TANDEM

USERS MANUAL

NonStop™
SYSTEMS

XRAY
USERS MANUAL

T16/8037A02
82037

TANDEM 16

XRAY

USERS MANUAL

UPDATE A03

TANDEM COMPUTERS INCORPORATED
19333 Vallco Parkway
Cupertino, California 95014

Reasons for Manual Update (Version A03):

Addition of new feature (LIST command) and miscellaneous changes.

Update insertion information on reverse side

Product No. T6/8037-A03
Part No. 82143

January 1980
Printed in U.S.A.

UPDATE INSERTION INSTRUCTIONS

This update package contains both replacement pages and new pages for your existing manual. When inserting a replacement page be sure to remove the old page.

Before updating your manual, be sure the manual is at the previous level. Your manual must be at the A02 level before this update package is inserted.

Replace	inner cover
"	(3)-1
"	3-3 through 3-24
Add	3-25 and 3-26
Replace	(4)-1
"	4-11 through 4-69
Add	4-70
Replace	Index-1 through Index-4

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.

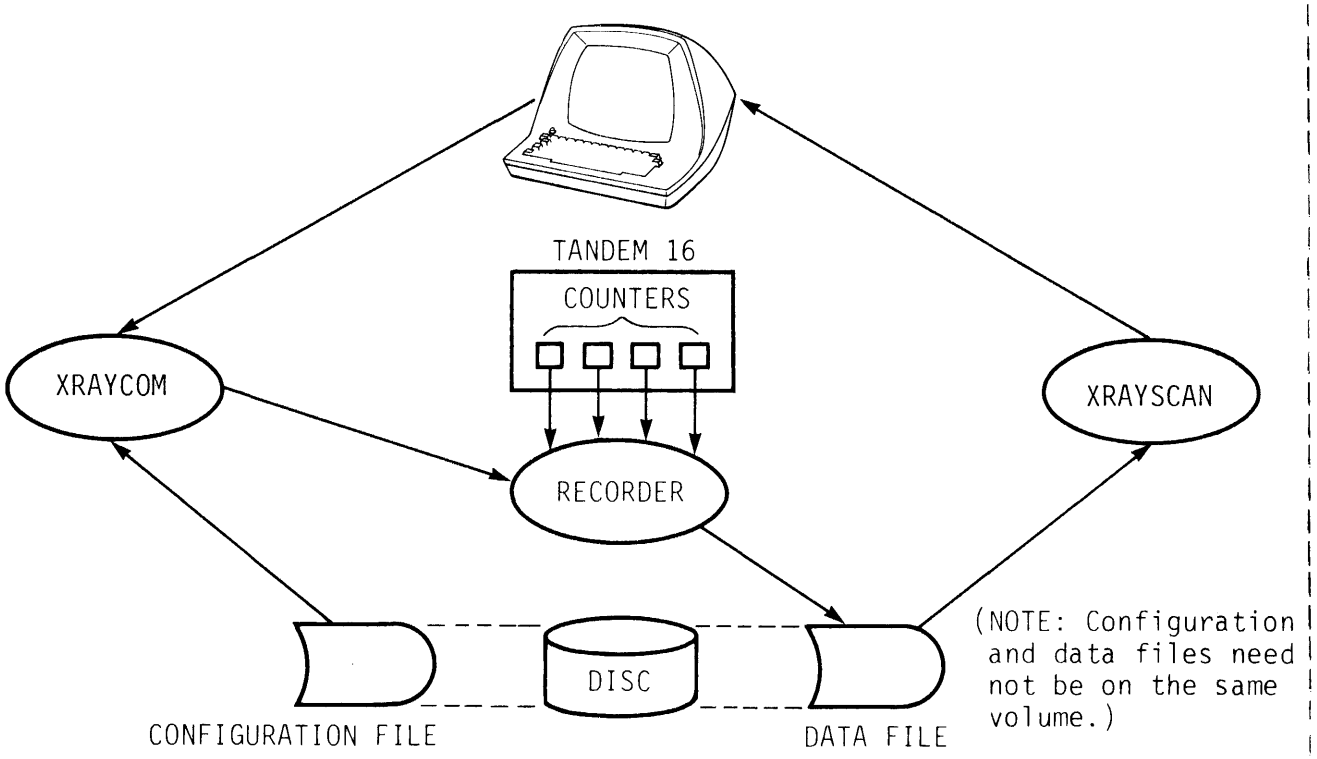
   XRAYSCAN is easy to use if you take it a step at a time. You will learn most quickly by using XRAY on some program you know well, in an isolated environment, i.e., "standalone", if possible. Explore this measurement in some detail, to become familiar with the analysis capabilities. Analysis is simple 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:

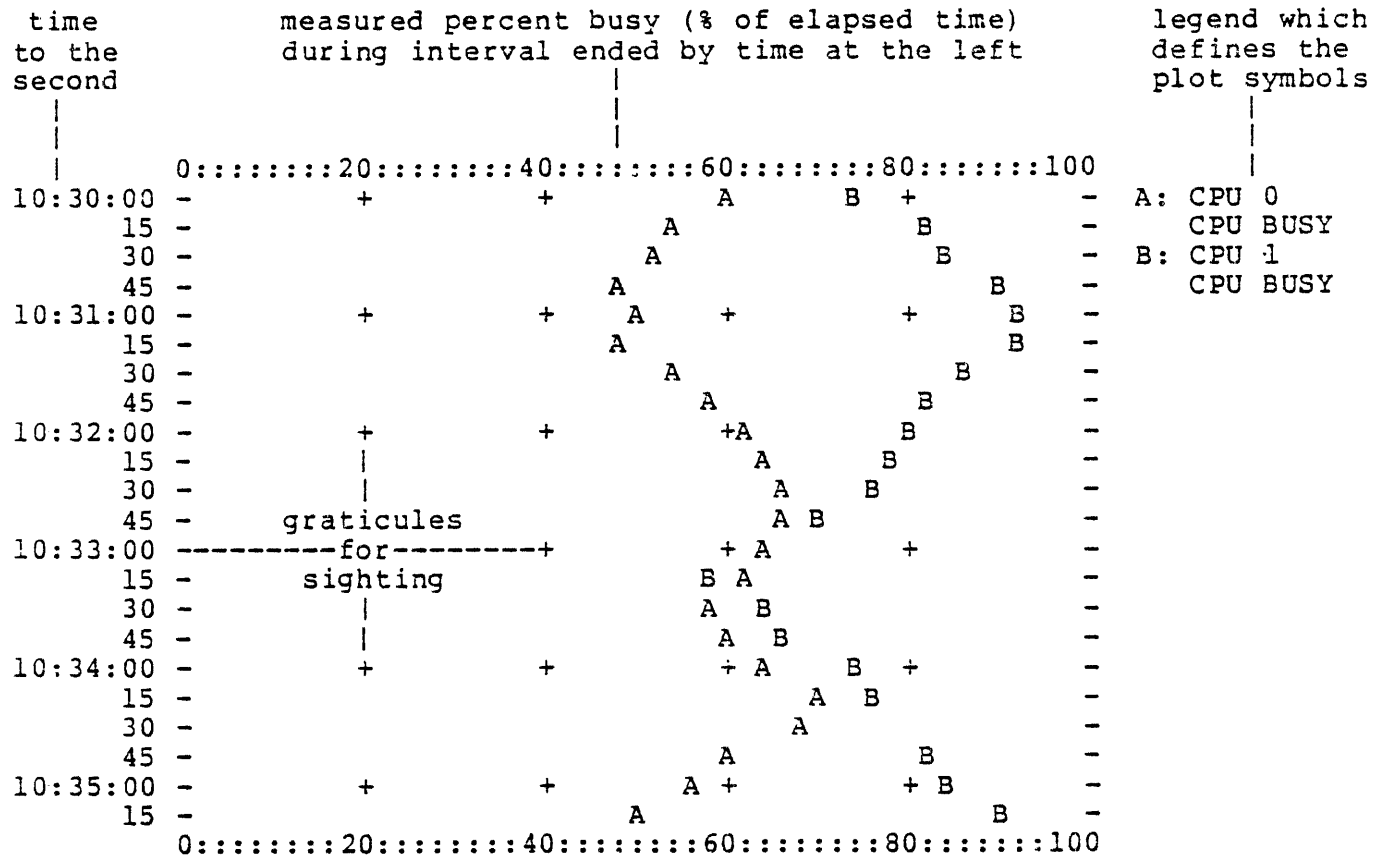
```
+FILE  $ORDERS.*.*, BY FILE RATE
```

