NonStop™ Systems

NonStop SQL Benchmark Workbook

Data Management Library

84160

TANDEM
SOFTWARE
PUBLICATIONS

TANDEM SOFTWARE NOMENCLATURE

The term "NonStop™ systems" refers to all configurable hardware-software combinations for NonStop II™, NonStop TXP™, NonStop VLX™, NonStop EXT™, NonStop EXT10™, and NonStop EXT25™ systems.

The term "NonStop 1+ ™ system" refers to the combination of NonStop 1+ processors with all software that runs on them.

Some software manuals pertain to the NonStop systems only, others pertain to the NonStop 1+ system only, and still others pertain both to the NonStop systems and to the NonStop 1+ system. The cover and title page of each manual indicate the system (or systems) to which the contents of the manual pertain.

Some older manuals were written for both the NonStop systems and the NonStop 1+ system but are now current only for the NonStop 1+ system. In these cases, a separate more recent manual exists for the NonStop systems only.

WORKBOOK OF THE TOPGUN BENCHMARK DEMONSTRATING NonStop SQL PERFORMANCE ON 32 TANDEM VLX PROCESSORS AND AUDITED BY CODD AND DATE CONSULTING GROUP

6 March 1987

AUDITOR:
Tom Sawyer

BENCHMARKERS:
Dan Adachi
Nhan Chu
Frank Clugage
Jim Enright
Ray Glasstone
Jim Gray
Gerhard Huff
Jeff Laplante
Jack Mauger
Frank O'Donnell
Harald Sammer
Praful Shah
Scott Sitler
Thomas Wilkens

Printed in the United States of America.
93 92 91 90 89 88 87 10 9 8 7 6 5 4 3 2 1

The following are trademarks or service marks of Tandem Computers Incorporated:

6AX	ENFORM	GUARDIAN	NonStop EXT	PS MAIL	Tandem
BINDER	ENSCRIBE	GUARDIAN 90	NonStop EXT10	PS TEXT EDIT	TGAL
CROSSREF	ENVOY	GUARDIAN 90XF	NonStop EXT25	PS TEXT FORMAT	THL
DATABOLT	ENVOYACP/XF	INSIGHT	NonStop SQL	RDF	TIL
DDL	EXCHANGE	INSPECT	NonStop II	SAFE	TMF
DYNABUS	EXPAND	MEASURE	NonStop TXP	SAFEGUARD	TRANSFER
DYNAMITE	FASTSORT	MULTILAN	NonStop VLX	SAFE-T-NET	VIEWPOINT
ENABLE	FAXLINK	NETBATCH	PATHMAKER	T-TEXT	WORDLINK
ENCOMPASS	FOX	NonStop	PCFORMAT	TACL	XL8
ENCORE	FOXII	NonStop 1+	PERUSE	TAL	XRAY

HYPERchannel is a trademark of Network Systems Corporation.

IBM and IBM PC are registered trademarks of International Business Machines Corporation.

Textpack4, Textpack6, IBM Displaywriter, DisplayWrite 2, DisplayWrite 3, Displaywriter communication software, and CICS are trademarks of International Business Machines Corporation.

LATTICE is a registered trademark of Lattice, Inc.

MultiMate is a trademark of Ashton-Tate, Inc.

UNIX is a trademark of AT&T.

Wang OIS workstation, Wang VS workstation, Wang OIS telecommunication software, and Wang VS telecommunication software are trademarks of Wang, Inc.

TABLE OF CONTENTS

12. 8 VLX RESULTS - ENSCRIBE
    * response time curves
    * 60TPS Driver system report
    * 59TPS Driver system report
    * 58TPS Driver system report
    * 56TPS Driver system report
    * 48TPS Driver system report
    * 40TPS Driver system report
    * CPU utilization by process type
    * CPU utilization by process type by cpu
    * Disc utilization per tansaction
    * Communication line utilization

13. LISTING OF AUDITOR'S TRANSACTION
14. LISTING OF SIMULATOR PROGRAM
15. LISTING OF SIMULATOR REPORT GENERATOR
16. LISTING OF SIMULATOR SCRIPT GENERATOR
17. A SAMPLE SIMULATOR SCRIPT
18. A SAMPLE SIMULATOR REPORT
19. LISTING OF REQUESTOR PROGRAM
20. LISTING OF SQL DATABASE DEFINITION
21. LISTING OF ENSCRIBE DATABASE DEFINITION
22. LISTING OF SQL SERVER PROGRAM
23. LISTING OF ENSCRIBE SERVER PROGRAM

Codd and Date
Consulting Group

Mr. Frank Clugage
Tandem Computers Incorporated
19191 Vallco Parkway
Cupertino CA 95014

March 6, 1987

Mr. Clugage:

The attached report describes the portions of the Debit/Credit benchmark audited by Codd and Date Consulting Group.

The 32-processor VLX system ran the benchmark in excess of 200 transactions per second for a period of 15 minutes.

The following additional attributes of the benchmark were verified:

- » The transaction is correctly implemented
- » The database is properly sized
- » The programs are unaware of data distribution
- » The percentage of inter-branch transactions is correct
- » The response time is correctly measured
- » The inter-arrival times are exponentially distributed
- » The recovery log is mirrored
- » Database updates are locked during a transaction
- » The performance is linear with system size
- » The application is implemented in COBOL and SQL
- » The network can mix small and large systems.

Respectfully yours,

Thomas H. Sawyer

Senior Consultant

408-997-3888 ■ TELEX 18-1159
Suite 109 ■ 6489 Camden Avenue
San Jose. Calif. 95120

Table of Contents

AUDIT REPORT OF TOPGUN BENCHMARK

Benchmark Synopsis

The Debit/Credit Transaction is a stylized automatic teller transaction. The process consists of updating the account balance of the teller user. The balance of the teller and branch are also updated. A history record is inserted to complete the database portion of the transaction.

The terminal portion of the transaction consists of receiving a 100 byte record from the automatic teller and returning a 200 byte record at transaction completion.

At least 15% of the accounts affected are in a different branch than the teller processing the transaction.

For a complete description of the debit/credit transaction, see "A Measure of Transaction Processing Power by Anon. el al." contained in the Auditor's Notebook.

Observed Results

Two hardware configurations were audited, each of which ran the debit/credit transaction implemented in COBOL and SQL. Both configurations delivered the performance documented elsewhere in this report. The verification section details the auditing techniques.

The larger hardware configuration consisted of four 8-processor VLX systems connected via a FOX ring. One VLX system was also connected to a remote EXT 10 via a 9.6K bit transmission line. Transactions were submitted directly to each system (including the EXT 10). The correct percentage of these were inter-branch. The EXT 10 also participated in inter-branch transactions at a rate commensurate with the size of its database.

Benchmark Implementation

The verification procedures are dependent on the Tandem implementation of the benchmark. Tandem implemented the benchmark with two sets of hardware - a driver system and the system to be measured.

The driver system (a 10-processor TXP system) was connected to the measured system via 56 K bit transmission lines. There was one line for each processor in the measured system. A transaction submitting program (the Driver Program) on the driver system simulated an Automatic Teller Network.

All measurements were taken, recorded and summarized on the driver system. The measured system contained two application programs. One program (the Requestor Program) received the transaction from the communication software, reformatted the transaction data and assigned it to a copy of the Server Program.

The Server Program performed the database updates and returned a confirmation to the Requestor Program. The latter program then sent a response to the driver system to complete the transaction.

Steps Involved in a Timing Run

Script Preparation

The data for each transaction is prepared by a script generation program. Each transaction record contains the account, branch, teller and amount involved.

The script generator uses a random number generator to determine the account number. This number is then used to determine the branch and teller. The random number generator is used a second time to determine if this transaction should be inter-branch. If so, the random number generator is used again to pick an account that is not in the branch.

Transaction Timing

Transaction timing is performed by the driver program as part of submitting transactions to the measured

system. The transaction is time-stamped when it is sent to the X.25 component on the driver machine. A second time-stamp is taken when the X.25 component returns the response to the driver software.

The response time is the difference of the two time-stamps. Note that the time-stamps include all X.25 processing and all transmission time in the response time. The standard benchmark response time requirement only counts the time spent in the measured system. Thus, the true response times are actually less than reported.

When the transaction completes, the driver calculates the inter-arrival rate of the next transaction relative to when it was submitted. If the inter-arrival time was exceeded by the response time, the next transaction is submitted immediately. If there is any remaining "think time", the transaction is queued until the think time elapses.

Tandem chose an exponential inter-arrival rate. This rate is the most difficult for transaction processing systems to accommodate. The easiest is a constant inter-arrival rate - e.g. each terminal submits one transaction every ten seconds with no variance.

The driver program writes a report record showing the response time, the calculated inter-arrival time of the next message and the delay factor if the response time exceeds the inter-arrival time.

Report preparation

After the transaction submission portion of the run is stopped, the report files from the driver system are collected and summarized. This summarization produces the transaction rate graph and inter-arrival rate graphs.

The inter-arrival rate graph shows the theoretical and the actual inter-arrival rate for each run. The theoretical rate is usually not met for extremely short inter-arrival times - the actual response times are longer. Where the computed times exceed the response time, the distribution takes on the correct shape.

The calculations and report logic were verified by inspecting the source code. A further check was made by spot-checking the dis-assembled code from the execution module with the listing.

Verification

Inter-branch transactions

The required percentage of 15% was verified two ways. The script generation code was inspected to validate the percentage generated and the output was spot checked to confirm the percentage. The spot check yielded 16%.

Response Time

Response time was verified two ways. The driver and reporter code was inspected for correctness. The driver code was modified to take additional transactions directly submitted by the auditor. During the transaction submission portion of the run, direct transactions were submitted. These transactions updated accounts on each of the nodes in the network. Upon receipt of the response, the computed response time was noted and later compared with the response times generated by the reports.

Linear Transaction Rate

The benchmark attempted to show an absolutely linear performance improvement as nodes were added to the system. The actual improvement is very close to linear but shows a growing divergence from true linearity as nodes are added. The auditor ascribes most of this fall-off to the effects of the inter-branch transactions.

In the benchmarked configuration, each VLX processor directly dealt with eight branches; a VLX system had eight processors. Thus in an 8 processor system, all inter-branch transactions would have their recovery scope contained on one system. The communication within a VLX system uses dual high speed buses (24Mbytes/second) and has no need to coordinate recovery scope with any other node on the system.

As nodes are added to the system, the percentage of transactions requiring inter-system coordination increases. For example, a system made up of two 8 processor VLXs would have one half of the inter-branch transactions spread across the network. As the number of nodes rises, the percentage increases until it approaches 100% of all inter-branch transactions or 15% of all transactions.

Indeed, the extrapolation of the 8 processor transaction rate to 16 and 32 processor systems predicts 116 and 232 tps. The interpolated rates were 106 and 208. These are 8 and 10% below linearity. The percentage of inter-node, inter-branch transactions for these two configurations is 7.5 and 11%.

Distributed Database Capability

The benchmark demonstrated three aspects of distributed data base support:

Program transparency

Distributed Updates

Asymmetric node support

Program Transparency

This feature allows applications to be coded as though all data is at one location. The system takes care of processing any data remote to the node executing the transaction.

The Tandem system permits spreading of a table across multiple systems based on the values of one column. This is commonly referred to as partitioning. For the benchmark, the accounts database was spread over 5 network nodes - 1 EXT 10 and 4 VLX systems. I inspected the application program (Server); there is no knowledge of this partitioning in the program.

Distributed Updates

This features coordinates updates that occur on more than one node for one transaction. This means the system must be prepared to abort all work done by a transaction on all nodes where it occurs.

The inter-branch transactions produce multiple node updates for eleven percent of the inter-branch transactions.

Additional tests were performed using the interactive SQL interface (SQLCI) where all updates were on a node different from the one processing the transaction.

Asymmetric Node Support

This feature allows processors of different power to participate in the same transaction. The EXT 10 participated in both local and inter-branch transactions even though the 2-processor EXT 10 has less than half the power of a single VLX processor. It should be noted that the debit/credit transaction is not a heavy duty test since it does not have much complex processing.

Duplexed or Mirrored Log File

The benchmark requires the log file to be duplexed. That is, that the system should be capable of losing one of the log files and still be able to recover or abort transactions.

This capability was demonstrated in the following manner. SQLCI was used to start a transaction and update an account. The power was turned off for one of the log disks. The transaction was aborted. The account balance was queried and shown to be restored to the balance prior to the start of the transaction.

Additional checks performed

Several additional checks were performed to insure the validity of the benchmark.

Database Locking

To prove that record locking was actually occurring, the following test was performed. SQLCI was used to start a transaction. An account balance was updated and queried to demonstrate the change. A second transaction was started on another terminal. The same account was queried. The balance was not returned to the second transaction which was timed out after waiting a decent interval. If there were no locking

mechanism, the balance would have been returned to the second transaction even though the first had not completed.

Transaction Message Size

The benchmark calls for an input message of 100 bytes and a 200 byte confirmation. This was verified in the following manner. The driver code was inspected to insure it was sending a 100 byte message. The code was also inspected to verify that it calculated the length of the response correctly. The auditor transaction submission code was inspected to verify the message length response was correct.

The length of the response was spot-checked during the submission of transactions during the test. Several additional transactions were submitted with invalid account numbers. These elicited error responses of a different length than the normal transactions.

Hardware configuration

Several checks were made to validate the hardware configuration.

The nodes on a Tandem system are connected with a fiber optic ring (FOX) when high transmission rates are desired. To verify that only the benchmark systems were on the ring, the tiles in the machine room were removed and the cabling traced.

All disks not directly needed for the benchmark were turned off.

All systems not used in the benchmark were inspected to insure no FOX connection.

The line speed setting between the measured system and the EXT 10 was verified at 9.6K bits/second.

Database sizing

The Tandem benchmark system was configured for a maximum transaction rate of 256 tps. This rate requires 25.6 million account records of 100 bytes each. In addition, the EXT 10 required 400,000 account records.

The above was verified by inspecting the data definition statements for the account, branch and teller tables. Additionally, accounts in each of the nodes were updated via transactions submitted during the timing portion. These accounts were then queried using SQLCI to verify the updates had taken place.

Inter-arrival rate

To verify the random inter-arrival rate, code was added to the reporting program to plot the theoretical and actual distribution. This code was inspected to insure correctness.

Re-verified all code each day

The timing runs were performed over several days. To insure the same software was used each day, the auditor kept a tape of the executable modules for the driver system. The contents of this tape were compared to the programs used for each run.

The code on the measured machine was inspected prior to each day's runs.

Verified use of generated scripts

After the script files were generated, one of them was picked and several of the transactions were altered to update balances not otherwise accessed. These transactions were scattered through the script to insure the entire script was used. After a timing run, the balances of these accounts were verified.

Conformance to Debit/Credit Benchmark

Deficiencies

Terminal Sizing

The benchmark calls for a configuration of ten terminals for every branch. Each terminal has a one hundred second think time. Tandem reduced the number of terminals by ten and reduced the think time by ten. This change retained the transaction rate but does

have the affect of reducing the amount of memory needed to support the terminal configuration.

Response Time Graph

The benchmark states that 95% of all transactions must complete in one second. Tandem uses two seconds for 90% of all transactions.

History Database

The history database did not have physical disks allocated to cover the ninety day requirement. This is a minor matter since this database is updated at the end. The priced configuration should have the correct amount of disk included.

Extensions

Mirrored Disks

All data is written to its primary location and its mirrored location. This doubles the number of disk I/Os for each transaction.

Message Inter-Arrival Rate

The benchmark does not specify the pattern of time between messages. The easiest is a constant time. For this benchmark that would be a rate of one message every ten seconds per terminal. Tandem used an inter-arrival rate that was exponential but had a mean of ten seconds per terminal. This rate is a worst case assumption.

Response Time Measurement

The benchmark definition of response time starts when the last bit of the transaction is received in the measured machine and ends when the first bit is sent out on the transmission line. It must include all time spent in the measured machine.

Tandem measured the response time from the time the message was accepted by the X.25 component on the driver side of the communication line until the last byte of the response had been processed by the X.25 component on the driver machine. Thus all message processing and queueing that occurred on the driver machine and all the line time was added to the response time.

Implementation Language

Tandem implemented all application dependent code in COBOL 85. The database processing was implemented in SQL rather than a machine specific access mechanism.

Account Balance Testing

The benchmark makes no requirement to test the account balance for overdrafts. Tandem included an overdraft protection check in the transaction. This made the transaction a bit more real life.

Conformance to ANSI Standard SQL

Three exceptions were noted.

The WHERE clause that specifies which teller or which branch is updated references a field called "syskey". This field is not a data value in the row; it is the relative record number - in effect a direct reference. This usage removes the access transparency normally gained by SQL. That is, the program is dependent on the underlying implementation. The same effect could be obtained by enhancing the CREATE TABLE statement to include the information that the branch_number is equal to syskey.

A second exception to the ANSI standard is the use of COBOL picture clauses in the column definitions under CREATE TABLE.

All table names in the application SQL statements are preceeded with equal signs ("="). This is a Tandem idiosyncrasy that concerns the mapping of table name to data file. Unfortunately, it makes the programs non-portable.

The Create Table statement contains one extension - the information needed to partition the table.

A BENCHMARK OF TANDEM'S NonStop SQL

DEMONSTRATING OVER 200 TRANSACTIONS PER SECOND

Anon Et Al

INTRODUCTION

NonStop SQL is Tandem's implementation of the ANSI SQL relational database language. In contrast to other SQL implementations, NonStop SQL is designed for on-line transaction processing as well as for information-center use. It delivers high performance at good price performance. In addition it supports distributed data with local autonomy.

THE PURPOSE OF THE BENCHMARKS

Part NonStop SQL's performance assurance compared the throughput of the Encompass system and the new SQL system. The benchmarks were done jointly by Software Development in Cupertino, the High Performance Research Center in Frankfurt, and the Benchmark Center in Sunnyvale.

Parts of the benchmark were audited and certified by the Codd and Date Consulting Group.

The benchmarks demonstrated the following aspects of NonStop SQL:

* FUNCTIONAL: NonStop SQL is functional and has been stress tested for high-volume OLTP applications.

* DISTRIBUTED: NonStop SQL allows distributed data and distributed transactions.

* SCALEABLE: NonStop SQL runs on small departmental systems as well as on large mainframe systems.

* LINEARITY: NonStop SQL demonstrates linear increases in throughput when the discs and processesors are added using the Dynabus or the FOX fiber optic ring.

* NO PERFORMANCE LIMIT: There is no apparent limit to the transaction throughput of NonStop SQL systems. In particular there are no bottle-necks in systems running hundreds of transactions per second.

* NO PERFORMANCE PENALTY: NonStop SQL performs as well as the record-at-a-time Encompass system on OLTP applications.

* GOOD PRICE PERFORMANCE: Both Encompass and NonStop SQL have impressive price performance at both low and high transaction volumes.

* SQL COSTS NO MORE: The price performance of Tandem systems is competitive for both departmental systems (EXT-10) and data center systems (VLX).

THE DEFINITION OF THROUGHPUT AND PRICE PERFORMANCE

The benchmarks were based on the DebitCredit OLTP transaction defined in Datamation "A Measure of Transaction Processing Power", [Anon]. That article defines a standard database, a standard terminal network and a way to scale them to larger systems. It defines a standard transaction, called the DebitCredit transaction and specifies how to measure the throughput and price/performance of the resulting system.

Briefly, the database consists of four SQL tables:

ACCOUNT: A file of bank accounts, each record is 100 bytes and holds the account number and balance among other things.

TELLER: A file of tellers, each record is 100 bytes and holds the teller number and cash position among other things.

BRANCH: A file of branches, each record is 100 bytes and holds the branch number and cash position of all tellers at the branch.

HISTORY: An entry sequence file containing a list of all transactions that have run. Each record is 50 bytes long and holds the account, teller, branch, delta, and timestamp.

The transaction coded in SQL is:

```
READ 100 BYTES FROM TERMINAL;
BEGIN WORK;
PERFORM PRESENTATION SERVICES GIVING account_number,
                                     teller_number,
                                     branch_number,
                                     delta;
UPDATE ACCOUNT
       SET  balance = balance + :delta
       WHERE  account_number = :account_number;
UPDATE TELLER
       SET  balance = balance + :delta
       WHERE teller_number = :teller_number;
UPDATE BRANCH
       SET  balance = balance + :delta
       WHERE branch_number = :branch_number;
INSERT INTO HISTORY VALUES
   (timestamp,account_number,teller_number,branch_number,delta);
```

```
PERFORM PRESENTATION SERVICES;
COMMIT WORK;
WRITE 200 BYTES TO TERMINAL;
```

The system is presented with various transaction rates and the response time is measured over ten-minute windows. This gives rise to a response time curve.

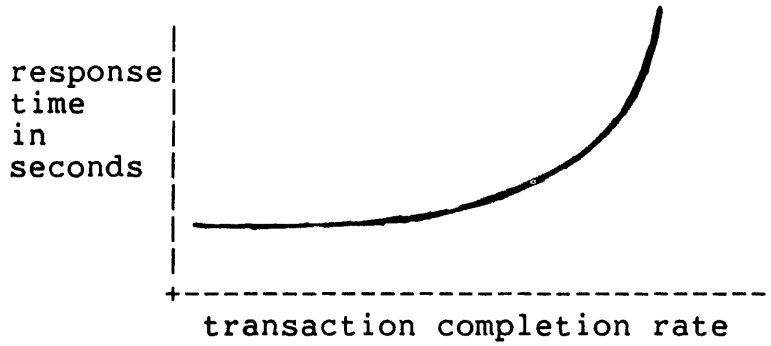


Figure 1. A typical response time curve showing how response time grows as the transaction load on the system increases.

A system that can run one such transaction per second giving less than one second response time to 95% of the transactions is called a 1TPS (transaction per second) system. For a 1TPS system the database is defined to have the following sizes:

```
   100,000 Accounts
       100 Tellers
        10 Branches
 2,590,000 History      ! space for a 90 day history assuming average
                        ! rate is 1/3 of peak rate of 1TPS and 24hrs
                        ! per day, seven days a week.
```

In addition, the standard specifies that a 1TPS system has 100 tellers at 10 branches. Each teller thinks for an average of 100 seconds and then submitts a transaction. The tellers use block mode terminals with ten fields of input and output. The input message is 100 bytes and the output message is 200 bytes. Messages are transmitted via the

X.25 communication protocol.

Systems which run more than 1TPS have the database sizes and network scaled linearly. For example a 100TPS system has a database and teller network one hundred times larger.

The standard specifies that accounts belong to branches and tellers belong to branches. If the database is distributed, then 15% of the transactions arrive at branches different from the account's home branch. These 15% are uniformly distributed among the other branches.

The benchmark requires that the transactions be run with UNDO/REDO transaction protection (abort, autorestart, and rollforward recovery). In addition it specifies that the transaction log must be duplexed.

DEPARTURES FROM THE STANDARD DEBIT CREDIT TRANSACTION

The NonStop SQL benchmark departed from the Datamation standard definition in the following ways:

1. Terminals were driven in record mode (Intelligent Device Support) rather than block mode. In effect this assumes that presentation services are done in the terminal.

2. It was assumed that each branch had a concentrator which multiplexed the teller terminals at the branch. This had the effect of reducing the number of sessions by a factor of 10 for the same transaction rate when compared to the standard.

3. TPS was measured as the throughput when 90% of the transactions get less than 2 second response time. The standard measures TPS at 1 second for 95% of the transactions.

4. The SQL system had the disc process check that debits did not cause account balances to become negative.

5. The system was measured over ten minute periods and the average for each period was used to compute the response time curves and consequent TPS ratings.

6. Response times were measured within the driver system. The standard specfies that response can be measured at the interface

to the service system.

7. All devices were driven by NonStop processes so that no single failure would cause a denial of service.

8. All discs were mirrored (duplexed) as is standard in Tandem.

9. Standard products were used. All applications were written in Cobol85.

Items 1, 2, and 3 have the effect of giving an optimistic TPS rating. Items 4, 5, 6, 7, 8, and 9 tend to reduce the system's TPS rating.

THE BENCHMARK SYSTEM DESIGN

The benchmark hardware consisted of 32 VLX processors. Each VLX procesor had a 5Mb/sec channel, 8Mb of main memory and is rated at about three Tandem mips. In addition, an EXT-10 system was included in the benchmark hardware to demonstrate scalability. The EXT-10 is Tandem's smallest system. It consists of two processors each with 4Mb of main memory and two discs. The EXT-10 processors together are appoximately as powerful as half a VLX cpu. Thus, the complex was sized as a 32.5 VLX processor system.

The VLX processors were decomposed into four nodes of eight VLX cpus each. Each node connects its eight processors via dual 20Mb/sec Dynabus, and the nodes are interconnected via a four fiber optic rings.

To model a distributed node, the EXT-10 was connected to VLXs via a 9.6Kb communication line.

The Guardian90XF system makes the entire 32VLX + EXT10 configuration appear to be a single system with location transparency.

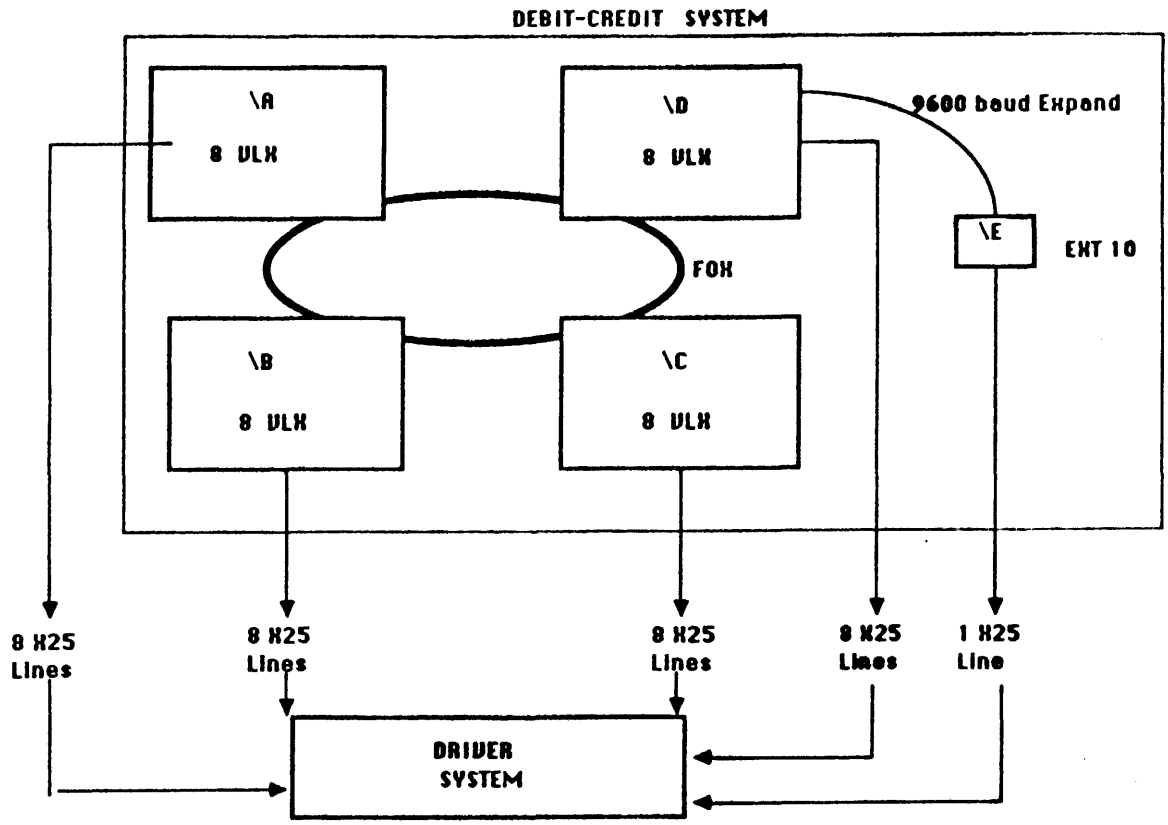


Figure 2. The benchmark configuration. The two processor EXT-10 is connected to one VLX via a 9.6KB line. The 32VXL are decomposed into four nodes of eight processors each. The nodes are connected by the high speed FOX ring. The whole complex looks like a single system to the application programmer. A separate driver system submitts transactions to this system via X.25 lines.

The database and transaction load were partitioned into 33 parts. Based on preliminary tests, each VLX processor could process up to eight transactions per second and the EXT-10 could process four trasactions per second. All sizings based on these preliminary measurements.

Each transaction input message is 100 bytes and the reply is 200 bytes. With X.25 overheads this translates to 340 bytes in all. At 8TPS, this is 22kbits/sec. A 56Kb line can handle this load quite comfortably. So each VLX processor was configured with a 56Kb line and the EXT-10 was configured with a single 56Kb line.

The database was sized as follows. Each eight VLX node had:

* A mirrored disc to store the programs and the transaction log (audit trail).
* A mirrored disc dedicated to the History table. In the benchmark only one disc was configured, but in pricing the system the 90-day history file of each node was sized at 6.7GB (14 mirrored volumes).

According to the standard, 8TPS implies:

80 branches	8Kb
800 tellers	80Kb
800,000 accounts	80Mb

These records along with their indicies and some slack fit comfortably on Tandem's smallest disc -- so each VLX processor was given a mirrored disc volume to hold its part of the Account-Branch-Teller (ABT) tables. A 1.6MB disc cache per mirrored disc partition was sufficient to keep all branches, tellers and the account index pages resident in main memory.

Overall then, the system configuration is:

NETWORK		DATABASE RECORDS	SIZE
25,600 Tellers		25,600 Tellers	2.6Mb
2560 branches	---------->	2,560 Branches	.3Mb
		25,600,000 Accounts	2.6Gb
		54,080,000,000 History	27.1Gb

The teller, branch and Account files were mirrored, but for economic reasons, the History file was not mirrored

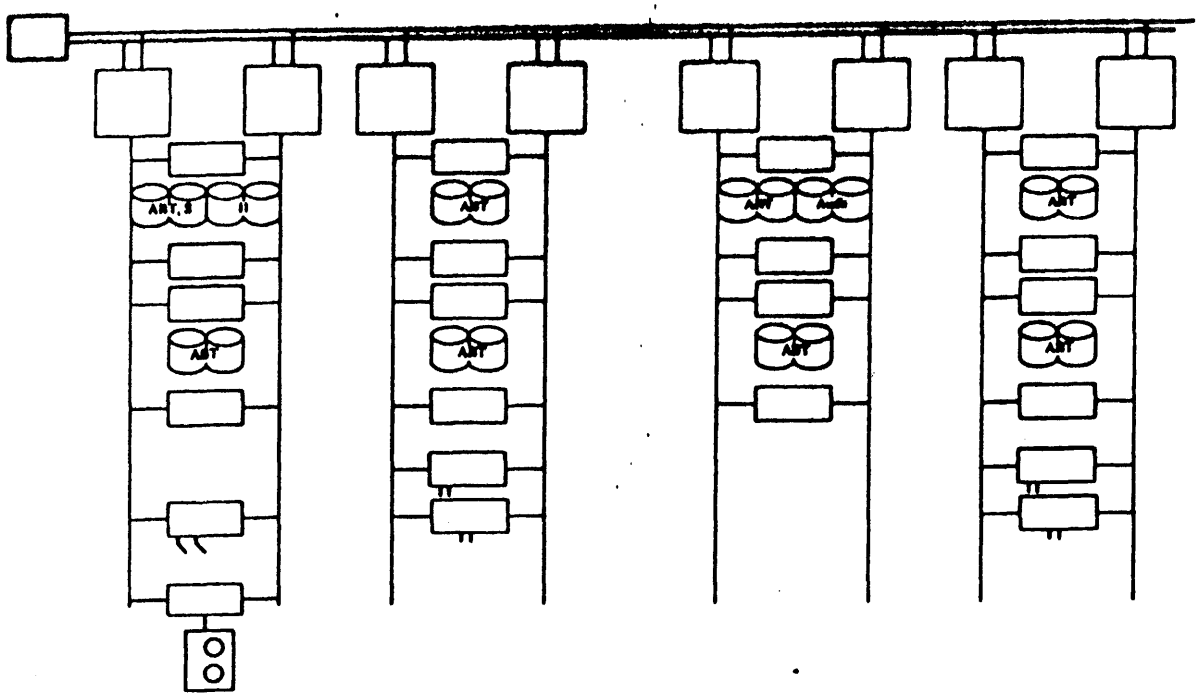


Figure 3. The configuration of each 8-processor VLX node in the FOX ring.

The SQL tables were defined to be partitioned by the appropriate key values. The approximate syntax for creating the ACCOUNT table is:

```
CREATE TABLE account (number   PIC 9(12),
                      balance  PIC S9(11)V(2),
                      ...
                      )
                      KEY number
                      PARTITION $vlx2 START KEY    800000
                      PARTITION $vlx3 START KEY   1600000
                      ...
                      PARTITION $vlx32 START KEY 24800000
                      PARTITION $ext   START KEY 25600000;
```

This creates a single table with 33 partitions. NonStop SQL hides this partitioning from the application programmer. The branch and teller files were partitioned in a similar way.

NonStop SQL imposes no practical limit on the number of partitions a table may have. Enscribe is limited to a maximum of 16 partitions. So the Enscribe database and Enscribe benchmark was limited to 16 VLX processors.

In the Tandem system, terminal control and presentaton services are done by a process called the Terminal Control Program (TCP). It is the logical equivalent of IMS/DC or CICS. Since each cpu was sized at 8TPS, each CPU controlled 800 tellers at 80 branches. It was decided to configure 32 branches (320 tellers) per TCP which gave rise to about 2.5 TCPs per VLX cpu.

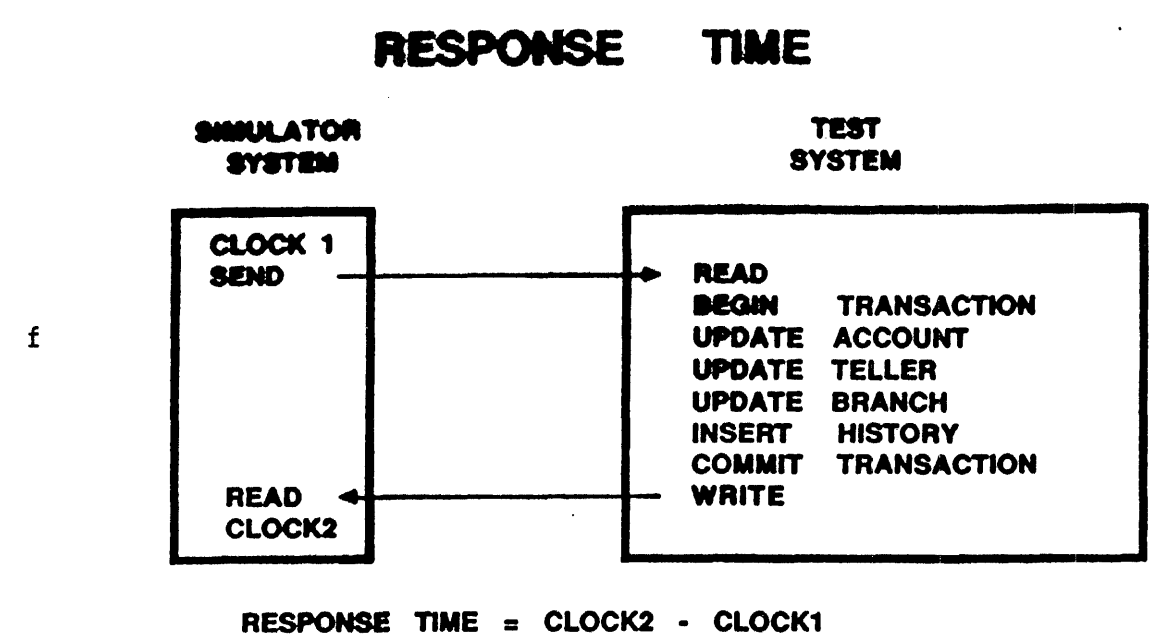
To avoid queueing on database servers at 8TPS with a 2 second average response time, 20 Debit/Credit servers were configured per cpu (20 ~ 2*8) The code and data for the TCPs came to about .5MB/VLX cpu. The Enscribe servers consumed .25MB/VLX. The NonStop SQL servers used about 10Kb more memory -- twenty NonStop SQL servers used about .5MB/VLX in all.

The EXT-10 system was sized at half of a VLX. The programs and audit trail were on one mirrored disc and the database resided on a second mirrored disc.

A second complex of 12 Tandem TXP processors simulated the network of 2560 branches (25,600 tellers) by submitting transactions via X.25 using modem eliminators to connect the X.25 lines. Typically, transactions were submitted to each VLX line at an average rate

between 4TPS and 8TPS with expenentially distributed interarrival times. The EXT-10 was driven at about 4TPS. Each transaction message named an Account, Branch, Teller and Debit amount. As specified by the standard, in 85% of the cases, the account and branch were local, in 15% of the cases the account was in a branch different from the branch and teller of this transaction. Some of these "non-local" transactions were in branches at the same node, but most went to other nodes of the network. For example, when the system was running at over 200 TPS the EXT-10 originated or received about 1 distributed transaction per second while each VLX node originated or received about 15 distributed transactions per second.

The driver system measured response time as the time between the send and the completion of the receive in the driver system.



f

Figure 4. The driver system submitts messages via X.25 and measures the transaction completion rate and response time.

In summary, the hardware configuration was:

DRIVER SYSTEM	LINES	PROCESSORS	DISCS
24mips	33*56Kb X.25	32VLX+EXT10	86 volumes
		100 mips	20Gb
		250Mbytes	+27GB history
			(when pricing)

Assembling this hardware was no small matter. Aside from the cost (about 8M$), the VLX has a backlog of orders. The benchmark was

allowed to use the equipment for a limited time. The hardware had to be assembled, benchmarked, and disassembled all in 50 days. Fortunately, the equipment was installed on schedule, there were no hardware errors during the benchmark, and no critical software errors were discovered in the SQL product itself. The only hardware problem was a double power failure early in the benchmark.

THE EXPERIMENTS

The benchmarks measured the ability to grow a Tandem VLX system from eight VLX processors to thirty two processors using the FOX fiber optic ring.

In addition, they demonstrate that NonStop SQL runs on Tandem's low-end EXT-10 system attached to the VLXs via an 9.6kb line. The EXT-10 had a proportionate part of the database and sent and received distributed transactions (15%).

First the throughput of single 8VLX system was measured for both Enscribe and SQL. Then a pair of 8VLX systems were connected via FOX and their performance measured for both SQL and Enscribe. Finally the 4-node 32-processor FOX connected network with attached EXT-10 was measured. The last experiment was only done for SQL because Enscribe is limited to 16 partitions per file.

Figure 5 shows the response time curves for the SQL and Enscribe tests on a eight processor VLX system. The curves show that for the same response time, SQL gives sligtly better throughput and conversely that SQL gives better response time for a fixed throughput.

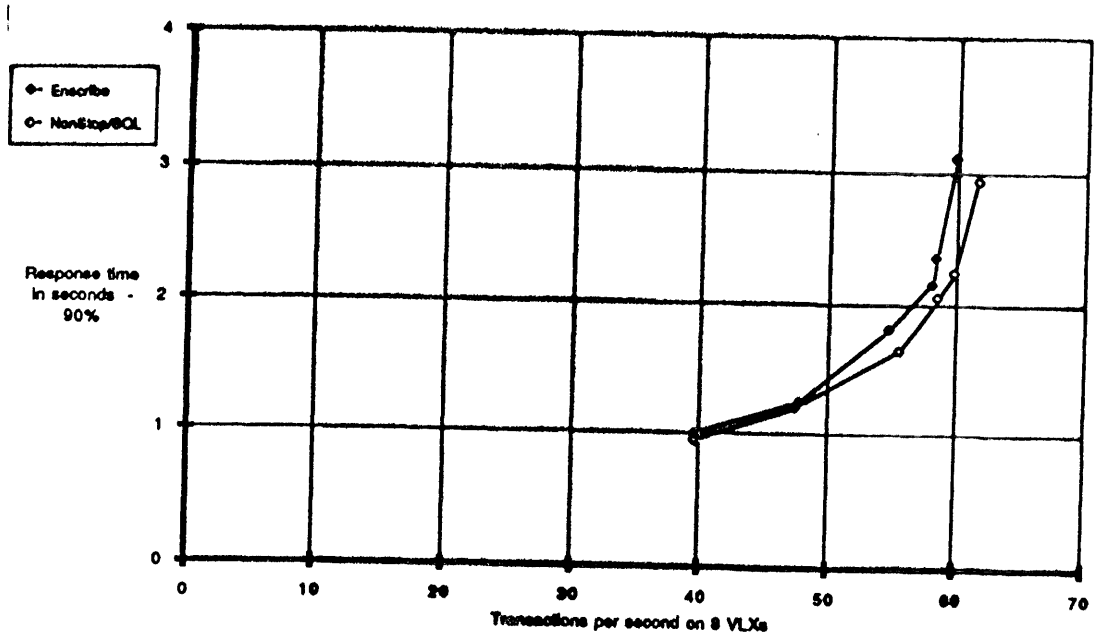


Figure 5. The response time curves for NonStop SQL and Tandem's record-at-a-time Enscribe system running on an 8 processor VLX system. Notice that NonStop SQL has slightly better performance than Enscribe.

The each transaction generated one physical read of the account file and two pyhsical writes of the account file (1 mirrored write). Disc reads and writes of the account, branch and history files are amortized over many transactions. In addition, each transaction contributes about .4 physical log IO per tansaction because group commits batch about five transactions per audit flush. A detailed breakdown of the cpu utilization by function is shown in figure 6. In that figure "INNT" signifies interrupt handing and the message system. "ATB" stands for the Account, Branch, and Teller disc servers. Both the primary ATB and backup (process pair) disc processes are separated. The "TCP" does presentation services and terminal handling, "X.25" is does the physical line handling for the input and

output messages. The disc server storing the History table is shown separately. Lastly, the transaction commit coordinator is represented by "TMF-MON".

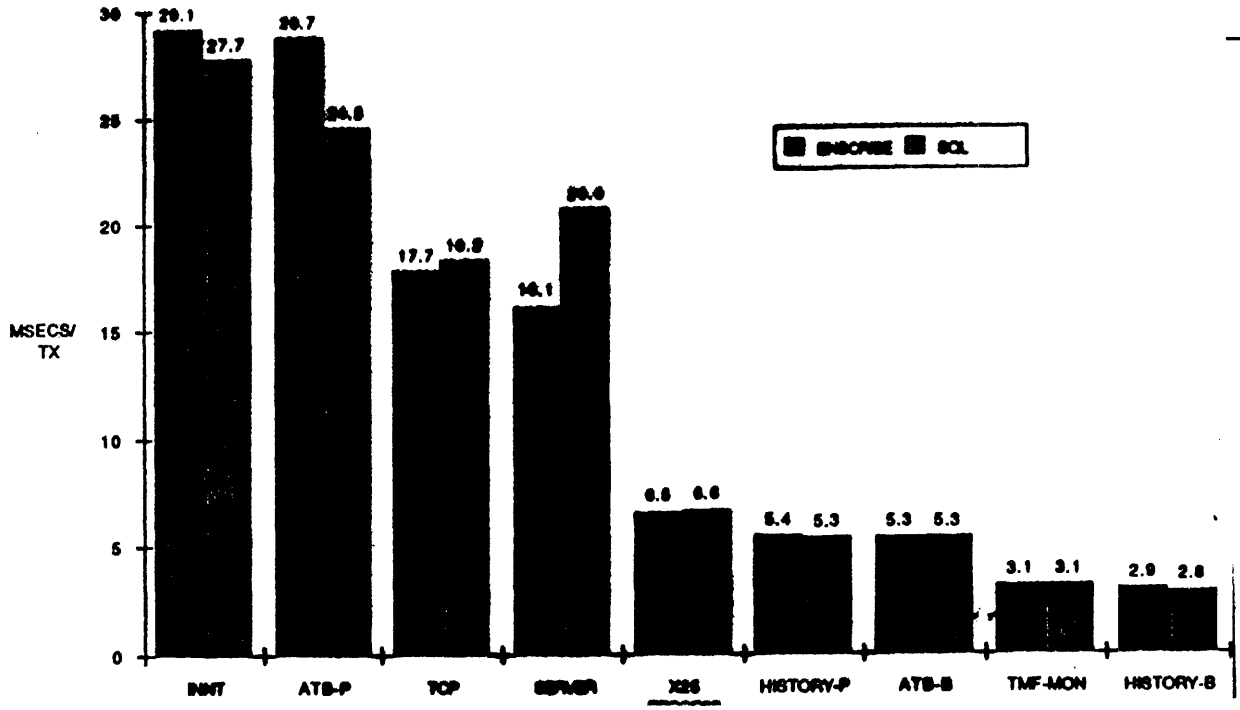


Figure 6. The detailed breakdown of cpu time spent by each component of the DebitCredit transaction on an 8 processor VLX system. Both the Enscribe and SQL costs are shown.

These experiments were repeated for 16 VLX processors and then the SQL experiments were repeated for 32 VLX processors (recall that Enscribe is limited to 16 partitions). The response times were plotted and are documented in [Sawyer]. The resulting throughput curves are shown in Figure 7.

These experiments were audited by Codd and Date. As explained in the Auditor's report [Sawyer], the auditor verified that:

* The code correctly implemented the standard transaction.
* The database was sized correctly
* 15% of the transactions were non-local
* Transactions were protected by a dual UNDO/REDO log
* The response times were measured correctly
* The measured response time curves matched the system
* SQL and Enscribe had comparable performance

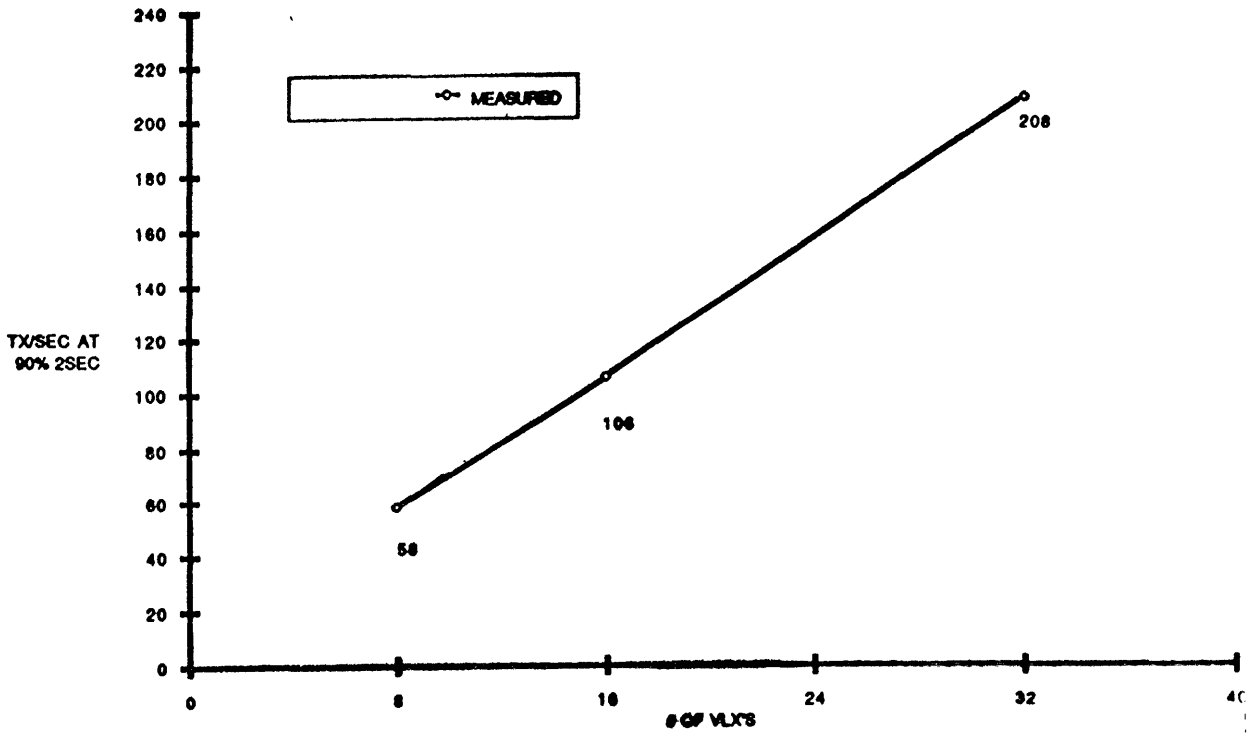


Figure 7. The TPS rating of VLX processors for NonStop SQL. At peak the system was running at a sustained rate of 208 SQL TPS.

Based on early measurements, the system was expected to perform about 7TPS per cpu and have linear growth from 8 to 32 cpus. That meant the 32 processor system would do about 256TPS. In fact the 8VLX system did 7.2TPS/VLX cpu and a there was a 10% dropoff as the system was scaled to 32 processors. This is shown in the figure 7. The dropoff is due to the increased cost of network distributed transactions. When transactions do work at multiple nodes, they cost extra instructions and IO traffic. This extra cost implies a reduction in throughput overall. Dispite this, the TPS curves are linear -- it is just that they are .9x rather than 1x.

In addition, NonStop SQL uses slightly fewer cpu instructions than does Enscribe. Consequently, NonStop SQL has slightly higher transaction throughput than Enscribe. Ignoring response time, the peak NonStop SQL throughput was 229 TPS.

Given this linear throughput vs hardware, it is possible to quote the price performance of the system in terms of the price performance of a single processor. To a first approximation, the five-year cost/transaction is the same for a small and a large system. The dominant issue is which processor line the user selects. Generally newer systems are less expensive. Table 1 shows the cost per transaction of the various Tandem processors. Two costs are quoted in table 1: the cost of a fully mirrored disc configuration (the typical Tandem configuration) and the cost of a un-mirrored disc configuration (1/2 as many discs).

	COST/TPS	
PROCESSOR	MIRRORED DISCS	UNMIRRORED DISCS
VLX	59K$	55K$
TXP	85K$	81K$
CLX	??	??
EXT-25	??	??
EXT-10	??	??

Table 1. The cost per transaction of various Tandem Processors.

WHY IS NonStop SQL SO FAST AND POWERFUL?

Historically, SQL has been confined to information center environments and to low-end systems where programmer productivity is more important than system performance. In these environmnts, the power of the SQL language and the relational model compensated for the lackluster performance of most relational systems.

NonStop SQL is the first SQL system to offer the high performance necessary for online transaction processing. As demonstrated by the benchmarks, it has performance comparable to Tandem's record-at-a-time Enscribe system. We believe both NonStop SQL and Enscribe have the best price performance of any full-function data management system. The cost per transaction is about 53K$ and the systems are capable of over 200TPS. In addition, NonStop SQL is a distributed relational system; whereas other systems do not offer full-function distributed data or distrubuted execution.

How did Tandem succeed where so many others failed? First, NonStop SQL benefited from the experience of its predecessors. The developers had collectively worked on System R, SQL/DS, DB2, R*, IDM, Encompass, Esvel, Wang VS, and several other systems. For some, this was their sixth, and hopefully last, SQL implementation. They did not repeat any of the mistakes of our earlier efforts.

A second benefit was the close cooperation between the database group and the operating system and languages group. Tandem is a transaction

processing company. So the operating system is geared for efficient processing of distributed transactions. NonStop SQL exploits the transaction mechanism of the Guardian90XF operating system to easily and efficiently get transaction protected distrubuted execution based on remote procedure calls.

Perhaps the most significant advantage of NonStop SQL is the SQL language itself. One can see the difference when contrasting the record-at-a-time interface of Enscribe, DL/1, or DBTG, with the set-oriented interface of SQL. Consider the UPDATE statements in the DebitCredit transaction. Ingnoring error handling, these statements are coded as:

COBOL- ENSCRIBE	SQL
START branch KEY IS EQUAL TO branch-number	
READ branch RECORD with LOCK	UPDATE branch
ADD debit TO balance OF branch	SET balance = balance + :debit
REWRITE branch	WHERE number = :branch-number
	AND balance >= debit;

In the Enscribe case, a message is sent to the appropriate part (disc process) of the database to get the designated branch record. The record is then returned to the program which examines it, alters it and then returns it to the database. This is two messages and a lot of data movement.

In the NonStop SQL case, a single message is sent to the data (disc process) requesting an update of the appropriate account by the appropriate amount. The disc process applies this update and returns a status message to the caller. In this case, NonStop SQL sends half as many messages and one quarter the number of message bytes.

In general, the NonStop SQL subcontracts single variable queries to remote servers. This allows the NonStop SQL disc process to act as a database machine which performs updates, deletes and filters data, returning only the desired rows and columns of a table.

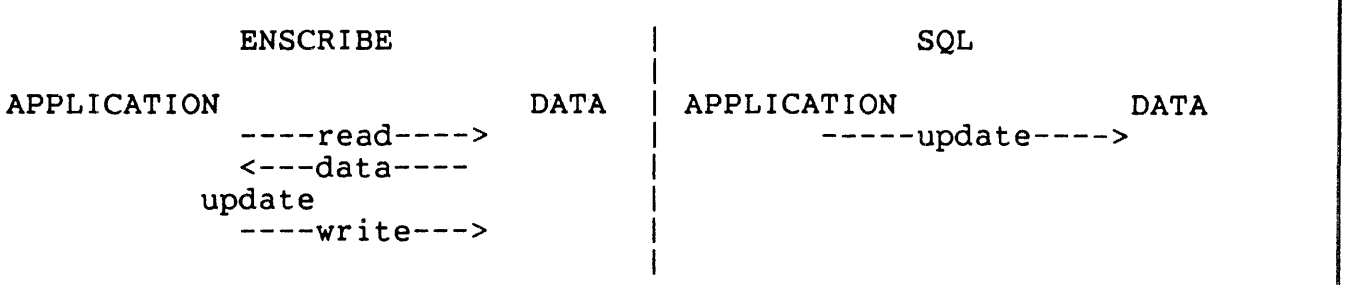


Figure 8. SQL can update or filter data at the source whereas record-at-a-time interfaces must fetch the data to the application program. In this way, SQL saves messages and cpu cycles over a conventional data management system.

This example hints at the synergy between SQL and distributed database systems. It has long been argued that SQL would be a good basis for distributed database systems. In all the benchmarks we have done, the message savings of SQL have compensated for the extra work the system must perform in order to implement the more complex semantics of the SQL language. For example, NonStop SQL is about two times faster than Enform on the "Wisconsin Benchmark" [Bitton]. Virtually all this speedup is due to the close integration of NonStop SQL with the operating system.

Even with all these nice features, a system might bottleneck on certain resources. For example, most transaction processing systems

bottleneck at 30TPS because they have not implemented group commit [Gawlick]. The fact that the NonStop SQL was benchmarked at 16 processors on a single dynabus and at over 200TPS without any bottlenecks suggests that there are no obvious bottlenecks. In fact we are unaware of any bottlenecks in the system. This is no accident, Tandem has spent the last few years eliminating bottlenecks.

