM                  DEFINE                            SCREEN[0005]
LICENSE^LICNUM^CHK              LITERAL                           %000003
LICENSE^LICNUM^LEN              LITERAL                           %000007
LICENSE^MONTH                   DEFINE                            SCREEN[0203]
LICENSE^MONTH^CHK               LITERAL                           %000003
LICENSE^MONTH^LEN               LITERAL                           %000002
LICENSE^NAME                    DEFINE                            SCREEN[0013]
LICENSE^NAME^CHK                LITERAL                           %000001
LICENSE^NAME^LEN                LITERAL                           %000074
LICENSE^OTHER                   DEFINE                            SCREEN[0218]
```

```
PAGE 11   $DATA.M020A00.SAMPLES   [1]          GLOBAL MAP

LICENSE^OTHER^CHK          LITERAL                             %177777
LICENSE^OTHER^LEN          LITERAL                             %000066
LICENSE^PREVEXP            DEFINE                              SCREEN[0200]
LICENSE^PREVEXP^CHK        LITERAL                             %000003
LICENSE^PREVEXP^LEN        LITERAL                             %000002
LICENSE^SBUF               LITERAL                             %000420
LICENSE^SEX                DEFINE                              SCREEN[0181]
LICENSE^SEX^CHK            LITERAL                             %000002
LICENSE^SEX^LEN            LITERAL                             %000001
LICENSE^STATE              DEFINE                              SCREEN[0172]
LICENSE^STATE^CHK          LITERAL                             %000002
LICENSE^STATE^LEN          LITERAL                             %000002
LICENSE^WEIGHT             DEFINE                              SCREEN[0196]
LICENSE^WEIGHT^CHK         LITERAL                             %000003
LICENSE^WEIGHT^LEN         LITERAL                             %000003
LICENSE^YEAR               DEFINE                              SCREEN[0209]
LICENSE^YEAR^CHK           LITERAL                             %000003
LICENSE^YEAR^LEN           LITERAL                             %000004
LICENSE^ZIP                DEFINE                              SCREEN[0175]
LICENSE^ZIP^CHK            LITERAL                             %000003
LICENSE^ZIP^LEN            LITERAL                             %000005
MYPID                      PROC        INT
MYTERM                     PROC
NUMIN                      PROC        INT
NUMOUT                     PROC
OPEN                       PROC
POSITION^SCREEN            PROC        INT
POSITION^^                 PROC        INT
READ                       PROC
READ^SCREEN                PROC        INT
SCREEN                     VARIABLE    STRING      G+000       INDIRECT
SCREENIO                   VARIABLE    INT         G+002       INDIRECT
SCREENIOS                  VARIABLE    STRING      G+001       INDIRECT
SCREENTEST                 PROC
SCREEN^VERSION^B01^^       PROC
SETMODE                    PROC
STOP                       PROC
TERM                       VARIABLE    INT         G+004       DIRECT
TRAPSOFF^^                 PROC
WORD^^                     PROC        INT
WRITE                      PROC
WRITEREAD                  PROC
```

```
PAGE 12   $DATA.M020A00.SAMPLES  [1]              LOAD MAP

PEP    BASE    LIMIT   ENTRY  ATTRIBUTES   NAME

002   001105  001263  001135               BLINK^SCREEN
003   001023  001031  001023               BYTE^^
004   001725  002374  001776               CHECK^SCREEN
005   001347  001673  001445     V         EXPAND^SCREEN
006   002504  003576  003160               FIELDCHECK
007   001074  001104  001074               FL^SCREEN
010   002375  002503  002375               IOCHK
011   001264  001346  001264               POSITION^SCREEN
012   001041  001073  001041               POSITION^^
013   001674  001724  001674               READ^SCREEN
014   003577  004023  003577     M         SCREENTEST
015   001016  001016  001016               SCREEN^VERSION^B01^^
016   001017  001022  001017               TRAPSOFF^^
017   001032  001040  001032               WORD^^
```