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
```



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.

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: $SPOOL
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: $SPOOL
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.

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

$$\frac{n * t}{g} * \text{multiplier}$$

where: multiplier = { 512 bytes for a null configuration
{ 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.

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.)

CONSIDERATIONS

* XRAYCOM should always be run in CPU 0. If the RECORDER is defective and stops the CPU, this will prevent XRAYCOM from creating RECORDERs in the remaining CPUs.

* SHADOW diagnostics and XRAY should never be run together since they both require exclusive use of several system resources. The program LOWLITE and XRAY are also mutually exclusive. Before running SHADOW or LOWLITE enter the EXIT! command in XRAYCOM to stop XRAY.

* After a power failure the LIGHTS command in XRAYCOM must be issued to restart the Diaglink controller(s) and the front panel lights.

* The use of (COMINT) SETTIME during a measurement will invalidate the analysis and may have otherwise unpredictable results.

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
```

CONSIDERATIONS

* 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.
```

The configuration file consists of six optional sections that describe the entities that XRAY is to measure. Each section is headed by one of eight keywords followed by a colon (:). The keywords are PROCESSORS, PROGRAMS, FILES, DEVICES, SYSTEMS, BUFFERS, PROCEDURES, and EXCLUDE.

Under each keyword is a list of entity descriptions, one for each entity to be measured (or, in the case of the EXCLUDE keyword, entities not to be measured).

These sections are described in greater detail in the next table. Lines beginning with an exclamation point (!) are comments.

CONFIGURATION FILE SECTIONS

PROGRAMS:	! Contains a list of the sets of programs ! and processes which are to be measured. ! The special keyword SYSPROCS can be used ! to specify that all system processes are ! to be measured.
FILES:	! Lists the sets of files to be measured. ! Each file open is measured separately. ! Calls to system procedures like READ, ! WRITE, etc. are counted. For disc files ! residing on the system under measurement, ! calls to the disc driver procedures made ! by the Disc Processes are also tallied. ! To measure activity against remote files, ! the system name must be included in the ! fileset. To measure Comm Access Method ! activity at the file level, the line and ! subdevice names must be included here.
DEVICES:	! Lists names or logical device numbers of ! physical devices to be measured. ! Additionally, lists the names of lines and ! subdevices for Comm Access Method ! measurement. The counters for physical ! devices typically include device ! utilization, logical requests to the ! managing process for the device, physical ! I/O commands, and byte transfer rates.
SYSTEMS:	! Includes a list of the systems in a ! network of TANDEM systems whose ! traffic flow is to be counted and ! timed.

-->

PROCESSORS:	! If this section is omitted, all cpus collect data. If this section is included, it contains a list of processors which are to collect data. Its purpose is to restrict the cpus participating in the collection of data.
POOLS:	! Includes a list of processors, for which system pool activity is to be measured.
BUFFERS:	! Lists names of programs or processes or process identifications of processes for which buffer activity is to measured. The special keyword SYSPROCS can be used to specify that buffer usage for all system processes be measured.
PROCEDURES:	! Lists the name of program files or edit files containing entry and subentry point names and addresses to be measured. The measurement is restricted to the named processes.
EXCLUDE:	! This section contains a list of entities that are NOT to be measured, even though they may be included in one of the above lists.

The next table contains a list of the set identifiers which may appear in each of the configuration sections. Lines beginning with an exclamation point (!) are comments.

CONFIGURATION FILE ENTITY SETS

PROGRAMS:

```
$<volume>.<subvol>.<filename>
  ! Measure all processes executing this program file.