SUMMARY

Running over 230 TPS on a 32VLX processor complex and an EXT-10 demonstes that NonStop SQL is

* FUNCTIONAL
* DISTRIBUTED
* SCALEABLE
* HAS LINEAR GROWTH
* HAS NO PERFORMANCE LIMIT
* HAS NO PERFORMACE PENANLTY FOR SQL
* HAS GOOD PRICE PERFORMANCE
* HAS FLAT PRICE PERFORMANCE FROM THE LOW END TO THE HIGH END

REFERENCES

[Anon] Anon, Et. Al., "A Measure of Transaction Processing Power", Datamation, April 1985.

[Sawyer] Sawyer, T., "Auditors Report on the TopGun Benchmark", Codd and Date Consulting Group, San Jose, CA. March 1987. Also available from Tandem Information Center, Cupertino CA.

[Gawlick] Gawlick, D. "Processing Hot Spots in High Performance Systems", Proceedings of IEEE COMPCON 85', 1985.

22 January 1987

Tom Sawyer
Codd and Date Consulting Group
6489 Camden Av #109
San Jose, CA. 95120

Frank Clugage
Tandem Computers Inc.
19333 Vallco Parkway
Cupertino, CA 95014

Dear Frank:

This letter reviews my plans to support Tandem's SQL announcement.

* I will audit and validate the results of the performance benchmark of SQL and Enscribe on a 32VXL+EXT10 configuration. This benchmark will run the standard Debit/Credit transaction as documented in "A Measure of Transaction Processing Power" [Datamation, April 1984] with exceptions.
* I will write a report with my conclusions on the performance of the benchmark and my confidence in the numbers.
* I will also assist C. J. Date in preparing a ten minute videotape on Tandem's SQL system.

In preparation for the audit I have reviewed the application (server) programs, the performance papers, the UFI Reference Manual, the recent Tandem Systems Review on performance, and the proposal for the benchmark prepared by Jim Enright on Jan. 15 1987.

So far, everything looks fine with one exception. I notice that the server programs always credit the account with 100$. The amount should be in the input message and should be used by the servers.

These are the aspects of the benchmark I expect to audit:

1. That the programs correctly implement the DebitCredit transaction.
2. That the database design is correct and that its size is correct for the transaction rate.
3. That the database is being updated by the transactions and that the inter-branch transaction rate is 15% of the total rate.
4. That the transaction rate and response times are as claimed by the monitoring software for both the VLX and EXT systems.
5. That the system is in steady-state.
6. That the transaction logs are duplexed and that they support transaction backout.

Each of these points must be established for both Enscribe and SQL.

In addition, you may want to demonstrate Tandem's ability to tolerate the following kinds of faults:

A. Processor failure
B. Disc controller failure
C. Disc failure
D. Log disc failure
E. Communications controller failure
F. Dynabus failure
G. Fox link failure.

In order to audit the benchmark I will need the following:

To establish points 1 and 2 above (the programs correctly implement the DebitCredit transaction, that the database design is correct, and that its size is correct for the transaction rate) I will need:

A. A formal description of the experiments to be performed. This should include the following:
   * A list of the exceptions to the standard for the benchmark.
   * Listings of the DRIVER software along with its manual.
   * Listing of the REQUESTOR software.
   * Listings of the SERVER software (SQL and Enscribe)
   * Listing of the Database Dictionary (SQL and Enscribe)
   * Listing of the physical database design (SQL and Enscribe)
   * Associated language manuals (Tal/Cobol).
B. Ability during the benchmark to verify that these listings correspond to the system being benchmarked. Examining the processes using the SOURCE command of INSPECT will prove that the programs are the same. FUP and UFI can be used to verify the file sizes and layouts.

To establish point 3 above (that the database is being updated by the transactions and that the inter-branch transaction rate is 15% of the total rate) I will need:

A. Ability to set the Account balances to zero for some key range.
B. Local updates will be X dollars (probably 1$), and global updates will be Y dollars (probably 1,000,000$). I will tell you X and Y at the time of the benchmark.
C. Ability with ENFORM and UFI to see account balances.
D. Ability to submit transactions to the system from the driver system given the account, branch, teller and delta. The program should be interactive and measure the response time. Enform/UFI will then be used to see that the data changed.

To establish point 4 above (that the transaction rate and response times are as claimed by the driver software for both the VLX and EXT) I will use the interactive program mentioned in point 3.D above.

I expect Tandem to present driver reports for two different ten minute intervals to prove that the system response time is stable. This will establish point 5.

To establish point 6, I will fail one of the log discs and then use the UFI or the interactive program to run and then abort a transaction. A special program will be required for Enscribe. UFI and ENFORM will be used to show that the transaction was indeed undone.

Note: This letter was typed from draft form by Jim Gray who took some editorial license with the draft.

A MEASURE OF TRANSACTION PROCESSING POWER

Anon Et Al

February 1985

Tandem Technical Report 85.2

Tandem TR 85.2

A Measure of Transaction Processing Power

Anon Et Al

February 1985

ABSTRACT

Three benchmarks are defined: Sort, Scan and DebitCredit. The first two benchmarks measure a system's input/output performance. DebitCredit is a simple transaction processing application used to define a throughput measure -- Transactions Per Second (TPS). These benchmarks measure the performance of diverse transaction processing systems. A standard system cost measure is stated and used to define price/performance metrics.

A condensed version of this paper appears in Datamation, April 1, 1985

TABLE OF CONTENTS

Who Needs Performance Metrics?

A measure of transaction processing power is needed -- a standard which can measure and compare the throughput and price/performance of various transaction processing systems.

Vendors of transaction processing systems quote Transaction Per Second (TPS) rates for their systems. But there isn't a standard transaction, so it is difficult to verify or compare these TPS claims. In addition, there is no accepted way to price a system supporting a desired TPS rate. This makes it impossible to compare the price/performance of different systems.

The performance of a transaction processing system depends heavily on the system input/output architecture, data communications architecture and even more importantly on the efficiency of the system software. Traditional computer performance metrics, Whetstones, MIPS, MegaFLOPS and GigaLIPS, focus on CPU speed. These measures do not capture the features that make one transaction processing system faster or cheaper than another.

This paper is an attempt by two dozen people active in transaction processing to write down the folklore we use to measure system performance. The authors include academics, vendors, and users. A condensation of this paper appears in Datamation (April 1, 1985).

We rate a transaction processing system's performance a price/performance by:

* Performance is quantified by measuring the elapsed time for t standard batch transactions and throughput for an interacti transaction.

* Price is quantified as the five-year capital cost of the syst equipment exclusive of communications lines, terminals, developme and operations.

* Price/Performance is the ratio Price over Performance.

These measures also gauge the peak performance and performance tren of a system as new hardware and software is introduced. This is valuable aid to system pricing, sales and purchase.

We rate a transaction processing system by its performance on thr generic operations:

* A simple interactive transaction.

* A minibatch transaction which updates a small batch of records.

* A utility which does bulk data movement.

This simplistic position is similar to Gibson's observation that if you can load and store quickly, you have a fast machine [Gibson].

We believe this simple benchmark is adequate because:

* The interactive transaction forms the basis for the TPS rating. It is also a litmus test for transaction processing systems -- it requires the system have at least minimal presentation services, transaction recovery, and data management.

* The minibatch transaction tells the IO performance available to the Cobol programmer. It tells us how fast the end-user IO software is.

* The utility program is included to show what a really tricky programmer can squeeze out of the system. It tells us how fast the real IO architecture is. On most systems, the utilities trick the IO software into giving the raw IO device performance with almost no software overhead.

In other words, we believe these three benchmarks indicate the performance of a transaction processing system because the utility benchmark gauges the IO hardware, the minibatch benchmark gauges the IO software and the interactive transaction gauges the performance of the online transaction processing system.

The particular programs chosen here have become part of the folklore of computing. Increasingly, they are being used to compare system

performance from release to release and in some cases, to compare the price/performance of different vendor's transaction processing systems.

The basic benchmarks are:

DebitCredit: A banking transaction interacts with a block-mode terminal connected via X.25. The system does presentation services to map the input for a Cobol program which in turn uses a database system to debit a bank account, do the standard double entry book-keeping and then reply to the terminal. 95% of the transactions must provide one second response time. Relevant measures are throughput and cost.

Scan: A minibatch Cobol transaction sequentially scans and updates one thousand records. A duplexed transaction log is automatically maintained for transaction recovery. Relevant measures are elapsed time, and cost.

Sort: A disc sort of one million records. The source and target files are sequential. Relevant measures are elapsed time, and cost.

A word of caution: these are performance metrics, not function metrics. They make minimal demands on the network (only x.25 and very minimal presentation services), transaction processing (no distributed data), data management (no complex data structures), and recovery

management (no duplexed or distributed data).

Most of us have spent our careers making high-function systems. It is painful to see a metric which rewards simplicity -- simple systems are faster than fancy ones. We really wish this were a function benchmark. It isn't.

Surprisingly, these minimal requirements disqualify many purported transaction processing systems, but there is a very wide spectrum of function and useability among the systems that have these minimal functions.

Our Performance and Price Metrics

What is meant by the terms: elapsed time, cost and throughput? Before getting into any discussion of these issues, you must get the right attitude. These measures are very rough. As the Environmental Protection Agency says about its milage ratings, "Your actual performance may vary depending on driving habits, road conditions and queue lengths -- use them for comparison purposes only". This cavalier attitude is required for the rest of this paper and for performance metrics in general -- if you don't believe this, reconsider EPA milage ratings for cars.

So, what is meant by the terms: elapsed time, cost and throughput?

ELAPSED TIME

Elapsed Time is the wall-clock time required to do the operation on an otherwise empty system. It is a very crude performance measure but it is both intuitive and indicative. It gives an optimistic performance measure. In a real system, things never go that fast, but someone got it to go that fast once.

COST

Cost is a much more complex measure. Anyone involved with an accounting system appreciates this. What should be included? Should

it include the cost of communications lines, terminals, application development, personnel, facilities, maintenance, etc.? Ideally, cost would capture the entire "cost-of-ownership". It is very hard to measure cost-of-ownership. We take a myopic vendor's view: cost is the 5-year capital cost of vendor supplied hardware and software in the machine room. It does not include terminal cost, communications costs, application development costs or operations costs. It does include hardware and software purchase, installation and maintenance charges.

This cost measure is typically one fifth of the total cost-of-ownership. We take this narrow view of cost because it is simple. One can count the hardware boxes and software packages. Each has a price in the price book. Computing this cost is a matter of inventory and arithmetic.

A benchmark is charged for the resources it uses rather than the entire system cost. For example, if the benchmark runs for an hour, we charge it for an hour. This in turn requires a way to measure system cost/hour rather than just system cost. Rather than get into discussions of the cost of money, we normalize the discussion by ignoring interest and imagine that the system is straight-line depreciated over 5 years. Hence an hour costs about 2E-5 of the five year cost and a second costs about 5E-9 of the five year cost.

Utilization is another tough issue. Who pays for overhead? The answer we adopt is a simple one: the benchmark is charged for all operating

system activity. Similarly, the disc is charged for all disc activity, either direct (e.g. application input/output) or indirect (e.g. paging).

To make this specific, lets compute the cost of a sort benchmark which runs for an hour, uses 2 megabytes of memory and two discs and their controllers.

Package	Package cost	Per hour cost	Benchmark cost
Processor	80K$	1.8$	1.8$
Memory	15K$	.3$	.3$
Disc	50K$	1.1$	1.1$
Software	50K$	1.1$	1.1$
			4.3$

So the cost is 4.3$ per sort.

The people who run the benchmark are free to configure it for minimum cost or minimum time. They may pick a fast processor, add or drop memory, channels or other accelerators. In general the minimum-elapsed-time system is not the minimum-cost system. For example, the minimum cost Tandem system for Sort is a one processor two disc system. Sort takes about 30 minutes at a cost of 1.5$. On the other hand, we believe a 16 processor two disc Tandem system with 8Mbytes per processor could do Sort within ten minutes for about 15$ -- six times faster and 10 times as expensive. In the IBM world, minimum cost generally comes with model 4300 processors, minimum time

generally comes with 308x processors.

The macho performance measure is throughput -- how much work the system can do per second. MIPS, GigaLIPS and MegaFLOPS are all throughput measures. For transaction processing, transactions per second (TPS) is the throughput measure.

A standard definition of the unit transaction is required to make the TPS metric concrete. We use the DebitCredit transaction as such a unit transaction.

To normalize the TPS measure, most of the transactions must have less than a specified response time. To eliminate the issue of communication line speed and delay, response time is defined as the time interval between the arrival of the last bit from the communications line and the sending of the first bit to the communications line. This is the metric used by most teleprocessing stress testers.

Hence the Transactions Per Second (TPS) unit is defined as:

TPS: Peak DebitCredit transactions per second with 95% of the transactions having less than one second response time.

Having defined the terms: elapsed time, cost and throughput, we can now define the various benchmarks.

The Sort Benchmark

The sort benchmark measures the performance possible with the best programmers using all the mean tricks in the system. It is an excellent test of the input-output architecture of a computer and its operating system.

The definition of the sort benchmark is simple. The input is one-million hundred-byte records stored in a sequential disc file. The first ten bytes of each record are the key. The keys of the input file are in random order. The sort program creates an output file and fills it with the input file sorted in key order. The sort may use as many scratch discs and as much memory as it likes.

Implementors of sort care about seeks, disc io, compares, and such. Users only care how long it takes and how much it costs. From the user's viewpoint, relevant metrics are:

Elapsed time: the time from the start to the end of the sort program.

Cost: the time weighted cost of the sort software, the software and hardware packages it uses.

In theory, a fast machine with 100mb memory could do the job in a minute at a cost of 20$. In practice, elapsed times range from 10 minutes to 10 hours and costs between 1$ and 100$. A one hour 10$ sort is typical of good commercial systems.

The Scan Benchmark

The Sort benchmark indicates what sequential performance a wizard can get out of the system. The Scan benchmark indicates the comparable performance available to end-users: Cobol programmers. The difference is frequently a factor of five or ten.

The Scan benchmark is based on a Cobol program which sequentially scans a sequential file, reading and updating each record. Such scans are typical of end-of-day processing in online transaction processing systems. The total scan is broken into minibatch transactions each of which scans one thousand records. Each minibatch transaction is a Scan transaction.

The input is a sequential file of 100 byte records stored on one disc. Because the data is online, Scan cannot get exclusive access to the file and cannot use old-master new-master recovery techniques. Scan must use fine granularity locking so that concurrent access to other parts of the file is possible while Scan is running. Updates to the file must be protected by a system maintained duplexed log which can be used to reconstruct the file in case of failure.

Scan must be written in Cobol, PLI or some other end-user application interface. It must use the standard IO library of the system and otherwise behave as a good citizen with portable and maintainable code. Scan cannot use features not directly supported by the language.

The transaction flow is:

```
     OPEN file SHARED, RECORD LOCKING
     PERFORM SCAN 1000 TIMES
                 BEGIN -- Start of Scan Transaction
                 BEGIN-TRANSACTION
                 PERFORM 1000 TIMES
                   READ file NEXT RECORD record WITH LOCK
                   REWRITE record
                 COMMIT-TRANSACTION
                 END  -- End of Scan Transaction
    CLOSE FILE
```

The relevant measures of Scan are:

Elapsed time: The average time between successive BeginTransaction steps. If the data is buffered in main memory, the flush to disc must be included.

Cost: the time weighted system cost of Scan.

In theory, a fast machine with a conventional disc and flawless software could do Scan in .1 second. In practice elapsed times range from 1 second to 100 seconds while costs range from .001$ to .1$. Commercial systems execute scan for a penny with ten second elapsed time.

The DebitCredit Benchmark

The Sort and Scan benchmarks have the virtue of simplicity. They can be ported to a system in a few hours if it has a reasonable software base -- a sort utility, Cobol compiler and a transactional file system. Without this base, there is not much sense considering the system for transaction processing.

The DebitCredit transaction is a more difficult benchmark to describe or port -- it can take a day or several months to install depending on the available tools. On the other hand, it is the simplest application we can imagine.

A little history explains how DebitCredit became a de facto standard. In 1973 a large retail bank wanted to put its 1,000 branches, 10,000 tellers and 10,000,000 accounts online. They wanted to run a peak load of 100 transactions per second against the system. They also wanted high availability (central system availability of 99.5%) with two data centers.

The bank got two bids, one for 5M$ from a minicomputer vendor and another for 25M$ from a major-computer vendor. The mini solution was picked and built [Good]. It had a 50K$/TPS cost whereas the other system had a 250K$/TPS cost. This event crystalized the concept of cost/TPS. A generalization (and elaboration) of the bread-and-butter transaction to support those 10,000 tellers has come to be variously known as the TP1, ET1 or DebitCredit transaction [Gray].

In order to make the transaction definition portable and explicit, we define some extra details, namely the communication protocol (x.25) and presentation services.

The DebitCredit application has a database consisting of four record types. History records are 50 bytes, others are 100 bytes.

```
  * 1,000 branches       (   .1 Mb,   random access )
  * 10,000 tellers       (   1  Mb     random access)
  * 10,000,000 accounts  (   1  Gb     random access)
  * a 90 day history     (  10  Gb     sequential   ).
```

The transaction has the flow:

```
DebitCredit:
  BEGIN-TRANSACTION
  READ    MESSAGE FROM TERMINAL (100 bytes)
  REWRITE ACCOUNT   (random)
  WRITE   HISTORY   (sequential)
  REWRITE TELLER    (random)
  REWRITE BRANCH    (random)
  WRITE   MESSAGE TO   TERMINAL (200 bytes)
  COMMIT-TRANSACTION
```

A few more things need to be said about the transaction. Branch keys are generated randomly. Then a teller within the branch is picked at random. Then a random account at the branch is picked 85% of the time and a random account at a different branch is picked 15% of the time. Account keys are 10 bytes, the other keys can be short. All data files must be protected by fine granularity locking and logging. The log file for transaction recovery must be duplexed to tolerate single failures, data files need not be duplexed. 95% of the transactions must give at least one second response time. Message handling should deal with a block-mode terminal (eg IBM 3270) with a base screen of 20 fields. Ten of these fields are read, mapped by presentation services

and then remapped and written as part of the reply. The line protocol is x.25.

The benchmark scales as follows. Tellers have 100 second think times on average. So at 10TPS, store only a tenth of the database. At 1TPS store one hundredth of the database. At one teller, store only one ten thousandth of the database and run .01 TPS.

Typical costs for DebitCredit appear below. These numbers come from real systems, hence the anomaly that the lean-and-mean system does too many disc ios. Identifying these systems makes an interesting parlor game.

	K-inst	IO	TPS	K$/TPS	¢/T	Packets
Lean and Mean	20	6	400	40	.02	2
Fast	50	4	100	60	.03	2
Good	100	10	50	80	.04	2
Common	300	20	15	150	.75	4
Funny	1000	20	1	400	2.0	8

The units in the table are:

K-inst: The number of thousands of instructions to run the transaction. You might think that adding 10$ to your bank account is a single instruction (add). Not so, one system needs a million instructions to do that add. Instructions are expressed in 370 instructions or their equivalent and are

fuzzy numbers for non-370 systems.

DiscIO: The number of disc io required to run the transaction. The fast system does two database IO and two log writes.

TPS: Maximum Transactions Per Second you can run before the largest system saturates (response time exceeds one second). This is a throughput measure. The good system peaks at 50 transactions per second.

K$/TPS: Cost per transaction per second. This is just system cost divided by TPS. It is a simple measure to compute. The funny system costs 400K$ per transaction per second. That is, it costs 400K$ over 5 years and can barely run one transaction per second with one second response time. The cost/transaction for these systems is .5E-8 times the K$/TPS.

¢/T: Cost per transaction (measured in pennies per transaction). This may be computed by multiplying the system $/TPS by 5E-9.

Packets: The number of X.25 packets exchanged per transaction. This charges for network traffic. A good system will send two X.25 packets per transaction. A bad one will send four times that many. This translates into larger demands for communications bandwidth, longer response times at the terminals and much higher costs. X.25 was chosen both because it is a standard and because it allows one to count packets.

Observations On The DebitCredit Benchmark

The numbers in the table on page 15 are ones achieved by vendors benchmarking their own systems. Strangely, customers rarely achieve these numbers -- typical customers report three to five times these costs and small fractions of the TPS rating. We suspect this is because vendor benchmarks are perfectly tuned while customers focus more on getting it to work at all and dealing with constant change and growth. If this explanation is correct, real systems are seriously out of tune and automatic system tuning will reap enormous cost savings.

The relatively small variation in costs is surprising -- the TPS range is 400 but the K$/TPS range is 10. In part the narrow cost range stems from the small systems being priced on the minicomputer curve and hence being much cheaper than the mainframe systems. Another factor is that disc capacity and access are a major part of the system cost. The disc storage scales with TPS and disc accesses only vary by a factor of 5. Perhaps the real determinant is that few people will pay 400 times more for one system over a competing system.

There are definite economies of scale in transaction processing -- high performance systems have very good price/performance.

It is also surprising to note that a personal computer with appropriate hardware and data management software supports one teller, scales to .01 TPS, and costs 8K$ -- about 800K$/TPS! Yes, that's an

unfair comparison. Performance comparisons are unfair.

There are many pitfalls for the data management system running DebitCredit. These pitfalls are typical of other applications. For example, the branch database is a high-traffic small database, the end of the history file is a hotspot, the log may grow rapidly at 100TPS unless it is compressed, the account file is large but it must be spread across many discs because of the high disc traffic to it, and so on. Most data management systems bottleneck on software performance bugs long before hardware limits are reached [Gawlick], [Gray, et al].

The system must be able to run the periodic reporting -- sort merge the history file with the other account activity to produce 1/20 of the monthly statements. This can be done as a collection of background batch jobs that run after the end-of-day processing and must complete before the next end-of-day. This accounts for the interest in the scan and sort benchmarks.

Criticism

Twenty four people wrote this paper. Each feels it fails to capture the performance bugs in his system. Each knows that systems have already evolved to make some of the assumptions irrelevant (e.g. intelligent terminals now do distributed presentation services). But these benchmarks have been with us for a long time and provide a static yardstick for our systems.

There is particular concern that we ignore the performance of system startup (after a crash or installation of new software), and transaction startup (the first time it is called). These are serious performance bugs in some systems. A system should restart in a minute, and should NEVER lose a 10,000 terminal network because restart would be unacceptably long. With the advent of the 64kbit memory chip (not to mention the 1mbit memory chip), program loading should be instantaneous.

The second major concern is that this is a performance benchmark. Most of us have spent our careers making high-function systems. It is painful to see a metric which rewards simplicity -- simple systems are faster than fancy ones. We really wish this were a function benchmark. It isn't.

In focusing on DebitCredit, we have ignored system features which pay off in more complex applications: e.g. clustering of detail records on the same page with the master record, sophisticated use of alternate

access paths, support for distributed data and distributed execution, and so on. Each of these features has major performance benefits. However, benchmarks to demonstrate them are too complex to be portable.

Lastly, we have grave reservations about our cost model.

First, our "cost" ignores communications costs and terminal costs. An ATM costs 50K$ over 5 years, the machine room hardware to support it costs 5K$. The communications costs are somewhere in between. Typically, the machine room cost is 10% of the system cost. But we can find no reasonable way to capture this "other 90%" of the cost. In defense of our cost metric, the other costs are fixed, while the central system cost does vary by an order of magnitude

Second, our "cost" ignores the cost of development and maintenance. One can implement the DebitCredit transaction in a day or two on some systems. On others it takes months to get started. There are huge differences in productivity between different systems. Implementing these benchmarks is a good test of a system's productivity tools. We have brought it up (from scratch) in a week, complete with test database and scripts for the network driver. We estimate the leanest-meanest system would require six months of expert time to get DebitCredit operational. What's more, it has no Sort utility or transaction logging.

Third, our "cost" ignores the cost of outages. People comprise 60% of most DP budgets. People costs do not enter into our calculations at all. We can argue that a system with 10,000 active users and a 30 minute outage each week costs 100K$/TPS just in lost labor over five years. Needless to say, this calculation is very controversial.

In defense of our myopic cost model, it is the vendor's model and the customer's model when money changes hands. Systems are sold (or not sold) based on the vendor's bid which is our cost number.

Summary

Computer performance is difficult to quantify. Different measures are appropriate to different application areas. None of the benchmarks described here use any floating point operations or logical inferences. Hence MegaFLOPS and GigaLIPS are not helpful on these applications. Even the MIPS measure is a poor metric -- one software system may use ten times the resources of another on the same hardware.

Cpu power measures miss an important trend in computer architecture: the emergence of parallel processing systems built out of modest processors which deliver impressive performance by using a large number of them. Cost and throughput are the only reasonable metrics for such computer architectures.

In addition, input-output architecture largely dominates the performance of most applications. Conventional measures ignore input-output completely.

We defined three benchmarks, Sort, Scan and DebitCredit. The first two benchmarks are really measure the system's input/output performance. DebitCredit is a very simple transaction processing application.

Based on the definition of DebitCredit we defined the Transactions Per Second (TPS) measure:

TPS: Peak DebitCredit transactions per second with 95% of the transactions having less than one second response time.

TPS is a good metric because it measures software and hardware performance including input-output.

These three benchmarks combined allow performance and price/performance comparisons of systems.

In closing, we restate our cavalier attitude about all this: "Actual performance may vary depending on driving habits, road conditions and queue lengths -- use these numbers for comparison purposes only". Put more bluntly, there are lies, damn lies and then there are performance measures.

References

[Gibson] Gibson, J.C., "The Gibson Mix", IBM TR00.2043, June 1970.

[Gawlick] Gawlick, D., "Processing of Hot Spots in Database Systems", Proceedings of IEEE COMPCON, San Francisco, IEEE Press, Feb. 1985.

[Gray] Gray, J., "Notes on Database Operating Systems", pp. 395-396, In Lecture Notes in Computer Science, Vol. 60, Bayer-Seegmuller eds., Springer Verlag, 1978.

[Gray2] Gray, J., Gawlick, D., Good, J.R., Homan, P., Sammer, H.R. "One Thousand Transactions Per Second", Proceedings of IEEE COMPCON, San Francisco, IEEE Press, Feb. 1985. Also, Tandem TR 85.1.

[Good] Good, J. R., "Experience With a Large Distributed Banking System", IEEE Computer Society on Database Engineering, Vol. 6, No. 2, June 1983.

[Anon Et Al] Dina Bitton of Cornell, Mark Brown of DEC, Rick Catell of Sun, Stefano Ceri of Milan, Tim Chou of Tandem, Dave DeWitt of Wisconsin, Dieter Gawlick of Amdahl, Hector Garcia-Molina of Princton, Bob Good of BofA, Jim Gray of Tandem, Pete Homan of Tandem, Bob Jolls of Tandem, Tony Lukes of HP, Ed Lawoska of U. Washington, John Nauman of 3Com, Mike Pong of Tandem, Alfred Spector of CMU, Kent Trieber of IBM, Harald Sammer of Tandem, Omri Serlin of FT News, Mike Stonebraker of Berkeley, Andras Reuter of U. Kaiserslutern, Peter Weinberger of ATT.

Starting February 16, 1987

Verify sources with inspect.
Generate script with new delta values.
Alter specific account and delta values throughout a script
Fup load script
Down six unused Enscribe volumes
Poke around SQL database at will

```
Begin SQL sessions on 32.5 VLX configuration
    Reset balance for specified key ranges on VLX and EXT
    Begin high tx rate run
        Monitor response time on interactive terminal
            for VLX and EXT partitions
        End response timings
        Audit proof at interactive terminal
            Begin transaction
            Modify account on node \x
            Select account and verify change
            Down primary or backup audit trail on node \x
            Select account and verify change
            Abort transaction
            Select account and verify change backout
        Revive downed disc
        Repeat audit proof on EXT?
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin high tx rate run
        Monitor response time on interactive terminal
            for VLX and EXT partitions
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin lower tx rate run
        Monitor response time on interactive terminal
            for VLX and EXT partitions
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin lowest tx rate run
        Monitor response time on interactive terminal
            for VLX and EXT partitions
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin highest tx rate run
        Monitor response time on interactive terminal
            for VLX and EXT partitions
    End test
```

```
Begin SQL sessions on 16 VLX configuration
Generate script with the delta values.
Alter specific account and delta values throughout a script
Fup load script
Poke around SQL database at will
    Reset balance for specified key ranges on VLX and EXT
    Begin high tx rate run
        Monitor response time on interactive terminal
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin high tx rate run
        Monitor response time on interactive terminal
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin lower tx rate run
        Monitor response time on interactive terminal
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin lowest tx rate run
        Monitor response time on interactive terminal
    End test

    Reset balance for specified key ranges on VLX and EXT
    Begin highest tx rate run
        Monitor response time on interactive terminal
    End test
```

```
Begin Enscribe sessions on 16 VLX configuration
Up six Enscribe volumes / Down six SQL volumes
Generate script with the delta values.
Alter specific account and delta values throughout a script
Fup load script
Load Enscribe database
Poke around Enscribe database at will
    Begin high tx rate run
        Monitor response time on interactive terminal
    End test

    Reload a partition to reset balances for specific key ranges
    Verify with Enform
    Begin high tx rate run
        Monitor response time on interactive terminal
    End test

    Reload a partition to reset balances for specific key ranges
    Verify with Enform
    Begin lower tx rate run
        Monitor response time on interactive terminal
    End test

    Reload a partition to reset balances for specific key ranges
    Verify with Enform
    Begin lowest tx rate run
        Monitor response time on interactive terminal
    End test

    Reload a partition to reset balances for specific key ranges
    Verify with Enform
    Begin highest tx rate run
        Monitor response time on interactive terminal
    End test
```

THE TOPGUN BENCHMARKS

WHY THE BENCHMARKS?

- TO DEMONSTRATE THAT NonStop/SQL :
  - WORKS
  - DISTRIBUTED
  - NO PERFORMANCE PENALTY USING SQL
  - LINEAR GROWTH
  - SCALEABLE (EXT TO VLX)
  - GOOD PRICE / PERFORMANCE
  - NO LIMIT ON PERFORMANCE

THE DEBITCREDIT "STANDARD"

- THE TRANSACTION

```
READ 100 BYTES FROM TERMINAL ;
BEGIN TRANSACTION ;
    UPDATE account BY DELTA ;
    UPDATE teller  BY DELTA ;
    UPDATE branch  BY DELTA ;
    INSERT INTO history ;
COMMIT TRANSACTION ;
WRITE 200 BYTES TO TERMINAL ;
```

THE DEBITCREDIT "STANDARD"

- THE DATABASE FOR A 1 TPS SYSTEM

  - THE "ABT" FILES :

    100,000 - 100 BYTE ACCOUNT RECORDS

    10 - 100 BYTE BRANCH RECORDS

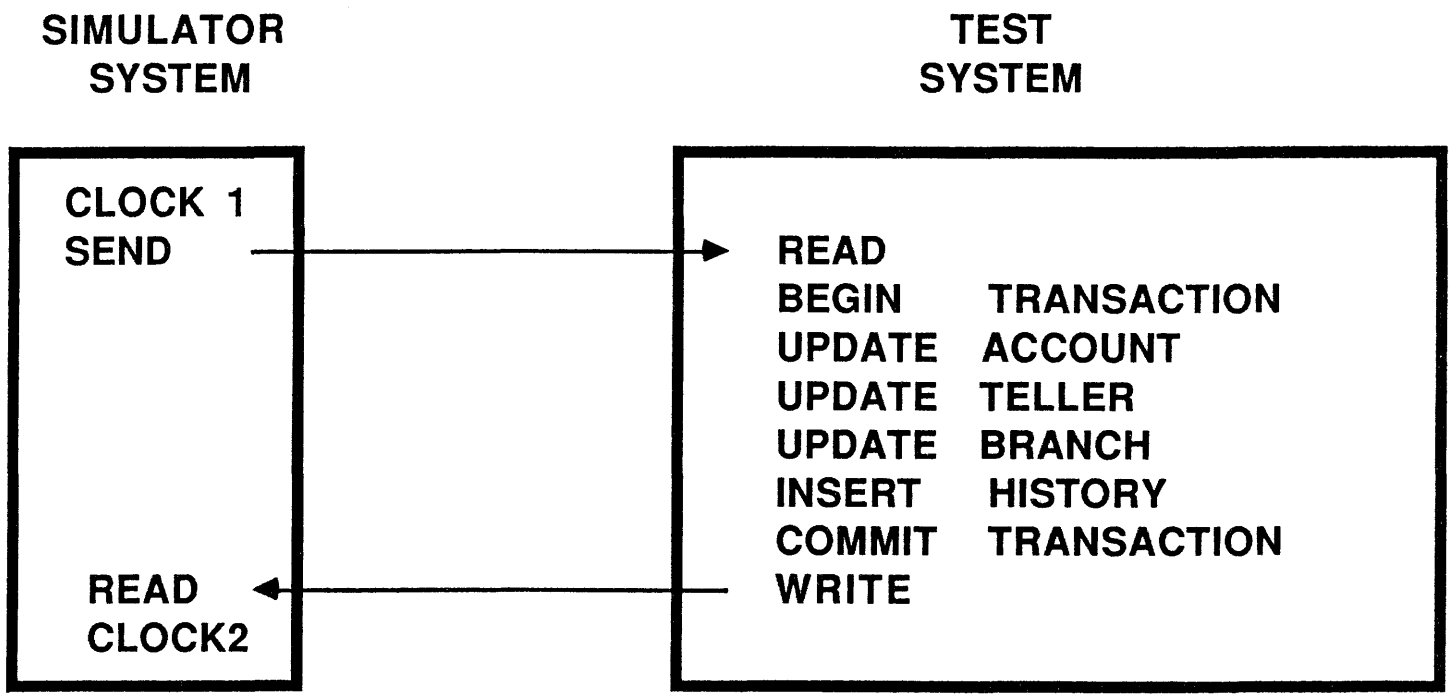
    100 - 100 BYTE TELLER RECORDS

    2,590,000 - 50 BYTE HISTORY RECORDS
    (90 days online history file, this is included in the "standard" to compare disk storage cost)

- THINK TIME - 100 seconds

  100 terminals / 100 seconds think time = 1 TPS

RESPONSE TIME



RESPONSE TIME = CLOCK2 - CLOCK1

NOTE :

THE STANDARD SPECIFIES THE RESPONSE TIME AS THE ELAPSED TIME BETWEEN THE LAST BIT ENTERING THE TEST SYSTEM TILL THE TIME THE FIRST BIT LEAVES THE SYSTEM.

THE DEBITCREDIT "STANDARD"

- OTHER DETAILS

  - THROUGHPUT OR TPS RATING

    TPS IS SPECIFIED AS THE TRANSACTION RATE AT WHICH 95% OF THE TRANSACTIONS HAVE A RESPONSE TIME OF LESS THAN OR EQUAL TO 1 SECOND.

  - DISTRIBUTED

    15 % OF TRANSACTIONS ARE INTER-BRANCH

  - COST

    MEASURED AS THE 5 YEAR COST OF HARDWARE AND SOFTWARE PURCHASE AND MAINTENANCE

  - THE BOTTOM LINE

    COST PER TPS

    e.g. A $ 250,000 system which can do 5 TPS has a cost per tps of $ 50,000. ($ 250,000 / 5 tps)

TANDEM'S TWO DATABASE SYSTEMS

ENSCRIBE	NonStop / SQL
RELATIONAL DDL	RELATIONAL DDL
RECORD-AT-A-TIME DML	RELATIONAL DML
DISTRIBUTED	DISTRIBUTED
TRANSACTION PROTECTED	TRANSACTION PROTECTED

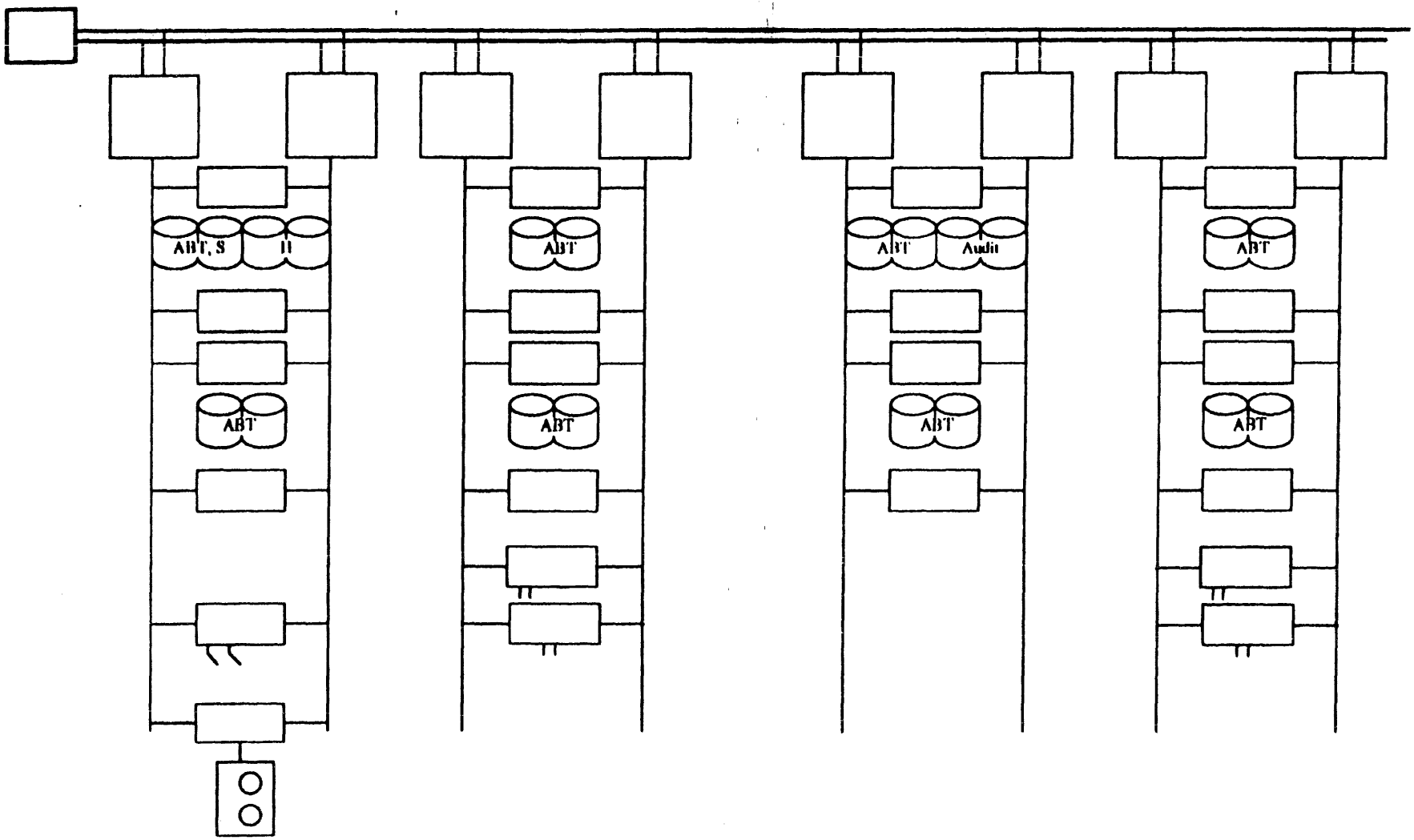
- **HARDWARE USED**

  - **FOR DEBIT-CREDIT SYSTEMS**
  - **FOR THE TERMINAL SIMULATORS**
  - **FOR COMMUNICATION**

• DEBIT-CREDIT SYSTEMS

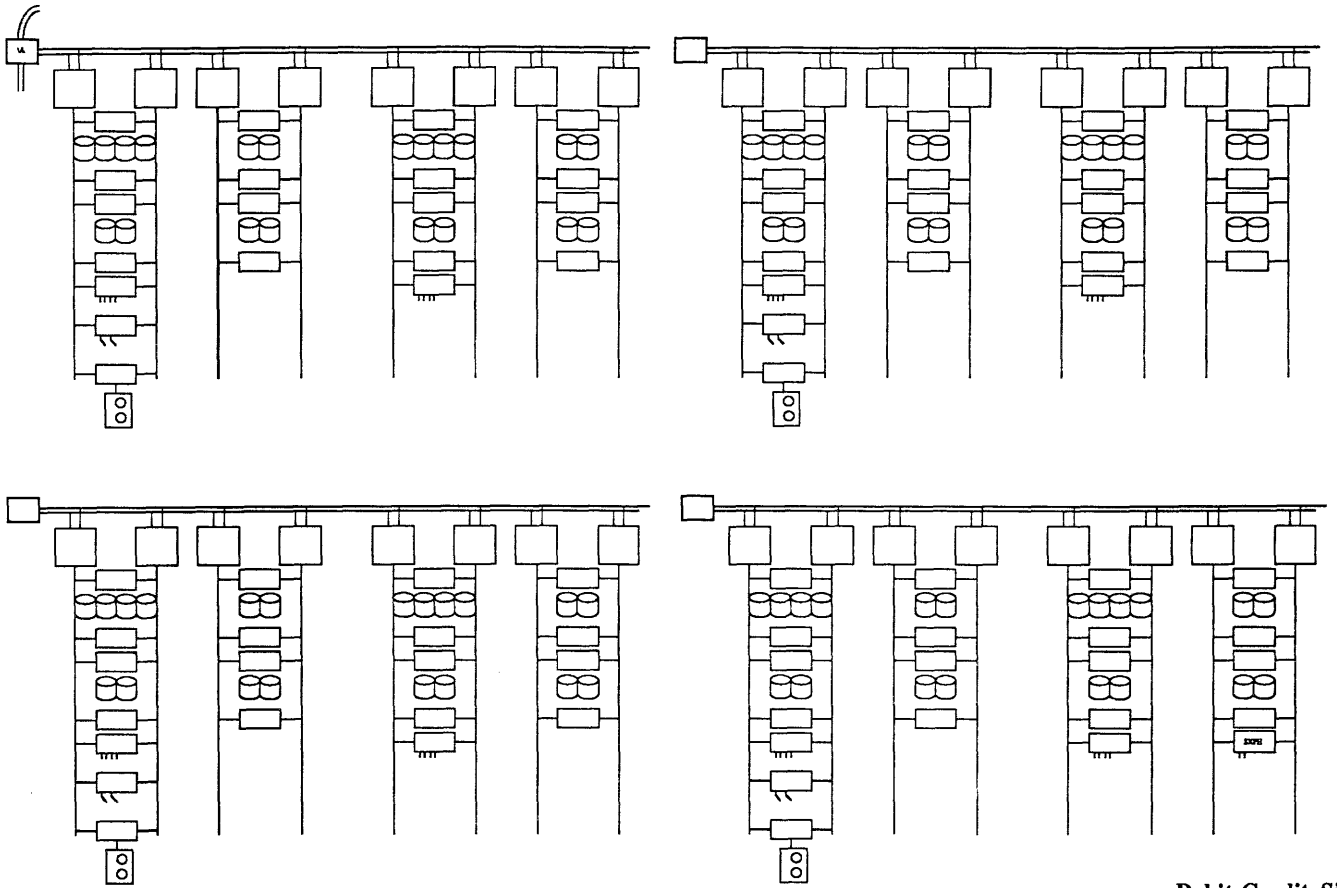
- 4 NODES CONNECTED VIA FOX
  - EACH NODE 8 VLX PROCESSORS
    - TOTAL OF 32 VLX PROCESSORS
  - 8 MBYTES MEMORY PER PROCESSOR
    - TOTAL OF 256 MBYTES MEMORY
  - 10 MIRRORED DISK VOLUMES PER NODE
    - TOTAL OF 40 MIRRORED DISK VOLUMES
- 1 EXT10 SYSTEM
  - 4 MBYTES MEMORY PER PROCESSOR
  - 2 MIRRORED DISK VOLUMES
  - CONNECTED VIA 9600 BAUD EXPAND LINE TO A VLX NODE

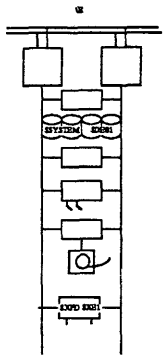
Example node: With Mirrored disk config.



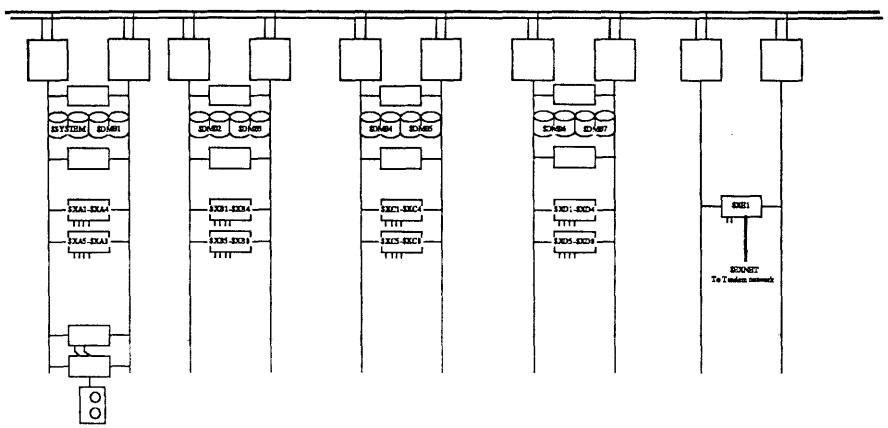
ABT Account Branch & Teller partitions
H History file (one per node)
S System disk

3 / 12/ 87 rev 6.1





Debit-Credit Side



Driver SIDE

- **FOR TERMINAL SIMULATORS**
  - **10 TXP PROCESSORS**
  - **4 MBYTES MEMORY PER PROCESSOR**
  - **8 MIRRORED DISK VOLUMES**

• COMMUNICATION HARDWARE

• 26 BITSYNC CONTROLLERS

- 4 BITSYNCS PER VLX NODE, 16 FOR 4 NODES
- 1 BITSYNC FOR THE EXT10 SYSTEM
- 9 BITSYNCS FOR THE SIMULATOR SYSTEM

• 33 X25 LINES

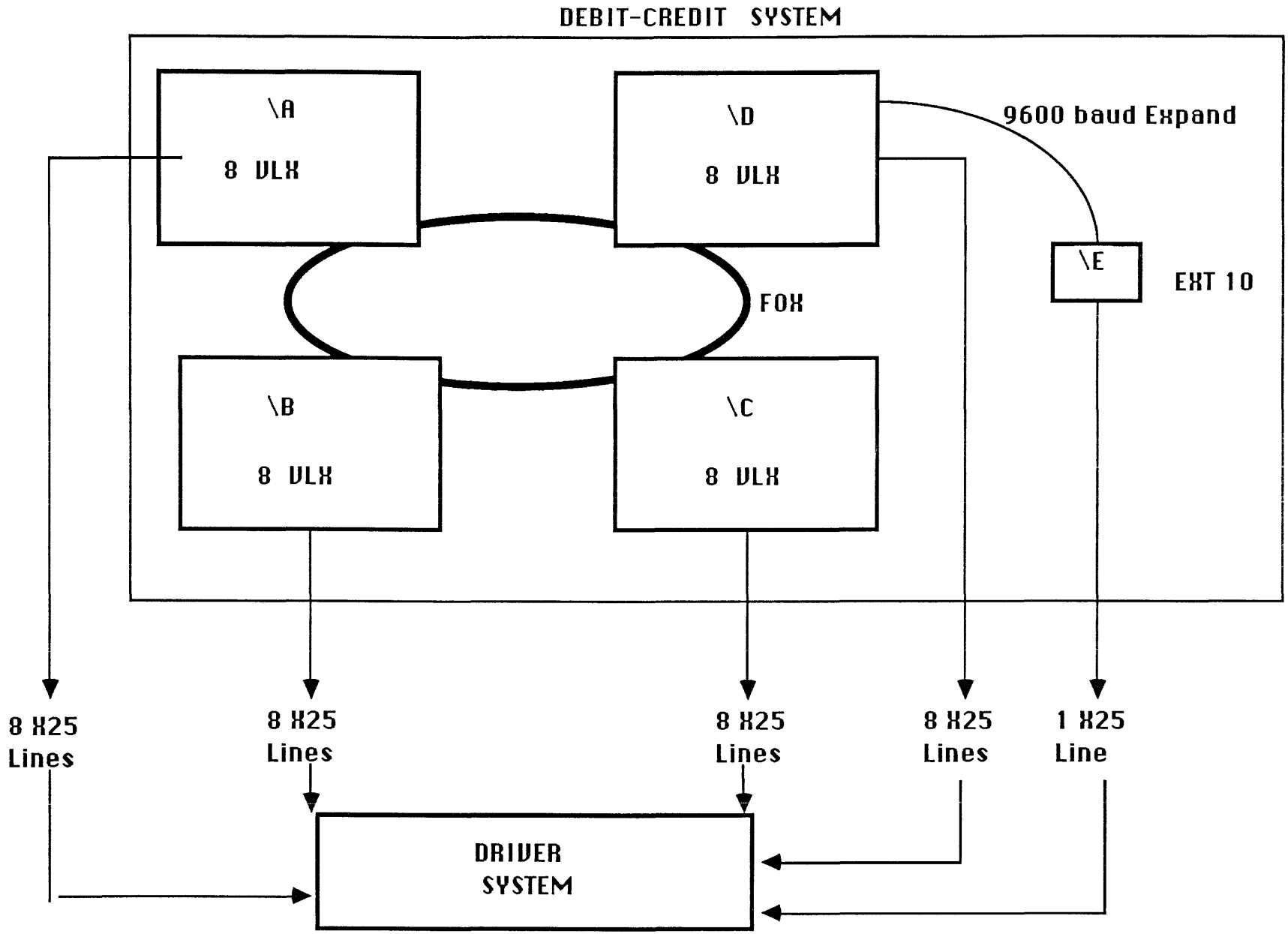
- 8 LINES PER VLX NODE, 32 FOR 4 NODES
- 1 LINE FOR THE EXT10 SYSTEM

• 33 MODEM ELIMINATORS (56 KB/SEC)

- FOR THE 33 X25 LINES

• 1 MODEM ELIMINATOR (9600 B/SEC)

- FOR THE EXPAND LINE CONNECTING THE EXT10


DEBIT-CREDIT SYSTEM
\A
8 VLX
\D
8 VLX
9600 baud Expand
\E
EXT 10
FOX
\B
8 VLX
\C
8 VLX
8 X25 Lines
8 X25 Lines
8 X25 Lines
8 X25 Lines
1 X25 Line
DRIVER SYSTEM

- DATABASE SIZE FOR WHOLE SYSTEM

  - ACCOUNT FILE
    - 33 PARTITIONS, 26 MILLION REC

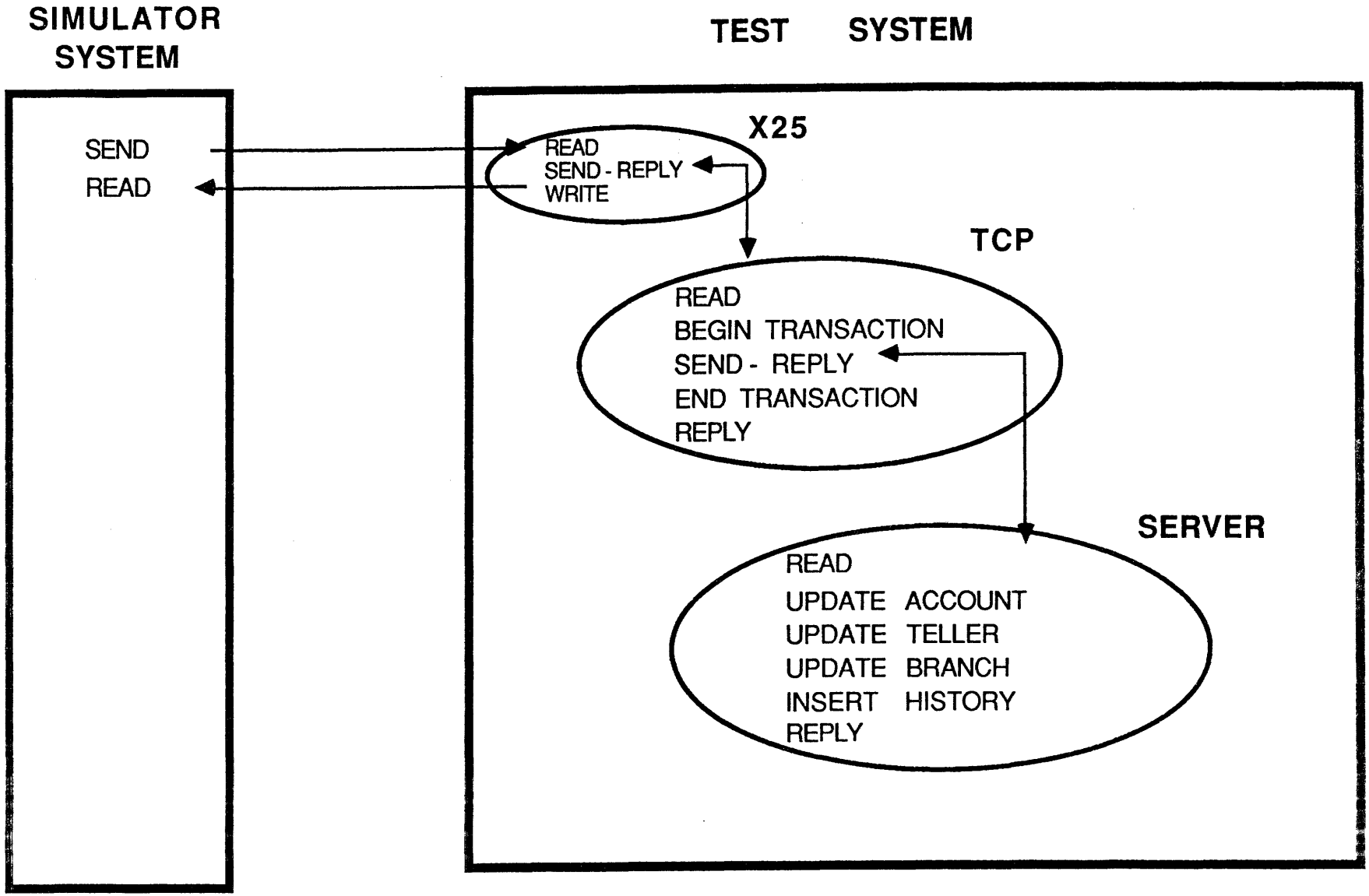
  - TELLER FILE
    - 33 PARTITIONS, 26000 RECORDS

  - BRANCH FILE
    - 33 PARTITIONS, 2600 RECORDS

  - HISTORY FILES
    - ONE PER NODE, TOTAL 5 FILES

- TOTAL DATABASE EXCLUDING HISTORY FILES WAS OVER 2,700 MBYTES.