```
OBJECT FILE NAME IS $DATA.M020A00.SAMPLE
NO. ERRORS=0   ; NO. WARNINGS=0
PRIMARY GLOBAL STORAGE=5
SECONDARY GLOBAL STORAGE=1159
CODE SIZE=1542
DATA AREA SIZE=3  PAGES
CODE AREA SIZE=3  PAGES
MAXIMUM SYMBOL TABLE SIZE=1621
NUMBER OF SOURCE LINES=2695
ELAPSED TIME -   0: 1:16
```

FILE MANAGEMENT ERROR LIST

<error>		description	<device type>
CCE			
0		operation successful	any
CCG			
1		end-of-file	3,4,6
2		operation not allowed on this type file	any
3		failure to open or purge a partition	3
4		failure to open an alternate key file	3
5		failure to provide sequential buffering	3
6		system message received	2
7		process not accepting CONTROL or SETMODE	0
CCL			
10	(%12)	file/record already exists	3
11	(%13)	file not in directory or record not in file	3
12	(%14)	file in use	3,4,5,6
13	(%15)	illegal filename specification	any
14	(%16)	device does not exist	3,4,5,6
15	(%17)	volume specification supplied does not match name of volume on which the file actually resides	3
16	(%20)	file number has not been opened	any
17	(%21)	paired-open was specified and the primary application process is not alive	3,4,5,6
21	(%25)	illegal <count> specified	any
22	(%26)	application parameter or buffer address out of bounds	any

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
23 (%27)	disc address out of bounds	3
24 (%30)	privileged mode required for this operation	any
25 (%31)	AWAITIO or CANCEL attempted on "wait" file	any
26 (%32)	AWAITIO or CANCEL attempted on a file with no outstanding operations	any
27 (%33)	"wait" operation attempted when outstanding requests pending	any
28 (%34)	number of outstanding no-wait operations would exceed that specified, or attempt to open a disc file or $RECEIVE with maximum no. of concurrent operations > 1	any
29 (%35)	missing parameter	any
30 (%36)	unable to obtain main memory space for a link control block	0,1,3,4,5, 6
31 (%37)	unable to obtain SHORTPOOL space for a file system buffer area	any
32 (%40)	unable to obtain LONGPOOL space for a file control block	any
33 (%41)	I/O process is unable to obtain IOPOOL space for an i/o buffer or <count> too large for dedicated i/o buffer	1,3,4,5,6
40 (%50)	operation timed out. AWAITIO did not complete within the time specified by its <time limit> parameter. If a 0D <time limit> (completion check) or -1 <file number> (any file) was specified, then the operation is considered incomplete. Otherwise, the operation is considered completed	any
42 (%52)	attempt to read from unallocated extent	3
43 (%53)	unable to obtain disc space for extent	3

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
44 (%54)	directory is full	
45 (%55)	file is full	3
46 (%56)	invalid key specified	3E
47 (%57)	key not consistent with file data	3E
48 (%60)	security violation	3
49 (%61)	access violation	any
50 (%62)	directory error	3
51 (%63)	directory is marked bad	3
52 (%64)	error in disc free space table	3
53 (%65)	file system internal error	3
54 (%66)	i/o error in disc free space table	3
55 (%67)	i/o error in directory	3
56 (%70)	i/o error on volume label	3
57 (%71)	i/o error in file label	3
58 (%72)	disc free space table is marked bad	3
59 (%73)	file is bad	3
60 (%74)	volume on which this file resides has been removed or device has been downed since the file was opened	3,4,5,6
61 (%75)	no file opens are permitted	3
62 (%76)	volume has been mounted, but mount order has not been given, file open not permitted	3
63 & 64 (%77 & %100)	volume has been mounted and mount is in progress, file open not permitted	3
65 (%101)	only special requests permitted	3

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
66 (%102)	device has been downed by operator	1,3,4,5,6
71 (%107)	duplicate record	3E
72 (%110)	attempt to access unmounted partition	3
73 (%111)	file/record locked	3/3E