$<volume>.<subvol>.*
  ! Measure all processes executing a program file in this
  ! subvolume.
```

-->

```
    $<volume>.*.*
      ! Measure all processes executing a program
      ! residing on this volume.

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

    <cpu, pin>
      ! Measure the specified process.

    SYSPROCS
      ! Measure all system processes (these manage virtual
      ! memory, I/O devices, the system console, and so on).

FILES:

    $<volume>.<subvol>.<filename>
       ! Measure the activity of this disc file.

    $<volume>.<subvol>.*
       ! Measure each file in the subvolume.

    $<volume>.*.*
       ! Measure every file opened on the volume, including all
       ! temporary as well as permanent files.

    $<volume>.#tempnum
       ! Measure the designated temporary file.

    $<device name>
       ! Measure every file opened on the device.

    $<process name>
       ! Measure every file opened on the process.

    \<sysname>.$<volume>.<subvol>.<filename>
       ! Measure the activity of this remote disc file opened
       ! by a process on the measured system.

    \<sysname>.$<volume>.<subvol>.*
       ! Measure each file in the subvolume opened by a
       ! process on the measured system.

    \<sysname>.$<volume>.*.*
       ! Measure files opened on the remote volume, including
       ! all temporary as well as permanent files.
```

-->

```
    \<sysname>.$<volume>.#tempnum
       ! Measure the designated temporary file.

    \<sysname>.$<device name>
       ! Measure every file opened locally on the device.

    \<sysname>.*.*.*
       ! Measure every file opened on the remote system by
       ! a process running on this system.

    $<linename>.*
       ! Measure Comm Access Method access to all subdevices on
       ! the line.

    $<linename>.#<subname>
       ! Measure Comm Access Method access to the specified
       ! subdevice.

  DEVICES:

    $<device name>
       ! Measure the named device.

    $<logical device number>
       ! Measure the device with the given logical device
       ! number.

    ASYNC
       ! Measure all the asynchronous, point-to-point
       ! terminals.

    $<linename>.*
       ! Measure Comm Access Method physical access to the
       ! specified line and all subdevices opn the line.

    $<linename>.#<subname>
       ! Measure Comm Access Method physical access to the
       ! specified line and the specified subdevice.

    $<linename>
       ! Measure Comm Access Method physical access to the
       ! specified line.


  SYSTEMS:

    \<system name>
       ! Measure traffic to the named system.

                                                          -->
```

```
    \<system number>
       ! Measure traffic to the system with the given system
       ! number.

  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.)


  POOLS:

     <cpunum>

        ! The Recorder in the cpus listed will emit pool data to
        ! the output file.


  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, iopool and control block space.

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.

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

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 no other configuration file was defined during this run of XRAYCOM, the subsequent GO command started 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.
```

CONSIDERATIONS

* 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?"

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
```

CONSIDERATIONS

* 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.

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.

The error messages output by XRAYCOM are listed below. Where the meaning is obvious, no explanation is given.

RECORDER MISSING EXTERNALS IN CPU <n>; PROCESSOR MAY HALT

The XRAY software may have been incorrectly installed on the system; <n> specifies the processor number.

CREATION FILE ERROR <e> IN CPU <n>; RECORDER NOT CREATED

A NEWPROCESS error occurred trying to create the Recorder; <e> specifies the error number and <n> specifies the processor number.

FORMAT ERROR IN CONFIGURATION FILE; MEASUREMENT NOT CONFIGURED

An EDITREAD problem occurred in the configuration file.