- ALL STANDARD SOFTWARE USED
  - B41 SOFTWARE
  - PATHWAY
  - COBOL SERVERS FOR ENSCRIBE
  - COBOL / SQL SERVERS FOR NonStop/SQL
  - SCOBOL TCP
  - TMF


SIMULATOR SYSTEM
TEST SYSTEM
SEND
READ
X25
READ
SEND - REPLY
WRITE
TCP
READ
BEGIN TRANSACTION
SEND - REPLY
END TRANSACTION
REPLY
SERVER
READ
UPDATE ACCOUNT
UPDATE TELLER
UPDATE BRANCH
INSERT HISTORY
REPLY

ET1 - SETUP

DEVIATIONS FROM "STANDARD" DURING TESTING

- OVERDONE
  - TRANSACTION ARRIVAL PATTERN
    - THE "STANDARD" DOES NOT SPECIFY THE DISTRIBUTION ARRIVALS FOR THE TRANSACTIONS. FOR THIS TESTING A RANDOM ARRIVAL WITH CONSTANT MEAN RATE WAS USED. (i.e. the distribution of the inter-arrival times was exponential)
    - RANDOM DISTRIBUTION OF ARRIVALS CAUSES GREATER FLUCTUATIONS IN THE RESPONSE TIMES OF THE TRANSACTIONS AS COMPARED TO THE A UNIFORM OR CONSTANT ARRIVALS.
    - RANDOM ARRIVALS IS MOST REALISTIC FOR OLTP SYSTEMS

- OVERDONE
  - RESPONSE TIME MEASUREMENTS
    - RESPONSE TIMES WERE MEASURED IN THE SIMULATOR SYSTEMS. THIS WOULD ADD BETWEEN 150 TO 180 MILLISECONDS TO THE RESPONSE TIMES AS COMPARED TO THE "STANDARD".
  - DATABASE SIZE
    - THE DATABASE WAS OVERSIZED. THE DATABASE WAS SIZED FOR 260 TPS.

- OVERDONE

  - ALL DISCS WERE MIRRORED

    - THE "STANDARD" STATES THAT ONLY LOG (TMF) DISK NEEDS TO BE MIRRORED.

    - ALL DISCS USED FOR THE BENCHMARK WERE MIRRORED.

- UNDERDONE

- THROUGHPUT MEASUREMENTS

  - ALL TPS MEASUREMENTS ARE COMPARED AT 90% OF TRANSACTIONS COMPLETING IN LESS THAN OR EQUAL TO TWO SECONDS.

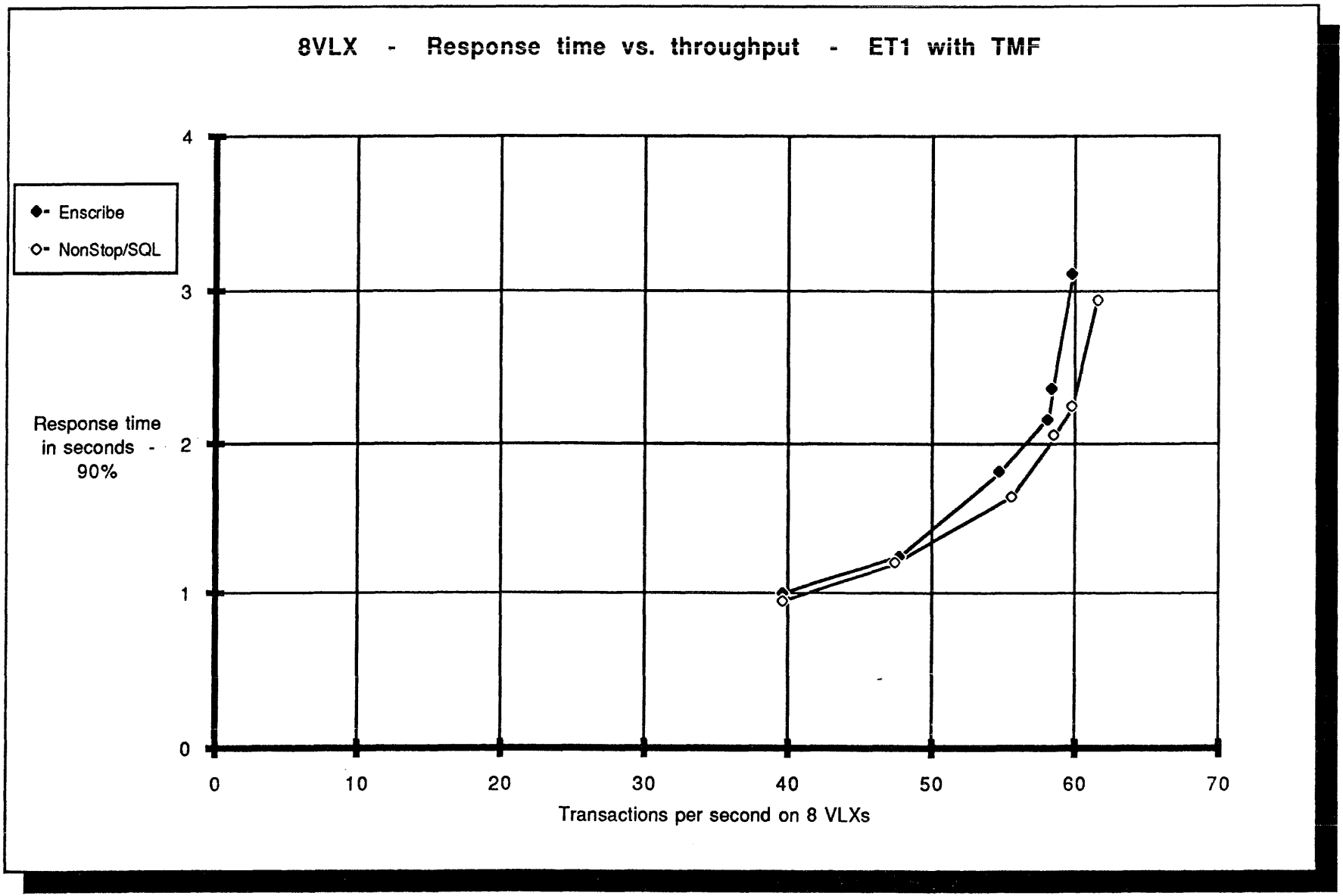
  - THIS WAS DONE SO THAT A COMPARISION ON THE SAME BASIS COULD BE DONE ACROSS THE TANDEM PROCESSOR LINE. (e.g. It would be unrealistic to measure the EXT10 system and compare its throughput at 95% transactions completing in less than or equal to 1 second.)

- UNDERDONE
  - NUMBER OF TERMINALS SIMULATED
    - THE NUMBER OF TERMINALS SIMULATED WERE EQUAL TO THE NUMBER OF BRANCHES (2,600) INSTEAD OF THE NUMBER OF TELLERS (26,000) AS SPECIFIED IN THE "STANDARD".
    - MOST CUSTOMERS WOULD USE CONCENTRATORS FROM THE BRANCHES TO THE DATABASE SYSTEMS. THE 10 TERMINALS AT EACH BRANCH WOULD NORMALLY BE CONNECTED THROUGH A SINGLE CONCENTRATOR. HENCE THE NUMBER OF TERMINALS SIMULATED WERE EQUAL TO THE NUMBER OF CONCENTRATORS (2,600).

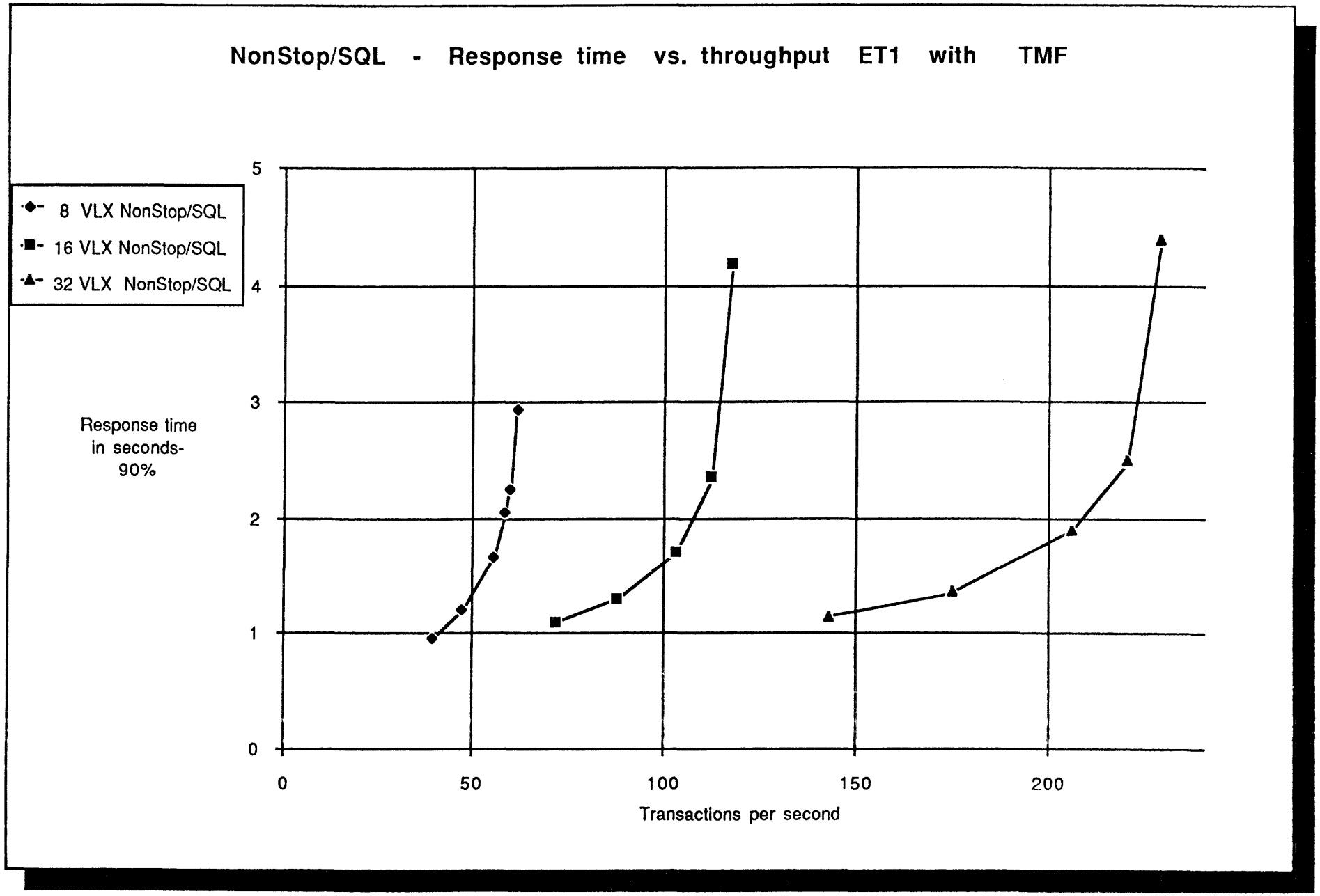
WHY THE AUDITOR?

- VALIDATE HARDWARE USED
- VALIDATE BENCHMARK IMPLEMENTATION
- VALIDATE MEASUREMENTS

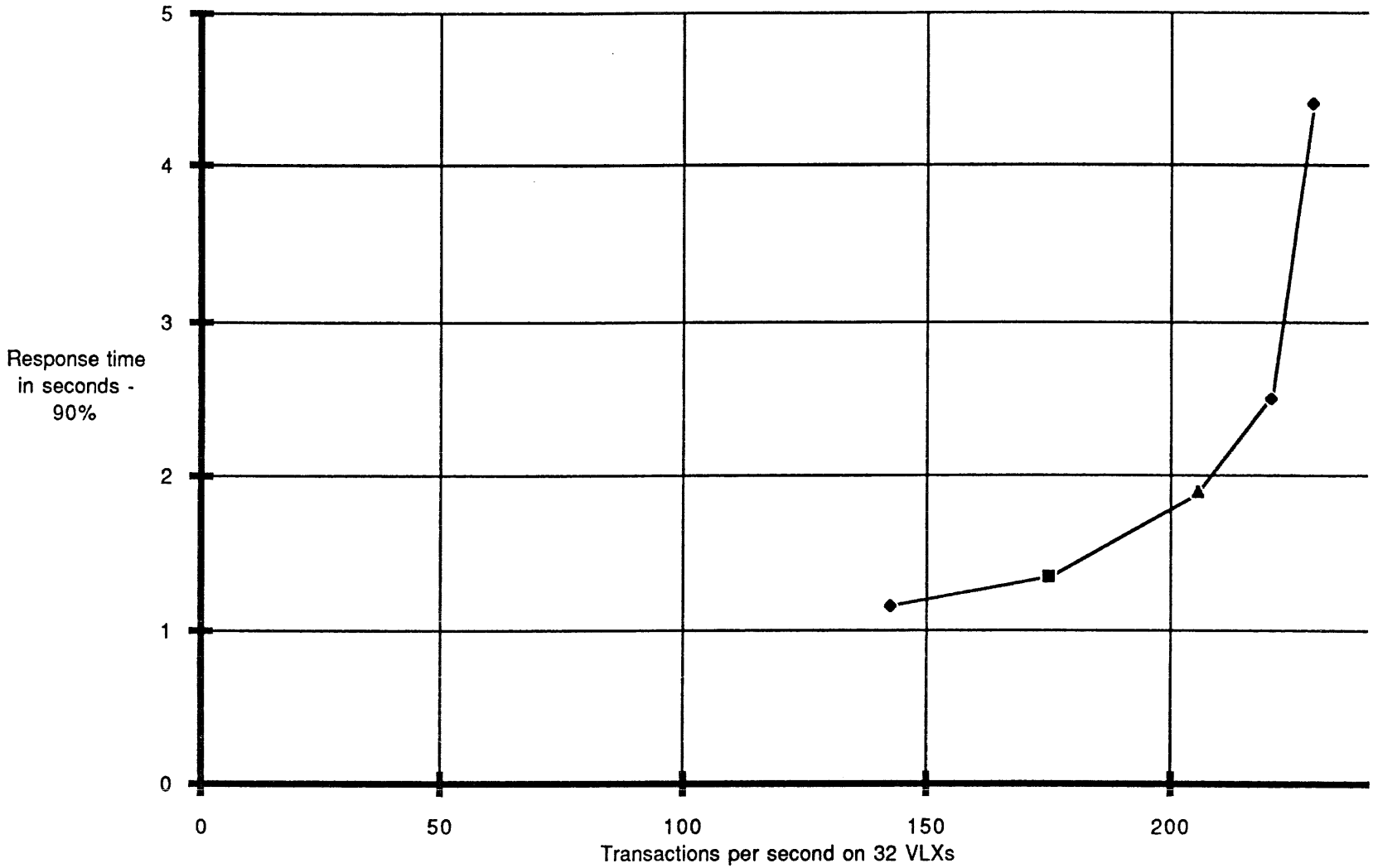
- AUDITOR - Codd & Date Consulting Group

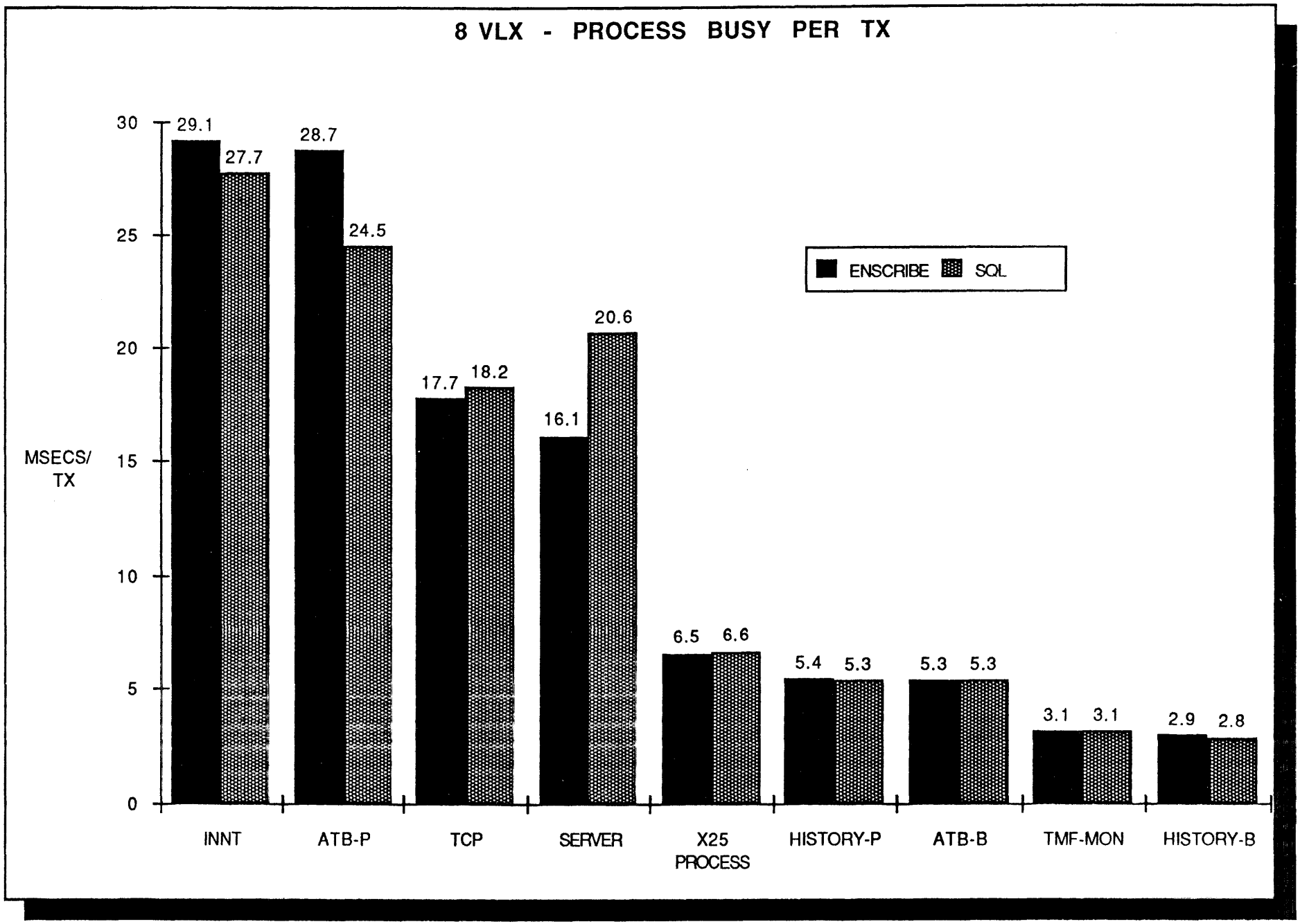


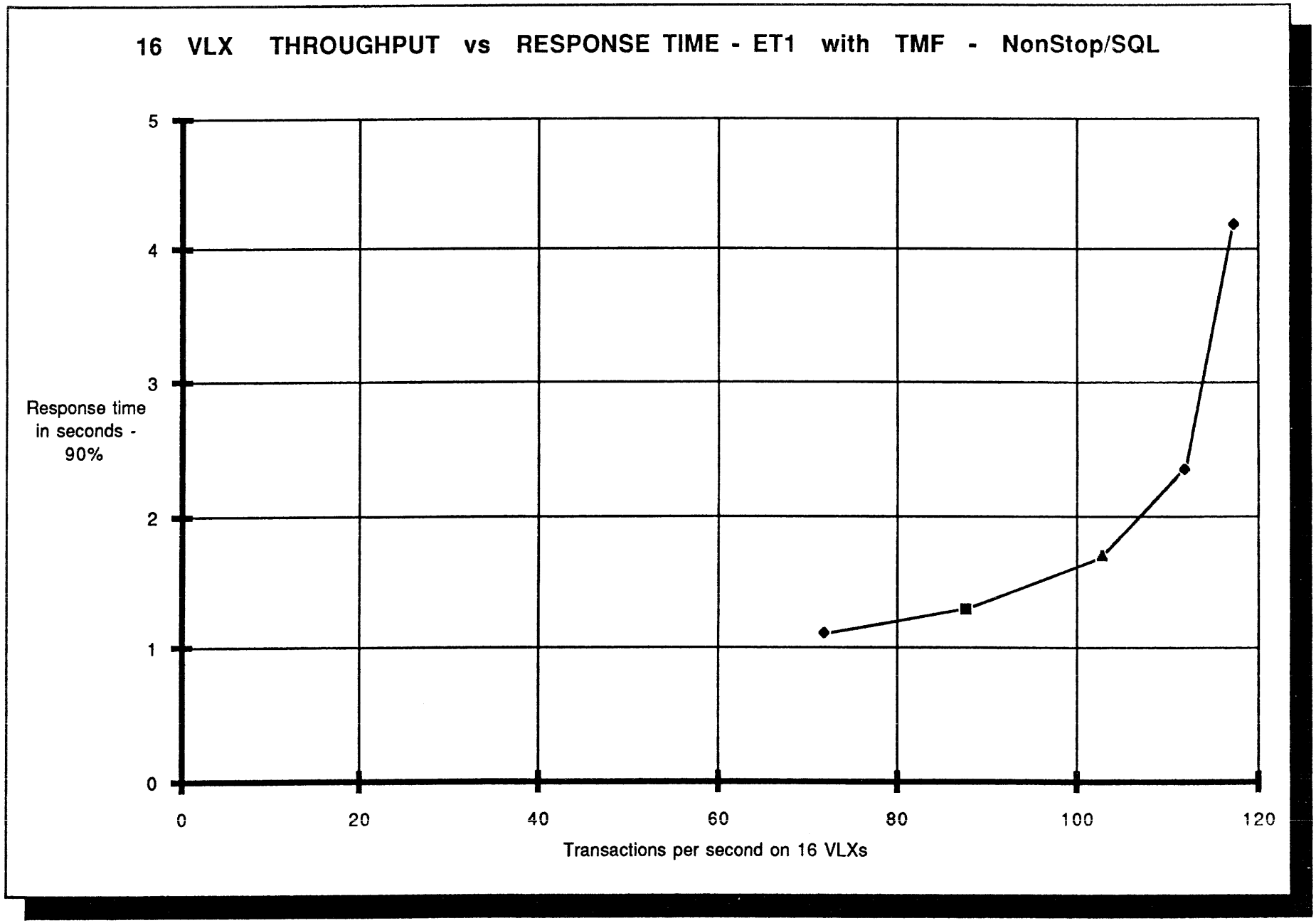
* For 90% of transactions


NonStop/SQL - Response time vs. throughput ET1 with TMF
8 VLX NonStop/SQL
16 VLX NonStop/SQL
32 VLX NonStop/SQL
Response time in seconds- 90%
0
1
2
3
4
5
0
50
100
150
200
Transactions per second

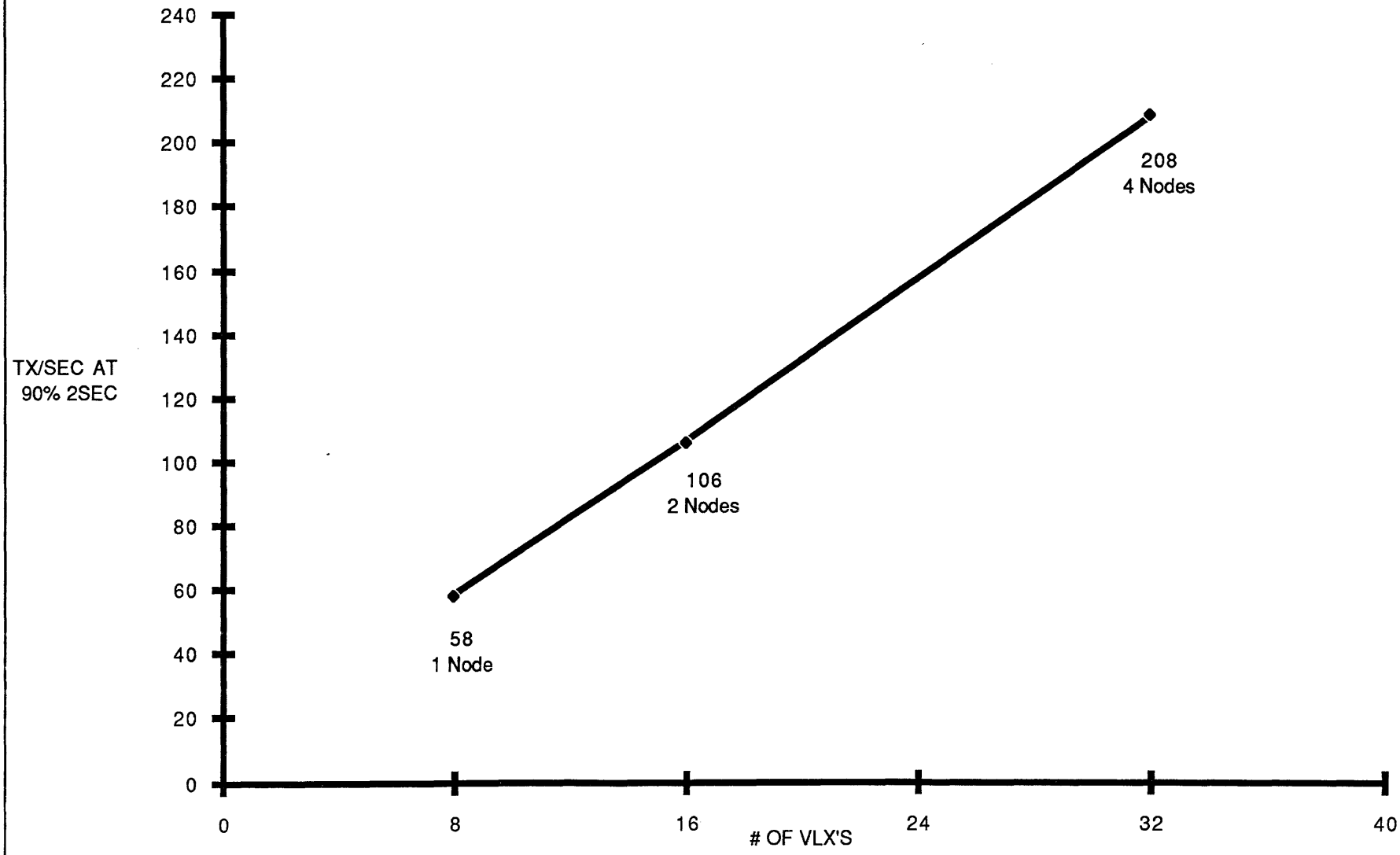
32 VLX - Response time vs. throughput ET1 - with TMF - NonStop/SQL




8 VLX - PROCESS BUSY PER TX
MSECS/ TX
30
25
20
15
10
5
0
ENSCRIBE
SQL
29.1
27.7
28.7
24.5
17.7
18.2
16.1
20.6
6.5
6.6
5.4
5.3
5.3
5.3
3.1
3.1
2.9
2.8
INNT
ATB-P
TCP
SERVER
X25 PROCESS
HISTORY-P
ATB-B
TMF-MON
HISTORY-B


16 VLX THROUGHPUT vs RESPONSE TIME - ET1 with TMF - NonStop/SQL
Response time in seconds - 90%
0
1
2
3
4
5
0
20
40
60
80
100
120
Transactions per second on 16 VLXs

TPS PER PROCESSOR - LINEARITY - 1 NODE TO 4 NODES
ET1 with TMF and Nonstop/SQL



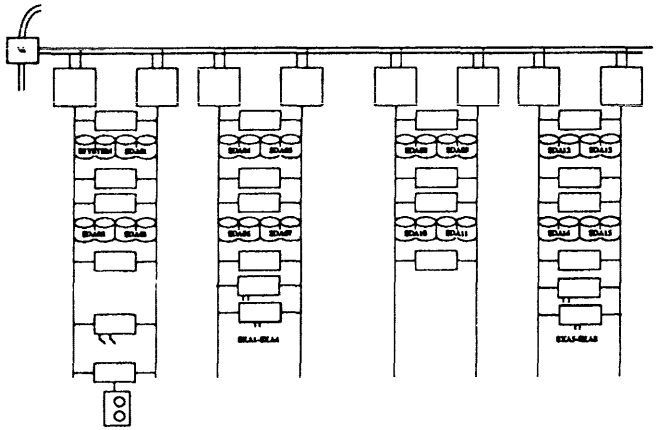
TOP GUN sizing

The 32 VLX processors were configured as four systems of eight processors each.

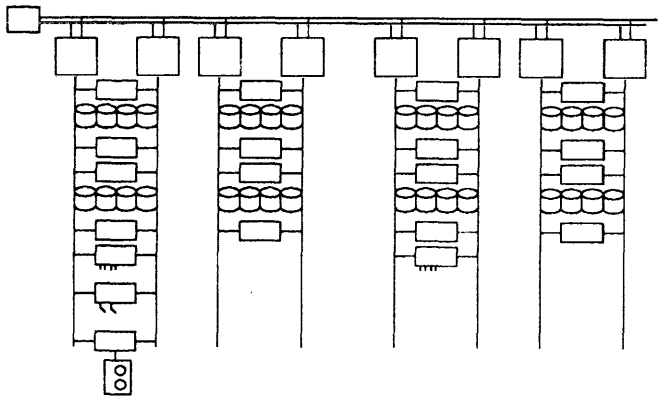
The next page shows the eight processor Mackie diagrams.

Having sized the VLX processors at 8 TPS each, the EXT10 system was sized as 1/2 of a VLX processor.

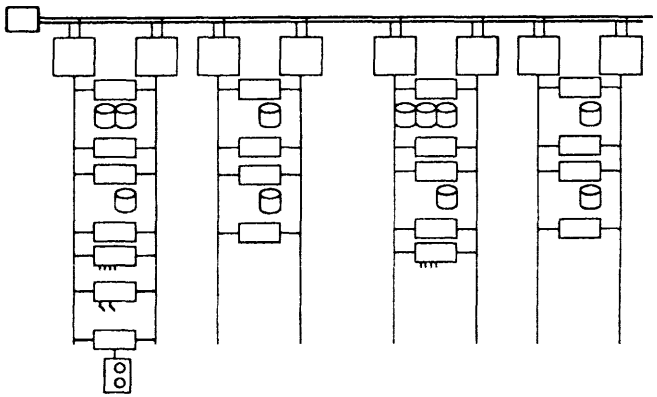
The following page shows a two processor EXT10 system Mackie diagram.



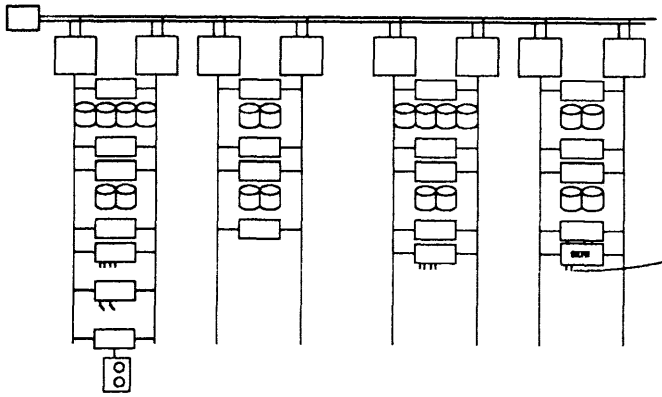
Example node: With full disk configuration (not all are used)



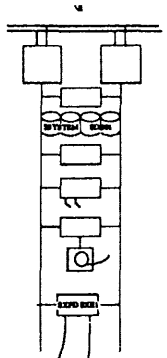
2/15/87 rev 6.0



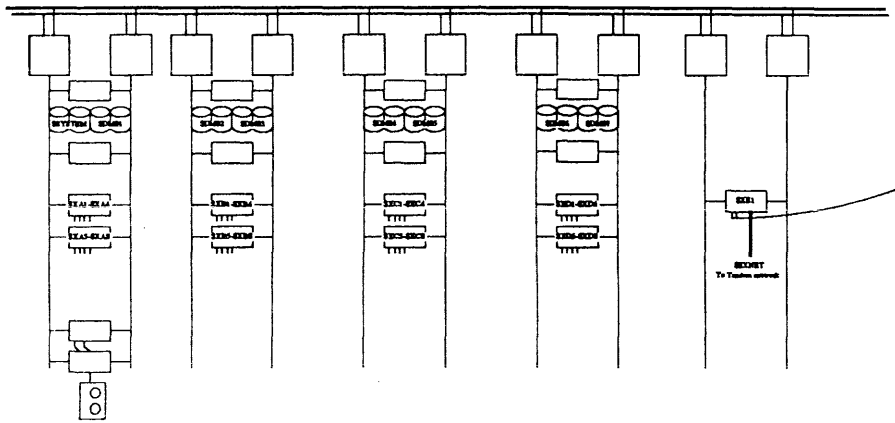
Example node: With Non-Mirrored disk config.



Example node: With Mirrored disk config.

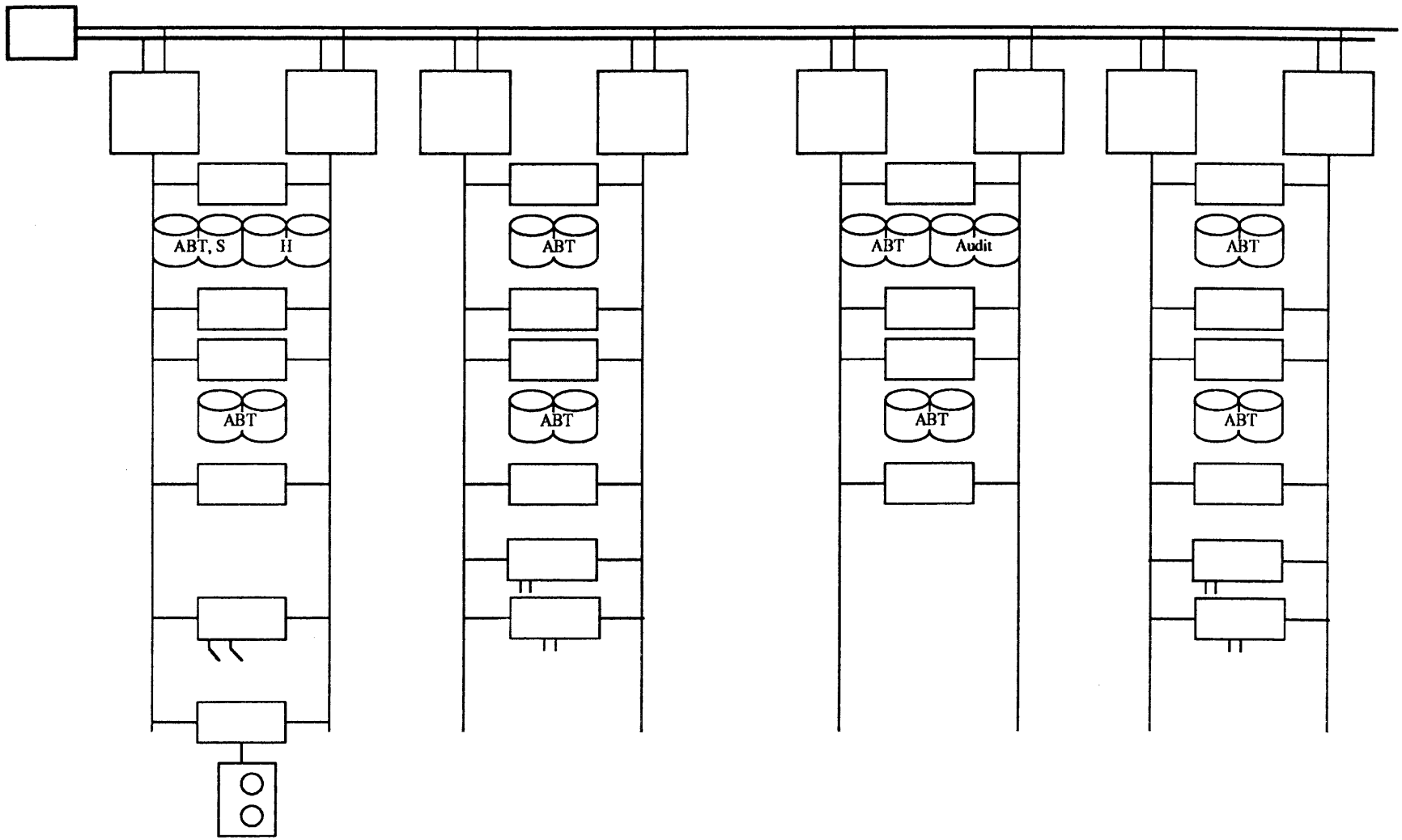


Debit-Credit Side



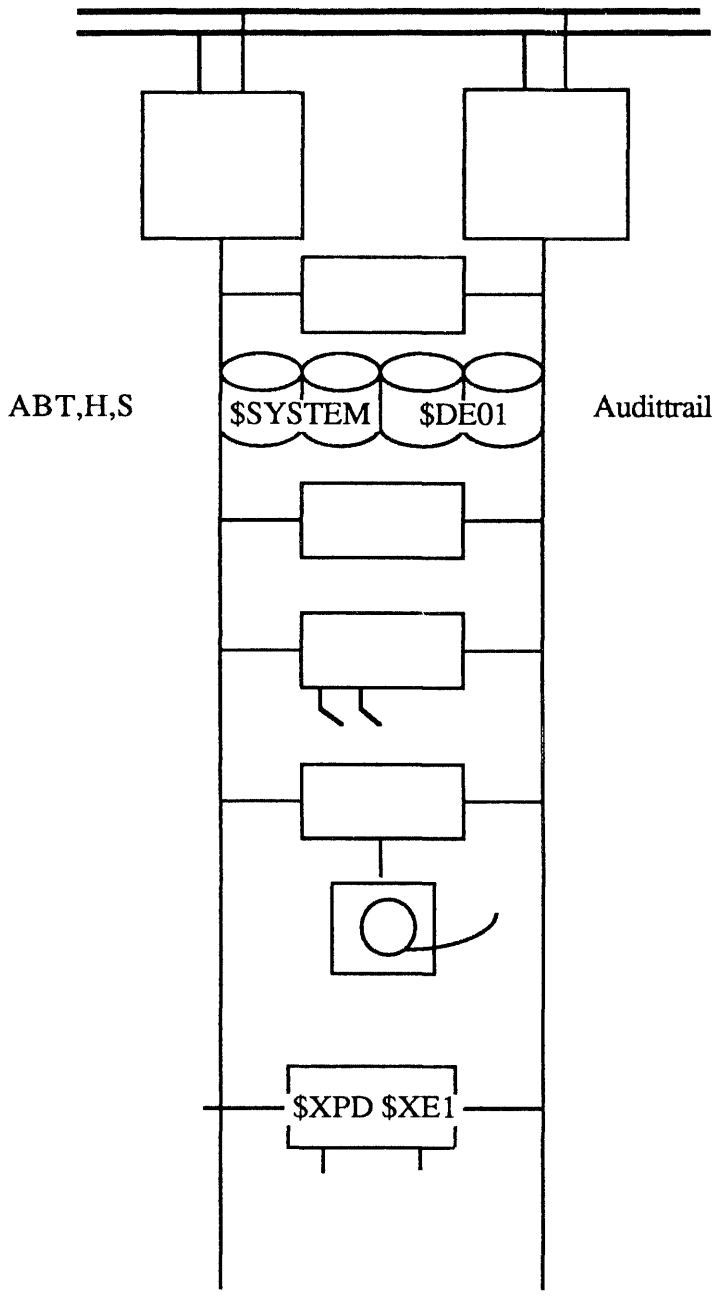
Driver SIDE

Example node: With Mirrored disk config.



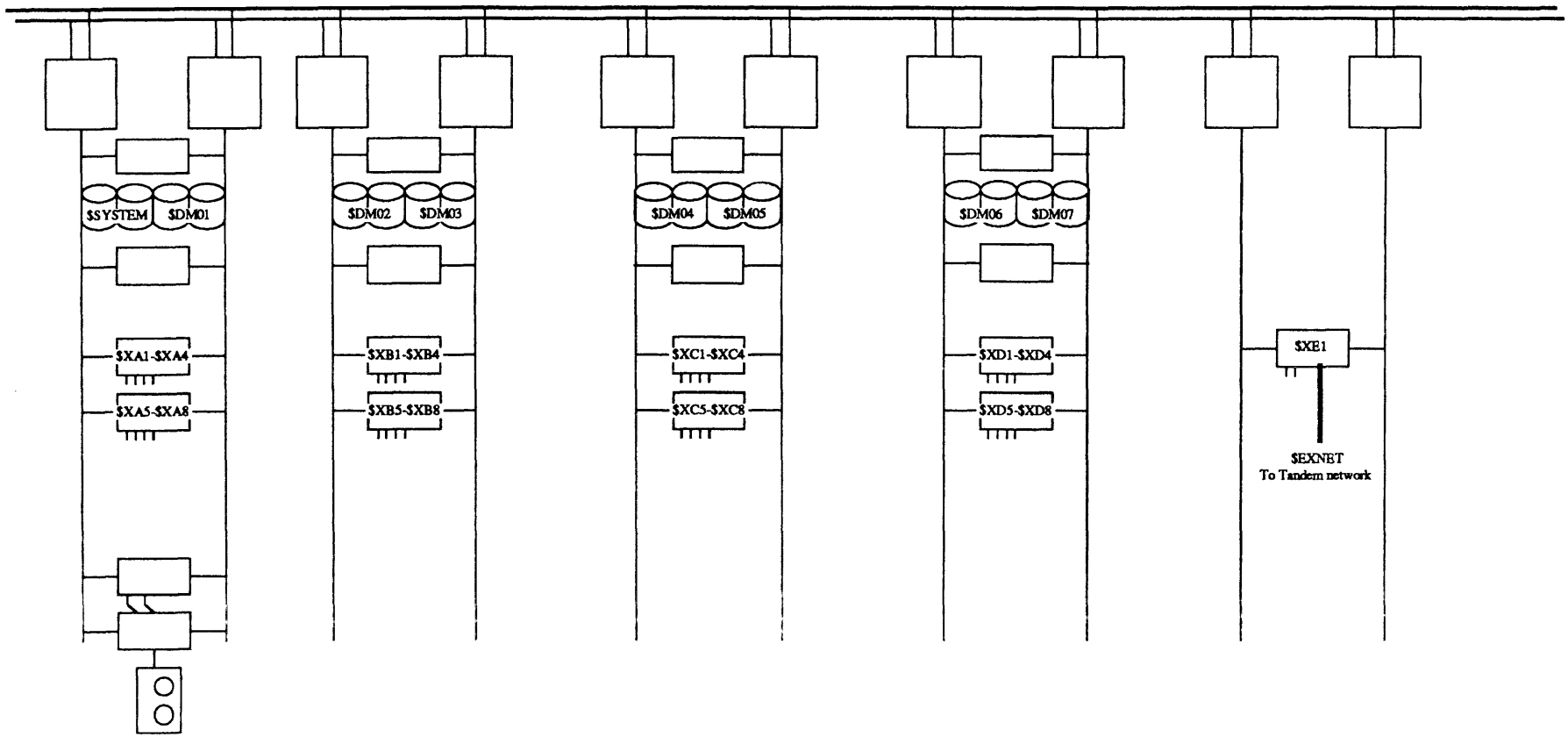
ABT Account Branch & Teller partitions
H History file (one per node)
S System disk

\E



ABT Account Branch & Teller partit
H History file (one per node)
S System disk
A Audit trail

Driver SIDE


$SYSTEM
$DM01
$DM02
$DM03
$DM04
$DM05
$DM06
$DM07
$XA1-$XA4
$XA5-$XA8
$XB1-$XB4
$XB5-$XB8
$XC1-$XC4
$XC5-$XC8
$XD1-$XD4
$XD5-$XD8
$XE1
$EXNET
To Tandem network

This document provides an overview of the SQL and ENSCRIBE benchmarks to be performed by Tandem. Included in this document are: 1) A high level description of the experiments to be run; 2) A sizing of the benchmark, including the equipment to be used; and 3) A schedule of events along the way. Currently open issues and status will appear in separate documents and mail messages.

Experiments

The experiments scheduled are based upon the ET1 standard published in Datamation on April 1, 1985. Exceptions from that standard are the use of conversational terminals instead of block mode devices and lines submit transactions once every ten seconds. This simulates ten attached terminals per X.25 line, each with a 100 second think time.

The benchmarks are intended to demonstrate the performance of Tandem/SQL and ENSRCIBE over a range of system sizes. Additional tests will be run up to a 32 VLX processor configuration which is expected to deliver the following TPS ratings. The intent of these tests is to demonstrate the linearity of Tandem systems. To demonstrate performance and function of remote systems, an EXT10 will be processing distributed transactions during some of the tests.

To size the benchmark the following areas are discussed here:

Driver system CPU requirements
Communication line requirements
Database sizing
disk and controller requirements
Memory requirements
System Configurations

Next the deliverables of the benchmark are discussed:
Description of the benchmark as run.
Disclosure details of tests.
Results.

Driver system CPU requirements.

Assumptions:

Must drive 256 TPS.
TXP processors.
X.25 protocol.
340 Bytes per transaction.
56kb lines.
System software on the driver may be "non released software".
256 Byte X.25 packet size.

Measured CPU components with TXGEN driver.

Description	TXP CPU ms
X.25 IO Process and Interrupts	20ms
Transaction Driver Process	3ms

Predicted CPU requirements for Driver (Remote Terminal Emulation or Driver) system.

23ms per TX total CPU requirement estimated for driver with C00.

23 * 256 = 5.888 TXP seconds per second.

Assuming 75% CPU busy: 5.888 / .75 = 7.8 or **8 TXPs**

This assumes CPU balancing is perfect. Ten TXPs will be used to drive 256 TPS. Sixteen processors are available and built as a single system. Each processor pair on the driver system will generate the transactions for an 8 processor VLX. The last CPU pair will drive the EXT10 system and provide spare capacity.

Communication line requirements

Assumptions:

Must drive 256 TPS.
TXP processors (driver side).
X.25 protocol.
340 Bytes per transaction.
56kb lines.
SBS-4 (6105) a/k/a Single Board Sage is planned
6104 bit sync controllers may be substituted for 6105s.

Required bandwidth

340 bytes per tx * 256 TPS = 87,040 Bytes per second
= 87.04 KB/s

Bandwidth per line
56Kbps line = 7KB/s (read or write)
At 40% capacity 2.8 KB/s

So 87.04 KB/s
2.8KB/s = 31.08 or 31 lines at 40%

If 4 lines per SBS-4 controller are assumed
This means 31 / 4 or 8 controllers = 16 Total (Driver + Debit-Credit at 40%)

Or if 2 lines per bit sync controller is assumed (SBS-4 controllers not available)
This means 31 / 2 or 16 controllers = 32 Total (Driver + Debit-Credit at 40%)

These figures do not include the 2 extra controllers to drive and foreward transactions to the EXT10. The EXT10 comes with it's own 6105 controller. However the Driver system will need a controller to drive the EXT10. One of the VLX systems will require an expand line for the distributed remote transactions from the EXT10.

The cost for a bit sync controller for 5 yrs is approximately $10,000. If we use eight additional controllers on the Debit-Credit (or 16 additional bit sync controllers in total) we will add $80k to our 5 year cost. This adds only $250 to $350 per transaction to our cost which is expected to be around $50k/t/s for 5 year system cost.

Each active line (32 on VLX side) will require a modem eliminator for clocking the bit sync controller ports. Joining these 32 will be two additional modem eliminators for the EXT10 Expand and X.25 lines.

Summary:

32 Active 56kb lines.
1 Active 9.6kb line for EXPAND between a VLX and the EXT10.
1 Active 37kb line for X.25 lines into the EXT10.
34 Modem eliminators

19 SBS-4 Controllers (10 on Debit-Credit side* and 9** on the Driver side)
or
33 Bit sync Controllers (17 on Debit-Credit side plus a 6105 on the EXT10 and 17 bit sync controllers on the Driver side)

* 8 on VLX, 1 on VLX for Expand to EXT10 and one in the EXT10.
** 8 to drive VLX, 1 one to drive EXT10

Database Sizing

Assumptions:

Datamation standard, sized to maximum TPS rating of system.
Expected VLX transactions per processor per second is eight.
Eight processor systems need a 64 TPS database per node.

A four processor system would have half the database spread over half the number of discs and CPUs. The partitions themselves would be identical.

The 260 TPS database **distributed over 5 systems** will look as follows:

File name	Record Size	Number of records	File Size	File Type
Account	100	26m	2.6gb	Key Seq partitioned - 12 byte key
Branch	100	2.6k	260kb	Relative partitioned
Teller	100	26k	2.6mb	Relative partitioned
History	50	1/tx	-	Entry Sequenced

A 64 TPS database on each VLX **node** will look as follows:

File name	Record Size	Number of records	File Size	File Type
Account	100	6.4m	640mb	Key Seq partitioned - 12 byte key
Branch	100	640	64k	Relative partitioned
Teller	100	6.4k	640k	Relative partitioned
History	50	1/tx	See Table	Entry Sequenced

A 8 TPS database on each VLX **partition** will look as follows:

File name	Record Size	Number of records	File Size	File Type
Account	100	800k	80mb	Key Seq partitioned - 12 byte key
Branch	100	80	8k	Relative partitioned
Teller	100	800	80k	Relative partitioned
History	Not partitioned		See sizing equation for History file.	

The 4 TPS database on the EXT10 partition will look as follows:

File name	Record Size	Number of records	File Size	File Type
Account	100	400k	40mb	Key Seq partitioned - 12 byte key
Branch	100	40	4k	Relative partitioned
Teller	100	400	40k	Relative partitioned
History	Not partitioned		See sizing equation for history file.	

Cache sizing

Account file cache calculation assumptions:

4k blocking on Account file
200 keys per index block
Indexes to be cache resident.

800000 records / 40 records per block = 20000 datablocks per Account partition.
20000 index pointers at 200 pointers per block = 100 4k index blocks per Account partition.
100 / 200 = < 1 root index block .

101 4kbyte index blocks per Account partition
250 4kbyte data blocks to assure data blocks are LRU.
Partitions with 500 blocks total cache.

Branch and Teller cache calculations:

512 Byte blocking
127KB Branch file + 127KB Teller file = 256 KB of Branch and Teller data
(These files are overpopulated to achieve the desired partitioning)
256000 / 512 = 500 .5k cache blocks per Branch/Teller partition

Total primary disk cache requirements: 351 4k blocks or 1.56 mb Account cache
500 .5k blocks or .256 mb Branch and Teller cache

Total primary cache per partition = 1.65mb or 3.3mb per CPU including the backup disk processes cache.

Assumptions:

8 TPS per partition (account, branch, teller) is peak access rate.
1 partition per CPU
1 Account read per tx
1 Account write per tx
16 IOs per partition (at 8 TPS / CPU) is acceptable and will provide reasonable response times.

1 History file disk per system (8 CPU or 4 CPU) shared with system disk.
1 Mirrored Audittrail per system. (Mirrored audittrails are a requirement of ET1)

Mirroring of the application database is not a requirement for ET1. Two different Pricing calculations will be made. If desired a non-mirrored test can be run to verify performance expectations. See "ET1 mirrored vs Non-mirrored" document.

Each system will have:

- eight mirrored volumes for each of the account, branch and teller partitions (V8 drives are cost-effective and suitable for the task with enough space)
- one volume containing both history and system files
- one audittrail volume

Summary:

A total of **10 mirrored volumes** or 20 disk drives per 8 CPU node. With this configuration the physical IOs on the account branch and teller partitions would consist of a read from the primary or mirror and a write to the primary and mirror disk. The history file with 4k blocks (Each containing 80 records per block) will require 2 IOs for every 80 transactions. At 64 TPS this is about 1.25 IOs / second.

For non-mirrored application database files, the disk compliment would be reduced in the following way. One less for each of the account, branch and teller partition disks or 8 fewer disks, also the history volume mirror could be deleted for a total reduction of 9 disks. This brings the total number of disks per node to 11. The history file disk space and controllers required to store 90 days of transactions must still be added.

Assumptions:

We will not exceed 8mb per CPU. The cost would be $6k per TPS to upgrade all CPUs to 16mb.

Caches are about 3.5mb/CPU

OS requirement 2+ mb

TCP requirements for 2.5 TCPs per CPU = .5mb

There will be 80 concurrent X.25 circuits per CPU or 640 per 8 processor node. At 8 TPS/CPU this would require an inter-circuit arrival time of 10 seconds.

The ET1 specification calls for a think time of 100 seconds per teller which would require one hundred terminals per transaction per second, or 800 per CPU for a total of 6400 terminals per node.

With 80 circuits per CPU and 8 CPUs per system this equals 640 circuits per system. With 32 branch circuits per TCP, 20 TCPs per system are required. This averages out to 2.5 TCPs per CPU.

Each TCP requires 100k bytes for stack plus 400 bytes per circuit for context or 100k + 1000 * 32 = 132k bytes of data per TCP.
132k * 2.5 = .33 mb TCP data per CPU + .1mb* TCP code per CPU for a total of .45 or **.5mb / CPU for TCPs.**

*Current measurements show the TCP code per CPU to be about 50 pages or 100k bytes.

The autorestart option will be for TCPs, so no backup TCP or checkpointing to the backup TCP will be required. Additionally IDS support will be used for the terminals.

Server requirement .48mb for 20 SQL servers.

COBOL/SQL Server process measurements.

Code space	120k / CPU
Stack & PFS	18k / server

Eight servers per TCP = 20 Servers per CPU.
120k + 18k * 20 = 480 or **.48mb of servers per CPU**

Enscribe Server process measurements.

Code space	40k / CPU
Stack & PFS	10k / server

Eight servers per TCP = 20 Servers per CPU.
40k + 10k * 20 = 240 or **.24mb of servers per CPU**

System Configuration

The Mackie diagram shows the full size of the benchmark. Each of the 4 VLX nodes is configured as shown by the systems at the top of the diagram. The one exception will be the forth system which will also require an Expand link to the EXT10. The example nodes demonstrate the following three system configurations:

Example node descriptions:

Full disk benchmark configuration
This is the basic system configuration, All systems are physically configured this way.

Mirrored disk configuration
This is the configuration of the systems as they are run. Only discs which are utilized are shown. This is NOT the pricing configuration. The 90 day history file requirement must be added onto the 5 year cost. This then is the 5 year price configuration (burdened with mirrors).

Un-mirrored disk configuration
This is the configuration of the systems if the database were not mirrored. The audittrail still must be mirrored. Once again , this is NOT the pricing configuration. The 90 day history file requirement must be added onto the 5 year cost. This then is the 5 year price configuration.

System software requirements

Required software by product;
Guardian, Pathway, TMF, DDL, COBOL, SQL stuff, X.25 and EXPAND
SORT should also be included.

Simple costing would include.
Guardian 90XF (Guardian, Pathway, TMF and DDL)
COBOL and SQL Stuff
X.25 and EXPAND

90 Day history file requirement

The amount of disk required to store 90 days of the History file online has been calculated with the following assumptions. The average transaction load is 1/3 of the peak. Also, 24 hour a day 7 day a week operation is assumed. The EXT10 is the most sensitive to changes because of low entry cost and not enough in-cabinet growth capacity.