74 (%112)	READUPDATE called for $RECEIVE and number of messages queued exceeds <receive depth> - or REPLY called with an invalid <message tag>	2
99 (%143)	Enscribe option not installed	3
100 (%144)	device not ready	3,4,5,6
101 (%145)	no write ring	4
102 (%146)	paper out	5
103 (%147)	disc not ready due to power fail	3
110 (%156)	only break access permitted	6
111 (%157)	operation aborted because of break	6
112 (%160)	READ or WRITEREAD preempted by operator message	6
120 (%170)	data parity error	1,3,4,5,6
121 (%171)	data overrun error	1,3,4,5,6
130 (%202)	illegal address to disc	3
131 (%203)	write check error from disc	3
132 (%204)	seek incomplete from disc	3
133 (%205)	access not ready on disc	3
134 (%206)	address compare error on disc	3
135 (%207)	write protect violation with disc	3

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
140 (%214)	modem error (communication link not yet established, modem failure, momentary loss of carrier, or disconnect)	6
150 (%226)	end-of-tape marker detected	4
151 (%227)	runaway tape detected	4
152 (%230)	unusual end - tape unit went off line	4
153 (%231)	tape drive power on	4
154 (%232)	BOT detected during backspace files or backspace records	4
160 (%240)	request is invalid for line state	7
161 (%241)	impossible event occurred for line state	7
162 (%242)	operation timed out	7
163 (%243)	EOT received	7
164 (%244)	disconnect received	7
165 (%245)	RVI received	7
166 (%246)	ENQ received	7
167 (%247)	EOT received on line bid	7
168 (%250)	NAK received on line bid	7
169 (%251)	WACK received on line bid	7
170 (%252)	no id sequence received during circuit assurance mode	7
171 (%253)	no response received	7
172 (%254)	reply not proper for protocol	7
173 (%255)	maximum allowable NAKs received	7
174 (%256)	WACK received	7

-->

FILE MANAGEMENT ERROR LIST (cont'd)

<error>	description	<device type>
175 (%257)	incorrect alternating ACK received	7
176 (%260)	poll sequence ended with no responder	7
177 (%261)	text overrun (insufficient buffer space for data transfer)	7
190 (%276)	invalid status received from device	1,3 - 7
200 (%310)	device is owned by alternate port	1,3 - 7
201 (%311)	the current path to the device is down or an attempt was made to write to a non-existent process	1,3 - 7 0
210 (%322)	device ownership changed during operation	1,3 - 7
211 (%323)	failure of CPU performing this operation	any
212 (%324)	EIO instruction failure	1,3 - 7
213 (%325)	channel data parity error	1,3 - 7
214 (%326)	channel timeout	1,3 - 7
215 (%327)	i/o attempted to absent memory page	1,3 - 7
216 (%330)	map parity error during this i/o	1,3 - 7
217 (%331)	memory parity error during this i/o	1,3 - 7
218 (%332)	interrupt timeout	1,3 - 7
219 (%333)	illegal device reconnect	1,3 - 7
220 (%334)	protect violation	1,3 - 7
221 (%335)	pad-in violation	1,3 - 7
222 (%336)	bad channel status from EIO instruction	1,3 - 7
223 (%337)	bad channel status from IIO instruction	1,3 - 7
230 (%346)	CPU power failed then restored	1,3 - 7
231 (%347)	controller power failed then restored	1,3 - 7

READER'S COMMENTS

Tandem welcomes your feedback on the quality and usefulness of its publications. Please indicate a specific *section* and *page* number when commenting on any manual. Does this manual have the desired completeness and flow of organization? Are the examples clear and useful? Is it easily understood? Does it have obvious errors? Are helpful additions needed?

Title of manual(s) ______________________________

FOLD ➤

FOLD ➤

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

City State ______________________ Zip ____________

A written response is requested. yes no ?

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 482 CUPERTINO, CA. U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM
COMPUTERS, INC.

Attn: Technical Publications
19333 Vallco Parkway
Cupertino, CA, U.S.A. 95014

FOLD

STAPLE HERE

102689439

1026 89439