ILLEGAL CONFIGURATION FILE NAME SUPPLIED; MEASUREMENT NOT CONFIGURED

ILLEGAL SYNTAX; MEASUREMENT NOT CONFIGURED

INPUT ERROR ON CONFIGURATION FILE; MEASUREMENT NOT CONFIGURED

An EDITREAD problem occurred in the configuration file.

PROGRAM FILE ILLEGAL FOR CPU n; RECORDER NOT CREATED

The Recorder has the wrong program file.

SEQUENCE ERROR IN CONFIGURATION FILE; MEASUREMENT NOT CONFIGURED

This is an EDITREAD error.

THE COMPILED MEASUREMENT IS TOO BIG; MEASUREMENT NOT CONFIGURED

The upper limit is approximately 10K nonblank bytes in the configuration file.

THE MEASUREMENT CANNOT BE CONFIGURED BECAUSE IT IS TOO BIG!

The upper limit is approximately 10K nonblank bytes in the configuration file.

THIS CHARACTER SEEMS OUT OF PLACE; MEASUREMENT NOT CONFIGURED

This is a command syntax error.

-->

THIS IS NOT A LEGAL FILENAME

THIS IS NOT THE NUMBER OF A PROPER CPU

THIS WORD SEEMS OUT OF PLACE; MEASUREMENT NOT CONFIGURED

This is a command syntax error.

TOO FEW PROCESS CONTROL BLOCKS IN CPU n; RECORDER NOT CREATED

The system is under-configured.

UNABLE TO ALLOCATE A MAP IN CPU n; RECORDER NOT CREATED

The system is under-configured.

UNABLE TO ALLOCATE VIRTUAL DISC SPACE FOR CPU n; RECORDER NOT CREATED

$SYSTEM has insufficient contiguous free space to run the Recorder.

UNABLE TO COMMUNICATE WITH CPU n; RECORDER NOT CREATED

UNABLE TO OPEN THE CONFIGURATION FILE; MEASUREMENT NOT CONFIGURED

UNEXPECTED CREATION ERROR IN CPU n; RECORDER NOT CREATED

A NEWPROCESS error occurred in creating the Recorder.

UNEXPECTED PROCESS NAME ERROR IN CPU n; RECORDER NOT CREATED

The Recorder must be run as an unnamed process. This message indicates an XRAYCOM internal error.

UNLICENSED PRIVILEGED PROGRAM; RECORDER NOT CREATED

The Recorder must be licensed.

WARNING: REMOTE MEASUREMENT ON LOCAL-ONLY SYSTEM; IGNORED

Attempt to invoke measurement on remote traffic when the system is incapable of network activity. The specific request is ignored and the remainder of the configuration is accepted.

-->

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>                         }
            { $<process name>                        }
            { \<sysname>.$<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> ...
                                                   -->
```