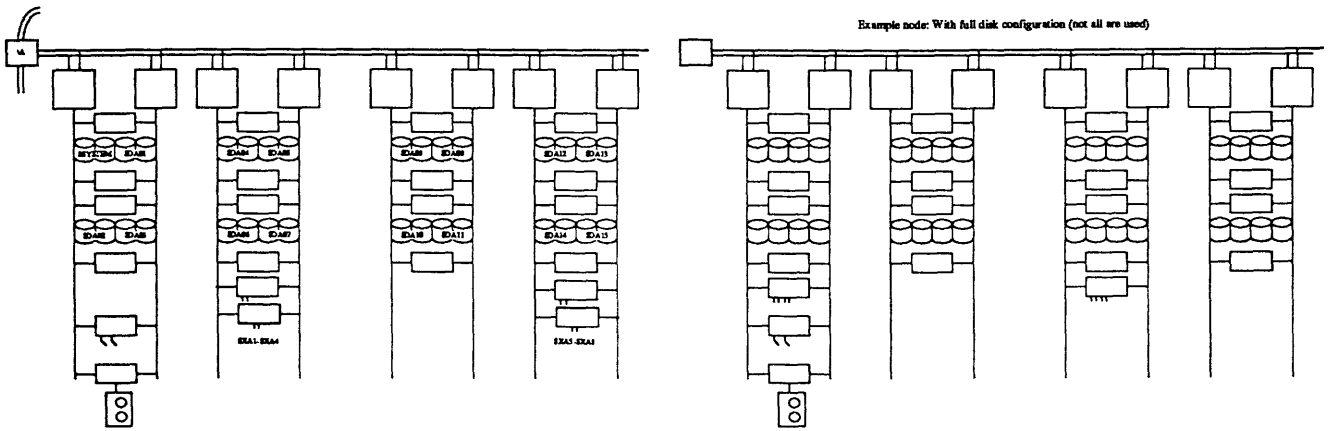
For the VLX nodes:
(208 TPS peak * 1/3 average) * 60 seconds * 60 minutes * 24 hours * 90 days * 50 bytes =

26,956,800,000 bytes or 27 GB for 4 VLX nodes or 6,750,000,000 bytes or 6.7 GB per node.

This should be about 6.7 GB per node, and require 16 XL8 volumes or 2 XL8 drives, assuming a non-mirrored long term history file. The initial cost is about 500k$ to support 90 days of 64tps banking for about 7,800$ per transaction per second.

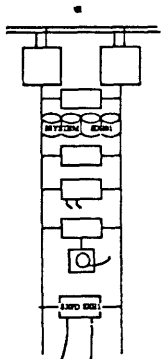
For the EXT10 node:
(3 TPS peak * 1/3 average) * 60 seconds * 60 minutes * 24 hours * 90 days * 50 bytes =

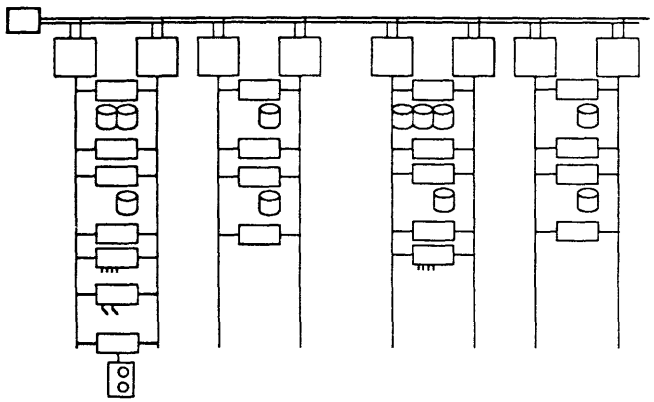
388,800,000 bytes or 389 MB for the EXT10. The best option is to replace the internal drives and controllers with an XL4. A mirrored volume configured as audittrail, the other with the entire online database and 90 days of history.



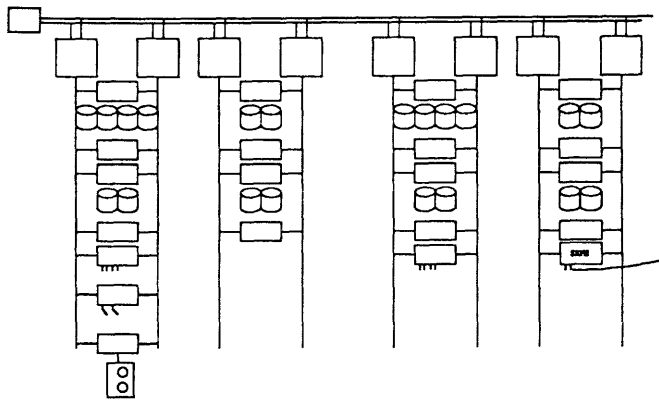
Example node: With full disk configuration (not all are used)

2 / 15/ 87 rev 6.0



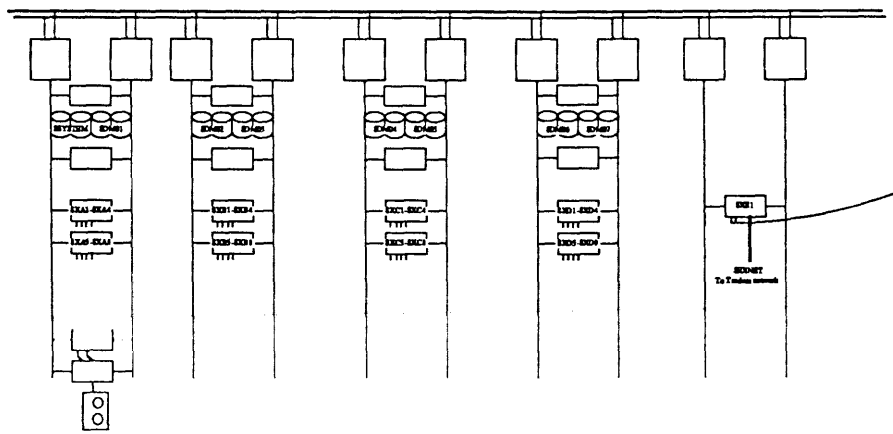


Example node: With Non-Mirrored disk config.



Example node: With Mirrored disk config.

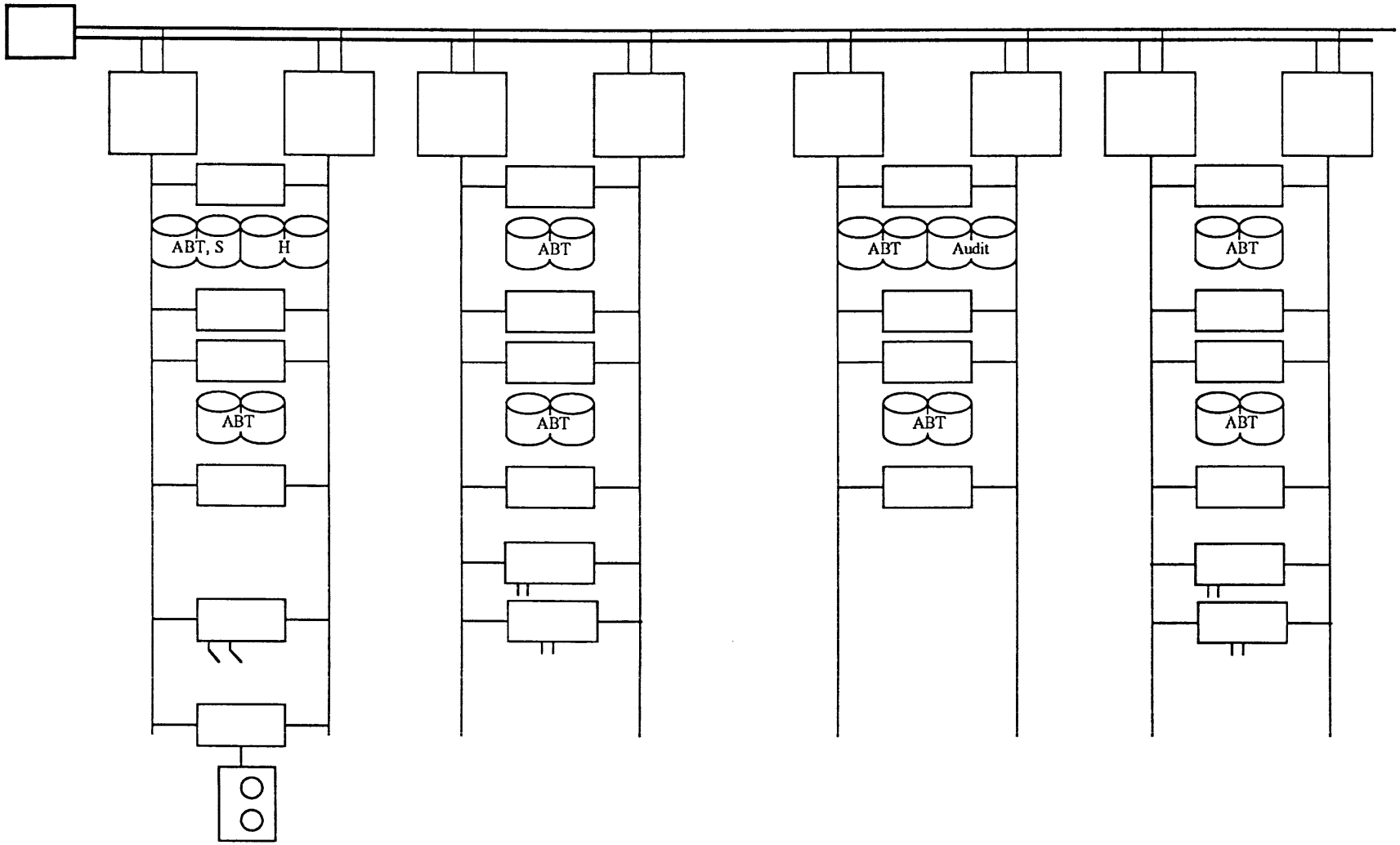
Debit-Credit Side



Driver SIDE

6.9

Example node: With Mirrored disk config.



ABT Account Branch & Teller partitions
H History file (one per node)
S System disk

The objective here is a full and complete description of the test run that was run and results that were obtained. To achieve this the following information will be made available:

Description of the benchmark

- Application description
  - Database description
  - Requestor description
  - Server description

Disclosure details of tests

- Implementation details of application
  - Number of terminals and think times
  - Database layout including locality of reference
    - physical size including indexes
  - TCP configuration
    - Terminals per TCP
    - TCP options selected
- Implementation details of system software configuration
  - OS revision level and relevant sysgen options
    - Expedited buffers, etc
  - Number of disk processes and type
  - Communication lines and type
- System hardware configuration
  - Including price of required configuration.
    - Initial cost including installation (free).
    - Cost of ownership (other Tandem charges) for five years

Results disclosure description

- Response time versus throughput curves.
  - 95th percentile
  - 90th percentile
  - Average
- CPU consumption per transaction
  - Total
  - By process
- IOs per transaction
  - By file type
  - By device
- Device utilizations
  - Discs
  - Processors
  - Comm Lines
- Driver system details
  - CPU per tx and total, disk IOs.

Schedule of events

Dec 31 Equipment ordered for Benchmark center.

Jan 5 Publish the following:
- This Schedule
- The Mackie diagrams for 8 x 4 configuration
  - The memory margin expected.
  - The comm sizing calculations.
  - The preliminary DB sizing.

Jan 5 Investigate changes to the driver to reduce the hardware resource requirements of the RTE system.
Jan 5 Have VLX four processor system up and running \Tino.
Jan 6 Have Build 12 installed on \Tino and \Cuper.
Jan 7 Unit test ET1 in a multi user environment.
Jan 7 Determine ET1 driver status; Distribute driver.
Jan 8 Complete a full ET1 SQL measurement.
Jan 12 Harald performing Enscribe tests.
Jan 14 Formalize the DB layout (send to Harald too) into a publishable description.
Jan 15 Meet w/ Codd-Date Auditor Tom Sawyer.
Jan 16 Checkpoint of plan to all involved (Will Jan 20th milestones be met?)
Jan 20 All three sites should be activly persuing:

B' Center	Continued system configurations comm, fox, etc.
Frankfurt	Proceeding with network enscribe on 1 x 4 thru 4 x 4. (2x4?)
Ridgeview	Complete 1 x 4 tests; Complete other tests; Test software fixes as delivered from Tandem/SQL group, etc. ReTest as necessary.

Jan 20 Publish results of 1 x 4. With only 500 terminals, this is to further assure the results for the 32 VLX benchmark.
Jan 20 Tom Sawyer familiar with the benchmark
Jan 21 Train arms and legs as needed (at least one Cupertino person to go to Germany).
Jan 28 Transmittal of updated version of plan to Cupertino from Germany

Jan 29 Network testing of Enscribe and SQL ET1 configurations should be complete.
Jan 29 Failure mode tests defined. Capabilities established and coded software.
Jan 29 Communication hardware decision (6105s if available).

Feb 2 Review formal presentation Gray & Enright (in Cupertino)
Feb 3 Have driver system and COMM hardware installed and tested at PSG.
Feb 3 Formal presentation to Tom Sawyer of tests to be run.
Feb 4 Begin configuring test on 4 x 8 VLX configuration. Enscribe & Tandem/SQL.

Feb 12 Bring in Tom Sawyer to begin review process.
Feb 12 Publish results of 1 x 8 SQL and Enscribe.
Feb 13 Review test plan with Tom Sawyer.
Feb 13 SQL database built on 34 processors (33 partitions on 5 nodes)
Feb 14 Build scripts and complete first 32.5 VLX SQL run.

Feb 15 Make series of 32.5 VLX SQL runs.
Feb 16 Vacation...Sure. (Dry run with Auditor)
Feb 17 External show for Auditors. Get final debit amounts to be applied to database.
Feb 18 Prepare preliminary reports on 32.5 VLX runs.
Feb 18 Internal failure mode tests.
Feb 20 Park all data on tapes...,etc.
Feb 20 Shutdown/Wrap at benchmark center.

Feb 24 Give Frank Clugage a final report...
Mar 16 Announce to public.
Mar 17 Have a cigar & watch the movie...

Price Performance Summary

Three hardware configurations are shown for price performance comparison purposes. They are:

- ◇ Benchmark compliance, plus additional fault tolerant capabilities
- ◇ Benchmark compliance
- ◇ Benchmark compliance without the 90 days of online history storage requirement

SUMMARY (NonStop / SQL)

ET1 - FULL TRANSACTION PROTECTION (TMF)

SYSTEMS	TPS - 90% AT 2 SEC RESPONSE TIME	5 YEAR COST OF OWNERSHIP IN K$	K$ / TPS
1 * 8 VLX	58	3,314	56.2
2 * 8 VLX	106	6,250	58.7
4 * 8 VLX	208	11,436	59.6
EXT10	4	255	63.8

NOTES: *"compliance plus"*

Above cost includes :

- 90 day HISTORY file disk storage requirement
- Dual data paths to 90 days of History data
- All database and Audittrail disks mirrored
- Full list Price (Not discounted)
- Maintenance cost with NPV @ 15% 5 years
- For multiple nodes, Distributed Service Option included in the pricing

SUMMARY (NonStop / SQL)

ET1 - FULL TRANSACTION PROTECTION (TMF)

SYSTEMS	TPS - 90% AT 2 SEC RESPONSE TIME	5 YEAR COST OF OWNERSHIP IN K$	K$ / TPS
1 * 8 VLX	58	2,995	51.6
2 * 8 VLX	106	5,775	54.5
4 * 8 VLX	208	11,436	55.0
EXT10	4	255	63.8

NOTES: *"straight compliance"*

Above cost includes :

- 90 day HISTORY file disk storage requirement
- Single data path to 90 days of History data
- Audittrail disk(s) mirrored, Database disks not mirrored
- Full list Price (Not discounted)
- Maintenance cost with NPV @ 15% 5 years
- For multiple nodes, Distributed Service Option included in the pricing

SUMMARY (NonStop / SQL)

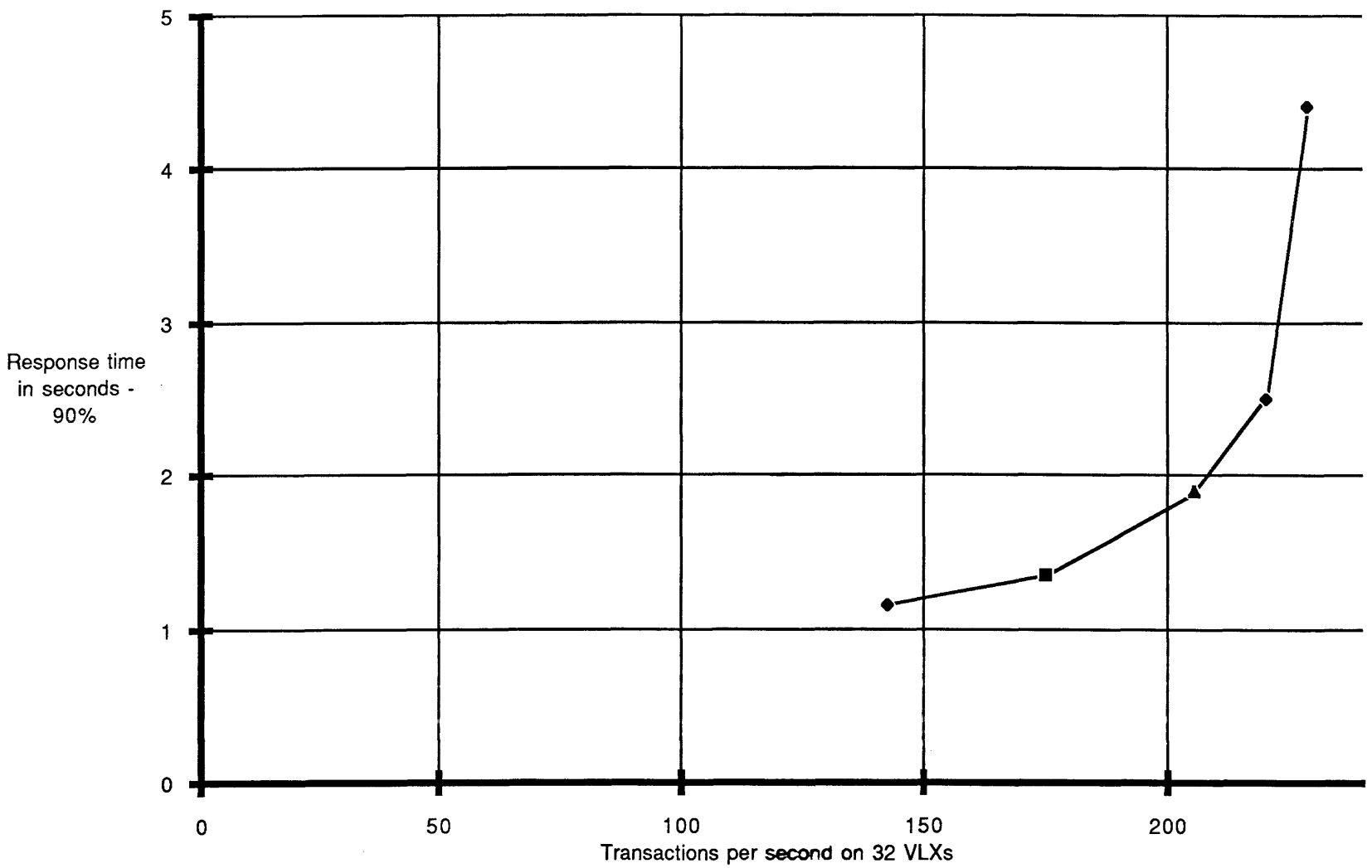
ET1 - FULL TRANSACTION PROTECTION (TMF)

SYSTEMS	TPS - 90% AT 2 SEC RESPONSE TIME	5 YEAR COST OF OWNERSHIP IN K$	K$ / TPS
1 * 8 VLX	58	2,485	42.8
2 * 8 VLX	106	4,911	46.3
4 * 8 VLX	208	9,709	46.7
EXT10	4	194	48.6

NOTES: *"compliance minus 90 day History requirement"*

Above cost includes :
- Audittrail disk(s) mirrored, Database disks not mirrored
- Full list Price (Not discounted)
- Maintenance cost with NPV @ 15% 5 years
- For multiple nodes, Distributed Service Option included in the pricin

32 VLX - Response time vs. throughput ET1 - with TMF - NonStop/SQL



Simulator Report

NonStop/SQL

4 * 8 VLX SYSTEMS

Throughput - 229 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test ALL;RUN06-32.5;SQL
Test Date                  =   87-02-19
Test Length                =   19:06:55 - 19:31:09  (1454 sec)
Window Length              =   19:13:00 - 19:28:00  (900 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TLOGSUMZ

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    8.500 tx/sec
Tx Arrival Rate/Terminal   =    0.106 tx/sec
Tx Inter-Arrival Time      =    9.412 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) ---------------------------------

Tx Throughput              =  229.353 tx/sec
Tx Response Time Average   =    2.759 sec
                 Std Dev   =    1.426 sec
                 Minimum   =    0.549 sec
                 Maximum   =  142.495 sec
                 Count     =   206418 #

Tx Response Time 10%       =    1.550 sec  ( 11.7%)
                 20%       =    1.800 sec  ( 21.3%)
                 30%       =    2.000 sec  ( 30.3%)
                 40%       =    2.250 sec  ( 41.3%)
                 50%       =    2.500 sec  ( 51.5%)
                 60%       =    2.750 sec  ( 60.0%)
                 70%       =    3.150 sec  ( 70.8%)
                 80%       =    3.650 sec  ( 80.7%)
                 90%       =    4.400 sec  ( 90.3%)
                 95%       =    5.100 sec  ( 95.0%)

       Tx Think Average    =    9.409 sec
                 Std Dev   =    9.389 sec
                 Minimum   =    0.001 sec
                 Maximum   =  112.020 sec
                 Count     =   155386 #    ( 75.3%)

       Tx Delay  Average   =    1.728 sec
                 Std Dev   =    1.737 sec
                 Minimum   =    0.000 sec
                 Maximum   =   94.971 sec
                 Count     =    51032 #    ( 24.7%)
```

```
------------------------------------------------------------------------
                  *  ET1 Response Time Distribution  *
------------------------------------------------------------------------
   Mean         Std          Min          Max          Count      Thruput
   2.759 s      1.426 s      0.549 s    142.495 s     206418      229.353 tx/s
------------------------------------------------------------------------

RespTime Count   Pct   CumPct
-------- -----  -----  ------
   0.000     0    0.0    0.0 |
   0.100     0    0.0    0.0 |
   0.200     0    0.0    0.0 |
   0.300     0    0.0    0.0 |
   0.400     0    0.0    0.0 |
   0.500     0    0.0    0.0 |
   0.600     9    0.0    0.0 |
   0.700    38    0.0    0.0 |
   0.800   135    0.1    0.1 |
   0.900   400    0.2    0.3 |*
   1.000   946    0.5    0.7 |****
   1.100  1638    0.8    1.5 |*******
   1.200  2611    1.3    2.8 |***********
   1.300  3832    1.9    4.7 |*****************
   1.400  4891    2.4    7.0 |**********************
   1.500  6078    2.9   10.0 |****************************
   1.600  7098    3.4   13.4 |********************************
   1.700  7739    3.7   17.2 |**********************************
   1.800  8600    4.2   21.3 |****************************************
   1.900  9135    4.4   25.7 |******************************************
   2.000  9356    4.5   30.3 |*******************************************
   2.100  9086    4.4   34.7 |******************************************
   2.200  9160    4.4   39.1 |******************************************
   2.300  8960    4.3   43.5 |*****************************************
   2.400  8536    4.1   47.6 |***************************************
   2.500  7979    3.9   51.5 |************************************
   2.600  7515    3.6   55.1 |**********************************
   2.700  6792    3.3   58.4 |*******************************
   2.800  6446    3.1   61.5 |******************************
   2.900  6019    2.9   64.4 |****************************
   3.000  5444    2.6   67.1 |*************************
   3.100  5286    2.6   69.6 |*************************
   3.200  4803    2.3   72.0 |**********************
   3.300  4455    2.2   74.1 |********************
   3.400  4318    2.1   76.2 |********************
   3.500  3920    1.9   78.1 |******************
   3.600  3682    1.8   79.9 |*****************
   3.700  3491    1.7   81.6 |****************
   3.800  3215    1.6   83.1 |***************
   3.900  2902    1.4   84.5 |*************
   4.000  2748    1.3   85.9 |************
   4.100  2628    1.3   87.1 |************
   4.200  2420    1.2   88.3 |***********
   4.300  2169    1.1   89.4 |*********
```

4.400	1918	0.9	90.3	********
4.500	1784	0.9	91.2	********
4.600	1651	0.8	92.0	*******
4.700	1531	0.7	92.7	******
4.800	1359	0.7	93.4	******
4.900	1230	0.6	94.0	*****
5.000	1173	0.6	94.5	*****
5.100	1045	0.5	95.0	****
5.200	956	0.5	95.5	****
5.300	853	0.4	95.9	***
5.400	791	0.4	96.3	***
5.500	711	0.3	96.6	***
5.600	626	0.3	96.9	**
5.700	581	0.3	97.2	**
5.800	513	0.2	97.5	**
5.900	469	0.2	97.7	**
6.000	428	0.2	97.9	*
6.100	374	0.2	98.1	*
6.200	382	0.2	98.3	*
6.300	318	0.2	98.4	*
6.400	297	0.1	98.6	*
6.500	258	0.1	98.7	*
6.600	226	0.1	98.8	*
6.700	230	0.1	98.9	*
6.800	200	0.1	99.0	
6.900	182	0.1	99.1	
7.000	146	0.1	99.2	
7.100	131	0.1	99.2	
7.200	109	0.1	99.3	
7.300	116	0.1	99.3	
7.400	104	0.1	99.4	
7.500	102	0.0	99.4	
7.600	85	0.0	99.5	
7.700	73	0.0	99.5	
7.800	67	0.0	99.6	
7.900	65	0.0	99.6	
8.000	60	0.0	99.6	
8.100	62	0.0	99.6	
8.200	35	0.0	99.7	
8.300	64	0.0	99.7	
8.400	40	0.0	99.7	
8.500	24	0.0	99.7	
8.600	58	0.0	99.8	
8.700	31	0.0	99.8	
8.800	27	0.0	99.8	
8.900	23	0.0	99.8	
9.000	29	0.0	99.8	
9.100	22	0.0	99.8	
9.200	22	0.0	99.8	
9.300	22	0.0	99.8	
9.400	28	0.0	99.9	
9.500	23	0.0	99.9	
9.600	14	0.0	99.9	
9.700	12	0.0	99.9	

9.800	8	0.0	99.9
9.900	9	0.0	99.9
10.000	10	0.0	99.9
10.100	5	0.0	99.9
10.200	9	0.0	99.9
10.300	6	0.0	99.9
10.400	10	0.0	99.9
10.500	12	0.0	99.9
10.600	6	0.0	99.9
10.700	5	0.0	99.9
10.800	8	0.0	99.9
10.900	6	0.0	99.9
11.000	4	0.0	99.9
11.100	3	0.0	99.9
11.200	7	0.0	99.9
11.300	3	0.0	99.9
11.400	5	0.0	99.9
11.500	1	0.0	99.9
11.600	2	0.0	99.9
11.700	4	0.0	99.9
11.800	3	0.0	99.9
11.900	1	0.0	99.9
12.000	3	0.0	99.9
12.100	1	0.0	99.9
12.200	6	0.0	99.9
12.300	1	0.0	99.9
12.400	4	0.0	99.9
12.500	3	0.0	99.9
12.600	1	0.0	99.9
12.700	2	0.0	99.9
12.800	4	0.0	99.9
12.900	1	0.0	99.9
13.000	2	0.0	100.0
13.100	3	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	2	0.0	100.0
13.500	1	0.0	100.0
13.600	0	0.0	100.0
13.700	1	0.0	100.0
13.800	1	0.0	100.0
13.900	2	0.0	100.0
14.000	2	0.0	100.0
14.100	3	0.0	100.0
14.200	1	0.0	100.0
14.300	1	0.0	100.0
14.400	3	0.0	100.0
14.500	4	0.0	100.0
14.600	2	0.0	100.0
14.700	3	0.0	100.0
14.800	0	0.0	100.0
14.900	1	0.0	100.0
15.000	0	0.0	100.0
15.100	1	0.0	100.0

15.200	3	0.0	100.0
15.300	2	0.0	100.0
15.400	0	0.0	100.0
15.500	0	0.0	100.0
15.600	2	0.0	100.0
15.700	2	0.0	100.0
15.800	0	0.0	100.0
15.900	1	0.0	100.0
16.000	3	0.0	100.0
16.100	0	0.0	100.0
16.200	1	0.0	100.0
16.300	0	0.0	100.0
16.400	0	0.0	100.0
16.500	0	0.0	100.0
16.600	3	0.0	100.0
16.700	4	0.0	100.0
16.800	1	0.0	100.0
16.900	1	0.0	100.0
17.000	3	0.0	100.0
17.100	0	0.0	100.0
17.200	3	0.0	100.0
17.300	2	0.0	100.0
17.400	2	0.0	100.0
17.500	1	0.0	100.0
17.600	0	0.0	100.0
17.700	0	0.0	100.0
17.800	1	0.0	100.0
17.900	1	0.0	100.0
18.000	0	0.0	100.0
18.100	0	0.0	100.0
18.200	2	0.0	100.0
18.300	0	0.0	100.0
18.400	0	0.0	100.0
18.500	0	0.0	100.0
18.600	0	0.0	100.0
18.700	3	0.0	100.0
18.800	0	0.0	100.0
18.900	1	0.0	100.0
19.000	0	0.0	100.0
19.100	0	0.0	100.0
19.200	2	0.0	100.0
19.300	2	0.0	100.0
19.400	0	0.0	100.0
19.500	0	0.0	100.0
19.600	0	0.0	100.0
19.700	0	0.0	100.0
19.800	0	0.0	100.0
19.900	0	0.0	100.0
20.000	0	0.0	100.0
20.100	1	0.0	100.0
20.200	0	0.0	100.0
20.300	1	0.0	100.0
20.400	0	0.0	100.0
20.500	0	0.0	100.0

20.600	0	0.0	100.0
20.700	0	0.0	100.0
20.800	0	0.0	100.0
20.900	1	0.0	100.0
21.000	0	0.0	100.0
21.100	0	0.0	100.0
21.200	0	0.0	100.0
21.300	0	0.0	100.0
21.400	1	0.0	100.0
21.500	0	0.0	100.0
21.600	0	0.0	100.0
21.700	0	0.0	100.0
21.800	2	0.0	100.0
21.900	0	0.0	100.0
22.000	0	0.0	100.0
22.100	0	0.0	100.0
22.200	0	0.0	100.0
22.300	1	0.0	100.0
22.400	0	0.0	100.0
22.500	0	0.0	100.0
22.600	0	0.0	100.0
22.700	2	0.0	100.0
22.800	0	0.0	100.0
22.900	0	0.0	100.0
23.000	0	0.0	100.0
23.100	0	0.0	100.0
23.200	0	0.0	100.0
23.300	0	0.0	100.0
23.400	0	0.0	100.0
23.500	0	0.0	100.0
23.600	0	0.0	100.0
23.700	1	0.0	100.0
23.800	0	0.0	100.0
23.900	0	0.0	100.0
24.000	0	0.0	100.0
24.100	0	0.0	100.0
24.200	0	0.0	100.0
24.300	0	0.0	100.0
24.400	0	0.0	100.0
24.500	1	0.0	100.0
24.600	0	0.0	100.0
24.700	0	0.0	100.0
24.800	0	0.0	100.0
24.900	0	0.0	100.0
25.000	0	0.0	100.0
25.100	0	0.0	100.0
25.200	0	0.0	100.0
25.300	0	0.0	100.0
25.400	0	0.0	100.0
25.500	0	0.0	100.0
25.600	0	0.0	100.0
25.700	0	0.0	100.0
25.800	0	0.0	100.0
25.900	0	0.0	100.0

26.000	0	0.0	100.0
26.100	0	0.0	100.0
26.200	0	0.0	100.0
26.300	0	0.0	100.0
26.400	0	0.0	100.0
26.500	0	0.0	100.0
26.600	0	0.0	100.0
26.700	0	0.0	100.0
26.800	0	0.0	100.0
26.900	0	0.0	100.0
27.000	0	0.0	100.0
27.100	0	0.0	100.0
27.200	0	0.0	100.0
27.300	0	0.0	100.0
27.400	0	0.0	100.0
27.500	0	0.0	100.0
27.600	1	0.0	100.0
27.700	0	0.0	100.0
27.800	0	0.0	100.0
27.900	0	0.0	100.0
28.000	0	0.0	100.0
28.100	0	0.0	100.0
28.200	0	0.0	100.0
28.300	0	0.0	100.0
28.400	0	0.0	100.0
28.500	0	0.0	100.0
28.600	1	0.0	100.0
28.700	0	0.0	100.0
28.800	0	0.0	100.0
28.900	0	0.0	100.0
29.000	0	0.0	100.0
29.100	0	0.0	100.0
29.200	0	0.0	100.0
29.300	0	0.0	100.0
29.400	0	0.0	100.0
29.500	0	0.0	100.0
29.600	0	0.0	100.0
29.700	0	0.0	100.0
29.800	0	0.0	100.0
29.900	0	0.0	100.0
30.000	0	0.0	100.0
30.100	0	0.0	100.0
30.200	0	0.0	100.0
30.300	0	0.0	100.0
30.400	0	0.0	100.0
30.500	0	0.0	100.0
30.600	0	0.0	100.0
30.700	0	0.0	100.0
30.800	0	0.0	100.0
30.900	0	0.0	100.0
31.000	0	0.0	100.0
31.100	0	0.0	100.0
31.200	0	0.0	100.0
31.300	0	0.0	100.0

31.400	0	0.0	100.0
31.500	0	0.0	100.0
31.600	0	0.0	100.0
31.700	0	0.0	100.0
31.800	0	0.0	100.0
31.900	0	0.0	100.0
32.000	0	0.0	100.0
32.100	0	0.0	100.0
32.200	0	0.0	100.0
32.300	0	0.0	100.0
32.400	0	0.0	100.0
32.500	0	0.0	100.0
32.600	0	0.0	100.0
32.700	0	0.0	100.0
32.800	0	0.0	100.0
32.900	0	0.0	100.0
33.000	0	0.0	100.0
33.100	0	0.0	100.0
33.200	1	0.0	100.0

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
9.842 s	9.094 s	0.671 s	142.495 s	206418	229.353 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	173	0.1	8.6	| T
2.000	11749	5.7	7.8	|****************** T
3.000	25651	12.4	7.2	|**********T**********
4.000	23813	11.5	6.5	|**********T**********
5.000	18788	9.1	6.0	|**********T*********
6.000	14856	7.2	5.5	|**********T*****
7.000	11985	5.8	5.0	|**********T**
8.000	10409	5.0	4.6	|**********T**
9.000	9290	4.5	4.2	|**********T*
10.000	8008	3.9	3.8	|**********T
11.000	7182	3.5	3.5	|**********T
12.000	6599	3.2	3.2	|**********T
13.000	5782	2.8	2.9	|*********T
14.000	5429	2.6	2.7	|********T
15.000	4757	2.3	2.4	|*******T
16.000	4242	2.1	2.2	|****** T
17.000	3781	1.8	2.0	|******T
18.000	3385	1.6	1.9	|*****T
19.000	3201	1.6	1.7	|*****T
20.000	2613	1.3	1.6	|****T
21.000	2586	1.3	1.4	|****T
22.000	2254	1.1	1.3	|***T
23.000	2021	1.0	1.2	|***T
24.000	1900	0.9	1.1	|***T
25.000	1668	0.8	1.0	|**T
26.000	1375	0.7	0.9	|**T
27.000	1344	0.7	0.8	|**T
28.000	1148	0.6	0.8	|* T
29.000	1031	0.5	0.7	|*T
30.000	927	0.4	0.6	|*T
31.000	894	0.4	0.6	|*T
32.000	791	0.4	0.5	|*T
33.000	728	0.4	0.5	|*T
34.000	602	0.3	0.4	| T
35.000	564	0.3	0.4	|T
36.000	514	0.2	0.4	|T
37.000	458	0.2	0.3	|T
38.000	417	0.2	0.3	|T
39.000	345	0.2	0.3	|T
40.000	317	0.2	0.3	|T
41.000	239	0.1	0.2	|T
42.000	257	0.1	0.2	|T
43.000	198	0.1	0.2	|T
44.000	235	0.1	0.2	|T

2)

45.000	144	0.1	0.2	T
46.000	163	0.1	0.2	T
47.000	175	0.1	0.1	
48.000	158	0.1	0.1	
49.000	137	0.1	0.1	
50.000	142	0.1	0.1	
51.000	137	0.1	0.1	
52.000	59	0.0	0.1	
53.000	62	0.0	0.1	
54.000	81	0.0	0.1	
55.000	72	0.0	0.1	
56.000	45	0.0	0.1	
57.000	53	0.0	0.1	
58.000	44	0.0	0.1	
59.000	29	0.0	0.0	
60.000	46	0.0	0.0	
61.000	31	0.0	0.0	
62.000	26	0.0	0.0	
63.000	16	0.0	0.0	
64.000	39	0.0	0.0	
65.000	14	0.0	0.0	
66.000	35	0.0	0.0	
67.000	7	0.0	0.0	
68.000	17	0.0	0.0	
69.000	19	0.0	0.0	
70.000	16	0.0	0.0	
71.000	16	0.0	0.0	
72.000	7	0.0	0.0	
73.000	9	0.0	0.0	
74.000	8	0.0	0.0	
75.000	8	0.0	0.0	
76.000	20	0.0	0.0	
77.000	2	0.0	0.0	
78.000	3	0.0	0.0	
79.000	0	0.0	0.0	
80.000	2	0.0	0.0	
81.000	8	0.0	0.0	
82.000	12	0.0	0.0	
83.000	5	0.0	0.0	
84.000	4	0.0	0.0	
85.000	4	0.0	0.0	
86.000	0	0.0	0.0	
87.000	0	0.0	0.0	
88.000	5	0.0	0.0	
89.000	0	0.0	0.0	
90.000	0	0.0	0.0	
91.000	2	0.0	0.0	
92.000	4	0.0	0.0	
93.000	3	0.0	0.0	
94.000	4	0.0	0.0	
95.000	0	0.0	0.0	
96.000	0	0.0	0.0	
97.000	4	0.0	0.0	
98.000	0	0.0	0.0	

99.000	0	0.0	0.0
100.000	0	0.0	0.0
101.000	1	0.0	0.0
102.000	1	0.0	0.0
103.000	1	0.0	0.0
104.000	0	0.0	0.0
105.000	1	0.0	0.0
106.000	0	0.0	0.0
107.000	1	0.0	0.0
108.000	0	0.0	0.0
109.000	0	0.0	0.0
110.000	0	0.0	0.0
111.000	0	0.0	0.0
112.000	4	0.0	0.0
113.000	1	0.0	0.0
114.000	0	0.0	0.0
115.000	4	0.0	0.0
116.000	0	0.0	0.0
117.000	0	0.0	0.0
118.000	0	0.0	0.0
119.000	0	0.0	0.0
120.000	0	0.0	0.0
121.000	0	0.0	0.0
122.000	0	0.0	0.0
123.000	0	0.0	0.0
124.000	0	0.0	0.0
125.000	0	0.0	0.0
126.000	0	0.0	0.0
127.000	0	0.0	0.0
128.000	0	0.0	0.0
129.000	0	0.0	0.0
130.000	0	0.0	0.0
131.000	0	0.0	0.0
132.000	0	0.0	0.0
133.000	0	0.0	0.0
134.000	0	0.0	0.0
135.000	0	0.0	0.0
136.000	0	0.0	0.0
137.000	0	0.0	0.0
138.000	0	0.0	0.0
139.000	0	0.0	0.0
140.000	0	0.0	0.0
141.000	0	0.0	0.0
142.000	0	0.0	0.0
143.000	1	0.0	0.0

Simulator Report

NonStop/SQL

4 * 8 VLX SYSTEMS

Throughput - 220 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                 =   Test ALL;RUN05-32.5;SQL
Test Date                  =   87-02-19
Test Length                =   18:06:59 - 18:32:13  (1513 sec)
Window Length              =   18:15:00 - 18:30:00  (900 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TLOGSUMZ

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    7.500 tx/sec
Tx Arrival Rate/Terminal   =    0.094 tx/sec
Tx Inter-Arrival Time      =   10.667 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) ---------------------------------

Tx Throughput              =  220.530 tx/sec
Tx Response Time Average   =    1.666 sec
                 Std Dev   =    0.649 sec
                 Minimum   =    0.405 sec
                 Maximum   =   58.224 sec
                 Count     =   198477 #

Tx Response Time 10%       =    1.050 sec  ( 11.1%)
                 20%       =    1.200 sec  ( 22.0%)
                 30%       =    1.300 sec  ( 30.1%)
                 40%       =    1.450 sec  ( 42.9%)
                 50%       =    1.550 sec  ( 50.7%)
                 60%       =    1.700 sec  ( 61.2%)
                 70%       =    1.900 sec  ( 72.3%)
                 80%       =    2.100 sec  ( 80.5%)
                 90%       =    2.500 sec  ( 90.9%)
                 95%       =    2.800 sec  ( 95.0%)

        Tx Think Average   =   10.655 sec
                 Std Dev   =   10.644 sec
                 Minimum   =    0.001 sec
                 Maximum   =  128.159 sec
                 Count     =   170328 #    ( 85.8%)

        Tx Delay Average   =    0.967 sec
                 Std Dev   =    0.842 sec
                 Minimum   =    0.000 sec
                 Maximum   =   52.599 sec
                 Count     =    28149 #    ( 14.2%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.666 s	0.649 s	0.405 s	58.224 s	198477	220.530 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	0	0.0	0.0	|
0.500	20	0.0	0.0	|
0.600	166	0.1	0.1	|
0.700	720	0.4	0.5	|*
0.800	2268	1.1	1.6	|*****
0.900	5183	2.6	4.2	|**********
1.000	8178	4.1	8.3	|********************
1.100	12151	6.1	14.5	|******************************
1.200	14881	7.5	22.0	|************************************
1.300	16243	8.2	30.1	|**
1.400	17165	8.6	38.8	|**
1.500	16193	8.2	46.9	|**
1.600	14818	7.5	54.4	|************************************
1.700	13497	6.8	61.2	|*********************************
1.800	11763	5.9	67.1	|*****************************
1.900	10271	5.2	72.3	|*************************
2.000	8750	4.4	76.7	|*********************
2.100	7519	3.8	80.5	|******************
2.200	6530	3.3	83.8	|****************
2.300	5510	2.8	86.6	|*************
2.400	4702	2.4	88.9	|***********
2.500	3897	2.0	90.9	|*********
2.600	3277	1.7	92.6	|********
2.700	2690	1.4	93.9	|******
2.800	2219	1.1	95.0	|*****
2.900	1802	0.9	95.9	|****
3.000	1497	0.8	96.7	|***
3.100	1211	0.6	97.3	|**
3.200	948	0.5	97.8	|**
3.300	786	0.4	98.2	|*
3.400	660	0.3	98.5	|*
3.500	509	0.3	98.8	|*
3.600	378	0.2	99.0	|
3.700	334	0.2	99.1	|
3.800	270	0.1	99.3	|
3.900	210	0.1	99.4	|
4.000	185	0.1	99.5	|
4.100	151	0.1	99.5	|
4.200	117	0.1	99.6	|
4.300	101	0.1	99.6	|

4.400	95	0.0	99.7
4.500	98	0.0	99.7
4.600	70	0.0	99.8
4.700	58	0.0	99.8
4.800	42	0.0	99.8
4.900	37	0.0	99.8
5.000	33	0.0	99.9
5.100	34	0.0	99.9
5.200	22	0.0	99.9
5.300	25	0.0	99.9
5.400	21	0.0	99.9
5.500	11	0.0	99.9
5.600	18	0.0	99.9
5.700	9	0.0	99.9
5.800	13	0.0	99.9
5.900	14	0.0	99.9
6.000	10	0.0	100.0
6.100	5	0.0	100.0
6.200	10	0.0	100.0
6.300	5	0.0	100.0
6.400	7	0.0	100.0
6.500	1	0.0	100.0
6.600	6	0.0	100.0
6.700	2	0.0	100.0
6.800	4	0.0	100.0
6.900	7	0.0	100.0
7.000	4	0.0	100.0
7.100	5	0.0	100.0
7.200	2	0.0	100.0
7.300	2	0.0	100.0
7.400	2	0.0	100.0
7.500	2	0.0	100.0
7.600	0	0.0	100.0
7.700	2	0.0	100.0
7.800	2	0.0	100.0
7.900	2	0.0	100.0
8.000	0	0.0	100.0
8.100	2	0.0	100.0
8.200	1	0.0	100.0
8.300	0	0.0	100.0
8.400	1	0.0	100.0
8.500	1	0.0	100.0
8.600	1	0.0	100.0
8.700	3	0.0	100.0
8.800	0	0.0	100.0
8.900	1	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	1	0.0	100.0
9.500	1	0.0	100.0
9.600	2	0.0	100.0
9.700	0	0.0	100.0

9.800	0	0.0	100.0
9.900	1	0.0	100.0
10.000	0	0.0	100.0
10.100	1	0.0	100.0
10.200	0	0.0	100.0
10.300	0	0.0	100.0
10.400	0	0.0	100.0
10.500	0	0.0	100.0
10.600	0	0.0	100.0
10.700	1	0.0	100.0
10.800	0	0.0	100.0
10.900	1	0.0	100.0
11.000	0	0.0	100.0
11.100	1	0.0	100.0
11.200	0	0.0	100.0
11.300	0	0.0	100.0
11.400	1	0.0	100.0
11.500	0	0.0	100.0
11.600	0	0.0	100.0
11.700	0	0.0	100.0
11.800	0	0.0	100.0
11.900	1	0.0	100.0
12.000	0	0.0	100.0
12.100	0	0.0	100.0
12.200	0	0.0	100.0
12.300	1	0.0	100.0
12.400	0	0.0	100.0
12.500	0	0.0	100.0
12.600	0	0.0	100.0
12.700	0	0.0	100.0
12.800	0	0.0	100.0
12.900	1	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	1	0.0	100.0

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
10.809 s	10.527 s	0.446 s	129.222 s	198477	220.530 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	1456	0.7	8.3	|** T
2.000	24458	12.3	7.6	|**********T**********
3.000	20911	10.5	6.9	|**********T**********
4.000	14851	7.5	6.4	|**********T***
5.000	12359	6.2	5.8	|**********T*
6.000	11025	5.6	5.4	|**********T
7.000	10048	5.1	4.9	|**********T
8.000	9113	4.6	4.5	|**********T
9.000	8496	4.3	4.1	|**********T
10.000	7777	3.9	3.8	|**********T
11.000	6920	3.5	3.5	|**********T
12.000	6321	3.2	3.2	|**********T
13.000	5637	2.8	2.9	|*********T
14.000	5508	2.8	2.7	|********T
15.000	4820	2.4	2.5	|*******T
16.000	4603	2.3	2.3	|*******T
17.000	3912	2.0	2.1	|******T
18.000	3626	1.8	1.9	|******T
19.000	3270	1.6	1.8	|*****T
20.000	2889	1.5	1.6	|****T
21.000	2772	1.4	1.5	|****T
22.000	2546	1.3	1.4	|****T
23.000	2246	1.1	1.2	|***T
24.000	2057	1.0	1.1	|***T
25.000	1927	1.0	1.0	|***T
26.000	1716	0.9	1.0	|**T
27.000	1589	0.8	0.9	|**T
28.000	1472	0.7	0.8	|**T
29.000	1201	0.6	0.7	|**T
30.000	1172	0.6	0.7	|*T
31.000	1114	0.6	0.6	|*T
32.000	904	0.5	0.6	|*T
33.000	851	0.4	0.5	|*T
34.000	764	0.4	0.5	|*T
35.000	758	0.4	0.4	|*T
36.000	721	0.4	0.4	|T
37.000	602	0.3	0.4	|T
38.000	575	0.3	0.3	|T
39.000	469	0.2	0.3	|T
40.000	481	0.2	0.3	|T
41.000	395	0.2	0.3	|T
42.000	398	0.2	0.2	|T
43.000	369	0.2	0.2	|T
44.000	285	0.1	0.2	|T

45.000	305	0.2	0.2	T
46.000	195	0.1	0.2	T
47.000	220	0.1	0.2	T
48.000	205	0.1	0.1	
49.000	174	0.1	0.1	
50.000	177	0.1	0.1	
51.000	124	0.1	0.1	
52.000	121	0.1	0.1	
53.000	168	0.1	0.1	
54.000	124	0.1	0.1	
55.000	142	0.1	0.1	
56.000	107	0.1	0.1	
57.000	112	0.1	0.1	
58.000	107	0.1	0.1	
59.000	58	0.0	0.1	
60.000	45	0.0	0.1	
61.000	66	0.0	0.0	
62.000	68	0.0	0.0	
63.000	53	0.0	0.0	
64.000	36	0.0	0.0	
65.000	39	0.0	0.0	
66.000	33	0.0	0.0	
67.000	29	0.0	0.0	
68.000	36	0.0	0.0	
69.000	23	0.0	0.0	
70.000	25	0.0	0.0	
71.000	21	0.0	0.0	
72.000	17	0.0	0.0	
73.000	26	0.0	0.0	
74.000	32	0.0	0.0	
75.000	15	0.0	0.0	
76.000	7	0.0	0.0	
77.000	17	0.0	0.0	
78.000	9	0.0	0.0	
79.000	21	0.0	0.0	
80.000	4	0.0	0.0	
81.000	21	0.0	0.0	
82.000	5	0.0	0.0	
83.000	8	0.0	0.0	
84.000	4	0.0	0.0	
85.000	8	0.0	0.0	
86.000	15	0.0	0.0	
87.000	4	0.0	0.0	
88.000	1	0.0	0.0	
89.000	2	0.0	0.0	
90.000	2	0.0	0.0	
91.000	0	0.0	0.0	
92.000	15	0.0	0.0	
93.000	4	0.0	0.0	
94.000	5	0.0	0.0	
95.000	4	0.0	0.0	
96.000	5	0.0	0.0	
97.000	0	0.0	0.0	
98.000	0	0.0	0.0	

99.000	5	0.0	0.0
100.000	0	0.0	0.0
101.000	0	0.0	0.0
102.000	0	0.0	0.0
103.000	1	0.0	0.0
104.000	4	0.0	0.0
105.000	0	0.0	0.0
106.000	3	0.0	0.0
107.000	4	0.0	0.0
108.000	0	0.0	0.0
109.000	0	0.0	0.0
110.000	4	0.0	0.0
111.000	0	0.0	0.0
112.000	0	0.0	0.0
113.000	0	0.0	0.0
114.000	0	0.0	0.0
115.000	0	0.0	0.0
116.000	0	0.0	0.0
117.000	0	0.0	0.0
118.000	0	0.0	0.0
119.000	0	0.0	0.0
120.000	0	0.0	0.0
121.000	0	0.0	0.0
122.000	0	0.0	0.0
123.000	0	0.0	0.0
124.000	0	0.0	0.0
125.000	0	0.0	0.0
126.000	0	0.0	0.0
127.000	4	0.0	0.0
128.000	1	0.0	0.0
129.000	0	0.0	0.0
130.000	3	0.0	0.0

Simulator Report

NonStop/SQL

4 * 8 VLX SYSTEMS

Throughput - 206 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test ALL;RUN02-32.5;SQL
Test Date                 =   87-02-19
Test Length               =   14:01:49 - 14:42:11  (2421 sec)
Window Length             =   14:21:00 - 14:36:00  (900 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TLOGSUMZ

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    6.500 tx/sec
Tx Arrival Rate/Terminal  =    0.081 tx/sec
Tx Inter-Arrival Time     =   12.308 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) ---------------------------------

Tx Throughput                =  206.040 tx/sec
Tx Response Time Average     =    1.271 sec
                 Std Dev     =    0.485 sec
                 Minimum     =    0.335 sec
                 Maximum     =   42.176 sec
                 Count       =   185436 #

Tx Response Time 10%         =    0.800 sec  ( 10.4%)
                 20%         =    0.950 sec  ( 25.2%)
                 30%         =    1.000 sec  ( 31.1%)
                 40%         =    1.100 sec  ( 43.1%)
                 50%         =    1.200 sec  ( 53.8%)
                 60%         =    1.300 sec  ( 62.7%)
                 70%         =    1.400 sec  ( 70.1%)
                 80%         =    1.600 sec  ( 80.8%)
                 90%         =    1.900 sec  ( 90.5%)
                 95%         =    2.200 sec  ( 95.5%)

        Tx Think Average     =   12.290 sec
                 Std Dev     =   12.222 sec
                 Minimum     =    0.001 sec
                 Maximum     =  147.797 sec
                 Count       =   167514 #    ( 90.3%)

        Tx Delay Average     =    0.737 sec
                 Std Dev     =    0.602 sec
                 Minimum     =    0.000 sec
                 Maximum     =   36.932 sec
                 Count       =    17922 #    (  9.7%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.271 s	0.485 s	0.335 s	42.176 s	185436	206.040 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	
0.100	0	0.0	0.0	
0.200	0	0.0	0.0	
0.300	0	0.0	0.0	
0.400	6	0.0	0.0	
0.500	245	0.1	0.1	
0.600	1801	1.0	1.1	***
0.700	5879	3.2	4.3	**********
0.800	11351	6.1	10.4	**********
0.900	17195	9.3	19.7	**********
1.000	21182	11.4	31.1	**********
1.100	22220	12.0	43.1	**********
1.200	19855	10.7	53.8	**********
1.300	16576	8.9	62.7	**********
1.400	13598	7.3	70.1	**********
1.500	11054	6.0	76.0	**********
1.600	8813	4.8	80.8	**********
1.700	7181	3.9	84.6	**********
1.800	5987	3.2	87.9	**********
1.900	4797	2.6	90.5	*********
2.000	3834	2.1	92.5	*******
2.100	3049	1.6	94.2	*****
2.200	2439	1.3	95.5	****
2.300	1928	1.0	96.5	***
2.400	1403	0.8	97.3	**
2.500	1047	0.6	97.8	*
2.600	851	0.5	98.3	*
2.700	640	0.3	98.6	*
2.800	469	0.3	98.9	
2.900	392	0.2	99.1	
3.000	295	0.2	99.3	
3.100	253	0.1	99.4	
3.200	201	0.1	99.5	
3.300	155	0.1	99.6	
3.400	118	0.1	99.7	
3.500	100	0.1	99.7	
3.600	61	0.0	99.8	
3.700	75	0.0	99.8	
3.800	62	0.0	99.8	
3.900	42	0.0	99.8	
4.000	35	0.0	99.9	
4.100	34	0.0	99.9	
4.200	24	0.0	99.9	
4.300	24	0.0	99.9	

4.400	21	0.0	99.9
4.500	15	0.0	99.9
4.600	15	0.0	99.9
4.700	15	0.0	99.9
4.800	11	0.0	100.0
4.900	9	0.0	100.0
5.000	5	0.0	100.0
5.100	5	0.0	100.0
5.200	6	0.0	100.0
5.300	5	0.0	100.0
5.400	1	0.0	100.0
5.500	6	0.0	100.0
5.600	4	0.0	100.0
5.700	5	0.0	100.0
5.800	6	0.0	100.0
5.900	5	0.0	100.0
6.000	6	0.0	100.0
6.100	1	0.0	100.0
6.200	3	0.0	100.0
6.300	1	0.0	100.0
6.400	2	0.0	100.0
6.500	1	0.0	100.0
6.600	1	0.0	100.0
6.700	1	0.0	100.0
6.800	1	0.0	100.0
6.900	1	0.0	100.0
7.000	0	0.0	100.0
7.100	1	0.0	100.0
7.200	0	0.0	100.0
7.300	0	0.0	100.0
7.400	0	0.0	100.0
7.500	1	0.0	100.0
7.600	1	0.0	100.0
7.700	1	0.0	100.0
7.800	1	0.0	100.0
7.900	0	0.0	100.0
8.000	0	0.0	100.0
8.100	0	0.0	100.0
8.200	1	0.0	100.0
8.300	0	0.0	100.0
8.400	0	0.0	100.0
8.500	1	0.0	100.0
8.600	0	0.0	100.0
8.700	0	0.0	100.0
8.800	0	0.0	100.0
8.900	0	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	1	0.0	100.0
9.500	0	0.0	100.0
9.600	1	0.0	100.0
9.700	0	0.0	100.0

9.800	0	0.0	100.0
9.900	1	0.0	100.0
10.000	0	0.0	100.0
10.100	0	0.0	100.0
10.200	0	0.0	100.0
10.300	0	0.0	100.0
10.400	2	0.0	100.0
10.500	0	0.0	100.0
10.600	0	0.0	100.0
10.700	0	0.0	100.0
10.800	0	0.0	100.0
10.900	0	0.0	100.0
11.000	0	0.0	100.0
11.100	0	0.0	100.0
11.200	0	0.0	100.0
11.300	0	0.0	100.0
11.400	0	0.0	100.0
11.500	0	0.0	100.0
11.600	0	0.0	100.0
11.700	0	0.0	100.0
11.800	0	0.0	100.0
11.900	0	0.0	100.0
12.000	0	0.0	100.0
12.100	0	0.0	100.0
12.200	0	0.0	100.0
12.300	0	0.0	100.0
12.400	0	0.0	100.0
12.500	0	0.0	100.0
12.600	0	0.0	100.0
12.700	0	0.0	100.0
12.800	0	0.0	100.0
12.900	0	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	0	0.0	100.0
13.500	0	0.0	100.0
13.600	0	0.0	100.0
13.700	0	0.0	100.0
13.800	0	0.0	100.0
13.900	0	0.0	100.0
14.000	0	0.0	100.0
14.100	0	0.0	100.0
14.200	0	0.0	100.0
14.300	0	0.0	100.0
14.400	0	0.0	100.0
14.500	0	0.0	100.0
14.600	0	0.0	100.0
14.700	0	0.0	100.0
14.800	0	0.0	100.0
14.900	0	0.0	100.0
15.000	0	0.0	100.0
15.100	0	0.0	100.0

15.200	0	0.0	100.0
15.300	0	0.0	100.0
15.400	0	0.0	100.0
15.500	0	0.0	100.0
15.600	0	0.0	100.0
15.700	0	0.0	100.0
15.800	0	0.0	100.0
15.900	0	0.0	100.0
16.000	0	0.0	100.0
16.100	0	0.0	100.0
16.200	0	0.0	100.0
16.300	0	0.0	100.0
16.400	0	0.0	100.0
16.500	0	0.0	100.0
16.600	0	0.0	100.0
16.700	0	0.0	100.0
16.800	0	0.0	100.0
16.900	0	0.0	100.0
17.000	0	0.0	100.0
17.100	0	0.0	100.0
17.200	0	0.0	100.0
17.300	0	0.0	100.0
17.400	0	0.0	100.0
17.500	0	0.0	100.0
17.600	0	0.0	100.0
17.700	0	0.0	100.0
17.800	0	0.0	100.0
17.900	0	0.0	100.0
18.000	0	0.0	100.0
18.100	0	0.0	100.0
18.200	0	0.0	100.0
18.300	0	0.0	100.0
18.400	0	0.0	100.0
18.500	0	0.0	100.0
18.600	0	0.0	100.0
18.700	0	0.0	100.0
18.800	0	0.0	100.0
18.900	0	0.0	100.0
19.000	0	0.0	100.0
19.100	0	0.0	100.0
19.200	0	0.0	100.0
19.300	0	0.0	100.0
19.400	0	0.0	100.0
19.500	0	0.0	100.0
19.600	0	0.0	100.0
19.700	0	0.0	100.0
19.800	0	0.0	100.0
19.900	0	0.0	100.0
20.000	0	0.0	100.0
20.100	0	0.0	100.0
20.200	0	0.0	100.0
20.300	0	0.0	100.0
20.400	0	0.0	100.0
20.500	0	0.0	100.0

20.600	0	0.0	100.0
20.700	0	0.0	100.0
20.800	0	0.0	100.0
20.900	0	0.0	100.0
21.000	0	0.0	100.0
21.100	0	0.0	100.0
21.200	0	0.0	100.0
21.300	0	0.0	100.0
21.400	0	0.0	100.0
21.500	0	0.0	100.0
21.600	0	0.0	100.0
21.700	0	0.0	100.0
21.800	0	0.0	100.0
21.900	0	0.0	100.0
22.000	0	0.0	100.0
22.100	0	0.0	100.0
22.200	0	0.0	100.0
22.300	0	0.0	100.0
22.400	0	0.0	100.0
22.500	0	0.0	100.0
22.600	0	0.0	100.0
22.700	0	0.0	100.0
22.800	0	0.0	100.0
22.900	0	0.0	100.0
23.000	0	0.0	100.0
23.100	0	0.0	100.0
23.200	0	0.0	100.0
23.300	0	0.0	100.0
23.400	0	0.0	100.0
23.500	0	0.0	100.0
23.600	0	0.0	100.0
23.700	0	0.0	100.0
23.800	0	0.0	100.0
23.900	0	0.0	100.0
24.000	0	0.0	100.0
24.100	0	0.0	100.0
24.200	0	0.0	100.0
24.300	0	0.0	100.0
24.400	0	0.0	100.0
24.500	0	0.0	100.0
24.600	0	0.0	100.0
24.700	0	0.0	100.0
24.800	0	0.0	100.0
24.900	0	0.0	100.0
25.000	0	0.0	100.0
25.100	0	0.0	100.0
25.200	0	0.0	100.0
25.300	0	0.0	100.0
25.400	0	0.0	100.0
25.500	0	0.0	100.0
25.600	0	0.0	100.0
25.700	0	0.0	100.0
25.800	0	0.0	100.0
25.900	0	0.0	100.0

26.000	0	0.0	100.0
26.100	0	0.0	100.0
26.200	0	0.0	100.0
26.300	0	0.0	100.0
26.400	0	0.0	100.0
26.500	0	0.0	100.0
26.600	0	0.0	100.0
26.700	0	0.0	100.0
26.800	0	0.0	100.0
26.900	0	0.0	100.0
27.000	0	0.0	100.0
27.100	0	0.0	100.0
27.200	0	0.0	100.0
27.300	0	0.0	100.0
27.400	0	0.0	100.0
27.500	0	0.0	100.0
27.600	0	0.0	100.0
27.700	0	0.0	100.0
27.800	0	0.0	100.0
27.900	0	0.0	100.0
28.000	0	0.0	100.0
28.100	0	0.0	100.0
28.200	0	0.0	100.0
28.300	0	0.0	100.0
28.400	0	0.0	100.0
28.500	0	0.0	100.0
28.600	0	0.0	100.0
28.700	0	0.0	100.0
28.800	0	0.0	100.0
28.900	0	0.0	100.0
29.000	0	0.0	100.0
29.100	0	0.0	100.0
29.200	0	0.0	100.0
29.300	0	0.0	100.0
29.400	0	0.0	100.0
29.500	0	0.0	100.0
29.600	0	0.0	100.0
29.700	0	0.0	100.0
29.800	0	0.0	100.0
29.900	0	0.0	100.0
30.000	0	0.0	100.0
30.100	0	0.0	100.0
30.200	0	0.0	100.0
30.300	0	0.0	100.0
30.400	0	0.0	100.0
30.500	0	0.0	100.0
30.600	0	0.0	100.0
30.700	0	0.0	100.0
30.800	0	0.0	100.0
30.900	0	0.0	100.0
31.000	0	0.0	100.0
31.100	0	0.0	100.0
31.200	0	0.0	100.0
31.300	0	0.0	100.0

31.400	0	0.0	100.0
31.500	0	0.0	100.0
31.600	0	0.0	100.0
31.700	0	0.0	100.0
31.800	0	0.0	100.0
31.900	0	0.0	100.0
32.000	0	0.0	100.0
32.100	0	0.0	100.0
32.200	0	0.0	100.0
32.300	0	0.0	100.0
32.400	0	0.0	100.0
32.500	0	0.0	100.0
32.600	0	0.0	100.0
32.700	0	0.0	100.0
32.800	0	0.0	100.0
32.900	0	0.0	100.0
33.000	0	0.0	100.0
33.100	0	0.0	100.0
33.200	0	0.0	100.0
33.300	0	0.0	100.0
33.400	0	0.0	100.0
33.500	0	0.0	100.0
33.600	0	0.0	100.0
33.700	0	0.0	100.0
33.800	0	0.0	100.0
33.900	0	0.0	100.0
34.000	0	0.0	100.0
34.100	0	0.0	100.0
34.200	0	0.0	100.0
34.300	0	0.0	100.0
34.400	0	0.0	100.0
34.500	0	0.0	100.0
34.600	0	0.0	100.0
34.700	0	0.0	100.0
34.800	0	0.0	100.0
34.900	0	0.0	100.0
35.000	0	0.0	100.0
35.100	0	0.0	100.0
35.200	0	0.0	100.0
35.300	0	0.0	100.0
35.400	0	0.0	100.0
35.500	0	0.0	100.0
35.600	0	0.0	100.0
35.700	0	0.0	100.0
35.800	0	0.0	100.0
35.900	0	0.0	100.0
36.000	0	0.0	100.0
36.100	0	0.0	100.0
36.200	0	0.0	100.0
36.300	0	0.0	100.0
36.400	0	0.0	100.0
36.500	0	0.0	100.0
36.600	0	0.0	100.0
36.700	0	0.0	100.0

36.800	0	0.0	100.0
36.900	0	0.0	100.0
37.000	0	0.0	100.0
37.100	0	0.0	100.0
37.200	0	0.0	100.0
37.300	0	0.0	100.0
37.400	0	0.0	100.0
37.500	0	0.0	100.0
37.600	0	0.0	100.0
37.700	0	0.0	100.0
37.800	0	0.0	100.0
37.900	0	0.0	100.0
38.000	0	0.0	100.0
38.100	0	0.0	100.0
38.200	0	0.0	100.0
38.300	0	0.0	100.0
38.400	0	0.0	100.0
38.500	0	0.0	100.0
38.600	0	0.0	100.0
38.700	0	0.0	100.0
38.800	0	0.0	100.0
38.900	0	0.0	100.0
39.000	0	0.0	100.0
39.100	0	0.0	100.0
39.200	0	0.0	100.0
39.300	0	0.0	100.0
39.400	0	0.0	100.0
39.500	0	0.0	100.0
39.600	0	0.0	100.0
39.700	0	0.0	100.0
39.800	0	0.0	100.0
39.900	0	0.0	100.0
40.000	0	0.0	100.0
40.100	0	0.0	100.0
40.200	0	0.0	100.0
40.300	0	0.0	100.0
40.400	0	0.0	100.0
40.500	0	0.0	100.0
40.600	0	0.0	100.0
40.700	0	0.0	100.0
40.800	0	0.0	100.0
40.900	0	0.0	100.0
41.000	0	0.0	100.0
41.100	0	0.0	100.0
41.200	0	0.0	100.0
41.300	0	0.0	100.0
41.400	0	0.0	100.0
41.500	0	0.0	100.0
41.600	0	0.0	100.0
41.700	0	0.0	100.0
41.800	0	0.0	100.0
41.900	0	0.0	100.0
42.000	0	0.0	100.0
42.100	0	0.0	100.0