```
        POOLS:  <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

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 (+).
```

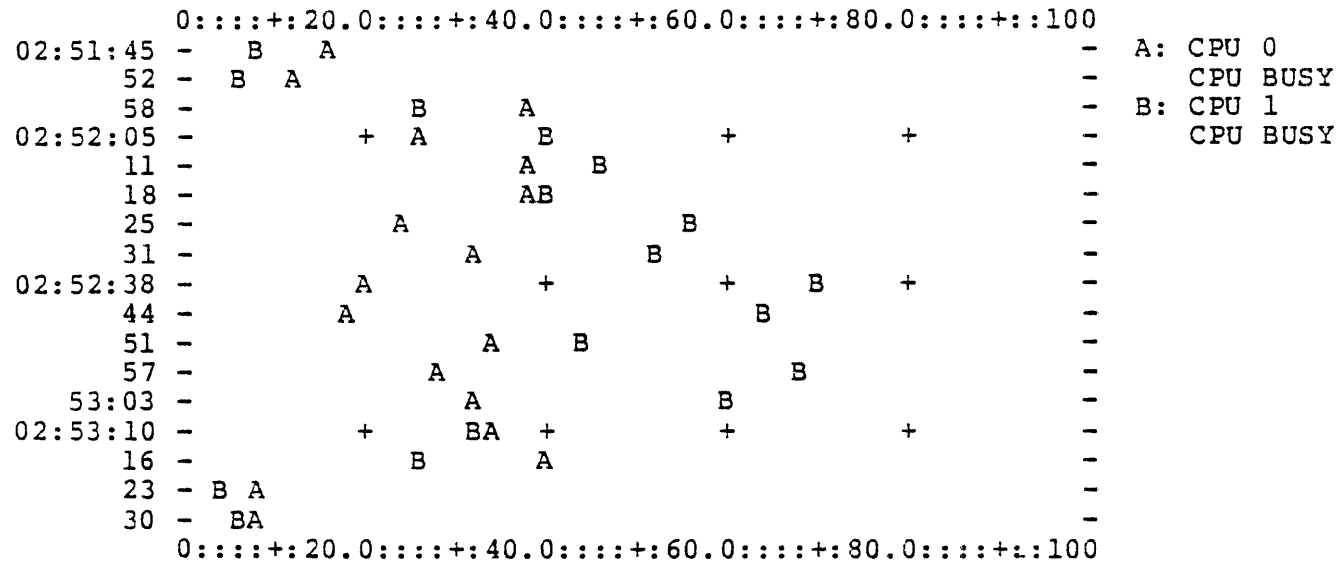
This command displays entities for all processors under measurement.

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



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.

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 four additional commands:

LIST	Lists entities measured.
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 RATE

   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 RATE

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 RATE

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 RATE

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.
```

CONSIDERATONS

* 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.

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 RATE

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.
```

CONSIDERATIONS

* 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.

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
```

DEVICE Reports

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)

FREE RATE

The count of the number of disc free space table I/Os.

-->

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.

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.

                                                              -->
```

```
\<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>
```

FILE Reports

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, but does not eliminate any items from the current plot (once an item is added to the plot, it 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 before items are added to the plot; thus, PLOT CPU BUSY adds the CPU BUSY 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.

LINE Reports

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 LIST command will generate a list of all entities measured for a given entity type.

The form of the LIST command is:

```
LIST <entity type>
     -------------

   <entity type> is one of:

      { BUFFER     }
      { CPU        }
      { DEVICE     }
      { DISC       }
      { DISCOPEN   }
      { FILE       }
      { LINE       }
      { NETLINE    }
      { POOL       }
      { PROCEDURE  }
      { PROCESS    }
      { SYSTEM     }
      { TERMINAL   }
```

CONSIDERATIONS

* Note that all entities measured will be listed, even though there may not be an item for that entity with a value large enough to generate a report or plot. XRAYSCAN only generates reports on entities with at least one item whose reported value is greater than 0.

* As with the PROCEDURE command, a PROCESS must be specified prior to a LIST PROCEDURE command.

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.

PLOT Command

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).
```

CONSIDERATIONS

* 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 RATE

   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 RATE

   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 RATE

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 RATE

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 noticeable 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.

* MAP BUSY, shown in the lower left hand corner to the left of the scale, gives the fraction of the process's cpu time spent in the sampled map. The balance is spent in the system map on behalf of the measured process. Multiplying MAP BUSY times CPU BUSY from the process report will yield the percent of elapsed time spent in the user code. Multiplying this figure times the percentage for an individual procedure will give the percent of elapsed time spent in that procedure. All three of these numbers are percents in the range of 0 to 100.

PROCESS Reports

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 RATE

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).

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 one of the following.

      (blank)     ! Report on all measured system traffic.

      <cpu>       ! Report on all systems accessed through the give
                    cpu.

      \<sysname>  ! Report on the specified system.

      \<sysname>,<cpu>    ! Report on the specified system when
                            accessed through the given cpu.

      \<systemnumber>     ! Report on the specified system.

      \<systemnumber>,<cpu> ! Report on the specified system when
                              accessed through the given cpu.