```
42.200      1    0.0 100.0 |
```

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
12.373 s	12.163 s	0.464 s	149.103 s	185436	206.040 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	4415	2.4	7.7	|******** T
2.000	21177	11.4	7.1	|**********T**********
3.000	13953	7.5	6.6	|**********T***
4.000	11582	6.2	6.1	|**********T
5.000	10471	5.6	5.6	|**********T
6.000	9676	5.2	5.2	|**********T
7.000	8953	4.8	4.8	|**********T
8.000	8095	4.4	4.4	|**********T
9.000	7482	4.0	4.1	|**********T
10.000	6992	3.8	3.7	|**********T
11.000	6494	3.5	3.5	|**********T
12.000	5994	3.2	3.2	|**********T
13.000	5293	2.9	2.9	|**********T
14.000	5170	2.8	2.7	|*********T
15.000	4541	2.4	2.5	|********T
16.000	4374	2.4	2.3	|********T
17.000	3998	2.2	2.1	|*******T
18.000	3856	2.1	2.0	|******T
19.000	3399	1.8	1.8	|******T
20.000	3121	1.7	1.7	|*****T
21.000	2888	1.6	1.5	|*****T
22.000	2575	1.4	1.4	|****T
23.000	2376	1.3	1.3	|****T
24.000	2320	1.3	1.2	|***T
25.000	2135	1.2	1.1	|***T
26.000	1821	1.0	1.0	|***T
27.000	1792	1.0	1.0	|***T
28.000	1591	0.9	0.9	|**T
29.000	1483	0.8	0.8	|**T
30.000	1406	0.8	0.7	|**T
31.000	1255	0.7	0.7	|**T
32.000	1275	0.7	0.6	|*T
33.000	1023	0.6	0.6	|*T
34.000	935	0.5	0.5	|*T
35.000	820	0.4	0.5	|*T
36.000	903	0.5	0.5	|*T
37.000	709	0.4	0.4	|*T
38.000	729	0.4	0.4	|T
39.000	672	0.4	0.4	|T
40.000	636	0.3	0.3	|T
41.000	545	0.3	0.3	|T
42.000	478	0.3	0.3	|T
43.000	512	0.3	0.3	|T
44.000	464	0.3	0.2	|T

45.000	387	0.2	0.2	T
46.000	356	0.2	0.2	T
47.000	381	0.2	0.2	T
48.000	313	0.2	0.2	T
49.000	303	0.2	0.2	T
50.000	241	0.1	0.1	T
51.000	230	0.1	0.1	T
52.000	266	0.1	0.1	
53.000	190	0.1	0.1	
54.000	182	0.1	0.1	
55.000	178	0.1	0.1	
56.000	161	0.1	0.1	
57.000	115	0.1	0.1	
58.000	135	0.1	0.1	
59.000	105	0.1	0.1	
60.000	109	0.1	0.1	
61.000	120	0.1	0.1	
62.000	94	0.1	0.1	
63.000	110	0.1	0.1	
64.000	102	0.1	0.0	
65.000	77	0.0	0.0	
66.000	98	0.1	0.0	
67.000	59	0.0	0.0	
68.000	59	0.0	0.0	
69.000	39	0.0	0.0	
70.000	42	0.0	0.0	
71.000	50	0.0	0.0	
72.000	54	0.0	0.0	
73.000	31	0.0	0.0	
74.000	38	0.0	0.0	
75.000	21	0.0	0.0	
76.000	34	0.0	0.0	
77.000	25	0.0	0.0	
78.000	41	0.0	0.0	
79.000	27	0.0	0.0	
80.000	32	0.0	0.0	
81.000	20	0.0	0.0	
82.000	16	0.0	0.0	
83.000	13	0.0	0.0	
84.000	17	0.0	0.0	
85.000	15	0.0	0.0	
86.000	13	0.0	0.0	
87.000	10	0.0	0.0	
88.000	4	0.0	0.0	
89.000	7	0.0	0.0	
90.000	10	0.0	0.0	
91.000	11	0.0	0.0	
92.000	11	0.0	0.0	
93.000	15	0.0	0.0	
94.000	7	0.0	0.0	
95.000	6	0.0	0.0	
96.000	2	0.0	0.0	
97.000	7	0.0	0.0	
98.000	5	0.0	0.0	

99.000	10	0.0	0.0
100.000	1	0.0	0.0
101.000	0	0.0	0.0
102.000	3	0.0	0.0
103.000	0	0.0	0.0
104.000	3	0.0	0.0
105.000	0	0.0	0.0
106.000	4	0.0	0.0
107.000	5	0.0	0.0
108.000	4	0.0	0.0
109.000	3	0.0	0.0
110.000	5	0.0	0.0
111.000	3	0.0	0.0
112.000	0	0.0	0.0
113.000	0	0.0	0.0
114.000	0	0.0	0.0
115.000	3	0.0	0.0
116.000	0	0.0	0.0
117.000	0	0.0	0.0
118.000	4	0.0	0.0
119.000	0	0.0	0.0
120.000	2	0.0	0.0
121.000	0	0.0	0.0
122.000	0	0.0	0.0
123.000	2	0.0	0.0
124.000	1	0.0	0.0
125.000	0	0.0	0.0
126.000	1	0.0	0.0
127.000	3	0.0	0.0
128.000	0	0.0	0.0
129.000	0	0.0	0.0
130.000	0	0.0	0.0
131.000	0	0.0	0.0
132.000	0	0.0	0.0
133.000	0	0.0	0.0
134.000	0	0.0	0.0
135.000	0	0.0	0.0
136.000	0	0.0	0.0
137.000	0	0.0	0.0
138.000	0	0.0	0.0
139.000	0	0.0	0.0
140.000	0	0.0	0.0
141.000	0	0.0	0.0
142.000	0	0.0	0.0
143.000	0	0.0	0.0
144.000	0	0.0	0.0
145.000	0	0.0	0.0
146.000	2	0.0	0.0
147.000	3	0.0	0.0
148.000	0	0.0	0.0
149.000	0	0.0	0.0
150.000	1	0.0	0.0

Simulator Report

NonStop/SQL

4 * 8 VLX S STEMS

Throughput - 175 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test ALL;RUN03-32.5;SQL
Test Date                 =   87-02-19
Test Length               =   15:53:53 - 16:20:09  (1576 sec)
Window Length             =   16:04:00 - 16:19:00  (900 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TLOGSUMZ

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    5.500 tx/sec
Tx Arrival Rate/Terminal  =    0.069 tx/sec
Tx Inter-Arrival Time     =   14.545 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) ---------------------------------

Tx Throughput              =  175.162 tx/sec
Tx Response Time Average   =    0.933 sec
                 Std Dev   =    0.331 sec
                 Minimum   =    0.293 sec
                 Maximum   =   16.605 sec
                 Count     =   157646 #

Tx Response Time 10%       =    0.650 sec  ( 13.1%)
                 20%       =    0.700 sec  ( 20.3%)
                 30%       =    0.800 sec  ( 37.4%)
                 40%       =    0.850 sec  ( 46.3%)
                 50%       =    0.900 sec  ( 54.6%)
                 60%       =    0.950 sec  ( 62.1%)
                 70%       =    1.050 sec  ( 73.9%)
                 80%       =    1.150 sec  ( 82.0%)
                 90%       =    1.350 sec  ( 91.3%)
                 95%       =    1.500 sec  ( 95.2%)

        Tx Think Average   =   14.567 sec
                 Std Dev   =   14.586 sec
                 Minimum   =    0.001 sec
                 Maximum   =  175.431 sec
                 Count     =   147983 #    ( 93.9%)

        Tx Delay Average   =    0.530 sec
                 Std Dev   =    0.523 sec
                 Minimum   =    0.000 sec
                 Maximum   =   16.060 sec
                 Count     =     9663 #    (  6.1%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.933 s	0.331 s	0.293 s	16.605 s	157646	175.162 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	2	0.0	0.0	|
0.400	184	0.1	0.1	|
0.500	2257	1.4	1.5	|***
0.600	9546	6.1	7.6	|**********
0.700	19969	12.7	20.3	|**********
0.800	26928	17.1	37.4	|**********
0.900	27260	17.3	54.6	|**********
1.000	21866	13.9	68.5	|**********
1.100	15568	9.9	78.4	|**********
1.200	10358	6.6	85.0	|**********
1.300	7223	4.6	89.5	|**********
1.400	5194	3.3	92.8	|********
1.500	3769	2.4	95.2	|*****
1.600	2501	1.6	96.8	|***
1.700	1699	1.1	97.9	|**
1.800	1027	0.7	98.5	|*
1.900	659	0.4	99.0	|*
2.000	405	0.3	99.2	|
2.100	282	0.2	99.4	|
2.200	181	0.1	99.5	|
2.300	149	0.1	99.6	|
2.400	116	0.1	99.7	|
2.500	87	0.1	99.7	|
2.600	77	0.0	99.8	|
2.700	54	0.0	99.8	|
2.800	41	0.0	99.8	|
2.900	33	0.0	99.9	|
3.000	28	0.0	99.9	|
3.100	24	0.0	99.9	|
3.200	19	0.0	99.9	|
3.300	14	0.0	99.9	|
3.400	14	0.0	99.9	|
3.500	11	0.0	99.9	|
3.600	12	0.0	99.9	|
3.700	6	0.0	99.9	|
3.800	12	0.0	100.0	|
3.900	3	0.0	100.0	|
4.000	9	0.0	100.0	|
4.100	7	0.0	100.0	|
4.200	2	0.0	100.0	|
4.300	2	0.0	100.0	|

4.400	3	0.0	100.0
4.500	5	0.0	100.0
4.600	3	0.0	100.0
4.700	3	0.0	100.0
4.800	1	0.0	100.0
4.900	2	0.0	100.0
5.000	5	0.0	100.0
5.100	3	0.0	100.0
5.200	3	0.0	100.0
5.300	2	0.0	100.0
5.400	1	0.0	100.0
5.500	1	0.0	100.0
5.600	0	0.0	100.0
5.700	0	0.0	100.0
5.800	0	0.0	100.0
5.900	0	0.0	100.0
6.000	0	0.0	100.0
6.100	0	0.0	100.0
6.200	1	0.0	100.0
6.300	0	0.0	100.0
6.400	1	0.0	100.0
		0.0	100.0
			100.0
6.700			100.0
6.800	0	0.0	100.0
6.900	0	0.0	100.0
	1	0.0	100.0
	0	0.0	100.0
		0.0	100.0
	1	0.0	100.0
		0.0	100.0
		0.0	100.0
		0.0	100.0
7.			100.0
7.800	0		100.0
7.900			100.0
8.000		0.0	100.0
8.100		0.0	100.0
8.200		0.0	100.0
8.300		0.0	100.0
8.400		0.0	100.0
8.500		0	100.0
			100.0
			00.0
			100.0

4.400	3	0.0	100.0
4.500	5	0.0	100.0
4.600	3	0.0	100.0
4.700	3	0.0	100.0
4.800	1	0.0	100.0
4.900	2	0.0	100.0
5.000	5	0.0	100.0
5.100	3	0.0	100.0
5.200	3	0.0	100.0
5.300	2	0.0	100.0
5.400	1	0.0	100.0
5.500	1	0.0	100.0
5.600	0	0.0	100.0
5.700	0	0.0	100.0
5.800	0	0.0	100.0
5.900	0	0.0	100.0
6.000	0	0.0	100.0
6.100	0	0.0	100.0
6.200	1	0.0	100.0
6.300	0	0.0	100.0
6.400	1	0.0	100.0
6.500	0	0.0	100.0
6.600	0	0.0	100.0
6.700	0	0.0	100.0
6.800	0	0.0	100.0
6.900	0	0.0	100.0
7.000	1	0.0	100.0
7.100	0	0.0	100.0
7.200	0	0.0	100.0
7.300	1	0.0	100.0
7.400	0	0.0	100.0
7.500	0	0.0	100.0
7.600	0	0.0	100.0
7.700	0	0.0	100.0
7.800	0	0.0	100.0
7.900	0	0.0	100.0
8.000	0	0.0	100.0
8.100	0	0.0	100.0
8.200	0	0.0	100.0
8.300	0	0.0	100.0
8.400	0	0.0	100.0
8.500	0	0.0	100.0
8.600	0	0.0	100.0
8.700	0	0.0	100.0
8.800	0	0.0	100.0
8.900	0	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	0	0.0	100.0
9.500	0	0.0	100.0
9.600	0	0.0	100.0
9.700	0	0.0	100.0

9.800	0	0.0	100.0	
9.900	0	0.0	100.0	
10.000	0	0.0	100.0	
10.100	0	0.0	100.0	
10.200	0	0.0	100.0	
10.300	0	0.0	100.0	
10.400	0	0.0	100.0	
10.500	0	0.0	100.0	
10.600	0	0.0	100.0	
10.700	0	0.0	100.0	
10.800	0	0.0	100.0	
10.900	0	0.0	100.0	
11.000	0	0.0	100.0	
11.100	0	0.0	100.0	
11.200	0	0.0	100.0	
11.300	0	0.0	100.0	
11.400	0	0.0	100.0	
11.500	0	0.0	100.0	
11.600	0	0.0	100.0	
11.700	0	0.0	100.0	
11.800	0	0.0	100.0	
11.900	0	0.0	100.0	
12.000	0	0.0	100.0	
12.100	0	0.0	100.0	
12.200	0	0.0	100.0	
12.300	0	0.0	100.0	
12.400	0	0.0	100.0	
12.500	0	0.0	100.0	
12.600	0	0.0	100.0	
12.700	0	0.0	100.0	
12.800	0	0.0	100.0	
12.900	0	0.0	100.0	
13.000	0	0.0	100.0	
13.100	0	0.0	100.0	
13.200	0	0.0	100.0	
13.300	0	0.0	100.0	
13.400	0	0.0	100.0	
13.500	0	0.0	100.0	
13.600	0	0.0	100.0	
13.700	0	0.0	100.0	
13.800	0	0.0	100.0	
13.900	0	0.0	100.0	
14.000	0	0.0	100.0	
14.100	1	0.0	100.0	
14.200	1	0.0	100.0	
14.300	0	0.0	100.0	
14.400	0	0.0	100.0	
14.500	0	0.0	100.0	
14.600	0	0.0	100.0	
14.700	1	0.0	100.0	
14.800	0	0.0	100.0	
14.900	1	0.0	100.0	
15.000	0	0.0	100.0	
15.100	0	0.0	100.0	

15.200	1	0.0	100.0
15.300	0	0.0	100.0
15.400	1	0.0	100.0
15.500	1	0.0	100.0
15.600	1	0.0	100.0
15.700	0	0.0	100.0
15.800	1	0.0	100.0
15.900	1	0.0	100.0
16.000	0	0.0	100.0
16.100	1	0.0	100.0
16.200	0	0.0	100.0
16.300	0	0.0	100.0
16.400	0	0.0	100.0
16.500	0	0.0	100.0
16.600	0	0.0	100.0
16.700	1	0.0	100.0

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
14.607 s	14.552 s	0.354 s	176.217 s	157646	175.162 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	7109	4.5	6.6	|********** T
2.000	12840	8.1	6.2	|**********T**********
3.000	9312	5.9	5.8	|*********T
4.000	8521	5.4	5.4	|*********T
5.000	8031	5.1	5.0	|*********T
6.000	7399	4.7	4.7	|*********T
7.000	6816	4.3	4.4	|*********T
8.000	6290	4.0	4.1	|*********T
9.000	6105	3.9	3.8	|*********T
10.000	5671	3.6	3.6	|*********T
11.000	5252	3.3	3.3	|*********T
12.000	4970	3.2	3.1	|*********T
13.000	4620	2.9	2.9	|*********T
14.000	4448	2.8	2.7	|*********T
15.000	3908	2.5	2.5	|*********T
16.000	3749	2.4	2.4	|*********T
17.000	3468	2.2	2.2	|*********T
18.000	3128	2.0	2.1	|*********T
19.000	3217	2.0	1.9	|*********T
20.000	2754	1.7	1.8	|********T
21.000	2755	1.7	1.7	|********T
22.000	2651	1.7	1.6	|*******T
23.000	2212	1.4	1.5	|*******T
24.000	2118	1.3	1.4	|******T
25.000	2069	1.3	1.3	|******T
26.000	1841	1.2	1.2	|*****T
27.000	1645	1.0	1.1	|*****T
28.000	1663	1.1	1.0	|****T
29.000	1499	1.0	1.0	|****T
30.000	1473	0.9	0.9	|****T
31.000	1270	0.8	0.8	|***T
32.000	1328	0.8	0.8	|***T
33.000	1132	0.7	0.7	|***T
34.000	1146	0.7	0.7	|***T
35.000	1030	0.7	0.6	|**T
36.000	890	0.6	0.6	|**T
37.000	978	0.6	0.6	|**T
38.000	856	0.5	0.5	|**T
39.000	741	0.5	0.5	|**T
40.000	753	0.5	0.5	|*T
41.000	666	0.4	0.4	|*T
42.000	652	0.4	0.4	|*T
43.000	554	0.4	0.4	|*T
44.000	534	0.3	0.3	|*T

45.000	474	0.3	0.3	*T
46.000	415	0.3	0.3	*T
47.000	490	0.3	0.3	T
48.000	401	0.3	0.3	T
49.000	441	0.3	0.2	T
50.000	352	0.2	0.2	T
51.000	357	0.2	0.2	T
52.000	330	0.2	0.2	T
53.000	260	0.2	0.2	T
54.000	306	0.2	0.2	T
55.000	249	0.2	0.2	T
56.000	197	0.1	0.2	T
57.000	248	0.2	0.1	T
58.000	231	0.1	0.1	T
59.000	168	0.1	0.1	T
60.000	183	0.1	0.1	T
61.000	179	0.1	0.1	T
62.000	117	0.1	0.1	T
63.000	111	0.1	0.1	
64.000	125	0.1	0.1	
65.000	116	0.1	0.1	
66.000	118	0.1	0.1	
67.000	89	0.1	0.1	
68.000	123	0.1	0.1	
69.000	87	0.1	0.1	
70.000	86	0.1	0.1	
71.000	42	0.0	0.1	
72.000	105	0.1	0.1	
73.000	83	0.1	0.0	
74.000	68	0.0	0.0	
75.000	83	0.1	0.0	
76.000	70	0.0	0.0	
77.000	52	0.0	0.0	
78.000	94	0.1	0.0	
79.000	63	0.0	0.0	
80.000	25	0.0	0.0	
81.000	25	0.0	0.0	
82.000	43	0.0	0.0	
83.000	51	0.0	0.0	
84.000	17	0.0	0.0	
85.000	50	0.0	0.0	
86.000	30	0.0	0.0	
87.000	24	0.0	0.0	
88.000	25	0.0	0.0	
89.000	22	0.0	0.0	
90.000	16	0.0	0.0	
91.000	12	0.0	0.0	
92.000	24	0.0	0.0	
93.000	13	0.0	0.0	
94.000	13	0.0	0.0	
95.000	17	0.0	0.0	
96.000	11	0.0	0.0	
97.000	10	0.0	0.0	
98.000	11	0.0	0.0	

99.000	24	0.0	0.0
100.000	9	0.0	0.0
101.000	25	0.0	0.0
102.000	9	0.0	0.0
103.000	3	0.0	0.0
104.000	8	0.0	0.0
105.000	13	0.0	0.0
106.000	7	0.0	0.0
107.000	12	0.0	0.0
108.000	7	0.0	0.0
109.000	0	0.0	0.0
110.000	11	0.0	0.0
111.000	4	0.0	0.0
112.000	4	0.0	0.0
113.000	8	0.0	0.0
114.000	0	0.0	0.0
115.000	4	0.0	0.0
116.000	10	0.0	0.0
117.000	10	0.0	0.0
118.000	4	0.0	0.0
119.000	0	0.0	0.0
120.000	0	0.0	0.0
121.000	3	0.0	0.0
122.000	0	0.0	0.0
123.000	2	0.0	0.0
124.000	0	0.0	0.0
125.000	6	0.0	0.0
126.000	10	0.0	0.0
127.000	0	0.0	0.0
128.000	4	0.0	0.0
129.000	0	0.0	0.0
130.000	4	0.0	0.0
131.000	4	0.0	0.0
132.000	0	0.0	0.0
133.000	0	0.0	0.0
134.000	0	0.0	0.0
135.000	3	0.0	0.0
136.000	0	0.0	0.0
137.000	0	0.0	0.0
138.000	0	0.0	0.0
139.000	0	0.0	0.0
140.000	0	0.0	0.0
141.000	0	0.0	0.0
142.000	4	0.0	0.0
143.000	0	0.0	0.0
144.000	3	0.0	0.0
145.000	0	0.0	0.0
146.000	4	0.0	0.0
147.000	0	0.0	0.0
148.000	0	0.0	0.0
149.000	0	0.0	0.0
150.000	4	0.0	0.0
151.000	0	0.0	0.0
152.000	0	0.0	0.0

153.000	0	0.0	0.0
154.000	0	0.0	0.0
155.000	0	0.0	0.0
156.000	0	0.0	0.0
157.000	0	0.0	0.0
158.000	0	0.0	0.0
159.000	0	0.0	0.0
160.000	0	0.0	0.0
161.000	0	0.0	0.0
162.000	0	0.0	0.0
163.000	0	0.0	0.0
164.000	0	0.0	0.0
165.000	0	0.0	0.0
166.000	0	0.0	0.0
167.000	0	0.0	0.0
168.000	0	0.0	0.0
169.000	0	0.0	0.0
170.000	0	0.0	0.0
171.000	0	0.0	0.0
172.000	0	0.0	0.0
173.000	4	0.0	0.0
174.000	0	0.0	0.0
175.000	0	0.0	0.0
176.000	0	0.0	0.0
177.000	3	0.0	0.0

Simulator Report

NonStop/SQL

4 * 8 VLX SYSTEMS

Throughput - 143 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test ALL;RUN04-32.5;SQL
Test Date                 =   87-02-19
Test Length               =   19:58:09 - 20:22:07  (1438 sec)
Window Length             =   20:04:00 - 20:19:00  (900 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TLOGSUMZ

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    4.500 tx/sec
Tx Arrival Rate/Terminal  =    0.056 tx/sec
Tx Inter-Arrival Time     =   17.778 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) ---------------------------------

Tx Throughput                =  142.944 tx/sec
Tx Response Time Average     =    0.827 sec
                 Std Dev     =    0.285 sec
                 Minimum     =    0.261 sec
                 Maximum     =   16.038 sec
                 Count       =   128650 #

Tx Response Time 10%         =    0.600 sec   ( 15.2%)
                 20%         =    0.650 sec   ( 23.0%)
                 30%         =    0.700 sec   ( 32.4%)
                 40%         =    0.750 sec   ( 42.9%)
                 50%         =    0.800 sec   ( 53.6%)
                 60%         =    0.850 sec   ( 63.4%)
                 70%         =    0.900 sec   ( 71.0%)
                 80%         =    1.000 sec   ( 81.9%)
                 90%         =    1.150 sec   ( 90.5%)
                 95%         =    1.300 sec   ( 95.1%)

        Tx Think Average     =   17.890 sec
                 Std Dev     =   17.858 sec
                 Minimum     =    0.001 sec
                 Maximum     =  214.754 sec
                 Count       =   122814 #     ( 95.5%)

        Tx Delay Average     =    0.463 sec
                 Std Dev     =    0.427 sec
                 Minimum     =    0.000 sec
                 Maximum     =   12.307 sec
                 Count       =     5836 #     (  4.5%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.827 s	0.285 s	0.261 s	16.038 s	128650	142.944 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	12	0.0	0.0	|
0.400	553	0.4	0.4	|
0.500	4803	3.7	4.2	|*******
0.600	14192	11.0	15.2	|**********
0.700	22061	17.1	32.4	|**********
0.800	27398	21.3	53.6	|**********
0.900	22306	17.3	71.0	|**********
1.000	14009	10.9	81.9	|**********
1.100	8298	6.5	88.3	|**********
1.200	5044	3.9	92.2	|*******
1.300	3650	2.8	95.1	|*****
1.400	2581	2.0	97.1	|***
1.500	1515	1.2	98.3	|**
1.600	875	0.7	98.9	|*
1.700	440	0.3	99.3	|
1.800	248	0.2	99.5	|
1.900	137	0.1	99.6	|
2.000	96	0.1	99.7	|
2.100	60	0.0	99.7	|
2.200	48	0.0	99.7	|
2.300	47	0.0	99.8	|
2.400	34	0.0	99.8	|
2.500	24	0.0	99.8	|
2.600	31	0.0	99.9	|
2.700	17	0.0	99.9	|
2.800	17	0.0	99.9	|
2.900	16	0.0	99.9	|
3.000	15	0.0	99.9	|
3.100	19	0.0	99.9	|
3.200	12	0.0	99.9	|
3.300	13	0.0	99.9	|
3.400	11	0.0	99.9	|
3.500	9	0.0	100.0	|
3.600	7	0.0	100.0	|
3.700	7	0.0	100.0	|
3.800	6	0.0	100.0	|
3.900	4	0.0	100.0	|
4.000	6	0.0	100.0	|
4.100	5	0.0	100.0	|
4.200	1	0.0	100.0	|
4.300	1	0.0	100.0	|

4.400	2	0.0	100.0
4.500	0	0.0	100.0
4.600	3	0.0	100.0
4.700	2	0.0	100.0
4.800	0	0.0	100.0
4.900	0	0.0	100.0
5.000	0	0.0	100.0
5.100	0	0.0	100.0
5.200	1	0.0	100.0
5.300	0	0.0	100.0
5.400	1	0.0	100.0
5.500	0	0.0	100.0
5.600	0	0.0	100.0
5.700	0	0.0	100.0
5.800	0	0.0	100.0
5.900	0	0.0	100.0
6.000	0	0.0	100.0
6.100	0	0.0	100.0
6.200	0	0.0	100.0
6.300	0	0.0	100.0
6.400	0	0.0	100.0
6.500	0	0.0	100.0
6.600	0	0.0	100.0
6.700	0	0.0	100.0
6.800	0	0.0	100.0
6.900	0	0.0	100.0
7.000	0	0.0	100.0
7.100	0	0.0	100.0
7.200	0	0.0	100.0
7.300	0	0.0	100.0
7.400	0	0.0	100.0
7.500	0	0.0	100.0
7.600	0	0.0	100.0
7.700	0	0.0	100.0
7.800	0	0.0	100.0
7.900	0	0.0	100.0
8.000	0	0.0	100.0
8.100	0	0.0	100.0
8.200	0	0.0	100.0
8.300	0	0.0	100.0
8.400	0	0.0	100.0
8.500	0	0.0	100.0
8.600	0	0.0	100.0
8.700	1	0.0	100.0
8.800	0	0.0	100.0
8.900	0	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	0	0.0	100.0
9.500	0	0.0	100.0
9.600	0	0.0	100.0
9.700	0	0.0	100.0

9.800	0	0.0	100.0
9.900	0	0.0	100.0
10.000	0	0.0	100.0
10.100	0	0.0	100.0
10.200	0	0.0	100.0
10.300	0	0.0	100.0
10.400	0	0.0	100.0
10.500	0	0.0	100.0
10.600	0	0.0	100.0
10.700	0	0.0	100.0
10.800	0	0.0	100.0
10.900	0	0.0	100.0
11.000	0	0.0	100.0
11.100	0	0.0	100.0
11.200	1	0.0	100.0
11.300	0	0.0	100.0
11.400	0	0.0	100.0
11.500	0	0.0	100.0
11.600	0	0.0	100.0
11.700	1	0.0	100.0
11.800	0	0.0	100.0
11.900	0	0.0	100.0
12.000	2	0.0	100.0
12.100	1	0.0	100.0
12.200	0	0.0	100.0
12.300	0	0.0	100.0
12.400	0	0.0	100.0
12.500	0	0.0	100.0
12.600	0	0.0	100.0
12.700	0	0.0	100.0
12.800	0	0.0	100.0
12.900	0	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	0	0.0	100.0
13.500	0	0.0	100.0
13.600	0	0.0	100.0
13.700	0	0.0	100.0
13.800	0	0.0	100.0
13.900	1	0.0	100.0
14.000	1	0.0	100.0
14.100	0	0.0	100.0
14.200	0	0.0	100.0
14.300	1	0.0	100.0
14.400	0	0.0	100.0
14.500	1	0.0	100.0
14.600	1	0.0	100.0
14.700	0	0.0	100.0
14.800	0	0.0	100.0
14.900	0	0.0	100.0
15.000	0	0.0	100.0
15.100	1	0.0	100.0

15.200	0	0.0	100.0
15.300	0	0.0	100.0
15.400	0	0.0	100.0
15.500	0	0.0	100.0
15.600	0	0.0	100.0
15.700	0	0.0	100.0
15.800	0	0.0	100.0
15.900	0	0.0	100.0
16.000	0	0.0	100.0
16.100	1	0.0	100.0

```
              *  ET1 Inter-Arrival Time Distribution  *

---------------------------------------------------------------------
   Mean        Std         Min        Max        Count      Thruput
 17.905 s   17.838 s    0.325 s  215.374 s    128650     142.944 tx/s
---------------------------------------------------------------------