 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.

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.
```

CONSIDERATIONS

* 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 RATE			
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 RATE	
		LINK FAIL	
		LINKS INUSE	

-->

XRAYSCAN ITEMS

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	04095 RATE	

BUFFER		POOL	
SHORT BUSY	LONG BUSY	SHORT BUSY	LONG BUSY
SHORT RATE	LONG RATE	SHORT RATE	LONG RATE
SHORT FAIL	LONG FAIL	SHORT FAIL	LONG FAIL
SHORT BYTES	LONG BYTES	SHORT BYTES	LONG BYTES
IO BUSY	CB BUSY	IO BUSY	CB BUSY
IO RATE	CB RATE	IO RATE	CB RATE
IO FAIL	CB FAIL	IO FAIL	CB FAIL
IO BYTES	CB WORDS	IO BYTES	CB WORDS

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> [ , <BY clause> ] [ , <IF clause> ]
------ ----------------------

    <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>     }


LIST <entity type>
---- -------------

   <entity type> is one of:

      { BUFFER    }
      { CPU       }
      { DEVICE    }
      { DISC      }
      { DISCOPEN  }
      { FILE      }
      { LINE      }
      { NETLINE   }
      { POOL      }
      { PROCEDURE }
      { PROCESS   }
      { SYSTEM    }
      { TERMINAL  }

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 [ , <BY clause> ]
----------

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

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:

$$\frac{util(j)}{demand(j)} = \frac{busy(j) / I}{busy(j) / trans} = trans / I = throughput$$

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.

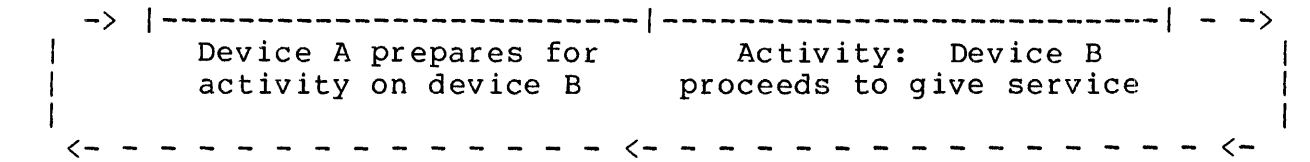
We may also discover that util(j) is less than maxutil(j) for all devices, showing hidden sequencing constraints between devices. Sometimes these can be eliminated (using no-wait I/O, for example). Thus, without changing demand(j), util(j) is forced closer to our conservative maxutil(j), and throughput 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.



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

$$\text{demand}(j, \text{class}) = \frac{\text{service}(j) * \text{visits}(j, \text{class})}{\text{transactions}(\text{class})}$$

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

$$P + Q = 1$$

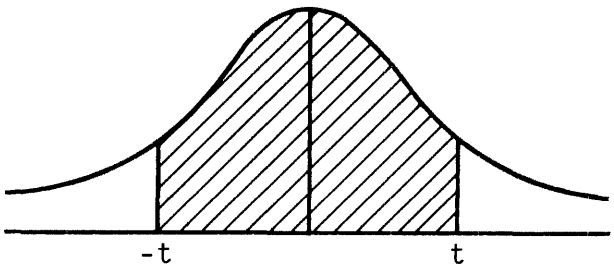
and both P and Q lie between 0 and 1, the maximum value of P * Q occurs when both are 1/2, and their product is 1/4 .

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.

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

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:

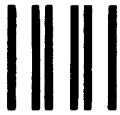
Name _____

Company _____

Address _____

City/State _____ Zip _____

A written response is requested, yes no ?



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

FOLD

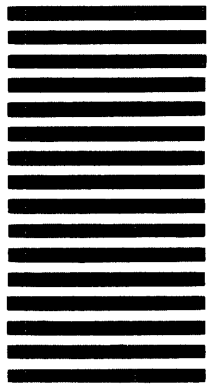
BUSINESS REPLY MAIL

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