ArrvTime Count  Pct  Theory
-------- ----- ----- ------
   0.000     0   0.0   0.0 |
   1.000  5700   4.4   5.4 |******************************       T
   2.000  7717   6.0   5.1 |***********************************T******
   3.000  6459   5.0   4.9 |*********************************T*
   4.000  5924   4.6   4.6 |*******************************T
   5.000  5537   4.3   4.3 |*****************************T
   6.000  5312   4.1   4.1 |****************************T
   7.000  5180   4.0   3.9 |**************************T*
   8.000  4660   3.6   3.7 |*************************T
   9.000  4320   3.4   3.5 |***********************T
  10.000  3960   3.1   3.3 |********************* T
  11.000  4087   3.2   3.1 |*********************T
  12.000  3840   3.0   2.9 |********************T
  13.000  3568   2.8   2.8 |******************T
  14.000  3466   2.7   2.6 |*****************T
  15.000  3229   2.5   2.5 |****************T
  16.000  2893   2.2   2.4 |***************T
  17.000  2999   2.3   2.2 |***************T
  18.000  2632   2.0   2.1 |**************T
  19.000  2559   2.0   2.0 |*************T
  20.000  2475   1.9   1.9 |************T
  21.000  2226   1.7   1.8 |***********T
  22.000  2086   1.6   1.7 |***********T
  23.000  2215   1.7   1.6 |**********T*
  24.000  1993   1.5   1.5 |**********T
  25.000  1688   1.3   1.4 |*********T
  26.000  1717   1.3   1.3 |********T
  27.000  1827   1.4   1.3 |********T
  28.000  1518   1.2   1.2 |*******T
  29.000  1452   1.1   1.1 |*******T
  30.000  1381   1.1   1.1 |*******T
  31.000  1328   1.0   1.0 |******T
  32.000  1179   0.9   1.0 |******T
  33.000  1146   0.9   0.9 |*****T
  34.000  1081   0.8   0.9 |*****T
  35.000  1054   0.8   0.8 |*****T
  36.000  1019   0.8   0.8 |****T
  37.000   884   0.7   0.7 |****T
  38.000   907   0.7   0.7 |****T
  39.000   865   0.7   0.7 |****T
  40.000   741   0.6   0.6 |***T
  41.000   761   0.6   0.6 |***T
  42.000   708   0.6   0.6 |***T
  43.000   720   0.6   0.5 |***T
  44.000   563   0.4   0.5 |**T
```

57

45.000	687	0.5	0.5	**T
46.000	624	0.5	0.4	**T
47.000	505	0.4	0.4	**T
48.000	532	0.4	0.4	**T
49.000	486	0.4	0.4	**T
50.000	487	0.4	0.4	*T
51.000	433	0.3	0.3	*T
52.000	407	0.3	0.3	*T
53.000	368	0.3	0.3	*T
54.000	335	0.3	0.3	*T
55.000	362	0.3	0.3	*T
56.000	263	0.2	0.3	*T
57.000	350	0.3	0.2	*T
58.000	315	0.2	0.2	*T
59.000	261	0.2	0.2	T
60.000	323	0.3	0.2	T
61.000	229	0.2	0.2	T
62.000	256	0.2	0.2	T
63.000	207	0.2	0.2	T
64.000	199	0.2	0.2	T
65.000	175	0.1	0.2	T
66.000	208	0.2	0.1	T
67.000	184	0.1	0.1	T
68.000	145	0.1	0.1	T
69.000	150	0.1	0.1	T
70.000	153	0.1	0.1	T
71.000	124	0.1	0.1	T
72.000	127	0.1	0.1	T
73.000	105	0.1	0.1	T
74.000	122	0.1	0.1	T
75.000	99	0.1	0.1	T
76.000	84	0.1	0.1	T
77.000	95	0.1	0.1	T
78.000	110	0.1	0.1	T
79.000	80	0.1	0.1	
80.000	71	0.1	0.1	
81.000	75	0.1	0.1	
82.000	59	0.0	0.1	
83.000	84	0.1	0.1	
84.000	50	0.0	0.1	
85.000	42	0.0	0.0	
86.000	62	0.0	0.0	
87.000	43	0.0	0.0	
88.000	47	0.0	0.0	
89.000	60	0.0	0.0	
90.000	43	0.0	0.0	
91.000	56	0.0	0.0	
92.000	39	0.0	0.0	
93.000	54	0.0	0.0	
94.000	32	0.0	0.0	
95.000	40	0.0	0.0	
96.000	50	0.0	0.0	
97.000	30	0.0	0.0	
98.000	12	0.0	0.0	

99.000	24	0.0	0.0
100.000	40	0.0	0.0
101.000	35	0.0	0.0
102.000	26	0.0	0.0
103.000	20	0.0	0.0
104.000	25	0.0	0.0
105.000	25	0.0	0.0
106.000	3	0.0	0.0
107.000	30	0.0	0.0
108.000	9	0.0	0.0
109.000	20	0.0	0.0
110.000	4	0.0	0.0
111.000	4	0.0	0.0
112.000	12	0.0	0.0
113.000	15	0.0	0.0
114.000	10	0.0	0.0
115.000	8	0.0	0.0
116.000	8	0.0	0.0
117.000	18	0.0	0.0
118.000	8	0.0	0.0
119.000	0	0.0	0.0
120.000	5	0.0	0.0
121.000	20	0.0	0.0
122.000	4	0.0	0.0
123.000	10	0.0	0.0
124.000	12	0.0	0.0
125.000	0	0.0	0.0
126.000	4	0.0	0.0
127.000	4	0.0	0.0
128.000	9	0.0	0.0
129.000	10	0.0	0.0
130.000	1	0.0	0.0
131.000	9	0.0	0.0
132.000	8	0.0	0.0
133.000	0	0.0	0.0
134.000	8	0.0	0.0
135.000	9	0.0	0.0
136.000	0	0.0	0.0
137.000	8	0.0	0.0
138.000	1	0.0	0.0
139.000	4	0.0	0.0
140.000	3	0.0	0.0
141.000	4	0.0	0.0
142.000	8	0.0	0.0
143.000	0	0.0	0.0
144.000	4	0.0	0.0
145.000	0	0.0	0.0
146.000	2	0.0	0.0
147.000	0	0.0	0.0
148.000	4	0.0	0.0
149.000	0	0.0	0.0
150.000	4	0.0	0.0
151.000	0	0.0	0.0
152.000	0	0.0	0.0

153.000	4	0.0	0.0
154.000	4	0.0	0.0
155.000	0	0.0	0.0
156.000	0	0.0	0.0
157.000	0	0.0	0.0
158.000	2	0.0	0.0
159.000	0	0.0	0.0
160.000	4	0.0	0.0
161.000	0	0.0	0.0
162.000	0	0.0	0.0
163.000	0	0.0	0.0
164.000	0	0.0	0.0
165.000	3	0.0	0.0
166.000	0	0.0	0.0
167.000	0	0.0	0.0
168.000	0	0.0	0.0
169.000	0	0.0	0.0
170.000	0	0.0	0.0
171.000	0	0.0	0.0
172.000	0	0.0	0.0
173.000	0	0.0	0.0
174.000	3	0.0	0.0
175.000	0	0.0	0.0
176.000	4	0.0	0.0
177.000	0	0.0	0.0
178.000	4	0.0	0.0
179.000	0	0.0	0.0
180.000	0	0.0	0.0
181.000	0	0.0	0.0
182.000	0	0.0	0.0
183.000	0	0.0	0.0
184.000	0	0.0	0.0
185.000	0	0.0	0.0
186.000	0	0.0	0.0
187.000	0	0.0	0.0
188.000	0	0.0	0.0
189.000	0	0.0	0.0
190.000	0	0.0	0.0
191.000	0	0.0	0.0
192.000	0	0.0	0.0
193.000	0	0.0	0.0
194.000	0	0.0	0.0
195.000	0	0.0	0.0
196.000	0	0.0	0.0
197.000	0	0.0	0.0
198.000	0	0.0	0.0
199.000	0	0.0	0.0
200.000	0	0.0	0.0
201.000	0	0.0	0.0
202.000	0	0.0	0.0
203.000	0	0.0	0.0
204.000	0	0.0	0.0
205.000	0	0.0	0.0
206.000	0	0.0	0.0

207.000	0	0.0	0.0
208.000	0	0.0	0.0
209.000	0	0.0	0.0
210.000	0	0.0	0.0
211.000	4	0.0	0.0
212.000	0	0.0	0.0
213.000	0	0.0	0.0
214.000	0	0.0	0.0
215.000	0	0.0	0.0
216.000	4	0.0	0.0

ENFORM Report

NonStop/SQL

4 * 8 VLX SYSTEMS + 1 EXT10

Throughput - 219 TPS

NODES	**A,B,C,D**	**-**	**8 VLXs**
NODE	**E**	**-**	**1 EXT10**

CPU UTILISATION PER TRANSACTION
SUMMARIZED FOR A 7 MINUTE PERIOD

LOADID	NUM-TRANS	TPS	AVG CPU BUSY %	CPU PER TRANS	INTRPT BUSY PER TRANS	DISC IOS PER TRANS	CHITS PER TRANS
SQL222 A	22691	54.167	88.351	.1304	.031	3.6	5.1
SQL222 B	22830	54.628	88.763	.1299	.031	3.5	5.1
SQL222 C	22740	54.185	88.531	.1307	.031	3.6	5.1
SQL222 D	21982	52.481	88.374	.1347	.032	3.6	5.1
SQL222 E	1523	3.626	21.105	.4656	.082	5.8	5.0

CPU UTILISATION BY PROCESS TYPE

PROCESS-TYPE	CPU PER TRANS	SENDS PER TRANS	RECVS PER TRANS	CPU BUSY PERCENT
ATB-B	5.475	.007	.534	29.656
ATB-P	28.091	.917	3.065	152.159
AUDIT-B	.680	.001	.616	3.685
AUDIT-P	1.908	.617	.618	10.336
FOX	.426	.179	.171	2.308
HISTORY-B	2.957	.000	.064	16.018
HISTORY-P	5.678	.123	1.001	30.758
INTERRUPTS	31.116	.000	.000	168.544
MISC	.359	.150	.269	1.947
PATHMON	.050	.006	.001	.270
SERVER	22.236	4.125	1.000	120.443
TCP	18.093	2.000	.000	98.006
TMF MONITOR	3.428	.000	1.424	18.570
TMP	3.232	.770	1.046	17.507
X.25	6.596	.000	1.001	35.727
	130.325	8.893	10.807	705.934

Loadid - SQL222 A

CPU UTILISATION BY PROCESS TYPE

PROCESS-TYPE	CPU PER TRANS	SENDS PER TRANS	RECVS PER TRANS	CPU BUSY PERCENT
ATB-B	5.445	.006	.539	29.744
ATB-P	27.866	.927	3.045	152.229
AUDIT-B	.669	.001	.616	3.656
AUDIT-P	1.894	.616	.618	10.347
FOX	.369	.174	.120	2.014
HISTORY-B	2.967	.000	.064	16.210
HISTORY-P	5.685	.122	1.000	31.055
INTERRUPTS	30.851	.000	.000	168.531
MISC	.347	.153	.277	1.897
PATHMON	.023	.003	.000	.127
SERVER	22.707	4.111	1.000	124.042
TCP	18.002	1.999	.000	98.339
TMF MONITOR	3.390	.000	1.421	18.519
TMP	3.147	.759	1.018	17.191
X.25	6.489	.000	.999	35.450
	129.850	8.871	10.717	709.352

Loadid - SQL222 B

CPU UTILISATION BY PROCESS TYPE

PROCESS-TYPE	CPU PER TRANS	SENDS PER TRANS	RECVS PER TRANS	CPU BUSY PERCENT
ATB-B	5.437	.008	.584	29.459
ATB-P	27.886	.977	3.058	151.098
AUDIT-B	.675	.001	.624	3.655
AUDIT-P	1.906	.624	.625	10.326
FOX	.432	.174	.193	2.342
HISTORY-B	2.995	.000	.064	16.229
HISTORY-P	5.610	.123	1.002	30.395
INTERRUPTS	31.054	.000	.000	168.264
MISC	.368	.168	.286	1.994
SERVER	23.269	4.139	1.000	126.081
TCP	17.963	2.002	.000	97.331
TMF MONITOR	3.318	.000	1.241	17.981
TMP	3.223	.779	1.052	17.461
X.25	6.439	.000	1.002	34.891
	130.573	8.994	10.729	707.507

Loadid - SQL222 C

CPU UTILISATION BY PROCESS TYPE

PROCESS-TYPE	CPU PER TRANS	SENDS PER TRANS	RECVS PER TRANS	CPU BUSY PERCENT
ATB-B	5.570	.009	.677	29.230
ATB-P	28.280	1.099	3.071	148.417
AUDIT-B	.703	.000	.666	3.691
AUDIT-P	2.005	.666	.666	10.521
FOX	.888	.405	.437	4.662
HISTORY-B	3.096	.000	.066	16.250
HISTORY-P	5.732	.126	1.001	30.081
INTERRUPTS	32.179	.000	.000	168.881
MISC	.363	.163	.269	1.906
PATHMON	.026	.003	.000	.138
SERVER	23.860	4.141	1.000	125.220
TCP	17.875	2.000	.000	93.809
TMF MONITOR	3.276	.000	1.078	17.192
TMP	3.364	.810	1.091	17.655
X.25	7.357	.166	1.145	38.610
	134.573	9.588	11.166	706.261

Loadid - SQL222 D

CPU UTILISATION BY PROCESS TYPE

PROCESS-TYPE	CPU PER TRANS	SENDS PER TRANS	RECVS PER TRANS	CPU BUSY PERCENT
ATB-B	31.893	.015	2.737	11.564
ATB-P	136.452	3.297	5.723	49.475
FOX	.018	.000	.013	.007
INTERRUPTS	81.709	.000	.000	29.626
MISC	2.002	.007	.295	.726
PATHMON	.000	.000	.000	.000
SERVER	86.987	4.179	1.000	31.540
TCP	54.085	2.001	.000	19.610
TMF MONITOR	6.508	.000	.055	2.360
TMP	18.530	2.232	1.739	6.719
X.25	44.676	.601	1.757	16.199
	462.860	12.332	13.320	167.825

Loadid - SQL222 E

DETAILED CPU UTILIZATION BY TRANSACTION

PROCESS-TYPE	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	CPU 6	CPU 7
ATB-B	3.510	3.500	3.730	3.640	3.810	3.850	3.790	3.770
ATB-P	18.450	18.340	18.650	18.510	19.780	18.870	20.590	18.920
AUDIT-B	.000	.000	.000	.000	.000	3.680	.000	.000
AUDIT-P	.000	.000	.000	.000	10.330	.000	.000	.000
FOX	.000	.000	.310	.000	.570	.000	1.410	.000
HISTORY-B	.000	16.010	.000	.000	.000	.000	.000	.000
HISTORY-P	30.750	.000	.000	.000	.000	.000	.000	.000
INTERRUPTS	19.880	17.290	25.590	18.830	23.190	21.530	21.120	21.080
MISC	1.450	.350	.020	.020	.020	.010	.020	.020
PATHMON	.260	.000	.000	.000	.000	.000	.000	.000
SERVER	13.400	18.760	14.130	15.550	14.080	13.930	15.830	14.740
TCP	.000	9.420	.000	14.960	14.930	24.580	14.730	19.350
TMF MONITOR	1.660	2.420	1.120	2.460	2.510	3.160	2.430	2.770
TMP	1.150	.000	16.350	.000	.000	.000	.000	.000
X.25	.000	.000	9.100	8.680	.000	.000	9.070	8.870
	90.510	86.090	89.000	82.650	89.220	89.610	88.990	89.520

LOADID - SQL222 A

DETAILED CPU UTILIZATION BY TRANSACTION

PROCESS-TYPE	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	CPU 6	CPU 7
ATB-B	3.620	3.750	3.750	3.700	3.670	3.750	3.690	3.780
ATB-P	19.200	19.020	18.690	18.730	19.170	18.660	20.060	18.650
AUDIT-B	.000	.000	.000	.000	.000	3.650	.000	.000
AUDIT-P	.000	.000	.000	.000	10.340	.000	.000	.000
FOX	.000	.000	.330	.000	.370	.000	1.300	.000
HISTORY-B	.000	16.210	.000	.000	.000	.000	.000	.000
HISTORY-P	31.050	.000	.000	.000	.000	.000	.000	.000
INTERRUPTS	20.240	17.510	25.640	18.910	23.020	21.330	20.880	20.960
MISC	1.400	.360	.020	.010	.020	.020	.020	.020
PATHMON	.120	.000	.000	.000	.000	.000	.000	.000
SERVER	13.930	19.630	15.070	16.360	16.030	13.560	15.180	14.240
TCP	.000	9.480	.000	14.800	14.920	24.500	14.870	19.740
TMF MONITOR	1.650	2.440	1.120	2.440	2.480	3.150	2.420	2.780
TMP	1.130	.000	16.050	.000	.000	.000	.000	.000
X.25	.000	.000	9.020	8.640	.000	.000	8.930	8.840
	92.340	88.400	89.690	83.590	90.020	88.620	87.350	89.010

LOADID - SQL222 B

DETAILED CPU UTILIZATION BY TRANSACTION

PROCESS-TYPE	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	CPU 6	CPU 7
ATB-B	3.650	3.500	3.610	3.630	3.750	3.840	3.710	3.730
ATB-P	18.170	18.810	18.490	18.330	19.840	18.680	20.280	18.460
AUDIT-B	.000	.000	.000	.000	.000	3.650	.000	.000
AUDIT-P	.000	.000	.000	.000	10.320	.000	.000	.000
FOX	.000	.000	.550	.000	.350	.000	1.430	.000
HISTORY-B	.000	16.220	.000	.000	.000	.000	.000	.000
HISTORY-P	30.390	.000	.000	.000	.000	.000	.000	.000
INTERRUPTS	19.870	18.060	25.780	18.900	23.050	21.100	20.940	20.530
MISC	1.480	.380	.020	.010	.020	.010	.010	.020
SERVER	12.650	20.510	14.660	17.990	17.500	13.260	15.430	14.050
TCP	.000	9.670	.000	15.050	14.580	24.220	14.390	19.380
TMF MONITOR	1.640	1.980	1.110	2.480	2.450	3.110	2.400	2.780
TMP	1.160	.000	16.300	.000	.000	.000	.000	.000
X.25	.000	.000	8.740	8.850	.000	.000	8.670	8.600
	89.010	89.130	89.260	85.240	91.860	87.870	87.260	87.550

LOADID - SQL222 C

DETAILED CPU UTILIZATION BY TRANSACTION

PROCESS-TYPE	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	CPU 6	CPU 7
ATB-B	3.610	3.510	3.820	3.560	3.670	3.580	3.600	3.830
ATB-P	18.100	18.480	17.600	18.370	18.810	18.290	20.340	18.380
AUDIT-B	.000	.000	.000	.000	.000	3.690	.000	.000
AUDIT-P	.000	.000	.000	.000	10.520	.000	.000	.000
FOX	.000	.000	1.490	.000	1.520	.000	1.630	.000
HISTORY-B	.000	16.250	.000	.000	.000	.000	.000	.000
HISTORY-P	30.080	.000	.000	.000	.000	.000	.000	.000
INTERRUPTS	19.690	17.570	26.040	18.570	23.570	20.640	20.520	22.250
MISC	1.420	.330	.020	.020	.020	.020	.020	.020
PATHMON	.130	.000	.000	.000	.000	.000	.000	.000
SERVER	14.410	20.370	13.200	16.930	17.080	12.860	15.060	15.270
TCP	.000	9.380	.000	14.680	13.820	23.740	14.060	18.100
TMF MONITOR	1.610	1.950	1.170	2.050	2.350	3.100	2.380	2.550
TMP	1.160	.000	16.480	.000	.000	.000	.000	.000
X.25	.000	.000	8.580	8.480	.000	.000	8.410	13.120
	90.210	87.840	88.400	82.660	91.360	85.920	86.020	93.520

LOADID - SQL222 D

27.

DETAILED CPU UTILIZATION BY TRANSACTION

PROCESS-TYPE	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	CPU 6	CPU 7
ATB-B	11.560	.000	.000	.000	.000	.000	.000	.000
ATB-P	.000	49.470	.000	.000	.000	.000	.000	.000
FOX	.000	.000	.000	.000	.000	.000	.000	.000
INTERRUPTS	16.230	13.390	.000	.000	.000	.000	.000	.000
MISC	.670	.050	.000	.000	.000	.000	.000	.000
PATHMON	.000	.000	.000	.000	.000	.000	.000	.000
SERVER	19.280	12.250	.000	.000	.000	.000	.000	.000
TCP	19.610	.000	.000	.000	.000	.000	.000	.000
TMF MONITOR	1.870	.480	.000	.000	.000	.000	.000	.000
TMP	6.270	.440	.000	.000	.000	.000	.000	.000
X.25	16.190	.000	.000	.000	.000	.000	.000	.000
	91.680	76.080	.000	.000	.000	.000	.000	.000

LOADID - SQL222 E

Disc Utilisation Summary per Transaction

USAGE	REQ PER TRANS	READS PER TRANS	WRITES PER TRANS	512 HITS PER TRANS	1024 HITS PER TRANS	2048 HITS PER TRANS	4096 HITS PER TRANS
ATB	3.116	1.003	2.097	1.038	.023	.989	2.068
AUDIT TRAILS	.618	.001	.376	.000	.000	.000	.001
HISTORY	1.000	.002	.027	.000	.000	.000	1.001
UNUSED	.000	.000	.000	.000	.000	.000	.000
XRAY	.030	.005	.105	.000	.000	.000	.000

Loadid - SQL222 A

Disc Utilisation Summary per Transaction

USAGE	REQ PER TRANS	READS PER TRANS	WRITES PER TRANS	512 HITS PER TRANS	1024 HITS PER TRANS	2048 HITS PER TRANS	4096 HITS PER TRANS
ATB	3.095	.990	2.082	1.035	.013	.987	2.042
AUDIT TRAILS	.617	.001	.366	.000	.000	.000	.001
HISTORY	1.000	.001	.027	.000	.000	.000	1.000
UNUSED	.000	.000	.000	.000	.000	.000	.000
XRAY	.029	.005	.073	.000	.000	.000	.000

Loadid - SQL222 B

Disc Utilisation Summary per Transaction

USAGE	REQ PER TRANS	READS PER TRANS	WRITES PER TRANS	512 HITS PER TRANS	1024 HITS PER TRANS	2048 HITS PER TRANS	4096 HITS PER TRANS
ATB	3.110	.995	2.081	1.035	.024	.988	2.053
AUDIT TRAILS	.625	.001	.371	.000	.000	.000	.001
HISTORY	1.002	.002	.027	.000	.000	.000	1.002
UNUSED	.000	.000	.000	.000	.000	.000	.000
XRAY	.030	.004	.098	.000	.000	.000	.000

Loadid - SQL222 C

Disc Utilisation Summary per Transaction

USAGE	REQ PER TRANS	READS PER TRANS	WRITES PER TRANS	512 HITS PER TRANS	1024 HITS PER TRANS	2048 HITS PER TRANS	4096 HITS PER TRANS
ATB	3.122	1.001	2.099	1.039	.028	.988	2.071
AUDIT TRAILS	.666	.001	.394	.000	.000	.000	.001
HISTORY	1.000	.002	.027	.000	.000	.000	1.000
UNUSED	.000	.000	.000	.000	.000	.000	.000
XRAY	.031	.005	.104	.000	.000	.000	.000

Loadid - SQL222 D

Disc Utilisation Summary per Transaction

USAGE	REQ PER TRANS	READS PER TRANS	WRITES PER TRANS	512 HITS PER TRANS	1024 HITS PER TRANS	2048 HITS PER TRANS	4096 HITS PER TRANS
ATB+HISTORY	4.093	1.081	2.171	1.029	.029	.953	3.014
AUDIT	1.710	.031	2.498	.000	.000	.000	.017

Loadid - SQL222 E

Disc Utilisation Summary

USAGE	SUM REQUESTS	SUM READS	SUM WRITES	512 HITS	1024 HITS	2048 HITS	4096 HITS
ATB	70702	22766	47588	23544	520	22438	46931
AUDIT TRAILS	14015	17	8537	0	0	0	19
HISTORY	22702	35	621	0	0	0	22703
UNUSED	0	0	0	0	0	0	0
XRAY	685	105	2373	0	0	0	0

Loadid - SQL222 A

Disc Utilisation Summary

USAGE	SUM REQUESTS	SUM READS	SUM WRITES	512 HITS	1024 HITS	2048 HITS	4096 HITS
ATB	70656	22610	47526	23637	296	22540	46614
AUDIT TRAILS	14093	17	8360	0	0	0	19
HISTORY	22837	23	610	0	0	0	22829
UNUSED	0	0	0	0	0	0	0
XRAY	670	123	1662	0	0	0	0

Loadid - SQL222 B

Disc Utilisation Summary

USAGE	SUM REQUESTS	SUM READS	SUM WRITES	512 HITS	1024 HITS	2048 HITS	4096 HITS
ATB	70722	22636	47327	23526	556	22470	46695
AUDIT TRAILS	14213	16	8436	0	0	0	19
HISTORY	22785	37	616	0	0	0	22778
UNUSED	0	0	0	0	0	0	0
XRAY	674	96	2236	0	0	0	0

Loadid - SQL222 C

Disc Utilisation Summary

USAGE	SUM REQUESTS	SUM READS	SUM WRITES	512 HITS	1024 HITS	2048 HITS	4096 HITS
ATB	68624	22001	46143	22833	606	21717	45530
AUDIT TRAILS	14641	12	8656	0	0	0	26
HISTORY	21991	40	603	0	0	0	21992
UNUSED	5	0	0	0	0	0	0
XRAY	692	102	2293	0	0	0	0

Loadid - SQL222 D

Disc Utilisation Summary

USAGE	SUM REQUESTS	SUM READS	SUM WRITES	512 HITS	1024 HITS	2048 HITS	4096 HITS
ATB+HISTORY	6233	1646	3307	1567	44	1452	4590
AUDIT	2604	47	3805	0	0	0	26

Loadid - SQL222 E

Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XA1	2840	314801	596820	20.300	10.708
$XA2	2832	313731	595256	20.247	10.671
$XA3	2819	312482	593533	20.188	10.629
$XA4	2770	307130	583166	19.835	10.447
$XA5	2886	319867	607406	20.658	10.879
$XA6	2859	316894	601601	20.461	10.778
$XA7	2858	317025	601598	20.462	10.783
$XA8	2839	314686	597701	20.330	10.704
	22703	2516616	4777081		

Loadid - SQL222 A

58

Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XB1	2870	318821	603234	20.518	10.844
$XB2	2830	313830	598423	20.355	10.675
$XB3	2818	312549	593314	20.181	10.631
$XB4	2821	312703	593855	20.199	10.636
$XB5	2899	321092	610181	20.756	10.922
$XB6	2850	316010	599866	20.405	10.749
$XB7	2856	316438	601244	20.450	10.763
$XB8	2874	318614	604946	20.576	10.837
	22818	2530057	4805063		

Loadid - SQL222 B

Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XC1	2841	314757	596609	20.293	10.706
$XC2	2861	316795	601023	20.443	10.775
$XC3	2859	316708	601765	20.468	10.772
$XC4	2853	316641	600495	20.425	10.770
$XC5	2834	313941	596446	20.287	10.678
$XC6	2867	317694	603238	20.518	10.806
$XC7	2848	315808	599476	20.392	10.743
$XC8	2815	311950	591725	20.128	10.611
	22778	2524294	4790777		

Loadid - SQL222 C

87

Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XD1	2726	301988	572887	19.486	10.272
$XD2	2741	304185	576263	19.601	10.346
$XD3	2725	309360	573939	19.522	10.523
$XD4	2776	307749	584488	19.881	10.468
$XD5	2751	304906	578993	19.692	10.370
$XD6	2714	300696	571294	19.430	10.227
$XD7	2789	309065	587079	19.968	10.512
$XD8	2769	306746	582872	19.825	10.433
	21991	2444695	4627815		

Loadid - SQL222 D

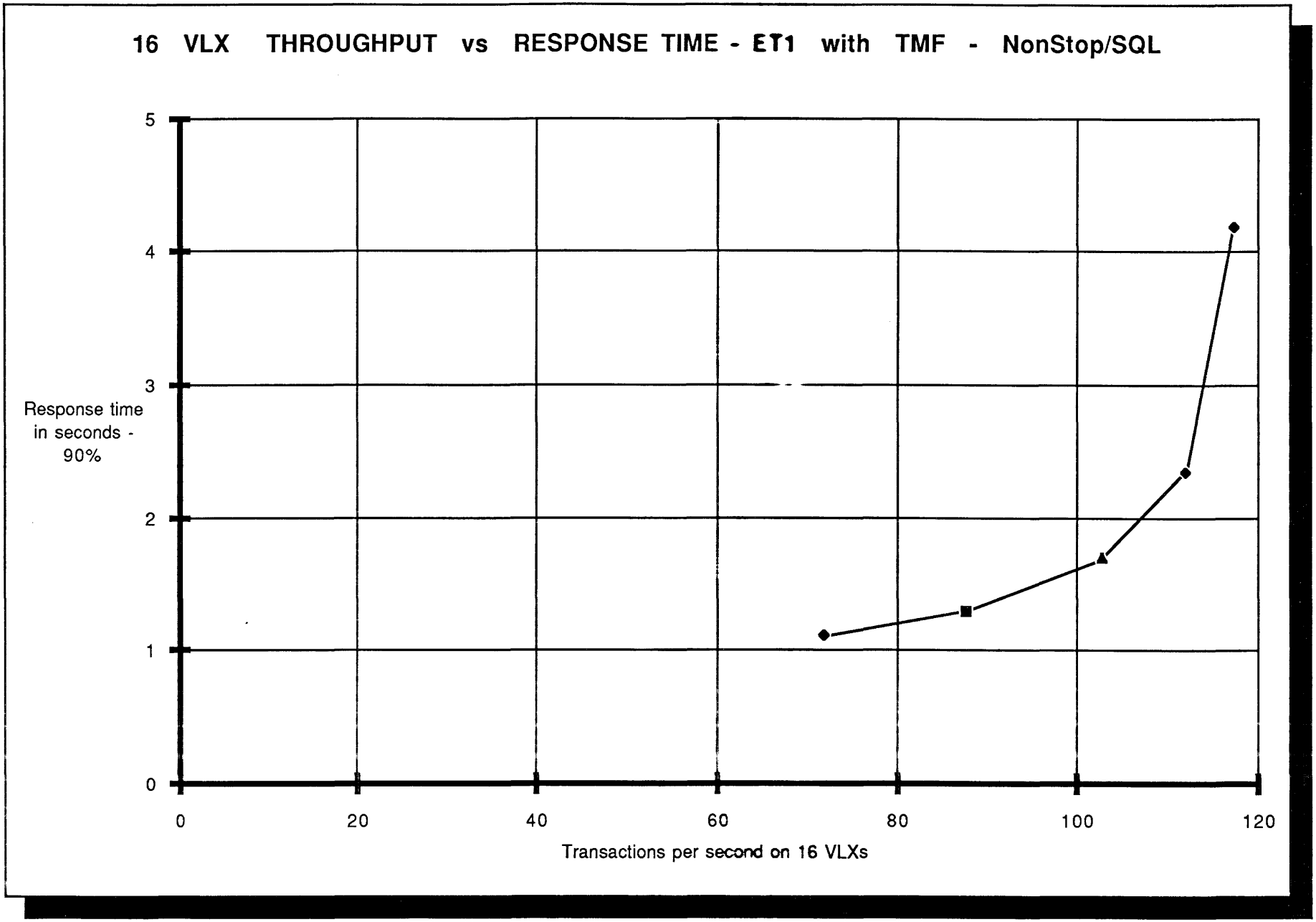
Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XE1	1525	169009	323432	11.001	5.748
	1525	169009	323432		

Loadid - SQL222 E

Communications line summary per transactions

LOADID	Transactions	Output per trans	input per trans
SQL222 A	22691	200.106	100.026
SQL222 B	22830	199.895	99.952
SQL222 C	22740	200.325	100.110
SQL222 D	21982	200.082	99.982
SQL222 E	1523	200.263	100.000


16 VLX THROUGHPUT vs RESPONSE TIME - ET1 with TMF - NonStop/SQL
Response time in seconds - 90%
5
4
3
2
1
0
0
20
40
60
80
100
120
Transactions per second on 16 VLXs

Simulator Report

NonStop/SQL

2 * 8 VLX SYSTEMS

Throughput - 117 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test ALL;RUN05-16;SQL
Test Date                  =   87-02-20
Test Length                =   20:02:38 - 20:22:18  (1180 sec)
Window Length              =   20:06:00 - 20:21:00  (900 sec)
Tx Log File                =   \DRIVER.$DM03.RU16N05.TLOGSUMZ

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    8.500 tx/sec
Tx Arrival Rate/Terminal   =    0.106 tx/sec
Tx Inter-Arrival Time      =    9.412 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) ---------------------------------

Tx Throughput                =  117.567 tx/sec
Tx Response Time Average     =    2.680 sec
                 Std Dev     =    1.119 sec
                 Minimum     =    0.540 sec
                 Maximum     =   27.125 sec
                 Count       =   105810 #

Tx Response Time 10%         =    1.550 sec  ( 11.1%)
                 20%         =    1.800 sec  ( 20.9%)
                 30%         =    2.000 sec  ( 30.1%)
                 40%         =    2.250 sec  ( 41.8%)
                 50%         =    2.450 sec  ( 50.4%)
                 60%         =    2.750 sec  ( 61.6%)
                 70%         =    3.050 sec  ( 70.6%)
                 80%         =    3.500 sec  ( 80.6%)
                 90%         =    4.200 sec  ( 90.5%)
                 95%         =    4.800 sec  ( 95.0%)

        Tx Think Average     =    9.426 sec
                 Std Dev     =    9.384 sec
                 Minimum     =    0.001 sec
                 Maximum     =  112.426 sec
                 Count       =    80073 #    ( 75.7%)

        Tx Delay Average     =    1.621 sec
                 Std Dev     =    1.219 sec
                 Minimum     =    0.000 sec
                 Maximum     =   26.744 sec
                 Count       =    25737 #    ( 24.3%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
2.680 s	1.119 s	0.540 s	27.125 s	105810	117.567 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	
0.100	0	0.0	0.0	
0.200	0	0.0	0.0	
0.300	0	0.0	0.0	
0.400	0	0.0	0.0	
0.500	0	0.0	0.0	
0.600	4	0.0	0.0	
0.700	20	0.0	0.0	
0.800	67	0.1	0.1	
0.900	190	0.2	0.3	*
1.000	385	0.4	0.6	***
1.100	784	0.7	1.4	******
1.200	1272	1.2	2.6	**********
1.300	1830	1.7	4.3	**********
1.400	2412	2.3	6.6	**********
1.500	3134	3.0	9.5	**********
1.600	3575	3.4	12.9	**********
1.700	4038	3.8	16.7	**********
1.800	4425	4.2	20.9	**********
1.900	4845	4.6	25.5	**********
2.000	4912	4.6	30.1	**********
2.100	4962	4.7	34.8	**********
2.200	4948	4.7	39.5	**********
2.300	4815	4.6	44.1	**********
2.400	4602	4.3	48.4	**********
2.500	4244	4.0	52.4	**********
2.600	4105	3.9	56.3	**********
2.700	3760	3.6	59.9	**********
2.800	3614	3.4	63.3	**********
2.900	3276	3.1	66.4	**********
3.000	3037	2.9	69.2	**********
3.100	2738	2.6	71.8	**********
3.200	2616	2.5	74.3	**********
3.300	2378	2.2	76.5	**********
3.400	2237	2.1	78.7	**********
3.500	2055	1.9	80.6	**********
3.600	2009	1.9	82.5	**********
3.700	1769	1.7	84.2	**********
3.800	1574	1.5	85.7	**********
3.900	1456	1.4	87.0	**********
4.000	1348	1.3	88.3	**********
4.100	1191	1.1	89.4	**********
4.200	1138	1.1	90.5	*********
4.300	1026	1.0	91.5	********

4.400	927	0.9	92.4	|*******
4.500	818	0.8	93.1	|******
4.600	726	0.7	93.8	|******
4.700	672	0.6	94.4	|*****
4.800	589	0.6	95.0	|****
4.900	553	0.5	95.5	|****
5.000	488	0.5	96.0	|****
5.100	455	0.4	96.4	|***
5.200	393	0.4	96.8	|***
5.300	364	0.3	97.1	|***
5.400	322	0.3	97.4	|**
5.500	264	0.2	97.7	|**
5.600	231	0.2	97.9	|*
5.700	214	0.2	98.1	|*
5.800	196	0.2	98.3	|*
5.900	199	0.2	98.5	|*
6.000	156	0.1	98.6	|*
6.100	131	0.1	98.8	|*
6.200	131	0.1	98.9	|*
6.300	115	0.1	99.0	|
6.400	105	0.1	99.1	|
6.500	89	0.1	99.2	|
6.600	85	0.1	99.2	|
6.700	92	0.1	99.3	|
6.800	74	0.1	99.4	|
6.900	62	0.1	99.5	|
7.000	65	0.1	99.5	|
7.100	44	0.0	99.6	|
7.200	48	0.0	99.6	|
7.300	42	0.0	99.7	|
7.400	47	0.0	99.7	|
7.500	42	0.0	99.7	|
7.600	32	0.0	99.8	|
7.700	16	0.0	99.8	|
7.800	18	0.0	99.8	|
7.900	23	0.0	99.8	|
8.000	17	0.0	99.8	|
8.100	17	0.0	99.9	|
8.200	14	0.0	99.9	|
8.300	18	0.0	99.9	|
8.400	14	0.0	99.9	|
8.500	14	0.0	99.9	|
8.600	12	0.0	99.9	|
8.700	9	0.0	99.9	|
8.800	8	0.0	99.9	|
8.900	9	0.0	99.9	|
9.000	7	0.0	100.0	|
9.100	7	0.0	100.0	|
9.200	5	0.0	100.0	|
9.300	1	0.0	100.0	|
9.400	2	0.0	100.0	|
9.500	2	0.0	100.0	|
9.600	3	0.0	100.0	|
9.700	2	0.0	100.0	|

9.800	1	0.0	100.0
9.900	2	0.0	100.0
10.000	0	0.0	100.0
10.100	4	0.0	100.0
10.200	1	0.0	100.0
10.300	0	0.0	100.0
10.400	2	0.0	100.0
10.500	3	0.0	100.0
10.600	3	0.0	100.0
10.700	3	0.0	100.0
10.800	0	0.0	100.0
10.900	1	0.0	100.0
11.000	3	0.0	100.0
11.100	2	0.0	100.0
11.200	1	0.0	100.0
11.300	0	0.0	100.0
11.400	0	0.0	100.0
11.500	0	0.0	100.0
11.600	0	0.0	100.0
11.700	0	0.0	100.0
11.800	0	0.0	100.0
11.900	0	0.0	100.0
12.000	0	0.0	100.0
12.100	0	0.0	100.0
12.200	0	0.0	100.0
12.300	0	0.0	100.0
12.400	0	0.0	100.0
12.500	0	0.0	100.0
12.600	0	0.0	100.0
12.700	0	0.0	100.0
12.800	0	0.0	100.0
12.900	0	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	0	0.0	100.0
13.500	0	0.0	100.0
13.600	0	0.0	100.0
13.700	0	0.0	100.0
13.800	0	0.0	100.0
13.900	0	0.0	100.0
14.000	0	0.0	100.0
14.100	0	0.0	100.0
14.200	0	0.0	100.0
14.300	0	0.0	100.0
14.400	0	0.0	100.0
14.500	0	0.0	100.0
14.600	0	0.0	100.0
14.700	0	0.0	100.0
14.800	0	0.0	100.0
14.900	0	0.0	100.0
15.000	0	0.0	100.0
15.100	0	0.0	100.0

15.200	0	0.0	100.0
15.300	0	0.0	100.0
15.400	0	0.0	100.0
15.500	0	0.0	100.0
15.600	0	0.0	100.0
15.700	0	0.0	100.0
15.800	0	0.0	100.0
15.900	0	0.0	100.0
16.000	0	0.0	100.0
16.100	0	0.0	100.0
16.200	0	0.0	100.0
16.300	0	0.0	100.0
16.400	0	0.0	100.0
16.500	0	0.0	100.0
16.600	0	0.0	100.0
16.700	0	0.0	100.0
16.800	0	0.0	100.0
16.900	0	0.0	100.0
17.000	0	0.0	100.0
17.100	0	0.0	100.0
17.200	0	0.0	100.0
17.300	0	0.0	100.0
17.400	0	0.0	100.0
17.500	0	0.0	100.0
17.600	0	0.0	100.0
17.700	0	0.0	100.0
17.800	0	0.0	100.0
17.900	0	0.0	100.0
18.000	0	0.0	100.0
18.100	0	0.0	100.0
18.200	0	0.0	100.0
18.300	0	0.0	100.0
18.400	0	0.0	100.0
18.500	0	0.0	100.0
18.600	0	0.0	100.0
18.700	0	0.0	100.0
18.800	0	0.0	100.0
18.900	0	0.0	100.0
19.000	0	0.0	100.0
19.100	0	0.0	100.0
19.200	0	0.0	100.0
19.300	0	0.0	100.0
19.400	0	0.0	100.0
19.500	0	0.0	100.0
19.600	0	0.0	100.0
19.700	0	0.0	100.0
19.800	0	0.0	100.0
19.900	0	0.0	100.0
20.000	0	0.0	100.0
20.100	0	0.0	100.0
20.200	0	0.0	100.0
20.300	0	0.0	100.0
20.400	0	0.0	100.0
20.500	0	0.0	100.0

20.600	0	0.0	100.0
20.700	0	0.0	100.0
20.800	0	0.0	100.0
20.900	0	0.0	100.0
21.000	0	0.0	100.0
21.100	0	0.0	100.0
21.200	0	0.0	100.0
21.300	0	0.0	100.0
21.400	0	0.0	100.0
21.500	0	0.0	100.0
21.600	0	0.0	100.0
21.700	0	0.0	100.0
21.800	0	0.0	100.0
21.900	0	0.0	100.0
22.000	0	0.0	100.0
22.100	0	0.0	100.0
22.200	0	0.0	100.0
22.300	0	0.0	100.0
22.400	0	0.0	100.0
22.500	0	0.0	100.0
22.600	0	0.0	100.0
22.700	0	0.0	100.0
22.800	0	0.0	100.0
22.900	0	0.0	100.0
23.000	0	0.0	100.0
23.100	0	0.0	100.0
23.200	0	0.0	100.0
23.300	0	0.0	100.0
23.400	0	0.0	100.0
23.500	0	0.0	100.0
23.600	0	0.0	100.0
23.700	0	0.0	100.0
23.800	0	0.0	100.0
23.900	0	0.0	100.0
24.000	1	0.0	100.0
24.100	0	0.0	100.0
24.200	0	0.0	100.0
24.300	0	0.0	100.0
24.400	0	0.0	100.0
24.500	0	0.0	100.0
24.600	0	0.0	100.0
24.700	0	0.0	100.0
24.800	0	0.0	100.0
24.900	0	0.0	100.0
25.000	0	0.0	100.0
25.100	0	0.0	100.0
25.200	0	0.0	100.0
25.300	0	0.0	100.0
25.400	0	0.0	100.0
25.500	0	0.0	100.0
25.600	0	0.0	100.0
25.700	0	0.0	100.0
25.800	0	0.0	100.0
25.900	0	0.0	100.0

26.000	0	0.0	100.0	|
26.100	0	0.0	100.0	|
26.200	0	0.0	100.0	|
26.300	0	0.0	100.0	|
26.400	0	0.0	100.0	|
26.500	0	0.0	100.0	|
26.600	0	0.0	100.0	|
26.700	0	0.0	100.0	|
26.800	1	0.0	100.0	|
26.900	1	0.0	100.0	|
27.000	0	0.0	100.0	|
27.100	0	0.0	100.0	|
27.200	1	0.0	100.0	|

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
9.813 s	9.086 s	0.605 s	114.017 s	105810	117.567 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	71	0.1	8.8	| T
2.000	6034	5.7	8.0	|****************** T
3.000	13832	13.1	7.3	|**********************T*********************
4.000	12492	11.8	6.7	|**********************T***************
5.000	9323	8.8	6.1	|********************T********
6.000	7301	6.9	5.5	|******************T****
7.000	5992	5.7	5.1	|****************T**
8.000	5244	5.0	4.6	|**************T
9.000	4742	4.5	4.2	|*************T
10.000	4061	3.8	3.8	|***********T
11.000	3653	3.5	3.5	|**********T
12.000	3353	3.2	3.2	|********\T
13.000	2950	2.8	2.9	|*******\T
14.000	2772	2.6	2.7	|*******\T
15.000	2442	2.3	2.4	|******\T
16.000	2179	2.1	2.2	|*****\T
17.000	1925	1.8	2.0	|****\T
18.000	1738	1.6	1.8	|****\T
19.000	1657	1.6	1.7	|****T
20.000	1350	1.3	1.5	|****T
21.000	1336	1.3	1.4	|***T
22.000	1148	1.1	1.3	|***T
23.000	1035	1.0	1.2	|***T
24.000	983	0.9	1.1	|**T
25.000	854	0.8	1.0	|**T
26.000	716	0.7	0.9	|**T
27.000	687	0.6	0.8	|**T
28.000	608	0.6	0.7	|*T
29.000	517	0.5	0.7	|*T
30.000	454	0.4	0.6	|*T
31.000	461	0.4	0.6	|*T
32.000	432	0.4	0.5	|*T
33.000	363	0.3	0.5	|T
34.000	308	0.3	0.4	|T
35.000	289	0.3	0.4	|T
36.000	273	0.3	0.4	|T
37.000	240	0.2	0.3	|T
38.000	211	0.2	0.3	|T
39.000	178	0.2	0.3	|T
40.000	156	0.1	0.2	|T
41.000	126	0.1	0.2	|T
42.000	136	0.1	0.2	|T
43.000	101	0.1	0.2	|T
44.000	127	0.1	0.2	|T

45.000	73	0.1	0.2
46.000	83	0.1	0.1
47.000	95	0.1	0.1
48.000	79	0.1	0.1
49.000	64	0.1	0.1
50.000	74	0.1	0.1
51.000	68	0.1	0.1
52.000	25	0.0	0.1
53.000	34	0.0	0.1
54.000	42	0.0	0.1
55.000	32	0.0	0.1
56.000	24	0.0	0.1
57.000	28	0.0	0.1
58.000	21	0.0	0.0
59.000	14	0.0	0.0
60.000	20	0.0	0.0
61.000	16	0.0	0.0
62.000	15	0.0	0.0
63.000	8	0.0	0.0
64.000	21	0.0	0.0
65.000	8	0.0	0.0
66.000	18	0.0	0.0
67.000	4	0.0	0.0
68.000	10	0.0	0.0
69.000	10	0.0	0.0
70.000	6	0.0	0.0
71.000	5	0.0	0.0
72.000	4	0.0	0.0
73.000	4	0.0	0.0
74.000	4	0.0	0.0
75.000	4	0.0	0.0
76.000	10	0.0	0.0
77.000	1	0.0	0.0
78.000	2	0.0	0.0
79.000	0	0.0	0.0
80.000	2	0.0	0.0
81.000	4	0.0	0.0
82.000	6	0.0	0.0
83.000	2	0.0	0.0
84.000	2	0.0	0.0
85.000	2	0.0	0.0
86.000	0	0.0	0.0
87.000	0	0.0	0.0
88.000	4	0.0	0.0
89.000	0	0.0	0.0
90.000	0	0.0	0.0
91.000	0	0.0	0.0
92.000	2	0.0	0.0
93.000	2	0.0	0.0
94.000	2	0.0	0.0
95.000	0	0.0	0.0
96.000	0	0.0	0.0
97.000	2	0.0	0.0
98.000	0	0.0	0.0

99.000	0	0.0	0.0
100.000	0	0.0	0.0
101.000	0	0.0	0.0
102.000	0	0.0	0.0
103.000	0	0.0	0.0
104.000	0	0.0	0.0
105.000	0	0.0	0.0
106.000	0	0.0	0.0
107.000	0	0.0	0.0
108.000	0	0.0	0.0
109.000	0	0.0	0.0
110.000	0	0.0	0.0
111.000	0	0.0	0.0
112.000	2	0.0	0.0
113.000	0	0.0	0.0
114.000	0	0.0	0.0
115.000	2	0.0	0.0

Simulator Report

NonStop/SQL

2 * 8 VLX SYSTEMS

Throughput - 112 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                 =   Test ALL;RUN01-16;SQL
Test Date                  =   87-02-20
Test Length                =   17:18:58 - 17:41:27  (1348 sec)
Window Length              =   17:24:00 - 17:39:00  (900 sec)
Tx Log File                =   \DRIVER.$DM03.RU16N01.TLOGSUMZ

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    7.500 tx/sec
Tx Arrival Rate/Terminal   =    0.094 tx/sec
Tx Inter-Arrival Time      =   10.667 sec
```

----- MEASUREMENT RESULTS (vs 2.03) ----------

Tx Throughput		=	112.177 tx/sec	
Tx Response Time	Average	=	1.607 sec	
	Std Dev	=	0.554 sec	
	Minimum	=	0.408 sec	
	Maximum	=	19.814 sec	
	Count	=	100959 #	
Tx Response Time	10%	=	1.050 sec	(12.1%)
	20%	=	1.200 sec	(23.8%)
	30%	=	1.300 sec	(32.4%)
	40%	=	1.400 sec	(41.4%)
	50%	=	1.550 sec	(53.8%)
	60%	=	1.650 sec	(61.2%)
	70%	=	1.800 sec	(70.6%)
	80%	=	2.000 sec	(80.1%)
	90%	=	2.350 sec	(90.4%)
	95%	=	2.650 sec	(95.0%)
Tx Think	Average	=	10.664 sec	
	Std Dev	=	10.626 sec	
	Minimum	=	0.001 sec	
	Maximum	=	127.996 sec	
	Count	=	87080 #	(86.3%)
Tx Delay	Average	=	0.911 sec	
	Std Dev	=	0.644 sec	
	Minimum	=	0.000 sec	
	Maximum	=	14.718 sec	
	Count	=	13879 #	(13.7%)

```
-------------------------------------------------------------------------
                  *  ET1 Response Time Distribution  *
-------------------------------------------------------------------------
   Mean         Std          Min          Max          Count      Thruput
   1.607 s      0.554 s      0.408 s     19.814 s     100959      112.177 tx/s
-------------------------------------------------------------------------

RespTime  Count   Pct   CumPct
--------  -----  -----  ------
   0.000      0    0.0     0.0 |
   0.100      0    0.0     0.0 |
   0.200      0    0.0     0.0 |
   0.300      0    0.0     0.0 |
   0.400      0    0.0     0.0 |
   0.500      8    0.0     0.0 |
   0.600    107    0.1     0.1 |
   0.700    379    0.4     0.5 |*
   0.800   1311    1.3     1.8 |******
   0.900   2847    2.8     4.6 |*************
   1.000   4416    4.4     9.0 |*******************
   1.100   6686    6.6    15.6 |*******************************
   1.200   8297    8.2    23.8 |****************************************
   1.300   8669    8.6    32.4 |******************************************
   1.400   9047    9.0    41.4 |*******************************************
   1.500   8580    8.5    49.9 |****************************************
   1.600   7734    7.7    57.5 |************************************
   1.700   7074    7.0    64.5 |*********************************
   1.800   6102    6.0    70.6 |****************************
   1.900   5112    5.1    75.6 |************************
   2.000   4467    4.4    80.1 |*********************
   2.100   3716    3.7    83.7 |*****************
   2.200   3088    3.1    86.8 |**************
   2.300   2580    2.6    89.4 |************
   2.400   2085    2.1    91.4 |*********
   2.500   1729    1.7    93.1 |********
   2.600   1363    1.4    94.5 |******
   2.700   1066    1.1    95.5 |****
   2.800    882    0.9    96.4 |****
   2.900    720    0.7    97.1 |***
   3.000    570    0.6    97.7 |**
   3.100    485    0.5    98.2 |**
   3.200    349    0.3    98.5 |*
   3.300    303    0.3    98.8 |*
   3.400    237    0.2    99.1 |*
   3.500    174    0.2    99.2 |
   3.600    150    0.1    99.4 |
   3.700    108    0.1    99.5 |
   3.800    101    0.1    99.6 |
   3.900     86    0.1    99.7 |
   4.000     81    0.1    99.8 |
   4.100     49    0.0    99.8 |
   4.200     31    0.0    99.8 |
   4.300     22    0.0    99.9 |
```

4.400	29	0.0	99.9
4.500	21	0.0	99.9
4.600	17	0.0	99.9
4.700	18	0.0	99.9
4.800	11	0.0	99.9
4.900	17	0.0	100.0
5.000	9	0.0	100.0
5.100	5	0.0	100.0
5.200	4	0.0	100.0
5.300	2	0.0	100.0
5.400	5	0.0	100.0
5.500	0	0.0	100.0
5.600	2	0.0	100.0
5.700	1	0.0	100.0
5.800	0	0.0	100.0
5.900	0	0.0	100.0
6.000	0	0.0	100.0
6.100	0	0.0	100.0
6.200	1	0.0	100.0
6.300	3	0.0	100.0
6.400	1	0.0	100.0
6.500	0	0.0	100.0
6.600	0	0.0	100.0
6.700	0	0.0	100.0
6.800	0	0.0	100.0
6.900	0	0.0	100.0
7.000	0	0.0	100.0
7.100	0	0.0	100.0
7.200	0	0.0	100.0
7.300	1	0.0	100.0
7.400	0	0.0	100.0
7.500	0	0.0	100.0
7.600	0	0.0	100.0
7.700	0	0.0	100.0
7.800	0	0.0	100.0
7.900	0	0.0	100.0
8.000	0	0.0	100.0
8.100	0	0.0	100.0
8.200	0	0.0	100.0
8.300	0	0.0	100.0
8.400	0	0.0	100.0
8.500	0	0.0	100.0
8.600	0	0.0	100.0
8.700	0	0.0	100.0
8.800	0	0.0	100.0
8.900	0	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	0	0.0	100.0
9.500	0	0.0	100.0
9.600	0	0.0	100.0
9.700	0	0.0	100.0

9.800	0	0.0	100.0
9.900	0	0.0	100.0
10.000	0	0.0	100.0
10.100	0	0.0	100.0
10.200	0	0.0	100.0
10.300	0	0.0	100.0
10.400	0	0.0	100.0
10.500	0	0.0	100.0
10.600	0	0.0	100.0
10.700	0	0.0	100.0
10.800	0	0.0	100.0
10.900	0	0.0	100.0
11.000	0	0.0	100.0
11.100	0	0.0	100.0
11.200	0	0.0	100.0
11.300	0	0.0	100.0
11.400	0	0.0	100.0
11.500	0	0.0	100.0
11.600	0	0.0	100.0
11.700	0	0.0	100.0
11.800	0	0.0	100.0
11.900	0	0.0	100.0
12.000	0	0.0	100.0
12.100	0	0.0	100.0
12.200	0	0.0	100.0
12.300	0	0.0	100.0
12.400	0	0.0	100.0
12.500	0	0.0	100.0
12.600	0	0.0	100.0
12.700	0	0.0	100.0
12.800	0	0.0	100.0
12.900	0	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	0	0.0	100.0
13.500	0	0.0	100.0
13.600	0	0.0	100.0
13.700	0	0.0	100.0
13.800	0	0.0	100.0
13.900	0	0.0	100.0
14.000	0	0.0	100.0
14.100	0	0.0	100.0
14.200	0	0.0	100.0
14.300	0	0.0	100.0
14.400	0	0.0	100.0
14.500	0	0.0	100.0
14.600	0	0.0	100.0
14.700	0	0.0	100.0
14.800	0	0.0	100.0
14.900	0	0.0	100.0
15.000	0	0.0	100.0
15.100	0	0.0	100.0

15.200	0	0.0	100.0
15.300	0	0.0	100.0
15.400	0	0.0	100.0
15.500	0	0.0	100.0
15.600	0	0.0	100.0
15.700	0	0.0	100.0
15.800	0	0.0	100.0
15.900	0	0.0	100.0
16.000	0	0.0	100.0
16.100	0	0.0	100.0
16.200	0	0.0	100.0
16.300	0	0.0	100.0
16.400	0	0.0	100.0
16.500	0	0.0	100.0
16.600	0	0.0	100.0
16.700	0	0.0	100.0
16.800	0	0.0	100.0
16.900	0	0.0	100.0
17.000	0	0.0	100.0
17.100	0	0.0	100.0
17.200	0	0.0	100.0
17.300	0	0.0	100.0
17.400	0	0.0	100.0
17.500	0	0.0	100.0
17.600	0	0.0	100.0
17.700	0	0.0	100.0
17.800	0	0.0	100.0
17.900	0	0.0	100.0
18.000	0	0.0	100.0
18.100	0	0.0	100.0
18.200	0	0.0	100.0
18.300	0	0.0	100.0
18.400	0	0.0	100.0
18.500	0	0.0	100.0
18.600	0	0.0	100.0
18.700	0	0.0	100.0
18.800	0	0.0	100.0
18.900	0	0.0	100.0
19.000	0	0.0	100.0
19.100	0	0.0	100.0
19.200	0	0.0	100.0
19.300	0	0.0	100.0
19.400	0	0.0	100.0
19.500	0	0.0	100.0
19.600	0	0.0	100.0
19.700	0	0.0	100.0
19.800	0	0.0	100.0
19.900	1	0.0	100.0

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
10.804 s	10.524 s	0.522 s	129.222 s	100959	112.177 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	806	0.8	8.4	|** T
2.000	12955	12.8	7.7	|************************T****************
3.000	10283	10.2	7.0	|**********************T**********
4.000	7428	7.4	6.5	|********************T***
5.000	6213	6.2	5.9	|******************T*
6.000	5577	5.5	5.4	|*****************T
7.000	5110	5.1	5.0	|****************T
8.000	4617	4.6	4.5	|***************T
9.000	4323	4.3	4.2	|**************T
10.000	3939	3.9	3.8	|***********T
11.000	3524	3.5	3.5	|***********T
12.000	3208	3.2	3.2	|*********T
13.000	2863	2.8	2.9	|*********T
14.000	2810	2.8	2.7	|********T
15.000	2448	2.4	2.5	|*******T
16.000	2329	2.3	2.3	|******T
17.000	1978	2.0	2.1	|******T
18.000	1846	1.8	1.9	|*****T
19.000	1672	1.7	1.7	|*****T
20.000	1474	1.5	1.6	|****T
21.000	1407	1.4	1.5	|****T
22.000	1302	1.3	1.3	|***T
23.000	1153	1.1	1.2	|***T
24.000	1043	1.0	1.1	|***T
25.000	978	1.0	1.0	|**T
26.000	870	0.9	0.9	|**T
27.000	823	0.8	0.9	|**T
28.000	769	0.8	0.8	|**T
29.000	615	0.6	0.7	|*T
30.000	604	0.6	0.7	|*T
31.000	568	0.6	0.6	|*T
32.000	470	0.5	0.6	|*T
33.000	429	0.4	0.5	|*T
34.000	387	0.4	0.5	|*T
35.000	388	0.4	0.4	|T
36.000	370	0.4	0.4	|T
37.000	312	0.3	0.4	|T
38.000	294	0.3	0.3	|T
39.000	238	0.2	0.3	|T
40.000	247	0.2	0.3	|T
41.000	200	0.2	0.3	|T
42.000	202	0.2	0.2	|T
43.000	187	0.2	0.2	|T
44.000	143	0.1	0.2	|T

45.000	148	0.1	0.2	T
46.000	93	0.1	0.2	T
47.000	112	0.1	0.1	
48.000	105	0.1	0.1	
49.000	87	0.1	0.1	
50.000	91	0.1	0.1	
51.000	63	0.1	0.1	
52.000	63	0.1	0.1	
53.000	88	0.1	0.1	
54.000	66	0.1	0.1	
55.000	74	0.1	0.1	
56.000	55	0.1	0.1	
57.000	58	0.1	0.1	
58.000	56	0.1	0.1	
59.000	24	0.0	0.1	
60.000	25	0.0	0.0	
61.000	32	0.0	0.0	
62.000	32	0.0	0.0	
63.000	29	0.0	0.0	
64.000	17	0.0	0.0	
65.000	21	0.0	0.0	
66.032	16	0.0	0.0	
67.000	13	0.0	0.0	
68.000	18	0.0	0.0	
69.000	12	0.0	0.0	
70.000	11	0.0	0.0	
71.000	10	0.0	0.0	
72.000	9	0.0	0.0	
73.000	12	0.0	0.0	
74.000	16	0.0	0.0	
75.000	8	0.0	0.0	
76.000	4	0.0	0.0	
77.000	9	0.0	0.0	
78.000	5	0.0	0.0	
79.000	10	0.0	0.0	
80.000	1	0.0	0.0	
81.000	10	0.0	0.0	
82.000	2	0.0	0.0	
83.000	4	0.0	0.0	
84.000	2	0.0	0.0	
85.000	4	0.0	0.0	
86.000	8	0.0	0.0	
87.000	2	0.0	0.0	
88.032	0	0.0	0.0	
89.000	1	0.0	0.0	
90.000	1	0.0	0.0	
91.000	0	0.0	0.0	
92.000	8	0.0	0.0	
93.000	2	0.0	0.0	
94.000	2	0.0	0.0	
95.000	2	0.0	0.0	
96.000	2	0.0	0.0	
97.000	0	0.0	0.0	
98.000	0	0.0	0.0	

99.000	2	0.0	0.0
100.000	0	0.0	0.0
101.000	0	0.0	0.0
102.000	0	0.0	0.0
103.000	0	0.0	0.0
104.000	2	0.0	0.0
105.000	0	0.0	0.0
106.000	2	0.0	0.0
107.000	2	0.0	0.0
108.000	0	0.0	0.0
109.032	0	0.0	0.0
110.000	2	0.0	0.0
111.000	0	0.0	0.0
112.000	0	0.0	0.0
113.000	0	0.0	0.0
114.000	0	0.0	0.0
115.000	0	0.0	0.0
116.000	0	0.0	0.0
117.000	0	0.0	0.0
118.000	0	0.0	0.0
119.000	0	0.0	0.0
120.000	0	0.0	0.0
121.000	0	0.0	0.0
122.000	0	0.0	0.0
123.000	0	0.0	0.0
124.000	0	0.0	0.0
125.000	0	0.0	0.0
126.000	0	0.0	0.0
127.000	2	0.0	0.0
128.000	0	0.0	0.0
129.000	0	0.0	0.0
130.032	2	0.0	0.0

Simulator - Report

NonStop/SQL

2 * 8 VLX SYSTEMS

Throughput - 103 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test ALL;RUN02-16;SQL
Test Date                  =   87-02-20
Test Length                =   18:04:35 - 18:24:53  (1218 sec)
Window Length              =   18:07:00 - 18:22:00  (900 sec)
Tx Log File                =   \DRIVER.$DM03.RU16N02.TLOGSUMZ

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    6.500 tx/sec
Tx Arrival Rate/Terminal   =    0.081 tx/sec
Tx Inter-Arrival Time      =   12.308 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) ---------------------------------

Tx Throughput              =  102.976 tx/sec
Tx Response Time Average   =    1.185 sec
                 Std Dev   =    0.384 sec
                 Minimum   =    0.382 sec
                 Maximum   =    4.919 sec
                 Count     =    92678 #

Tx Response Time 10%       =    0.800 sec  ( 12.2%)
                 20%       =    0.900 sec  ( 23.0%)
                 30%       =    1.000 sec  ( 35.3%)
                 40%       =    1.050 sec  ( 42.1%)
                 50%       =    1.150 sec  ( 54.9%)
                 60%       =    1.200 sec  ( 60.4%)
                 70%       =    1.350 sec  ( 73.4%)
                 80%       =    1.500 sec  ( 82.6%)
                 90%       =    1.700 sec  ( 90.5%)
                 95%       =    1.950 sec  ( 95.6%)

        Tx Think Average   =   12.325 sec
                 Std Dev   =   12.338 sec
                 Minimum   =    0.001 sec
                 Maximum   =  147.891 sec
                 Count     =    84301 #    ( 91.0%)

        Tx Delay Average   =    0.667 sec
                 Std Dev   =    0.461 sec
                 Minimum   =    0.000 sec
                 Maximum   =    3.881 sec
                 Count     =     8377 #    (  9.0%)

-----------------------------------------------------------------------
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.185 s	0.384 s	0.382 s	4.919 s	92678	102.976 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	4	0.0	0.0	|
0.500	198	0.2	0.2	|
0.600	1109	1.2	1.4	|***
0.700	3114	3.4	4.8	|**********
0.800	6900	7.4	12.2	|**********
0.900	9962	10.7	23.0	|**********
1.000	11445	12.3	35.3	|**********
1.100	12344	13.3	48.6	|**********
1.200	10885	11.7	60.4	|**********
1.300	8489	9.2	69.5	|**********
1.400	6718	7.2	76.8	|**********
1.500	5340	5.8	82.6	|**********
1.600	4208	4.5	87.1	|**********
1.700	3115	3.4	90.5	|**********
1.800	2323	2.5	93.0	|*******
1.900	1733	1.9	94.8	|*****
2.000	1299	1.4	96.2	|****
2.100	931	1.0	97.2	|***
2.200	718	0.8	98.0	|**
2.300	475	0.5	98.5	|*
2.400	370	0.4	98.9	|*
2.500	278	0.3	99.2	|
2.600	166	0.2	99.4	|
2.700	133	0.1	99.5	|
2.800	107	0.1	99.7	|
2.900	80	0.1	99.7	|
3.000	55	0.1	99.8	|
3.100	47	0.1	99.9	|
3.200	32	0.0	99.9	|
3.300	25	0.0	99.9	|
3.400	16	0.0	99.9	|
3.500	14	0.0	100.0	|
3.600	12	0.0	100.0	|
3.700	5	0.0	100.0	|
3.800	6	0.0	100.0	|
3.900	4	0.0	100.0	|
4.000	4	0.0	100.0	|
4.100	2	0.0	100.0	|
4.200	2	0.0	100.0	|
4.300	3	0.0	100.0	|
4.400	0	0.0	100.0	|

4.500	3	0.0	100.0
4.600	1	0.0	100.0
4.700	0	0.0	100.0
4.800	1	0.0	100.0
4.900	1	0.0	100.0
5.000	1	0.0	100.0

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
12.396 s	12.280 s	0.442 s	149.103 s	92678	102.976 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	2480	2.7	7.7	|******** T
2.000	10909	11.8	7.1	|***********************T****************
3.000	6528	7.0	6.6	|**********************T**
4.000	5758	6.2	6.1	|********************T
5.000	5269	5.7	5.6	|******************T
6.000	4790	5.2	5.2	|*****************T
7.000	4255	4.6	4.8	|****************T
8.000	4130	4.5	4.4	|***************T
9.000	3736	4.0	4.1	|*************T
10.000	3572	3.9	3.7	|************T
11.000	3159	3.4	3.5	|***********T
12.000	3026	3.3	3.2	|**********T
13.000	2735	3.0	2.9	|**********T
14.000	2481	2.7	2.7	|*********T
15.000	2288	2.5	2.5	|********T
16.000	2210	2.4	2.3	|*******T
17.000	1926	2.1	2.1	|*******T
18.000	1853	2.0	2.0	|******T
19.000	1754	1.9	1.8	|*****T
20.000	1518	1.6	1.7	|*****T
21.000	1451	1.6	1.5	|*****T
22.000	1303	1.4	1.4	|****T
23.000	1196	1.3	1.3	|****T
24.000	1098	1.2	1.2	|***T
25.000	1073	1.2	1.1	|***T
26.000	893	1.0	1.0	|***T
27.000	930	1.0	1.0	|**T
28.000	774	0.8	0.9	|**T
29.000	784	0.8	0.8	|**T
30.000	677	0.7	0.7	|**T
31.000	640	0.7	0.7	|*T
32.000	630	0.7	0.6	|*T
33.000	537	0.6	0.6	|*T
34.000	512	0.6	0.5	|*T
35.000	465	0.5	0.5	|*T
36.000	435	0.5	0.5	|*T
37.000	373	0.4	0.4	|*T
38.000	322	0.3	0.4	|T
39.000	305	0.3	0.4	|T
40.000	318	0.3	0.3	|T
41.000	299	0.3	0.3	|T
42.000	274	0.3	0.3	|T
43.000	241	0.3	0.3	|T
44.000	218	0.2	0.2	|T

45.000	184	0.2	0.2	T
46.000	187	0.2	0.2	T
47.000	181	0.2	0.2	T
48.000	160	0.2	0.2	T
49.000	151	0.2	0.2	T
50.000	123	0.1	0.2	T
51.000	132	0.1	0.1	
52.000	109	0.1	0.1	
53.000	77	0.1	0.1	
54.000	97	0.1	0.1	
55.000	80	0.1	0.1	
56.000	81	0.1	0.1	
57.000	58	0.1	0.1	
58.000	82	0.1	0.1	
59.000	57	0.1	0.1	
60.000	47	0.1	0.1	
61.000	71	0.1	0.1	
62.000	60	0.1	0.1	
63.000	51	0.1	0.1	
64.000	41	0.0	0.0	
65.000	42	0.0	0.0	
66.032	56	0.1	0.0	
67.000	35	0.0	0.0	
68.000	19	0.0	0.0	
69.000	28	0.0	0.0	
70.000	35	0.0	0.0	
71.000	12	0.0	0.0	
72.000	32	0.0	0.0	
73.000	17	0.0	0.0	
74.000	24	0.0	0.0	
75.000	11	0.0	0.0	
76.000	14	0.0	0.0	
77.000	9	0.0	0.0	
78.000	15	0.0	0.0	
79.000	6	0.0	0.0	
80.000	15	0.0	0.0	
81.000	9	0.0	0.0	
82.000	6	0.0	0.0	
83.000	5	0.0	0.0	
84.000	16	0.0	0.0	
85.000	9	0.0	0.0	
86.000	13	0.0	0.0	
87.032	3	0.0	0.0	
88.032	6	0.0	0.0	
89.000	8	0.0	0.0	
90.000	5	0.0	0.0	
91.000	6	0.0	0.0	
92.000	4	0.0	0.0	
93.000	4	0.0	0.0	
94.000	6	0.0	0.0	
95.000	4	0.0	0.0	
96.000	3	0.0	0.0	
97.000	2	0.0	0.0	
98.000	3	0.0	0.0	

99.000	6	0.0	0.0
100.000	2	0.0	0.0
101.000	2	0.0	0.0
102.000	2	0.0	0.0
103.000	0	0.0	0.0
104.000	2	0.0	0.0
105.000	0	0.0	0.0
106.000	5	0.0	0.0
107.000	2	0.0	0.0
108.000	0	0.0	0.0
109.032	2	0.0	0.0
110.000	2	0.0	0.0
111.000	2	0.0	0.0
112.000	0	0.0	0.0
113.000	0	0.0	0.0
114.000	2	0.0	0.0
115.000	1	0.0	0.0
116.000	0	0.0	0.0
117.000	0	0.0	0.0
118.000	0	0.0	0.0
119.000	0	0.0	0.0
120.000	2	0.0	0.0
121.000	0	0.0	0.0
122.000	2	0.0	0.0
123.000	2	0.0	0.0
124.000	0	0.0	0.0
125.000	0	0.0	0.0
126.000	0	0.0	0.0
127.000	2	0.0	0.0
128.000	0	0.0	0.0
129.000	0	0.0	0.0
130.032	0	0.0	0.0
131.000	0	0.0	0.0
132.064	0	0.0	0.0
133.064	0	0.0	0.0
134.000	0	0.0	0.0
135.000	0	0.0	0.0
136.000	0	0.0	0.0
137.000	0	0.0	0.0
138.000	0	0.0	0.0
139.000	0	0.0	0.0
140.000	0	0.0	0.0
141.000	0	0.0	0.0
142.000	0	0.0	0.0
143.000	0	0.0	0.0
144.000	0	0.0	0.0
145.000	0	0.0	0.0
146.000	2	0.0	0.0
147.000	0	0.0	0.0
148.000	0	0.0	0.0
149.000	0	0.0	0.0
150.000	2	0.0	0.0

Simulator Report

NonStop/SQL

2 * 8 VLX SYSTEMS

Throughput - 87 TPS

```
--------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                   *
--------------------------------------------------------------------

----- PARAMETERS -----------------------------------------------------

Test Title                =   Test ALL;RUN03-16;SQL
Test Date                 =   87-02-20
Test Length               =   18:43:04 - 19:15:57  (1972 sec)
Window Length             =   18:58:00 - 19:13:00  (900 sec)
Tx Log File               =   \DRIVER.$DM03.RU16N03.TLOGSUMZ

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    5.500 tx/sec
Tx Arrival Rate/Terminal  =    0.069 tx/sec
Tx Inter-Arrival Time     =   14.545 sec
```

----- MEASUREMENT RESULTS (vs 2.03) ----------

Tx Throughput		=	87.713 tx/sec	
Tx Response Time	Average	=	0.934 sec	
	Std Dev	=	0.364 sec	
	Minimum	=	0.266 sec	
	Maximum	=	14.351 sec	
	Count	=	78942 #	
Tx Response Time	10%	=	0.650 sec	(12.0%)
	20%	=	0.750 sec	(26.4%)
	30%	=	0.800 sec	(35.9%)
	40%	=	0.850 sec	(45.2%)
	50%	=	0.900 sec	(53.6%)
	60%	=	0.950 sec	(61.4%)
	70%	=	1.050 sec	(74.3%)
	80%	=	1.150 sec	(83.2%)
	90%	=	1.300 sec	(90.7%)
	95%	=	1.450 sec	(95.0%)
Tx Think	Average	=	14.552 sec	
	Std Dev	=	14.522 sec	
	Minimum	=	0.001 sec	
	Maximum	=	175.401 sec	
	Count	=	74058 #	(93.8%)
Tx Delay	Average	=	0.542 sec	
	Std Dev	=	0.567 sec	
	Minimum	=	0.000 sec	
	Maximum	=	11.015 sec	
	Count	=	4884 #	(6.2%)

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.934 s	0.364 s	0.266 s	14.351 s	78942	87.713 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	3	0.0	0.0	|
0.400	95	0.1	0.1	|
0.500	1145	1.5	1.6	|***
0.600	4484	5.7	7.3	|**********
0.700	8635	10.9	18.2	|**********
0.800	13943	17.7	35.9	|**********
0.900	14031	17.8	53.6	|**********
1.000	11521	14.6	68.2	|**********
1.100	8604	10.9	79.1	|**********
1.200	5634	7.1	86.3	|**********
1.300	3543	4.5	90.7	|**********
1.400	2451	3.1	93.9	|*******
1.500	1591	2.0	95.9	|****
1.600	1058	1.3	97.2	|***
1.700	741	0.9	98.1	|**
1.800	466	0.6	98.7	|*
1.900	304	0.4	99.1	|
2.000	205	0.3	99.4	|
2.100	126	0.2	99.5	|
2.200	73	0.1	99.6	|
2.300	70	0.1	99.7	|
2.400	36	0.0	99.8	|
2.500	27	0.0	99.8	|
2.600	20	0.0	99.8	|
2.700	26	0.0	99.9	|
2.800	15	0.0	99.9	|
2.900	13	0.0	99.9	|
3.000	5	0.0	99.9	|
3.100	5	0.0	99.9	|
3.200	6	0.0	99.9	|
3.300	1	0.0	99.9	|
3.400	1	0.0	99.9	|
3.500	2	0.0	99.9	|
3.600	0	0.0	99.9	|
3.700	2	0.0	99.9	|
3.800	0	0.0	99.9	|
3.900	1	0.0	99.9	|
4.000	0	0.0	99.9	|
4.100	1	0.0	99.9	|
4.200	1	0.0	99.9	|
4.300	1	0.0	99.9	|

4.400	0	0.0	99.9
4.500	2	0.0	99.9
4.600	2	0.0	99.9
4.700	0	0.0	99.9
4.800	0	0.0	99.9
4.900	0	0.0	99.9
5.000	0	0.0	99.9
5.100	0	0.0	99.9
5.200	0	0.0	99.9
5.300	0	0.0	99.9
5.400	2	0.0	99.9
5.500	0	0.0	99.9
5.600	0	0.0	99.9
5.700	0	0.0	99.9
5.800	0	0.0	99.9
5.900	1	0.0	99.9
6.000	1	0.0	99.9
6.100	1	0.0	99.9
6.200	1	0.0	99.9
6.300	4	0.0	99.9
6.400	1	0.0	99.9
6.500	0	0.0	99.9
6.600	0	0.0	99.9
6.700	0	0.0	99.9
6.800	0	0.0	99.9
6.900	0	0.0	99.9
7.000	0	0.0	99.9
7.100	0	0.0	99.9
7.200	2	0.0	100.0
7.300	1	0.0	100.0
7.400	0	0.0	100.0
7.500	0	0.0	100.0
7.600	1	0.0	100.0
7.700	2	0.0	100.0
7.800	1	0.0	100.0
7.900	0	0.0	100.0
8.000	0	0.0	100.0
8.100	0	0.0	100.0
8.200	0	0.0	100.0
8.300	0	0.0	100.0
8.400	1	0.0	100.0
8.500	0	0.0	100.0
8.600	0	0.0	100.0
8.700	1	0.0	100.0
8.800	0	0.0	100.0
8.900	1	0.0	100.0
9.000	1	0.0	100.0
9.100	1	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	0	0.0	100.0
9.500	0	0.0	100.0
9.600	0	0.0	100.0
9.700	1	0.0	100.0

9.800	1	0.0	100.0
9.900	0	0.0	100.0
10.000	0	0.0	100.0
10.100	0	0.0	100.0
10.200	1	0.0	100.0
10.300	1	0.0	100.0
10.400	1	0.0	100.0
10.500	2	0.0	100.0
10.600	0	0.0	100.0
10.700	1	0.0	100.0
10.800	0	0.0	100.0
10.900	1	0.0	100.0
11.000	0	0.0	100.0
11.100	0	0.0	100.0
11.200	0	0.0	100.0
11.300	1	0.0	100.0
11.400	0	0.0	100.0
11.500	2	0.0	100.0
11.600	1	0.0	100.0
11.700	1	0.0	100.0
11.800	0	0.0	100.0
11.900	1	0.0	100.0
12.000	1	0.0	100.0
12.100	1	0.0	100.0
12.200	1	0.0	100.0
12.300	1	0.0	100.0
12.400	4	0.0	100.0
12.500	0	0.0	100.0
12.600	1	0.0	100.0
12.700	0	0.0	100.0
12.800	1	0.0	100.0
12.900	1	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	0	0.0	100.0
13.500	2	0.0	100.0
13.600	0	0.0	100.0
13.700	0	0.0	100.0
13.800	0	0.0	100.0
13.900	0	0.0	100.0
14.000	0	0.0	100.0
14.100	0	0.0	100.0
14.200	0	0.0	100.0
14.300	0	0.0	100.0
14.400	1	0.0	100.0

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
14.586 s	14.492 s	0.316 s	176.217 s	78942	87.713 tx/s

ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	|
1.000	3517	4.5	6.6	|********** T
2.000	6471	8.2	6.2	|**********T**********
3.000	4608	5.8	5.8	|**********T
4.000	4274	5.4	5.4	|**********T
5.000	4026	5.1	5.0	|**********T
6.000	3750	4.8	4.7	|**********T
7.000	3458	4.4	4.4	|**********T
8.000	3152	4.0	4.1	|**********T
9.000	3008	3.8	3.8	|**********T
10.000	2800	3.5	3.6	|**********T
11.000	2683	3.4	3.3	|**********T
12.000	2469	3.1	3.1	|**********T
13.000	2311	2.9	2.9	|**********T
14.000	2192	2.8	2.7	|**********T
15.000	1950	2.5	2.5	|**********T
16.000	1857	2.4	2.4	|**********T
17.000	1762	2.2	2.2	|**********T
18.000	1577	2.0	2.1	|**********T
19.000	1622	2.1	1.9	|*********T
20.000	1401	1.8	1.8	|********T
21.000	1377	1.7	1.7	|********T
22.000	1298	1.6	1.6	|*******T
23.000	1180	1.5	1.5	|*******T
24.000	1056	1.3	1.4	|******T
25.000	1047	1.3	1.3	|******T
26.000	927	1.2	1.2	|*****T
27.000	846	1.1	1.1	|*****T
28.000	855	1.1	1.0	|****T
29.000	775	1.0	1.0	|****T
30.000	743	0.9	0.9	|****T
31.000	649	0.8	0.8	|***T
32.000	660	0.8	0.8	|***T
33.000	561	0.7	0.7	|***T
34.000	556	0.7	0.7	|***T
35.000	530	0.7	0.6	|**T
36.000	420	0.5	0.6	|**T
37.000	469	0.6	0.6	|**T
38.000	432	0.5	0.5	|**T
39.000	346	0.4	0.5	|**T
40.000	356	0.5	0.5	|*T
41.000	316	0.4	0.4	|*T
42.000	310	0.4	0.4	|*T
43.000	262	0.3	0.4	|*T
44.000	256	0.3	0.3	|*T

45.000	249	0.3	0.3	*T
46.000	230	0.3	0.3	*T
47.000	253	0.3	0.3	T
48.000	203	0.3	0.3	T
49.000	206	0.3	0.2	T
50.000	172	0.2	0.2	T
51.000	174	0.2	0.2	T
52.000	160	0.2	0.2	T
53.000	136	0.2	0.2	T
54.000	142	0.2	0.2	T
55.000	135	0.2	0.2	T
56.000	104	0.1	0.2	T
57.000	124	0.2	0.1	T
58.000	112	0.1	0.1	T
59.000	91	0.1	0.1	T
60.000	88	0.1	0.1	T
61.000	97	0.1	0.1	T
62.000	66	0.1	0.1	T
63.000	67	0.1	0.1	
64.000	67	0.1	0.1	
65.000	60	0.1	0.1	
66.032	58	0.1	0.1	
67.000	48	0.1	0.1	
68.000	56	0.1	0.1	
69.000	39	0.0	0.1	
70.000	49	0.1	0.1	
71.000	29	0.0	0.1	
72.000	46	0.1	0.1	
73.000	37	0.0	0.0	
74.000	33	0.0	0.0	
75.000	45	0.1	0.0	
76.000	28	0.0	0.0	
77.000	28	0.0	0.0	
78.000	39	0.0	0.0	
79.000	26	0.0	0.0	
80.000	13	0.0	0.0	
81.000	16	0.0	0.0	
82.000	19	0.0	0.0	
83.000	18	0.0	0.0	
84.000	14	0.0	0.0	
85.000	24	0.0	0.0	
86.000	13	0.0	0.0	
87.032	12	0.0	0.0	
88.032	13	0.0	0.0	
89.000	10	0.0	0.0	
90.000	10	0.0	0.0	
91.000	11	0.0	0.0	
92.000	14	0.0	0.0	
93.000	10	0.0	0.0	
94.000	7	0.0	0.0	
95.000	10	0.0	0.0	
96.000	5	0.0	0.0	
97.000	5	0.0	0.0	
98.000	5	0.0	0.0	

99.000	11	0.0	0.0
100.000	5	0.0	0.0
101.000	8	0.0	0.0
102.000	3	0.0	0.0
103.000	1	0.0	0.0
104.000	2	0.0	0.0
105.000	5	0.0	0.0
106.000	3	0.0	0.0
107.000	5	0.0	0.0
108.000	3	0.0	0.0
109.032	3	0.0	0.0
110.000	6	0.0	0.0
111.000	4	0.0	0.0
112.000	2	0.0	0.0
113.000	3	0.0	0.0
114.000	1	0.0	0.0
115.000	3	0.0	0.0
116.000	3	0.0	0.0
117.000	3	0.0	0.0
118.000	1	0.0	0.0
119.000	1	0.0	0.0
120.000	0	0.0	0.0
121.000	1	0.0	0.0
122.000	0	0.0	0.0
123.000	1	0.0	0.0
124.000	0	0.0	0.0
125.000	2	0.0	0.0
126.000	3	0.0	0.0
127.000	1	0.0	0.0
128.000	1	0.0	0.0
129.000	0	0.0	0.0
130.032	3	0.0	0.0
131.000	1	0.0	0.0
132.064	0	0.0	0.0
133.064	0	0.0	0.0
134.000	0	0.0	0.0
135.000	2	0.0	0.0
136.000	0	0.0	0.0
137.000	0	0.0	0.0
138.000	0	0.0	0.0
139.000	0	0.0	0.0
140.000	1	0.0	0.0
141.000	0	0.0	0.0
142.000	2	0.0	0.0
143.000	0	0.0	0.0
144.000	1	0.0	0.0
145.000	0	0.0	0.0
146.000	1	0.0	0.0
147.000	0	0.0	0.0
148.000	0	0.0	0.0
149.000	0	0.0	0.0
150.000	1	0.0	0.0
151.000	0	0.0	0.0
152.000	0	0.0	0.0

153.000	0	0.0	0.0
154.000	0	0.0	0.0
155.000	1	0.0	0.0
156.000	0	0.0	0.0
157.000	0	0.0	0.0
158.000	0	0.0	0.0
159.000	0	0.0	0.0
160.000	0	0.0	0.0
161.000	0	0.0	0.0
162.000	0	0.0	0.0
163.000	0	0.0	0.0
164.000	0	0.0	0.0
165.000	0	0.0	0.0
166.000	0	0.0	0.0
167.000	0	0.0	0.0
168.000	0	0.0	0.0
169.000	0	0.0	0.0
170.000	0	0.0	0.0
171.000	0	0.0	0.0
172.000	0	0.0	0.0
173.000	1	0.0	0.0
174.064	1	0.0	0.0
175.064	0	0.0	0.0
176.064	0	0.0	0.0
177.000	1	0.0	0.0

Simulator Report

NonStop/SQL

2 * 8 VLX SYSTEMS

Throughput - 72 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test ALL;RUN04-16;SQL
Test Date                  =   87-02-20
Test Length                =   19:33:43 - 19:51:33  (1070 sec)
Window Length              =   19:35:00 - 19:50:00  (900 sec)
Tx Log File                =   \DRIVER.$DM03.RU16N04.TLOGSUMZ

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    4.500 tx/sec
Tx Arrival Rate/Terminal   =    0.056 tx/sec
Tx Inter-Arrival Time      =   17.778 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.03) --------------------------------

Tx Throughput              =   72.111 tx/sec
Tx Response Time Average   =    0.797 sec
                 Std Dev   =    0.226 sec
                 Minimum   =    0.262 sec
                 Maximum   =    3.326 sec
                 Count     =    64900 #

Tx Response Time 10%       =    0.550 sec  ( 10.2%)
                 20%       =    0.650 sec  ( 25.5%)
                 30%       =    0.700 sec  ( 35.4%)
                 40%       =    0.750 sec  ( 46.5%)
                 50%       =    0.800 sec  ( 57.8%)
                 60%       =    0.850 sec  ( 67.6%)
                 70%       =    0.900 sec  ( 75.1%)
                 80%       =    0.950 sec  ( 80.9%)
                 90%       =    1.100 sec  ( 91.3%)
                 95%       =    1.250 sec  ( 95.7%)

        Tx Think Average   =   17.815 sec
                 Std Dev   =   17.728 sec
                 Minimum   =    0.001 sec
                 Maximum   =  214.537 sec
                 Count     =    61993 #    ( 95.5%)

        Tx Delay Average   =    0.431 sec
                 Std Dev   =    0.290 sec
                 Minimum   =    0.000 sec
                 Maximum   =    2.100 sec
                 Count     =     2907 #    (  4.5%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.797 s	0.226 s	0.262 s	3.326 s	64900	72.111 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	16	0.0	0.0	|
0.400	413	0.6	0.7	|*
0.500	2841	4.4	5.0	|********
0.600	7871	12.1	17.2	|**********
0.700	11850	18.3	35.4	|**********
0.800	14542	22.4	57.8	|**********
0.900	11232	17.3	75.1	|**********
1.000	6712	10.3	85.5	|**********
1.100	3781	5.8	91.3	|**********
1.200	2155	3.3	94.6	|******
1.300	1307	2.0	96.6	|***
1.400	936	1.4	98.1	|**
1.500	516	0.8	98.9	|*
1.600	284	0.4	99.3	|
1.700	162	0.2	99.6	|
1.800	88	0.1	99.7	|
1.900	48	0.1	99.8	|
2.000	38	0.1	99.8	|
2.100	20	0.0	99.9	|
2.200	21	0.0	99.9	|
2.300	22	0.0	99.9	|
2.400	14	0.0	100.0	|
2.500	8	0.0	100.0	|
2.600	2	0.0	100.0	|
2.700	5	0.0	100.0	|
2.800	4	0.0	100.0	|
2.900	5	0.0	100.0	|
3.000	3	0.0	100.0	|
3.100	1	0.0	100.0	|
3.200	1	0.0	100.0	|
3.300	0	0.0	100.0	|
3.400	2	0.0	100.0	|

* ET1 Inter-Arrival Time Distribution *

Mean	Std	Min	Max	Count	Thruput
17.813 s	17.711 s	0.350 s	215.374 s	64900	72.111 tx/s

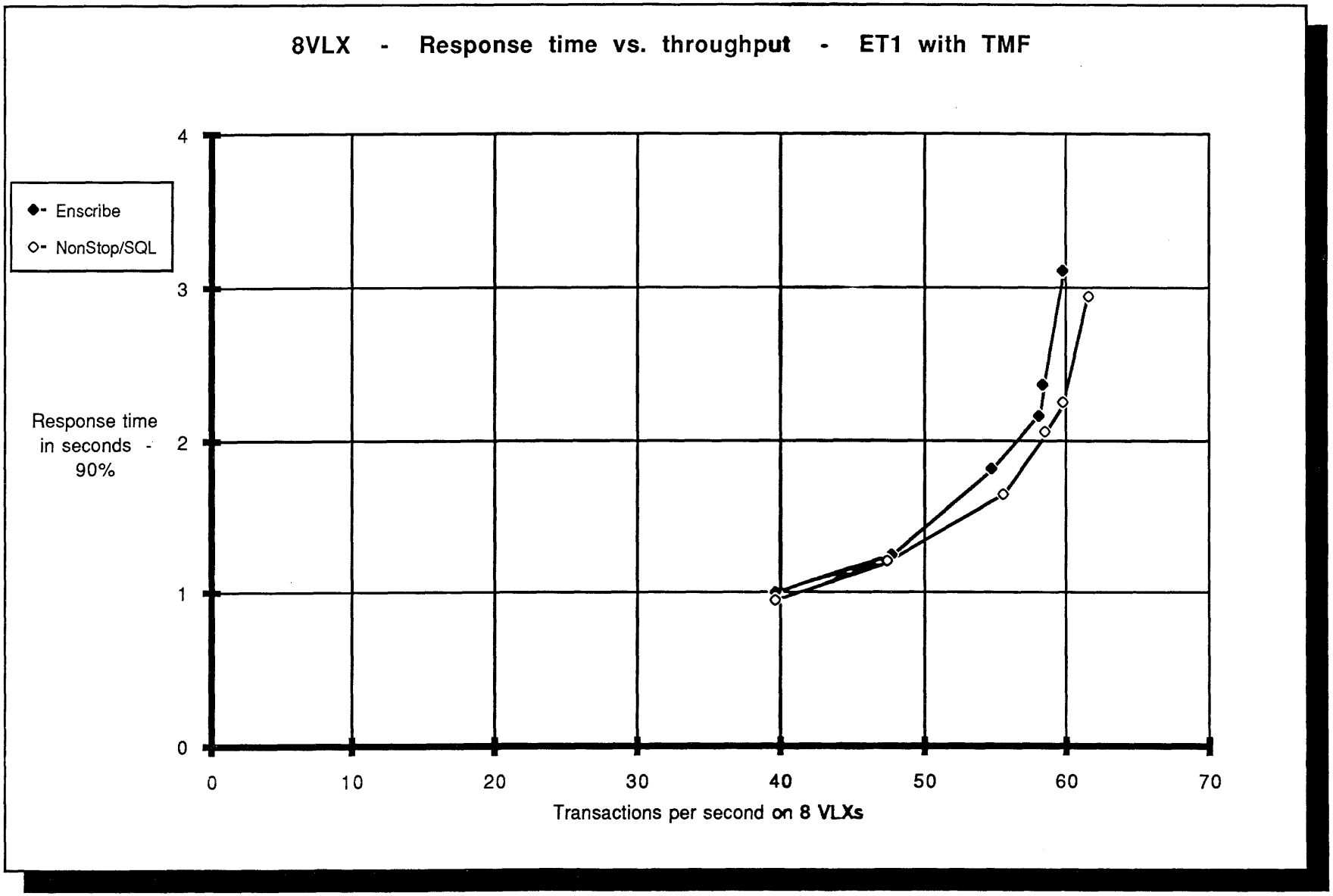
ArrvTime	Count	Pct	Theory	
0.000	0	0.0	0.0	
1.000	3064	4.7	5.5	********************************* T
2.000	3824	5.9	5.2	************************************T*****
3.000	3212	4.9	4.9	**********************************T
4.000	3035	4.7	4.6	********************************T
5.000	2783	4.3	4.4	******************************T
6.000	2708	4.2	4.1	****************************T
7.000	2642	4.1	3.9	***************************T*
8.000	2308	3.6	3.7	*************************T
9.000	2164	3.3	3.5	*********************** T
10.000	1982	3.1	3.3	********************* T
11.000	2039	3.1	3.1	*********************T
12.000	1958	3.0	2.9	********************T
13.000	1805	2.8	2.8	*******************T
14.000	1732	2.7	2.6	******************T
15.000	1561	2.4	2.5	*****************T
16.000	1445	2.2	2.4	*************** T
17.000	1509	2.3	2.2	***************T
18.000	1341	2.1	2.1	**************T
19.000	1299	2.0	2.0	*************T
20.000	1261	1.9	1.9	************T
21.000	1140	1.8	1.8	************T
22.000	1055	1.6	1.7	***********T
23.000	1152	1.8	1.6	**********T*
24.000	989	1.5	1.5	**********T
25.000	864	1.3	1.4	*********T
26.000	885	1.4	1.3	*********T
27.000	919	1.4	1.3	********T*
28.000	776	1.2	1.2	********T
29.000	761	1.2	1.1	*******T
30.000	714	1.1	1.1	*******T
31.000	652	1.0	1.0	******T
32.000	585	0.9	1.0	******T
33.000	584	0.9	0.9	*****T
34.000	529	0.8	0.9	*****T
35.000	513	0.8	0.8	*****T
36.000	507	0.8	0.8	****T
37.000	469	0.7	0.7	****T
38.000	447	0.7	0.7	****T
39.000	442	0.7	0.6	****T
40.000	367	0.6	0.6	***T
41.000	371	0.6	0.6	***T
42.000	345	0.5	0.5	***T
43.000	353	0.5	0.5	***T
44.000	270	0.4	0.5	**T

45.000	362	0.6	0.5	**T
46.000	317	0.5	0.4	**T
47.000	255	0.4	0.4	**T
48.000	264	0.4	0.4	**T
49.000	266	0.4	0.4	**T
50.000	227	0.3	0.3	*T
51.000	211	0.3	0.3	*T
52.000	200	0.3	0.3	*T
53.000	185	0.3	0.3	*T
54.000	168	0.3	0.3	*T
55.000	184	0.3	0.3	*T
56.000	118	0.2	0.2	*T
57.000	176	0.3	0.2	*T
58.000	166	0.3	0.2	*T
59.000	120	0.2	0.2	T
60.000	151	0.2	0.2	T
61.000	121	0.2	0.2	T
62.000	115	0.2	0.2	T
63.000	111	0.2	0.2	T
64.000	101	0.2	0.2	T
65.000	92	0.1	0.1	T
66.000	107	0.2	0.1	T
67.000	95	0.1	0.1	T
68.000	72	0.1	0.1	T
69.000	75	0.1	0.1	T
70.000	76	0.1	0.1	T
71.000	57	0.1	0.1	T
72.000	70	0.1	0.1	T
73.000	46	0.1	0.1	T
74.000	66	0.1	0.1	T
75.000	57	0.1	0.1	T
76.000	43	0.1	0.1	T
77.000	42	0.1	0.1	T
78.000	64	0.1	0.1	T
79.000	46	0.1	0.1	
80.000	31	0.0	0.1	
81.000	34	0.1	0.1	
82.000	36	0.1	0.1	
83.000	40	0.1	0.1	
84.000	26	0.0	0.1	
85.000	25	0.0	0.0	
86.000	26	0.0	0.0	
87.000	25	0.0	0.0	
88.000	26	0.0	0.0	
89.000	31	0.0	0.0	
90.000	24	0.0	0.0	
91.000	28	0.0	0.0	
92.000	19	0.0	0.0	
93.000	22	0.0	0.0	
94.000	15	0.0	0.0	
95.000	21	0.0	0.0	
96.000	21	0.0	0.0	
97.000	12	0.0	0.0	
98.000	6	0.0	0.0	

99.000	8	0.0	0.0
100.000	17	0.0	0.0
101.000	14	0.0	0.0
102.000	9	0.0	0.0
103.000	5	0.0	0.0
104.000	14	0.0	0.0
105.000	12	0.0	0.0
106.000	0	0.0	0.0
107.000	18	0.0	0.0
108.000	7	0.0	0.0
109.000	8	0.0	0.0
110.000	5	0.0	0.0
111.000	3	0.0	0.0
112.000	6	0.0	0.0
113.000	6	0.0	0.0
114.000	4	0.0	0.0
115.000	5	0.0	0.0
116.000	3	0.0	0.0
117.000	10	0.0	0.0
118.000	2	0.0	0.0
119.000	0	0.0	0.0
120.000	4	0.0	0.0
121.000	10	0.0	0.0
122.000	4	0.0	0.0
123.000	4	0.0	0.0
124.000	6	0.0	0.0
125.000	0	0.0	0.0
126.000	2	0.0	0.0
127.000	0	0.0	0.0
128.000	4	0.0	0.0
129.000	3	0.0	0.0
130.000	2	0.0	0.0
131.000	4	0.0	0.0
132.000	2	0.0	0.0
133.000	0	0.0	0.0
134.000	2	0.0	0.0
135.000	8	0.0	0.0
136.000	0	0.0	0.0
137.000	4	0.0	0.0
138.000	2	0.0	0.0
139.000	0	0.0	0.0
140.000	0	0.0	0.0
141.000	1	0.0	0.0
142.000	2	0.0	0.0
143.000	0	0.0	0.0
144.000	2	0.0	0.0
145.000	0	0.0	0.0
146.000	2	0.0	0.0
147.000	0	0.0	0.0
148.000	2	0.0	0.0
149.000	0	0.0	0.0
150.000	2	0.0	0.0
151.000	0	0.0	0.0
152.000	0	0.0	0.0

153.000	2	0.0	0.0
154.000	2	0.0	0.0
155.000	0	0.0	0.0
156.000	0	0.0	0.0
157.000	0	0.0	0.0
158.000	0	0.0	0.0
159.000	0	0.0	0.0
160.000	2	0.0	0.0
161.000	0	0.0	0.0
162.000	0	0.0	0.0
163.000	0	0.0	0.0
164.000	0	0.0	0.0
165.000	2	0.0	0.0
166.000	0	0.0	0.0
167.000	0	0.0	0.0
168.000	0	0.0	0.0
169.000	0	0.0	0.0
170.000	0	0.0	0.0
171.000	0	0.0	0.0
172.000	0	0.0	0.0
173.000	0	0.0	0.0
174.000	0	0.0	0.0
175.000	0	0.0	0.0
176.000	2	0.0	0.0
177.000	0	0.0	0.0
178.000	2	0.0	0.0
179.000	0	0.0	0.0
180.000	0	0.0	0.0
181.000	0	0.0	0.0
182.000	0	0.0	0.0
183.000	0	0.0	0.0
184.000	0	0.0	0.0
185.000	0	0.0	0.0
186.000	0	0.0	0.0
187.000	0	0.0	0.0
188.000	0	0.0	0.0
189.000	0	0.0	0.0
190.000	2	0.0	0.0
191.000	0	0.0	0.0
192.000	0	0.0	0.0
193.000	0	0.0	0.0
194.000	0	0.0	0.0
195.000	0	0.0	0.0
196.000	0	0.0	0.0
197.000	0	0.0	0.0
198.000	0	0.0	0.0
199.000	0	0.0	0.0
200.000	0	0.0	0.0
201.000	0	0.0	0.0
202.000	0	0.0	0.0
203.000	0	0.0	0.0
204.000	0	0.0	0.0
205.000	0	0.0	0.0
206.000	0	0.0	0.0

207.000	0	0.0	0.0
208.000	0	0.0	0.0
209.000	0	0.0	0.0
210.000	0	0.0	0.0
211.000	1	0.0	0.0
212.000	0	0.0	0.0
213.000	0	0.0	0.0
214.000	0	0.0	0.0
215.000	0	0.0	0.0
216.000	2	0.0	0.0



* For 90% of transactions

Simulator Report

NonStop/SQL

1 * 8 VLX SYSTEMS

Throughput - 61 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test 07; TPS 80/15
Test Date                  =   87-02-11
Test Length                =   23:11:13 - 23:23:21  (727 sec)
Window Length              =   23:15:00 - 23:22:00  (420 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =   10.000 tx/sec
Tx Arrival Rate/Terminal   =    0.125 tx/sec
Tx Inter-Arrival Time      =    8.000 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ---------------------------------

Tx Throughput                =   61.643 tx/sec
Tx Response Time Average     =    1.940 sec
                 Std Dev     =    0.800 sec
                 Minimum     =    0.404 sec
                 Maximum     =    9.464 sec
                 Count       =    25890 #

Tx Response Time 10%         =    1.150 sec   ( 10.8%)
                 20%         =    1.350 sec   ( 22.5%)
                 30%         =    1.500 sec   ( 32.5%)
                 40%         =    1.650 sec   ( 42.3%)
                 50%         =    1.800 sec   ( 51.7%)
                 60%         =    1.950 sec   ( 60.2%)
                 70%         =    2.200 sec   ( 71.5%)
                 80%         =    2.450 sec   ( 80.0%)
                 90%         =    2.950 sec   ( 90.1%)
                 95%         =    3.450 sec   ( 95.1%)

        Tx Think Average     =    7.991 sec
                 Std Dev     =    8.062 sec
                 Minimum     =    0.001 sec
                 Maximum     =   68.586 sec
                 Count       =    20284 #     ( 78.3%)

        Tx Delay Average     =    1.169 sec
                 Std Dev     =    0.864 sec
                 Minimum     =    0.000 sec
                 Maximum     =    7.626 sec
                 Count       =     5606 #     ( 21.7%)

-----------------------------------------------------------------------
```

* ET1 Response Time Distribution *

```
----------------------------------------------------------------------
   Mean        Std        Min        Max        Count      Thruput
   1.940 s    0.800 s    0.404 s    9.464 s    25890       61.643 tx/s
----------------------------------------------------------------------

RespTime Count   Pct  CumPct
-------- -----  ----- ------
   0.000     0    0.0    0.0 |
   0.100     0    0.0    0.0 |
   0.200     0    0.0    0.0 |
   0.300     0    0.0    0.0 |
   0.400     0    0.0    0.0 |
   0.500     2    0.0    0.0 |
   0.600    18    0.1    0.1 |
   0.700    47    0.2    0.3 |*
   0.800   154    0.6    0.9 |***
   0.900   365    1.4    2.3 |********
   1.000   641    2.5    4.7 |***************
   1.100   962    3.7    8.5 |***********************
   1.200  1289    5.0   13.4 |*******************************
   1.300  1520    5.9   19.3 |*************************************
   1.400  1682    6.5   25.8 |*****************************************
   1.500  1722    6.7   32.5 |*****************************************
   1.600  1723    6.7   39.1 |******************************************
   1.700  1664    6.4   45.5 |****************************************
   1.800  1602    6.2   51.7 |***************************************
   1.900  1497    5.8   57.5 |************************************
   2.000  1348    5.2   62.7 |********************************
   2.100  1219    4.7   67.4 |*****************************
   2.200  1054    4.1   71.5 |*************************
   2.300   983    3.8   75.3 |***********************
   2.400   847    3.3   78.6 |********************
   2.500   748    2.9   81.4 |******************
   2.600   637    2.5   83.9 |***************
   2.700   530    2.0   86.0 |************
   2.800   473    1.8   87.8 |***********
   2.900   420    1.6   89.4 |**********
   3.000   336    1.3   90.7 |********
   3.100   321    1.2   91.9 |*******
   3.200   272    1.1   93.0 |******
   3.300   250    1.0   94.0 |******
   3.400   209    0.8   94.8 |*****
   3.500   172    0.7   95.4 |****
   3.600   138    0.5   96.0 |***
   3.700   109    0.4   96.4 |**
   3.800   107    0.4   96.8 |**
   3.900   103    0.4   97.2 |**
   4.000    82    0.3   97.5 |*
   4.100    86    0.3   97.8 |**
   4.200    73    0.3   98.1 |*
   4.300    58    0.2   98.4 |*
   4.400    46    0.2   98.5 |*
```

4.500	50	0.2	98.7	*
4.600	40	0.2	98.9	
4.700	28	0.1	99.0	
4.800	36	0.1	99.1	
4.900	27	0.1	99.2	
5.000	21	0.1	99.3	
5.100	27	0.1	99.4	
5.200	20	0.1	99.5	
5.300	10	0.0	99.5	
5.400	12	0.0	99.6	
5.500	10	0.0	99.6	
5.600	13	0.1	99.7	
5.700	10	0.0	99.7	
5.800	4	0.0	99.7	
5.900	9	0.0	99.8	
6.000	7	0.0	99.8	
6.100	5	0.0	99.8	
6.200	3	0.0	99.8	
6.300	9	0.0	99.8	
6.400	4	0.0	99.9	
6.500	1	0.0	99.9	
6.600	3	0.0	99.9	
6.700	4	0.0	99.9	
6.800	7	0.0	99.9	
6.900	2	0.0	99.9	
7.000	2	0.0	99.9	
7.100	1	0.0	99.9	
7.200	1	0.0	99.9	
7.300	2	0.0	99.9	
7.400	3	0.0	100.0	
7.500	0	0.0	100.0	
7.600	0	0.0	100.0	
7.700	0	0.0	100.0	
7.800	2	0.0	100.0	
7.900	1	0.0	100.0	
8.000	2	0.0	100.0	
8.100	2	0.0	100.0	
8.200	0	0.0	100.0	
8.300	0	0.0	100.0	
8.400	0	0.0	100.0	
8.500	0	0.0	100.0	
8.600	0	0.0	100.0	
8.700	1	0.0	100.0	
8.800	0	0.0	100.0	
8.900	1	0.0	100.0	
9.000	0	0.0	100.0	
9.100	0	0.0	100.0	
9.200	0	0.0	100.0	
9.300	0	0.0	100.0	
9.400	0	0.0	100.0	
9.500	1	0.0	100.0	

Simulator Report

NonStop/SQL

1 * 8 VLX SYSTEMS

Throughput - 60 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                 =   Test 05; 64 TPS
Test Date                  =   87-02-11
Test Length                =   21:16:12 - 21:31:40  (927 sec)
Window Length              =   21:20:00 - 21:27:00  (420 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    8.000 tx/sec
Tx Arrival Rate/Terminal   =    0.100 tx/sec
Tx Inter-Arrival Time      =   10.000 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ---------------------------------

Tx Throughput                =   59.795 tx/sec
Tx Response Time Average     =    1.525 sec
                 Std Dev     =    0.650 sec
                 Minimum     =    0.486 sec
                 Maximum     =   13.675 sec
                 Count       =    25114 #

Tx Response Time 10%         =    0.950 sec   ( 10.2%)
                 20%         =    1.100 sec   ( 22.4%)
                 30%         =    1.200 sec   ( 31.8%)
                 40%         =    1.300 sec   ( 41.7%)
                 50%         =    1.400 sec   ( 50.9%)
                 60%         =    1.550 sec   ( 62.7%)
                 70%         =    1.700 sec   ( 72.2%)
                 80%         =    1.900 sec   ( 81.5%)
                 90%         =    2.250 sec   ( 90.6%)
                 95%         =    2.650 sec   ( 95.4%)

        Tx Think Average     =    9.969 sec
                 Std Dev     =   10.047 sec
                 Minimum     =    0.001 sec
                 Maximum     =   85.901 sec
                 Count       =    21499 #     ( 85.6%)

        Tx Delay Average     =    0.919 sec
                 Std Dev     =    0.761 sec
                 Minimum     =    0.000 sec
                 Maximum     =   10.311 sec
                 Count       =     3615 #     ( 14.4%)

-----------------------------------------------------------------------
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.525 s	0.650 s	0.486 s	13.675 s	25114	59.795 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	0	0.0	0.0	|
0.500	3	0.0	0.0	|
0.600	49	0.2	0.2	|
0.700	226	0.9	1.1	|***
0.800	563	2.2	3.3	|*********
0.900	1001	4.0	7.3	|******************
1.000	1657	6.6	13.9	|****************************
1.100	2118	8.4	22.4	|**********************************
1.200	2371	9.4	31.8	|**
1.300	2473	9.8	41.7	|**
1.400	2333	9.3	50.9	|**
1.500	2019	8.0	59.0	|**********************************
1.600	1871	7.5	66.4	|*******************************
1.700	1447	5.8	72.2	|************************
1.800	1319	5.3	77.4	|**********************
1.900	1008	4.0	81.5	|******************
2.000	793	3.2	84.6	|*************
2.100	693	2.8	87.4	|***********
2.200	553	2.2	89.6	|*********
2.300	492	2.0	91.5	|********
2.400	356	1.4	93.0	|******
2.500	281	1.1	94.1	|****
2.600	218	0.9	94.9	|***
2.700	188	0.7	95.7	|***
2.800	179	0.7	96.4	|***
2.900	135	0.5	96.9	|**
3.000	91	0.4	97.3	|*
3.100	88	0.4	97.7	|*
3.200	72	0.3	97.9	|*
3.300	74	0.3	98.2	|*
3.400	54	0.2	98.5	|
3.500	44	0.2	98.6	|
3.600	43	0.2	98.8	|
3.700	31	0.1	98.9	|
3.800	23	0.1	99.0	|
3.900	32	0.1	99.1	|
4.000	16	0.1	99.2	|
4.100	22	0.1	99.3	|
4.200	17	0.1	99.4	|
4.300	21	0.1	99.4	|
4.400	13	0.1	99.5	|

4.500	21	0.1	99.6
4.600	10	0.0	99.6
4.700	6	0.0	99.6
4.800	5	0.0	99.7
4.900	14	0.1	99.7
5.000	4	0.0	99.7
5.100	3	0.0	99.7
5.200	7	0.0	99.8
5.300	2	0.0	99.8
5.400	3	0.0	99.8
5.500	5	0.0	99.8
5.600	2	0.0	99.8
5.700	2	0.0	99.8
5.800	0	0.0	99.8
5.900	1	0.0	99.8
6.000	2	0.0	99.8
6.100	0	0.0	99.8
6.200	0	0.0	99.8
6.300	1	0.0	99.8
6.400	0	0.0	99.8
6.500	3	0.0	99.9
6.600	0	0.0	99.9
6.700	0	0.0	99.9
6.800	2	0.0	99.9
6.900	1	0.0	99.9
7.000	2	0.0	99.9
7.100	0	0.0	99.9
7.200	0	0.0	99.9
7.300	0	0.0	99.9
7.400	1	0.0	99.9
7.500	0	0.0	99.9
7.600	1	0.0	99.9
7.700	1	0.0	99.9
7.800	3	0.0	99.9
7.900	0	0.0	99.9
8.000	0	0.0	99.9
8.100	2	0.0	99.9
8.200	1	0.0	99.9
8.300	2	0.0	99.9
8.400	1	0.0	99.9
8.500	1	0.0	99.9
8.600	1	0.0	99.9
8.700	1	0.0	99.9
8.800	2	0.0	99.9
8.900	1	0.0	99.9
9.000	0	0.0	99.9
9.100	1	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	1	0.0	100.0
9.500	0	0.0	100.0
9.600	0	0.0	100.0
9.700	0	0.0	100.0
9.800	3	0.0	100.0

9.900	0	0.0	100.0
10.000	0	0.0	100.0
10.100	1	0.0	100.0
10.200	0	0.0	100.0
10.300	0	0.0	100.0
10.400	0	0.0	100.0
10.500	0	0.0	100.0
10.600	2	0.0	100.0
10.700	0	0.0	100.0
10.800	0	0.0	100.0
10.900	0	0.0	100.0
11.000	0	0.0	100.0
11.100	0	0.0	100.0
11.200	0	0.0	100.0
11.300	0	0.0	100.0
11.400	0	0.0	100.0
11.500	0	0.0	100.0
11.600	0	0.0	100.0
11.700	1	0.0	100.0
11.800	0	0.0	100.0
11.900	0	0.0	100.0
12.000	0	0.0	100.0
12.100	0	0.0	100.0
12.200	1	0.0	100.0
12.300	0	0.0	100.0
12.400	0	0.0	100.0
12.500	1	0.0	100.0
12.600	0	0.0	100.0
12.700	0	0.0	100.0
12.800	0	0.0	100.0
12.900	0	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	1	0.0	100.0
13.500	0	0.0	100.0
13.600	0	0.0	100.0
13.700	1	0.0	100.0

Simulator Report

NonStop/SQL

1 * 8 VLX SYSTEMS

Throughput - 59 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test 04; 60 TPS
Test Date                  =   87-02-11
Test Length                =   22:27:59 - 22:40:29  (749 sec)
Window Length              =   22:32:00 - 22:39:00  (420 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    7.500 tx/sec
Tx Arrival Rate/Terminal   =    0.094 tx/sec
Tx Inter-Arrival Time      =   10.667 sec
```

----- MEASUREMENT RESULTS (vs 2.02) ----------

Tx Throughput		=	58.529 tx/sec	
Tx Response Time	Average	=	1.390 sec	
	Std Dev	=	0.544 sec	
	Minimum	=	0.384 sec	
	Maximum	=	6.658 sec	
	Count	=	24582 #	
Tx Response Time	10%	=	0.900 sec	(12.0%)
	20%	=	1.000 sec	(21.0%)
	30%	=	1.100 sec	(31.5%)
	40%	=	1.200 sec	(42.4%)
	50%	=	1.300 sec	(52.8%)
	60%	=	1.400 sec	(61.9%)
	70%	=	1.550 sec	(72.7%)
	80%	=	1.700 sec	(80.5%)
	90%	=	2.050 sec	(90.7%)
	95%	=	2.400 sec	(95.2%)
Tx Think	Average	=	10.631 sec	
	Std Dev	=	10.694 sec	
	Minimum	=	0.001 sec	
	Maximum	=	92.103 sec	
	Count	=	21474 #	(87.4%)
Tx Delay	Average	=	0.815 sec	
	Std Dev	=	0.613 sec	
	Minimum	=	0.000 sec	
	Maximum	=	5.552 sec	
	Count	=	3108 #	(12.6%)

---

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.390 s	0.544 s	0.384 s	6.658 s	24582	58.529 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	1	0.0	0.0	|
0.500	9	0.0	0.0	|
0.600	117	0.5	0.5	|*
0.700	393	1.6	2.1	|******
0.800	862	3.5	5.6	|*************
0.900	1565	6.4	12.0	|************************
1.000	2227	9.1	21.0	|**********************************
1.100	2565	10.4	31.5	|**
1.200	2678	10.9	42.4	|**
1.300	2564	10.4	52.8	|**
1.400	2245	9.1	61.9	|**********************************
1.500	1868	7.6	69.5	|*****************************
1.600	1480	6.0	75.6	|************************
1.700	1209	4.9	80.5	|*******************
1.800	977	4.0	84.5	|***************
1.900	695	2.8	87.3	|**********
2.000	624	2.5	89.8	|*********
2.100	435	1.8	91.6	|******
2.200	354	1.4	93.0	|*****
2.300	290	1.2	94.2	|****
2.400	232	0.9	95.2	|***
2.500	168	0.7	95.8	|**
2.600	148	0.6	96.4	|**
2.700	120	0.5	96.9	|*
2.800	92	0.4	97.3	|*
2.900	82	0.3	97.6	|*
3.000	83	0.3	98.0	|*
3.100	57	0.2	98.2	|
3.200	60	0.2	98.4	|
3.300	70	0.3	98.7	|*
3.400	50	0.2	98.9	|
3.500	33	0.1	99.1	|
3.600	29	0.1	99.2	|
3.700	42	0.2	99.4	|
3.800	17	0.1	99.4	|
3.900	23	0.1	99.5	|
4.000	15	0.1	99.6	|
4.100	14	0.1	99.6	|
4.200	12	0.0	99.7	|
4.300	8	0.0	99.7	|
4.400	7	0.0	99.7	|

4.500	7	0.0	99.8
4.600	7	0.0	99.8
4.700	10	0.0	99.8
4.800	2	0.0	99.9
4.900	6	0.0	99.9
5.000	5	0.0	99.9
5.100	3	0.0	99.9
5.200	7	0.0	99.9
5.300	4	0.0	100.0
5.400	2	0.0	100.0
5.500	3	0.0	100.0
5.600	0	0.0	100.0
5.700	2	0.0	100.0
5.800	1	0.0	100.0
5.900	1	0.0	100.0
6.000	1	0.0	100.0
6.100	0	0.0	100.0
6.200	0	0.0	100.0
6.300	0	0.0	100.0
6.400	0	0.0	100.0
6.500	0	0.0	100.0
6.600	0	0.0	100.0
6.700	1	0.0	100.0

Simulator Report

NonStop/SQL

1 * 8 VLX SYSTEMS

Throughput - 56 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test 03; 56 TPS
Test Date                 =   87-02-11
Test Length               =   20:32:34 - 20:46:15  (820 sec)
Window Length             =   20:36:00 - 20:43:00  (420 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    7.000 tx/sec
Tx Arrival Rate/Terminal  =    0.088 tx/sec
Tx Inter-Arrival Time     =   11.429 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ---------------------------------

Tx Throughput                =   55.588 tx/sec
Tx Response Time Average     =    1.150 sec
                 Std Dev     =    0.437 sec
                 Minimum     =    0.381 sec
                 Maximum     =    5.026 sec
                 Count       =    23347 #

Tx Response Time 10%         =    0.750 sec   ( 10.6%)
                 20%         =    0.850 sec   ( 21.4%)
                 30%         =    0.950 sec   ( 35.1%)
                 40%         =    1.000 sec   ( 42.1%)
                 50%         =    1.100 sec   ( 55.4%)
                 60%         =    1.150 sec   ( 61.5%)
                 70%         =    1.250 sec   ( 71.5%)
                 80%         =    1.400 sec   ( 81.4%)
                 90%         =    1.650 sec   ( 90.4%)
                 95%         =    1.950 sec   ( 95.1%)

        Tx Think Average     =   11.452 sec
                 Std Dev     =   11.448 sec
                 Minimum     =    0.001 sec
                 Maximum     =   99.531 sec
                 Count       =    21043 #     ( 90.1%)

        Tx Delay Average     =    0.657 sec
                 Std Dev     =    0.488 sec
                 Minimum     =    0.000 sec
                 Maximum     =    4.555 sec
                 Count       =     2304 #     (  9.9%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.150 s	0.437 s	0.381 s	5.026 s	23347	55.588 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	3	0.0	0.0	|
0.500	63	0.3	0.3	|
0.600	357	1.5	1.8	|****
0.700	1152	4.9	6.7	|**********
0.800	2039	8.7	15.5	|*************************
0.900	2931	12.6	28.0	|************************************
1.000	3278	14.0	42.1	|**
1.100	3117	13.4	55.4	|************************************
1.200	2702	11.6	67.0	|********************************
1.300	1923	8.2	75.2	|************************
1.400	1447	6.2	81.4	|******************
1.500	993	4.3	85.7	|************
1.600	784	3.4	89.0	|**********
1.700	569	2.4	91.5	|*******
1.800	435	1.9	93.3	|*****
1.900	280	1.2	94.5	|***
2.000	224	1.0	95.5	|**
2.100	190	0.8	96.3	|**
2.200	159	0.7	97.0	|**
2.300	124	0.5	97.5	|*
2.400	99	0.4	98.0	|*
2.500	85	0.4	98.3	|*
2.600	66	0.3	98.6	|
2.700	48	0.2	98.8	|
2.800	37	0.2	99.0	|
2.900	29	0.1	99.1	|
3.000	30	0.1	99.2	|
3.100	25	0.1	99.3	|
3.200	18	0.1	99.4	|
3.300	15	0.1	99.5	|
3.400	21	0.1	99.6	|
3.500	13	0.1	99.6	|
3.600	11	0.0	99.7	|
3.700	12	0.1	99.7	|
3.800	11	0.0	99.8	|
3.900	5	0.0	99.8	|
4.000	10	0.0	99.8	|
4.100	10	0.0	99.9	|
4.200	11	0.0	99.9	|
4.300	1	0.0	99.9	|

4.400	3	0.0	99.9
4.500	4	0.0	99.9
4.600	3	0.0	100.0
4.700	2	0.0	100.0
4.800	1	0.0	100.0
4.900	4	0.0	100.0
5.000	1	0.0	100.0
5.100	2	0.0	100.0

Simulator Report

NonStop/SQL

1 * 8 VLX SYSTEMS

Throughput - 48 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                =   Test 02; 48 TPS
Test Date                 =   87-02-11
Test Length               =   22:06:12 - 22:19:53  (821 sec)
Window Length             =   22:10:00 - 22:17:00  (420 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    6.000 tx/sec
Tx Arrival Rate/Terminal  =    0.075 tx/sec
Tx Inter-Arrival Time     =   13.333 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ----------------------------------

Tx Throughput              =    47.498 tx/sec
Tx Response Time Average   =     0.903 sec
                 Std Dev   =     0.406 sec
                 Minimum   =     0.276 sec
                 Maximum   =     9.515 sec
                 Count     =     19949 #

Tx Response Time 10%       =     0.650 sec  ( 14.3%)
                 20%       =     0.700 sec  ( 22.5%)
                 30%       =     0.750 sec  ( 31.9%)
                 40%       =     0.800 sec  ( 41.6%)
                 50%       =     0.850 sec  ( 51.5%)
                 60%       =     0.900 sec  ( 60.7%)
                 70%       =     1.000 sec  ( 75.3%)
                 80%       =     1.050 sec  ( 80.6%)
                 90%       =     1.200 sec  ( 90.4%)
                 95%       =     1.400 sec  ( 95.3%)

        Tx Think Average   =    13.351 sec
                 Std Dev   =    13.350 sec
                 Minimum   =     0.002 sec
                 Maximum   =   116.045 sec
                 Count     =     18596 #    ( 93.2%)

        Tx Delay Average   =     0.534 sec
                 Std Dev   =     0.571 sec
                 Minimum   =     0.001 sec
                 Maximum   =     6.935 sec
                 Count     =      1353 #    (  6.8%)

-----------------------------------------------------------------------
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.903 s	0.406 s	0.276 s	9.515 s	19949	47.498 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	1	0.0	0.0	|
0.400	34	0.2	0.2	|
0.500	353	1.8	1.9	|***
0.600	1272	6.4	8.3	|************
0.700	2821	14.1	22.5	|******************************
0.800	3827	19.2	41.6	|**
0.900	3802	19.1	60.7	|**
1.000	2909	14.6	75.3	|*******************************
1.100	1918	9.6	84.9	|********************
1.200	1088	5.5	90.4	|**********
1.300	585	2.9	93.3	|******
1.400	394	2.0	95.3	|****
1.500	244	1.2	96.5	|**
1.600	169	0.8	97.3	|*
1.700	134	0.7	98.0	|*
1.800	85	0.4	98.4	|
1.900	70	0.4	98.8	|
2.000	50	0.3	99.0	|
2.100	50	0.3	99.3	|
2.200	19	0.1	99.4	|
2.300	14	0.1	99.4	|
2.400	16	0.1	99.5	|
2.500	11	0.1	99.6	|
2.600	5	0.0	99.6	|
2.700	9	0.0	99.7	|
2.800	7	0.0	99.7	|
2.900	4	0.0	99.7	|
3.000	1	0.0	99.7	|
3.100	2	0.0	99.7	|
3.200	1	0.0	99.7	|
3.300	1	0.0	99.7	|
3.400	1	0.0	99.7	|
3.500	0	0.0	99.7	|
3.600	1	0.0	99.7	|
3.700	1	0.0	99.7	|
3.800	0	0.0	99.7	|
3.900	1	0.0	99.8	|
4.000	0	0.0	99.8	|
4.100	1	0.0	99.8	|
4.200	0	0.0	99.8	|
4.300	0	0.0	99.8	|
4.400	0	0.0	99.8	|

4.500	0	0.0	99.8
4.600	0	0.0	99.8
4.700	0	0.0	99.8
4.800	0	0.0	99.8
4.900	0	0.0	99.8
5.000	0	0.0	99.8
5.100	0	0.0	99.8
5.200	1	0.0	99.8
5.300	1	0.0	99.8
5.400	0	0.0	99.8
5.500	1	0.0	99.8
5.600	0	0.0	99.8
5.700	0	0.0	99.8
5.800	0	0.0	99.8
5.900	1	0.0	99.8
6.000	2	0.0	99.8
6.100	1	0.0	99.8
6.200	2	0.0	99.8
6.300	3	0.0	99.8
6.400	3	0.0	99.8
6.500	1	0.0	99.8
6.600	3	0.0	99.9
6.700	0	0.0	99.9
6.800	3	0.0	99.9
6.900	2	0.0	99.9
7.000	5	0.0	99.9
7.100	0	0.0	99.9
7.200	0	0.0	99.9
7.300	1	0.0	99.9
7.400	1	0.0	99.9
7.500	1	0.0	99.9
7.600	2	0.0	99.9
7.700	1	0.0	99.9
7.800	1	0.0	99.9
7.900	4	0.0	100.0
8.000	2	0.0	100.0
8.100	2	0.0	100.0
8.200	2	0.0	100.0
8.300	0	0.0	100.0
8.400	1	0.0	100.0
8.500	0	0.0	100.0
8.600	0	0.0	100.0
8.700	0	0.0	100.0
8.800	0	0.0	100.0
8.900	0	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	0	0.0	100.0
9.500	0	0.0	100.0
9.600	1	0.0	100.0

Simulator Report

NonStop/SQL

1 * 8 VLX SYSTEMS

Throughput - 40 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test 01; 40 TPS
Test Date                 =   87-02-11
Test Length               =   21:46:24 - 21:58:48  (744 sec)
Window Length             =   21:50:00 - 21:57:00  (420 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    5.000 tx/sec
Tx Arrival Rate/Terminal  =    0.063 tx/sec
Tx Inter-Arrival Time     =   16.000 sec
```

----- MEASUREMENT RESULTS (vs 2.02) ----------

Tx Throughput		=	39.781	tx/sec	
Tx Response Time	Average	=	0.740	sec	
	Std Dev	=	0.189	sec	
	Minimum	=	0.272	sec	
	Maximum	=	3.697	sec	
	Count	=	16708	#	
Tx Response Time	10%	=	0.550	sec	(12.2%)
	20%	=	0.600	sec	(20.5%)
	30%	=	0.650	sec	(31.5%)
	40%	=	0.700	sec	(44.1%)
	50%	=	0.750	sec	(56.8%)
	60%	=	0.800	sec	(69.1%)
	70%	=	0.850	sec	(78.6%)
	80%	=	0.900	sec	(85.6%)
	90%	=	0.950	sec	(90.2%)
	95%	=	1.050	sec	(95.3%)
Tx Think	Average	=	16.089	sec	
	Std Dev	=	16.125	sec	
	Minimum	=	0.001	sec	
	Maximum	=	155.104	sec	
	Count	=	15895	#	(95.1%)
Tx Delay	Average	=	0.387	sec	
	Std Dev	=	0.257	sec	
	Minimum	=	0.000	sec	
	Maximum	=	2.561	sec	
	Count	=	813	#	(4.9%)

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.740 s	0.189 s	0.272 s	3.697 s	16708	39.781 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	3	0.0	0.0	|
0.400	141	0.8	0.9	|*
0.500	924	5.5	6.4	|*********
0.600	2363	14.1	20.5	|**********
0.700	3944	23.6	44.1	|**********
0.800	4178	25.0	69.1	|**********
0.900	2748	16.4	85.6	|**********
1.000	1316	7.9	93.5	|**********
1.100	552	3.3	96.8	|*****
1.200	248	1.5	98.3	|**
1.300	112	0.7	98.9	|*
1.400	68	0.4	99.3	|
1.500	35	0.2	99.5	|
1.600	23	0.1	99.7	|
1.700	11	0.1	99.7	|
1.800	11	0.1	99.8	|
1.900	5	0.0	99.8	|
2.000	5	0.0	99.9	|
2.100	1	0.0	99.9	|
2.200	5	0.0	99.9	|
2.300	1	0.0	99.9	|
2.400	1	0.0	99.9	|
2.500	1	0.0	99.9	|
2.600	1	0.0	99.9	|
2.700	0	0.0	99.9	|
2.800	3	0.0	100.0	|
2.900	1	0.0	100.0	|
3.000	1	0.0	100.0	|
3.100	1	0.0	100.0	|
3.200	0	0.0	100.0	|
3.300	0	0.0	100.0	|
3.400	0	0.0	100.0	|
3.500	1	0.0	100.0	|
3.600	1	0.0	100.0	|
3.700	3	0.0	100.0	|

ENFORM Report

NonStop/SQL

1 * 8 VLX SYSTEM

Throughput - 59 TPS

CPU UTILISATION PER TRANSACTION
SUMMARIZED FOR A 7 MINUTE PERIOD

LOADID	NUM-TRANS	TPS	AVG CPU BUSY %	CPU PER TRANS	INTRPT BUSY PER TRANS	DISC IOS PER TRANS	CHITS PER TRANS
SQL59	24596	58.562	86.310	.1179	.028	3.5	5.0

CPU UTILISATION BY PROCESS TYPE

PROCESS-TYPE	CPU PER TRANS	SENDS PER TRANS	RECVS PER TRANS	CPU BUSY PERCENT
ATB-B	5.287	.000	.516	30.962
ATB-P	24.541	.894	3.033	143.718
AUDIT-B	.591	.001	.529	3.459
AUDIT-P	1.400	.529	.530	8.198
FOX	.681	.374	.390	3.986
HISTORY-B	2.764	.000	.058	16.189
HISTORY-P	5.328	.112	1.000	31.204
INTERRUPTS	27.728	.000	.000	162.380
MISC	.243	.136	.209	1.425
PATHMON	.000	.000	.000	.000
SERVER	20.556	4.000	1.000	120.378
TCP	18.201	2.000	.000	106.592
TMF MONITOR	3.139	.000	1.329	18.385
TMP	.783	.096	.001	4.584
X.25	6.574	.000	1.000	38.500
	117.816	8.140	9.594	689.960

DETAILED CPU UTILIZATION BY TRANSACTION

PROCESS-TYPE	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	CPU 6	CPU 7
ATB-B	3.720	3.690	3.900	3.900	3.920	3.990	3.800	4.000
ATB-P	16.980	17.650	17.730	17.690	18.530	17.900	19.690	17.510
AUDIT-B	.000	.000	.000	.000	.000	3.450	.000	.000
AUDIT-P	.000	.000	.000	.000	8.190	.000	.000	.000
FOX	3.980	.000	.000	.000	.000	.000	.000	.000
HISTORY-B	.000	16.180	.000	.000	.000	.000	.000	.000
HISTORY-P	31.200	.000	.000	.000	.000	.000	.000	.000
INTERRUPTS	20.970	18.430	17.440	19.540	22.090	21.680	20.700	21.490
MISC	1.010	.320	.010	.010	.010	.010	.010	.010
PATHMON	.000	.000	.000	.000	.000	.000	.000	.000
SERVER	12.780	20.480	19.490	15.540	15.240	8.760	15.490	12.560
TCP	.000	10.440	.000	15.990	16.580	27.360	15.410	20.790
TMF MONITOR	1.530	2.520	.880	2.390	2.530	3.310	2.380	2.800
TMP	.000	.000	4.580	.000	.000	.000	.000	.000
X.25	.000	.000	9.590	9.530	.000	.000	9.650	9.710
	92.170	89.710	73.620	84.590	87.090	86.460	87.130	88.870

LOADID - SQL59

Disc Utilisation Summary per Transaction

USAGE	REQ PER TRANS	READS PER TRANS	WRITES PER TRANS	512 HITS PER TRANS	1024 HITS PER TRANS	2048 HITS PER TRANS	4096 HITS PER TRANS
ATB	3.085	1.001	2.114	1.029	.000	.984	2.011
AUDIT TRAILS	.530	.001	.208	.000	.000	.000	.001
HISTORY	1.001	.002	.023	.000	.000	.000	1.000
UNUSED	.000	.006	.006	.000	.000	.000	.003
XRAY	.027	.004	.091	.000	.000	.000	.000

Loadid - SQL59

Disc Utilisation Summary

USAGE	SUM REQUESTS	SUM READS	SUM WRITES	512 HITS	1024 HITS	2048 HITS	4096 HITS
ATB	75878	24617	52008	25307	0	24209	49454
AUDIT TRAILS	13038	17	5121	0	0	0	19
HISTORY	24610	41	572	0	0	0	24597
UNUSED	0	156	150	0	0	0	70
XRAY	672	107	2228	0	0	0	0

Loadid - SQL59

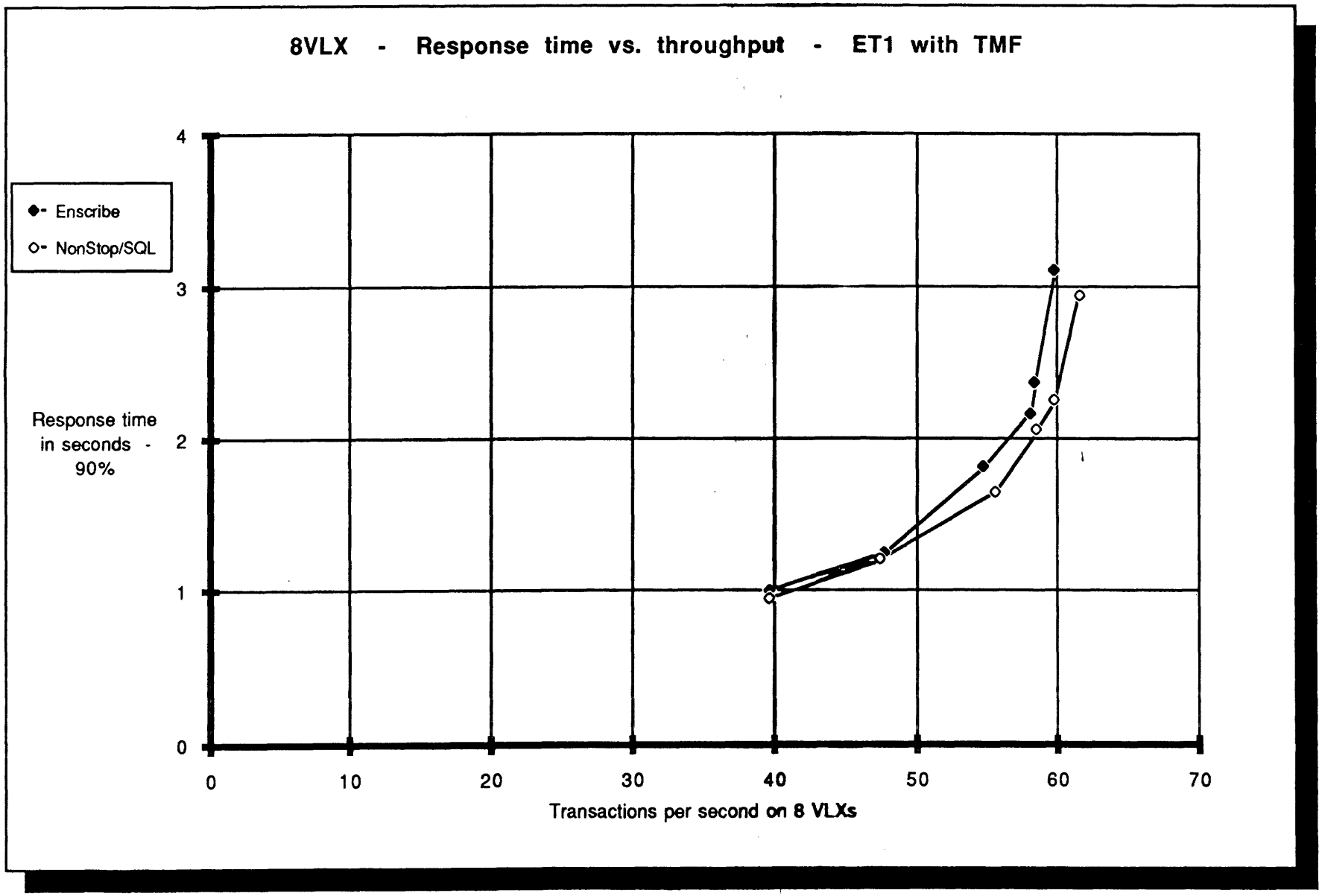
Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XA1	3181	352711	669163	22.761	11.997
$XA2	3078	341771	648049	22.043	11.625
$XA3	3042	336835	640122	21.773	11.457
$XA4	3038	336997	639162	21.740	11.462
$XA5	2982	330876	627474	21.344	11.255
$XA6	3082	342285	648434	22.057	11.643
$XA7	3073	341414	646731	21.998	11.613
$XA8	3110	345045	654450	22.260	11.736
	24586	2727934	5173585		

Loadid - SQL59

Communications line summary per transactions

LOADID	Transactions	Output per trans	input per trans
SQL59	24596	199.927	100.012



* For 90% of transactions

Simulator Report

ENSCRIBE

1 * 8 VLX SYSTEMS

Throughput - 60 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test 07; 80 TPS; COB
Test Date                  =   87-02-12
Test Length                =   02:21:40 - 02:33:15  (695 sec)
Window Length              =   02:25:00 - 02:32:00  (420 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =   10.000 tx/sec
Tx Arrival Rate/Terminal   =    0.125 tx/sec
Tx Inter-Arrival Time      =    8.000 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ---------------------------------

Tx Throughput              =   59.855 tx/sec
Tx Response Time Average   =    2.000 sec
                 Std Dev   =    0.825 sec
                 Minimum   =    0.518 sec
                 Maximum   =    9.229 sec
                 Count     =    25139 #

Tx Response Time 10%       =    1.150 sec  ( 10.3%)
                 20%       =    1.350 sec  ( 20.8%)
                 30%       =    1.550 sec  ( 33.2%)
                 40%       =    1.700 sec  ( 42.6%)
                 50%       =    1.850 sec  ( 51.3%)
                 60%       =    2.050 sec  ( 61.6%)
                 70%       =    2.300 sec  ( 71.8%)
                 80%       =    2.600 sec  ( 80.8%)
                 90%       =    3.100 sec  ( 90.4%)
                 95%       =    3.550 sec  ( 95.0%)

        Tx Think Average   =    8.034 sec
                 Std Dev   =    8.028 sec
                 Minimum   =    0.001 sec
                 Maximum   =   67.690 sec
                 Count     =    19498 #    ( 77.6%)

        Tx Delay Average   =    1.191 sec
                 Std Dev   =    0.882 sec
                 Minimum   =    0.001 sec
                 Maximum   =    7.133 sec
                 Count     =     5641 #    ( 22.4%)

-----------------------------------------------------------------------
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
2.000 s	0.825 s	0.518 s	9.229 s	25139	59.855 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	0	0.0	0.0	|
0.500	0	0.0	0.0	|
0.600	19	0.1	0.1	|
0.700	59	0.2	0.3	|*
0.800	174	0.7	1.0	|****
0.900	378	1.5	2.5	|*********
1.000	577	2.3	4.8	|***************
1.100	913	3.6	8.4	|***********************
1.200	1057	4.2	12.6	|**************************
1.300	1301	5.2	17.8	|**********************************
1.400	1465	5.8	23.6	|**************************************
1.500	1587	6.3	30.0	|**
1.600	1601	6.4	36.3	|**
1.700	1588	6.3	42.6	|**
1.800	1445	5.7	48.4	|************************************
1.900	1406	5.6	54.0	|***********************************
2.000	1338	5.3	59.3	|**********************************
2.100	1108	4.4	63.7	|****************************
2.200	1070	4.3	68.0	|***************************
2.300	964	3.8	71.8	|************************
2.400	848	3.4	75.2	|**********************
2.500	752	3.0	78.2	|*******************
2.600	673	2.7	80.8	|*****************
2.700	628	2.5	83.3	|****************
2.800	511	2.0	85.4	|*************
2.900	509	2.0	87.4	|*************
3.000	375	1.5	88.9	|*********
3.100	391	1.6	90.4	|**********
3.200	341	1.4	91.8	|********
3.300	286	1.1	92.9	|*******
3.400	239	1.0	93.9	|******
3.500	210	0.8	94.7	|*****
3.600	160	0.6	95.4	|****
3.700	149	0.6	96.0	|***
3.800	141	0.6	96.5	|***
3.900	116	0.5	97.0	|***
4.000	109	0.4	97.4	|**
4.100	93	0.4	97.8	|**
4.200	81	0.3	98.1	|**
4.300	63	0.3	98.4	|*
4.400	47	0.2	98.5	|*

4.500	40	0.2	98.7	*
4.600	36	0.1	98.8	
4.700	34	0.1	99.0	
4.800	35	0.1	99.1	
4.900	21	0.1	99.2	
5.000	23	0.1	99.3	
5.100	18	0.1	99.4	
5.200	19	0.1	99.4	
5.300	13	0.1	99.5	
5.400	14	0.1	99.5	
5.500	14	0.1	99.6	
5.600	10	0.0	99.6	
5.700	7	0.0	99.7	
5.800	7	0.0	99.7	
5.900	5	0.0	99.7	
6.000	4	0.0	99.7	
6.100	10	0.0	99.8	
6.200	7	0.0	99.8	
6.300	6	0.0	99.8	
6.400	4	0.0	99.8	
6.500	7	0.0	99.9	
6.600	3	0.0	99.9	
6.700	2	0.0	99.9	
6.800	5	0.0	99.9	
6.900	5	0.0	99.9	
7.000	3	0.0	99.9	
7.100	0	0.0	99.9	
7.200	1	0.0	99.9	
7.300	1	0.0	99.9	
7.400	3	0.0	100.0	
7.500	1	0.0	100.0	
7.600	1	0.0	100.0	
7.700	2	0.0	100.0	
7.800	0	0.0	100.0	
7.900	1	0.0	100.0	
8.000	0	0.0	100.0	
8.100	0	0.0	100.0	
8.200	0	0.0	100.0	
8.300	1	0.0	100.0	
8.400	1	0.0	100.0	
8.500	0	0.0	100.0	
8.600	0	0.0	100.0	
8.700	1	0.0	100.0	
8.800	1	0.0	100.0	
8.900	0	0.0	100.0	
9.000	0	0.0	100.0	
9.100	0	0.0	100.0	
9.200	0	0.0	100.0	
9.300	1	0.0	100.0	

Simulator Report

ENSCRIBE

1 * 8 VLX SYSTEMS

Throughput - 59 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test 05; 64 TPS; COB
Test Date                 =   87-02-12
Test Length               =   02:03:54 - 02:15:32  (698 sec)
Window Length             =   02:07:00 - 02:14:00  (420 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    8.000 tx/sec
Tx Arrival Rate/Terminal  =    0.100 tx/sec
Tx Inter-Arrival Time     =   10.000 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ---------------------------------

Tx Throughput              =    58.464 tx/sec
Tx Response Time Average   =     1.582 sec
                 Std Dev   =     0.561 sec
                 Minimum   =     0.469 sec
                 Maximum   =     4.911 sec
                 Count     =     24555 #

Tx Response Time 10%       =     1.000 sec  ( 11.2%)
                 20%       =     1.150 sec  ( 22.4%)
                 30%       =     1.250 sec  ( 30.5%)
                 40%       =     1.400 sec  ( 43.6%)
                 50%       =     1.500 sec  ( 51.9%)
                 60%       =     1.650 sec  ( 62.8%)
                 70%       =     1.800 sec  ( 71.8%)
                 80%       =     2.000 sec  ( 81.0%)
                 90%       =     2.350 sec  ( 90.9%)
                 95%       =     2.650 sec  ( 95.1%)

        Tx Think Average   =     9.982 sec
                 Std Dev   =    10.018 sec
                 Minimum   =     0.001 sec
                 Maximum   =    86.490 sec
                 Count     =     20958 #    ( 85.4%)

        Tx Delay Average   =     0.927 sec
                 Std Dev   =     0.635 sec
                 Minimum   =     0.000 sec
                 Maximum   =     4.063 sec
                 Count     =      3597 #    ( 14.6%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.582 s	0.561 s	0.469 s	4.911 s	24555	58.464 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	
0.100	0	0.0	0.0	
0.200	0	0.0	0.0	
0.300	0	0.0	0.0	
0.400	0	0.0	0.0	
0.500	4	0.0	0.0	
0.600	47	0.2	0.2	
0.700	174	0.7	0.9	***
0.800	426	1.7	2.7	********
0.900	823	3.4	6.0	**********
1.000	1269	5.2	11.2	************************
1.100	1753	7.1	18.3	*********************************
1.200	1966	8.0	26.3	**************************************
1.300	2105	8.6	34.9	**
1.400	2135	8.7	43.6	**
1.500	2031	8.3	51.9	**************************************
1.600	1864	7.6	59.4	**********************************
1.700	1638	6.7	66.1	*******************************
1.800	1400	5.7	71.8	**************************
1.900	1194	4.9	76.7	**********************
2.000	1065	4.3	81.0	********************
2.100	879	3.6	84.6	*****************
2.200	720	2.9	87.5	**************
2.300	573	2.3	89.9	**********
2.400	460	1.9	91.7	*********
2.500	393	1.6	93.3	*******
2.600	302	1.2	94.6	*****
2.700	250	1.0	95.6	****
2.800	195	0.8	96.4	***
2.900	165	0.7	97.1	***
3.000	137	0.6	97.6	**
3.100	112	0.5	98.1	**
3.200	85	0.3	98.4	*
3.300	74	0.3	98.7	*
3.400	46	0.2	98.9	
3.500	47	0.2	99.1	
3.600	33	0.1	99.2	
3.700	37	0.2	99.4	
3.800	40	0.2	99.5	
3.900	22	0.1	99.6	
4.000	20	0.1	99.7	
4.100	13	0.1	99.8	
4.200	16	0.1	99.8	
4.300	14	0.1	99.9	

4.400	7	0.0	99.9
4.500	3	0.0	99.9
4.600	5	0.0	99.9
4.700	7	0.0	100.0
4.800	3	0.0	100.0
4.900	1	0.0	100.0
5.000	2	0.0	100.0

Simulator Report

ENSCRIBE

1 * 8 VLX SYSTEMS

Throughput - 58 TPS

```
--------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                         *
--------------------------------------------------------------------------

----- PARAMETERS ---------------------------------------------------------

Test Title                 =   Test 04; 60 TPS; COB
Test Date                  =   87-02-12
Test Length                =   01:39:16 - 01:53:20  (844 sec)
Window Length              =   01:44:00 - 01:51:00  (420 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    7.500 tx/sec
Tx Arrival Rate/Terminal   =    0.094 tx/sec
Tx Inter-Arrival Time      =   10.667 sec
```

----- MEASUREMENT RESULTS (vs 2.02) ----------

Tx Throughput		=	58.050	tx/sec	
Tx Response Time	Average	=	1.454	sec	
	Std Dev	=	0.511	sec	
	Minimum	=	0.380	sec	
	Maximum	=	5.220	sec	
	Count	=	24381	#	
Tx Response Time	10%	=	0.950	sec	(13.0%)
	20%	=	1.050	sec	(21.0%)
	30%	=	1.150	sec	(30.3%)
	40%	=	1.250	sec	(40.2%)
	50%	=	1.400	sec	(53.8%)
	60%	=	1.500	sec	(61.7%)
	70%	=	1.650	sec	(71.6%)
	80%	=	1.850	sec	(81.8%)
	90%	=	2.150	sec	(90.7%)
	95%	=	2.450	sec	(95.4%)
Tx Think	Average	=	10.635	sec	
	Std Dev	=	10.722	sec	
	Minimum	=	0.002	sec	
	Maximum	=	99.599	sec	
	Count	=	21209	#	(87.0%)
Tx Delay	Average	=	0.846	sec	
	Std Dev	=	0.584	sec	
	Minimum	=	0.000	sec	
	Maximum	=	4.574	sec	
	Count	=	3172	#	(13.0%)

---

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
1.454 s	0.511 s	0.380 s	5.220 s	24381	58.050 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	
0.100	0	0.0	0.0	
0.200	0	0.0	0.0	
0.300	0	0.0	0.0	
0.400	1	0.0	0.0	
0.500	6	0.0	0.0	
0.600	84	0.3	0.4	*
0.700	314	1.3	1.7	*****
0.800	689	2.8	4.5	**********
0.900	1218	5.0	9.5	**********
1.000	1811	7.4	16.9	**********
1.100	2107	8.6	25.6	**********
1.200	2328	9.5	35.1	**********
1.300	2413	9.9	45.0	**********
1.400	2136	8.8	53.8	**********
1.500	1925	7.9	61.7	**********
1.600	1657	6.8	68.5	**********
1.700	1488	6.1	74.6	**********
1.800	1237	5.1	79.6	**********
1.900	980	4.0	83.6	**********
2.000	783	3.2	86.9	**********
2.100	651	2.7	89.5	**********
2.200	539	2.2	91.7	**********
2.300	435	1.8	93.5	*******
2.400	322	1.3	94.8	*****
2.500	268	1.1	95.9	****
2.600	224	0.9	96.9	***
2.700	154	0.6	97.5	**
2.800	129	0.5	98.0	**
2.900	96	0.4	98.4	*
3.000	70	0.3	98.7	*
3.100	65	0.3	99.0	*
3.200	50	0.2	99.2	
3.300	33	0.1	99.3	
3.400	32	0.1	99.4	
3.500	30	0.1	99.6	
3.600	20	0.1	99.6	
3.700	17	0.1	99.7	
3.800	13	0.1	99.8	
3.900	10	0.0	99.8	
4.000	10	0.0	99.9	
4.100	3	0.0	99.9	
4.200	4	0.0	99.9	
4.300	5	0.0	99.9	
4.400	3	0.0	99.9	

4.500	7	0.0	99.9
4.600	2	0.0	100.0
4.700	3	0.0	100.0
4.800	4	0.0	100.0
4.900	3	0.0	100.0
5.000	0	0.0	100.0
5.100	0	0.0	100.0
5.200	1	0.0	100.0
5.300	1	0.0	100.0

Simulator Report

ENSCRIBE

1 * 8 VLX SYSTEMS

Throughput - 55 TPS

```
--------------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                               *
--------------------------------------------------------------------------------

----- PARAMETERS -----------------------------------------------------------

Test Title                =   Test 03; 56 TPS; COB
Test Date                 =   87-02-12
Test Length               =   01:19:37 - 01:31:38  (721 sec)
Window Length             =   01:22:00 - 01:29:00  (420 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    7.000 tx/sec
Tx Arrival Rate/Terminal  =    0.088 tx/sec
Tx Inter-Arrival Time     =   11.429 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ----------------------------------

Tx Throughput              =    54.748 tx/sec
Tx Response Time Average   =     1.243 sec
                 Std Dev   =     0.525 sec
                 Minimum   =     0.372 sec
                 Maximum   =     9.749 sec
                 Count     =     22994 #

Tx Response Time 10%       =     0.800 sec  ( 12.1%)
                 20%       =     0.900 sec  ( 21.7%)
                 30%       =     1.000 sec  ( 33.5%)
                 40%       =     1.100 sec  ( 45.7%)
                 50%       =     1.150 sec  ( 51.5%)
                 60%       =     1.250 sec  ( 61.6%)
                 70%       =     1.400 sec  ( 73.5%)
                 80%       =     1.550 sec  ( 81.7%)
                 90%       =     1.800 sec  ( 90.1%)
                 95%       =     2.150 sec  ( 95.5%)

        Tx Think Average   =    11.494 sec
                 Std Dev   =    11.514 sec
                 Minimum   =     0.001 sec
                 Maximum   =   108.485 sec
                 Count     =     20566 #    ( 89.4%)

        Tx Delay Average   =     0.726 sec
                 Std Dev   =     0.593 sec
                 Minimum   =     0.000 sec
                 Maximum   =     7.661 sec
                 Count     =      2428 #    ( 10.6%)
```

```
----------------------------------------------------------------------
                  *  ET1 Response Time Distribution  *
----------------------------------------------------------------------
   Mean         Std         Min         Max         Count      Thruput
   1.243 s     0.525 s     0.372 s     9.749 s     22994       54.748 tx/s
----------------------------------------------------------------------

RespTime Count  Pct   CumPct
-------- ----- -----  ------
   0.000     0   0.0    0.0 |
   0.100     0   0.0    0.0 |
   0.200     0   0.0    0.0 |
   0.300     0   0.0    0.0 |
   0.400     4   0.0    0.0 |
   0.500    42   0.2    0.2 |
   0.600   299   1.3    1.5 |****
   0.700   842   3.7    5.2 |************
   0.800  1602   7.0   12.1 |************************
   0.900  2210   9.6   21.7 |*********************************
   1.000  2708  11.8   33.5 |****************************************
   1.100  2799  12.2   45.7 |******************************************
   1.200  2524  11.0   56.7 |*************************************
   1.300  2110   9.2   65.8 |*******************************
   1.400  1753   7.6   73.5 |**************************
   1.500  1305   5.7   79.1 |*******************
   1.600  1064   4.6   83.8 |***************
   1.700   775   3.4   87.1 |***********
   1.800   689   3.0   90.1 |**********
   1.900   474   2.1   92.2 |*******
   2.000   344   1.5   93.7 |*****
   2.100   298   1.3   95.0 |****
   2.200   221   1.0   96.0 |***
   2.300   201   0.9   96.8 |***
   2.400   132   0.6   97.4 |*
   2.500   113   0.5   97.9 |*
   2.600    79   0.3   98.2 |*
   2.700    67   0.3   98.5 |*
   2.800    64   0.3   98.8 |
   2.900    38   0.2   99.0 |
   3.000    40   0.2   99.1 |
   3.100    27   0.1   99.3 |
   3.200    24   0.1   99.4 |
   3.300    18   0.1   99.4 |
   3.400    13   0.1   99.5 |
   3.500    12   0.1   99.6 |
   3.600    13   0.1   99.6 |
   3.700     7   0.0   99.6 |
   3.800     4   0.0   99.7 |
   3.900     4   0.0   99.7 |
   4.000     0   0.0   99.7 |
   4.100     2   0.0   99.7 |
   4.200     1   0.0   99.7 |
   4.300     2   0.0   99.7 |
```

4.400	3	0.0	99.7
4.500	3	0.0	99.7
4.600	2	0.0	99.7
4.700	2	0.0	99.7
4.800	2	0.0	99.7
4.900	9	0.0	99.8
5.000	5	0.0	99.8
5.100	3	0.0	99.8
5.200	1	0.0	99.8
5.300	0	0.0	99.8
5.400	4	0.0	99.8
5.500	0	0.0	99.8
5.600	2	0.0	99.9
5.700	2	0.0	99.9
5.800	0	0.0	99.9
5.900	1	0.0	99.9
6.000	1	0.0	99.9
6.100	2	0.0	99.9
6.200	3	0.0	99.9
6.300	1	0.0	99.9
6.400	1	0.0	99.9
6.500	0	0.0	99.9
6.600	0	0.0	99.9
6.700	0	0.0	99.9
6.800	0	0.0	99.9
6.900	1	0.0	99.9
7.000	0	0.0	99.9
7.100	0	0.0	99.9
7.200	1	0.0	99.9
7.300	0	0.0	99.9
7.400	0	0.0	99.9
7.500	0	0.0	99.9
7.600	0	0.0	99.9
7.700	0	0.0	99.9
7.800	0	0.0	99.9
7.900	1	0.0	99.9
8.000	2	0.0	99.9
8.100	2	0.0	99.9
8.200	2	0.0	99.9
8.300	2	0.0	99.9
8.400	3	0.0	100.0
8.500	3	0.0	100.0
8.600	2	0.0	100.0
8.700	1	0.0	100.0
8.800	1	0.0	100.0
8.900	0	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	1	0.0	100.0
9.300	0	0.0	100.0
9.400	0	0.0	100.0
9.500	0	0.0	100.0
9.600	0	0.0	100.0
9.700	0	0.0	100.0

9.800 1 0.0 100.0 |

Simulator Report

ENSCRIBE

1 * 8 VLX SYSTEMS

Throughput - 48 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test 02; 48 TPS; COB
Test Date                 =   87-02-12
Test Length               =   00:59:37 - 01:10:42  (665 sec)
Window Length             =   01:02:00 - 01:09:00  (420 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    6.000 tx/sec
Tx Arrival Rate/Terminal  =    0.075 tx/sec
Tx Inter-Arrival Time     =   13.333 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ---------------------------------

Tx Throughput                =   47.845 tx/sec
Tx Response Time Average     =    0.894 sec
                 Std Dev     =    0.259 sec
                 Minimum     =    0.301 sec
                 Maximum     =    3.542 sec
                 Count       =    20095 #

Tx Response Time 10%         =    0.650 sec  ( 14.4%)
                 20%         =    0.700 sec  ( 21.8%)
                 30%         =    0.750 sec  ( 30.7%)
                 40%         =    0.800 sec  ( 40.2%)
                 50%         =    0.900 sec  ( 58.9%)
                 60%         =    0.950 sec  ( 66.6%)
                 70%         =    1.000 sec  ( 73.0%)
                 80%         =    1.100 sec  ( 82.8%)
                 90%         =    1.250 sec  ( 91.5%)
                 95%         =    1.400 sec  ( 95.7%)

        Tx Think Average     =   13.397 sec
                 Std Dev     =   13.461 sec
                 Minimum     =    0.002 sec
                 Maximum     =  129.230 sec
                 Count       =    18738 #    ( 93.2%)

        Tx Delay Average     =    0.487 sec
                 Std Dev     =    0.315 sec
                 Minimum     =    0.000 sec
                 Maximum     =    1.828 sec
                 Count       =     1357 #    (  6.8%)

-----------------------------------------------------------------------
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.894 s	0.259 s	0.301 s	3.542 s	20095	47.845 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	18	0.1	0.1	|
0.500	355	1.8	1.9	|***
0.600	1335	6.6	8.5	|**************
0.700	2669	13.3	21.8	|*****************************
0.800	3692	18.4	40.2	|**
0.900	3759	18.7	58.9	|***
1.000	2847	14.2	73.0	|********************************
1.100	1963	9.8	82.8	|**********************
1.200	1331	6.6	89.4	|**************
1.300	805	4.0	93.4	|********
1.400	448	2.2	95.7	|*****
1.500	291	1.4	97.1	|***
1.600	196	1.0	98.1	|**
1.700	136	0.7	98.8	|*
1.800	72	0.4	99.1	|
1.900	51	0.3	99.4	|
2.000	46	0.2	99.6	|
2.100	35	0.2	99.8	|
2.200	13	0.1	99.8	|
2.300	12	0.1	99.9	|
2.400	6	0.0	99.9	|
2.500	1	0.0	99.9	|
2.600	6	0.0	100.0	|
2.700	1	0.0	100.0	|
2.800	3	0.0	100.0	|
2.900	1	0.0	100.0	|
3.000	1	0.0	100.0	|
3.100	0	0.0	100.0	|
3.200	1	0.0	100.0	|
3.300	0	0.0	100.0	|
3.400	0	0.0	100.0	|
3.500	0	0.0	100.0	|
3.600	1	0.0	100.0	|

Simulator Report

ENSCRIBE

1 * 8 VLX SYSTEMS

Throughput - 40 TPS

```
------------------------------------------------------------------------
*   ET1 Report (Driver System) -  TOPGUN PROJECT                       *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                =   Test 01; 40 TPS; COB
Test Date                 =   87-02-12
Test Length               =   00:26:26 - 00:39:17  (771 sec)
Window Length             =   00:30:00 - 00:37:00  (420 sec)
Tx Log File               =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line        =       80 #
Tx Arrival Process        =   RANDOM
Tx Arrival Rate/X25 Line  =    5.000 tx/sec
Tx Arrival Rate/Terminal  =    0.063 tx/sec
Tx Inter-Arrival Time     =   16.000 sec
```

```
----- MEASUREMENT RESULTS  (vs 2.02) ---------------------------------

Tx Throughput              =   39.807 tx/sec
Tx Response Time Average   =    0.755 sec
                 Std Dev   =    0.197 sec
                 Minimum   =    0.275 sec
                 Maximum   =    2.432 sec
                 Count     =    16719 #

Tx Response Time 10%       =    0.550 sec  ( 12.4%)
                 20%       =    0.600 sec  ( 20.5%)
                 30%       =    0.650 sec  ( 30.6%)
                 40%       =    0.700 sec  ( 42.4%)
                 50%       =    0.750 sec  ( 54.0%)
                 60%       =    0.800 sec  ( 65.0%)
                 70%       =    0.850 sec  ( 74.8%)
                 80%       =    0.900 sec  ( 82.2%)
                 90%       =    1.000 sec  ( 90.9%)
                 95%       =    1.150 sec  ( 96.1%)

        Tx Think Average   =   16.105 sec
                 Std Dev   =   16.141 sec
                 Minimum   =    0.003 sec
                 Maximum   =  155.191 sec
                 Count     =    15888 #    ( 95.0%)

        Tx Delay Average   =    0.397 sec
                 Std Dev   =    0.263 sec
                 Minimum   =    0.000 sec
                 Maximum   =    1.422 sec
                 Count     =      831 #    (  5.0%)
```

* ET1 Response Time Distribution *

Mean	Std	Min	Max	Count	Thruput
0.755 s	0.197 s	0.275 s	2.432 s	16719	39.807 tx/s

RespTime	Count	Pct	CumPct	
0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	5	0.0	0.0	|
0.400	146	0.9	0.9	|*
0.500	989	5.9	6.8	|*********
0.600	2283	13.7	20.5	|************************
0.700	3659	21.9	42.4	|**
0.800	3783	22.6	65.0	|***
0.900	2877	17.2	82.2	|*****************************
1.000	1449	8.7	90.9	|****************
1.100	670	4.0	94.9	|*******
1.200	372	2.2	97.1	|****
1.300	204	1.2	98.3	|**
1.400	111	0.7	99.0	|*
1.500	66	0.4	99.4	|
1.600	44	0.3	99.6	|
1.700	22	0.1	99.8	|
1.800	15	0.1	99.9	|
1.900	10	0.1	99.9	|
2.000	5	0.0	99.9	|
2.100	2	0.0	100.0	|
2.200	4	0.0	100.0	|
2.300	1	0.0	100.0	|
2.400	1	0.0	100.0	|
2.500	1	0.0	100.0	|

ENFORM Report

ENSCRIBE

1 * 8 VLX SYSTEM

Throughput - 59 TPS

CPU UTILISATION PER TRANSACTION
SUMMARIZED FOR A 7 MINUTE PERIOD

LOADID	NUM-TRANS	TPS	AVG CPU BUSY %	CPU PER TRANS	INTRPT BUSY PER TRANS	DISC IOS PER TRANS	CHITS PER TRANS
ENS58	24408	58.528	86.056	.1176	.029	3.4	9.1

CPU UTILISATION BY PROCESS TYPE

PROCESS-TYPE	CPU PER TRANS	SENDS PER TRANS	RECVS PER TRANS	CPU BUSY PERCENT
ATB-B	5.264	.003	.477	30.807
ATB-P	28.417	.828	6.040	166.319
AUDIT-B	.566	.001	.505	3.314
AUDIT-P	1.378	.506	.513	8.064
FOX	.062	.021	.034	.361
HISTORY-B	2.822	.000	.061	16.516
HISTORY-P	5.260	.116	1.000	30.788
INTERRUPTS	29.084	.000	.000	170.225
MISC	.255	.146	.223	1.492
PATHMON	.067	.008	.001	.392
SERVER	16.095	7.009	1.000	94.200
TCP	17.845	1.999	.000	104.443
TMF MONITOR	3.110	.000	1.322	18.200
TMP	.773	.096	.000	4.527
X.25	6.535	.000	.999	38.251
	117.533	10.733	12.175	687.899

Loadid - ENS58

DETAILED CPU UTILIZATION BY TRANSACTION

PROCESS-TYPE	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	CPU 6	CPU 7
ATB-B	3.830	3.780	3.800	3.850	3.860	3.950	3.760	3.940
ATB-P	19.860	20.680	20.350	20.470	21.370	21.040	22.200	20.320
AUDIT-B	.000	.000	.000	.000	.000	3.310	.[illegible]	.000
AUDIT-P	.000	.000	.000	.000	8.060	.000	.000	.000
FOX	.360	.000	.000	.000	.000	.000	.000	.000
HISTORY-B	.000	16.510	.000	.000	.000	.000	.000	.000
HISTORY-P	30.780	.000	.000	.000	.000	.000	.000	.000
INTERRUPTS	20.020	19.370	18.360	20.800	23.070	23.370	22.120	23.070
MISC	1.040	.340	.020	.010	.010	.010	.010	.010
PATHMON	.370	.010	.000	.000	.000	.000	.000	.000
SERVER	10.300	14.950	12.130	11.650	11.240	9.920	12.740	11.230
TCP	.000	10.180	.000	15.910	15.750	26.260	15.670	20.630
TMF MONITOR	1.520	2.500	.870	2.420	2.450	3.200	2.410	2.800
TMP	.000	.000	4.520	.000	.000	.000	.000	.000
X.25	.000	.000	9.820	9.380	.000	.000	9.460	9.560
	88.080	88.320	69.870	84.490	85.810	91.060	88.370	91.560

LOADID - ENS58

Disc Utilisation Summary per Transaction

USAGE	REQ PER TRANS	READS PER TRANS	WRITES PER TRANS	512 HITS PER TRANS	1024 HITS PER TRANS	2048 HITS PER TRANS	4096 HITS PER TRANS
ATB	7.881	1.124	2.337	2.318	.000	2.248	5.534
AUDIT TRAILS	.512	.006	.206	.000	.000	.000	.002
HISTORY	1.000	.001	.027	.000	.000	.000	1.000
UNUSED	.000	.008	.007	.000	.000	.000	.000
XRAY	.026	.005	.074	.000	.000	.000	.000

Loadid - ENS58

Disc Utilisation Summary

USAGE	SUM REQUESTS	SUM READS	SUM WRITES	512 HITS	1024 HITS	2048 HITS	4096 HITS
ATB	192355	27440	57041	56569	0	54871	135081
AUDIT TRAILS	12504	148	5030	0	0	0	53
HISTORY	24397	14	657	0	0	0	24404
UNUSED	0	190	181	0	0	0	0
XRAY	633	111	1802	0	0	0	0

Loadid - ENS58

Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XA1	3112	344853	654472	22.261	11.730
$XA2	3077	341042	647303	22.017	11.600
$XA3	3029	335452	637259	21.676	11.410
$XA4	3015	333981	634425	21.579	11.360
$XA5	2989	331075	629047	21.396	11.261
$XA6	3059	339245	643501	21.888	11.539
$XA7	3049	337581	641543	21.821	11.482
$XA8	3062	339808	644370	21.917	11.558
	24392	2703037	5131920		

Loadid - ENS58

Communications line utilisation summary

DEVICE-NAME	SUM REQUESTS	input bytes	output bytes	write utilisation	input utilisation
$XA1	3097	343322	651231	22.151	11.678
$XA2	3070	340799	646164	21.979	11.592
$XA3	3050	338964	641426	21.817	11.530
$XA4	3080	341715	647860	22.036	11.623
$XA5	3007	333473	632803	21.523	11.342
$XA6	3051	338991	642012	21.837	11.530
$XA7	3082	341846	648226	22.047	11.627
$XA8	3114	345497	654818	22.271	11.751
	24551	2724607	5164540		

Loadid - ENS59

Communications line summary per transactions

LOADID	Transactions	Output per trans	input per trans
ENS58	24408	199.869	99.848

TAL - T9250C00 - (15JUL87) Source language: TAL - Target machine: Tandem NonStop II System
Date - Time : 18FEB87 - 19:52:32 Default options: On (LIST,CODE,MAP,WARN,LMAP) - Off (ICODE,INNERLIST)

```
1.    000000 0  0   ?SYMBOLS, nocode, noprintsym
2.    000000 0  0   ?NOLIST, source $system.system.extdecs
```

```
 7.  000000 0  0   Proc ET1^demo main;
 8.  000000 1  0   begin
 9.  000000 1  1     int    .buf  [-1:100],
10.  000000 1  1           .driver^fname [0:11],
11.  000000 1  1           .driver^tname [0:11],
12.  000000 1  1           .driver^buf   [0:100],
13.  000000 1  1           rnum, term^num, driver^fnum, 1, error;
14.  000000 1  1
15.  000000 1  1     fixed my^time;
16.  000000 1  1
17.  000000 1  1     string .s^buf := @buf '<<' 1;
18.  000000 1  1
19.  000000 1  1     struct .Tx^data;
20.  000000 1  1     begin
21.  000000 1  2        string f1;
22.  000000 1  2        string account [0:11];
23.  000000 1  2        string f2;
24.  000000 1  2        string teller  [0:11];
25.  000000 1  2        string f3;
26.  000000 1  2        string branch  [0:11];
27.  000000 1  2        string f4;
28.  000000 1  2        string amount  [0:11];
29.  000000 1  2     end;
30.  000000 1  1
31.  000000 1  1     struct .Tx^resp;
32.  000000 1  1     begin
33.  000000 1  2        int length;
34.  000000 1  2        int term^name [0:11];
35.  000000 1  2        fixed start^time;
36.  000000 1  2        int (32) xtime [0:1];
37.  000000 1  2        fixed resp^time = xtime;
38.  000000 1  2     end;
39.  000000 1  1
40.  000000 1  1     subproc new^time (time^field);
41.  000000 2  1   ! ----------------------------
42.  000000 2  1     string .time^field;
43.  000000 2  1     begin
44.  000000 2  2        int time^array [0:7], m;
45.  000000 2  2        string .month = 'P' := "JanFebMarAprMayJunJulAugSepOctNovDec";
46.  000022 2  2        entry start^time;
47.  000022 2  2
48.  000022 2  2        call time (time^array);
49.  000031 2  2        time^array [6] := time^array [6] * 10;  ! make it milli seconds
50.  000035 2  2        goto all;
51.  000036 2  2
52.  000036 2  2        start^time:
53.  000040 2  2        -----------
54.  000040 2  2        call interprettimestamp ( converttimestamp (tx^resp.start^time, 1),
55.  000040 2  2                                  time^array);
56.  000056 2  2
57.  000056 2  2        all:
58.  000056 2  2        ----
59.  000056 2  2  !                        0123456789+123456789+123
60.  000056 2  2        time^field ':=' ["DD MMM 1982, HH:MM:SS.hhh", 0];
61.  000066 2  2        call numout (time^field [ 0], time^array [2], 10, 2); !Day
62.  000074 2  2        call numout (time^field [ 7], time^array [0], 10, 4); !Year
63.  000103 2  2        call numout (time^field [13], time^array [3], 10, 2); !Hour
```

```
    64.  000112 2  2        call numout (time^field [16], time^array [4], 10, 2); !Min
    65.  000121 2  2        call numout (time^field [19], time^array [5], 10, 2); !Sec
    66.  000130 2  2        call numout (time^field [22], time^array [6], 10, 3); !milli
    67.  000137 2  2        m := time^array [1] - 1;
    68.  000142 2  2        time^field [3] ':=' month [m*3] for 3;                !month
    69.  000155 2  2     end; !subproc new^time;

ALL                             Label                           %000056
M                               Variable        INT             Direct          S-000
MONTH                           Variable        STRING          Indirect        P+000
START^TIME                      Entry                           %000037
TIME^ARRAY                      Variable        INT             Direct          S-010
TIME^FIELD                      Variable        STRING          Indirect        S-012

    70.  000174 1  1
    71.  000174 1  1     subproc response^time (time, field);
    72.  000174 2  1   ! ----------------------------------
    73.  000174 2  1     fixed time; string .field;
    74.  000174 2  1     begin
    75.  000174 2  2        int hour, min , sec, mil, mic;
    76.  000174 2  2
    77.  000174 2  2        call convertprocesstime (time, hour, min, sec, mil, mic);
    78.  000211 2  2
    79.  000211 2  2   !                  0123456789+123456789+123
    80.  000211 2  2        field ':=' ["MM:SS.mmm", 0];
    81.  000221 2  2        call numout (field [ 0], min, 10, 2);
    82.  000227 2  2        call numout (field [ 3], sec, 10, 2);
    83.  000236 2  2        call numout (field [ 6], mil, 10, 3);
    84.  000245 2  2     end;

FIELD                           Variable        STRING          Indirect        S-006
HOUR                            Variable        INT             Direct          S-004
MIC                             Variable        INT             Direct          S-000
MIL                             Variable        INT             Direct          S-001
MIN                             Variable        INT             Direct          S-003
SEC                             Variable        INT             Direct          S-002
TIME                            Variable        FIXED (0)       Direct          S-012

    85.  000254 1  1
    86.  000254 1  1     int subproc length (b);
    87.  000254 2  1   ! ----------------------
    88.  000254 2  1     string .b;
    89.  000254 2  1     begin
    90.  000254 2  2        string .l;
    91.  000254 2  2        scan b until 0 -> @l;
    92.  000261 2  2        return @l - @b;
    93.  000266 2  2     end; !subproc length

B                               Variable        STRING          Indirect        S-002
L                               Variable        STRING          Indirect        S-000

    94.  000266 1  1
    95.  000266 1  1     subproc update^time;
    96.  000266 2  1   ! -------------------
    97.  000266 2  1     begin
    98.  000266 2  2        call new^time (s^buf);
    99.  000271 2  2        call term^io (term^num, buf, 25, 2, 52, write^);
```

```
100.  000303 2  2    end;
101.  000305 1  1
102.  000305 1  1    subproc write^menu;
103.  000305 2  1  ! -------------------
104.  000305 2  1    begin
105.  000305 2  2       int r, c, length, all^done, i;
106.  000305 2  2       string .buf^p, .old^p;
107.  000305 2  2       string .menu = 'P' :=
108.  000305 2  2       [01, 02, "=",                                                    0,
109.  000307 2  2        02, 04, "First NonStop Bank   Debit/Credit Application",        0,
110.  000337 2  2        03, 02, "=",                                                    0,
111.  000341 2  2        05, 04, "Line Driver : ", !              Change line driver :",!0,
112.  000352 2  2        06, 04, "Branch      : ", !              Change branch      :",!0,
113.  000362 2  2        07, 04, "Teller      : ", !              Change teller      :",!0,
114.  000373 2  2        09, 02, "=",                                                    0,
115.  000375 2  2        11, 04, "Account     :                   Amount :",          0,
116.  000423 2  2        11, 66, "Ok: ",                                             0,
117.  000427 2  2        13, 02, "=",                                                    0,
118.  000431 2  2        16, 04, "Terminal used                :",                      0,
119.  000452 2  2        17, 04, "Response length              :     Bytes",            0,
120.  000501 2  2        18, 04, "Transaction started          :",                      0,
121.  000522 2  2        19, 04, "Transaction response time    :",                      0,
122.  000544 2  2        20, 04, "Total response time          :",                      0,
123.  000565 2  2        22, 70, "Next : ",                                          0,
124.  000572 2  2        23, 02, "=",                                                    0,
125.  000574 2  2        %377];
126.  000575 2  2
127.  000575 2  2       @buf^p := @menu '-' 1;
128.  000604 2  2       do
129.  000604 2  2       begin
130.  000604 2  3          @old^p := @buf^p '+' 1;
131.  000607 2  3          STACK @old^p, 0; CODE (SBU %660); ! scan text until 0 -> @buf^p;
132.  000612 2  3          STORE @buf^p;
133.  000613 2  3          length := @buf^p '-' @old^p;      ! length includ. row & col;
134.  000617 2  3
135.  000617 2  3          STACK @s^buf, @old^p, length + 2; ! s^buf ':=' text for length;
136.  000623 2  3          CODE (MOVB %67);
137.  000624 2  3          all^done := s^buf [length + 1] = %377;
138.  000635 2  3
139.  000635 2  3          r := s^buf [0];
140.  000637 2  3          c := s^buf [1];
141.  000642 2  3          if s^buf [2] = "=" then
142.  000646 2  3          begin
143.  000646 2  4             length := 78;
144.  000650 2  4             s^buf [3] ':=' s^buf [2] for 77;
145.  000655 2  4          -- for i := (c+10-2) to 70 by 10 do s^buf [i] := "+";
146.  000655 2  4          end;
147.  000655 2  3          call term^io (term^num, buf [1] , length - 2, r, c, write^);
148.  000671 2  3       end
149.  000671 2  2       until all^done;
150.  000673 2  2       call update^time;
151.  000674 2  2    end; !subproc write^menu
```

ALL^DONE	Variable	INT	Direct	S-003
BUF^P	Variable	STRING	Indirect	S-001
C	Variable	INT	Direct	S-005
I	Variable	INT	Direct	S-002

LENGTH	Variable	INT	Direct	S-004
MENU	Variable	STRING	Indirect	P+000
OLD^P	Variable	STRING	Indirect	S-000
R	Variable	INT	Direct	S-006

```
  153.  000700 1  1    subproc clear;
  154.  000700 2  1  ! -------------
  155.  000700 2  1    begin
  156.  000700 2  2       int i := 0;
  157.  000700 2  2       int .clear^rows = 'P' := [  5,  6,  7, 20, 22, 0];
  158.  000706 2  2       int .clear^cols = 'P' := [ 59, 59, 59, 48, 77, 0];
  159.  000714 2  2
  160.  000714 2  2       s^buf ':=' " " & s^buf for 34;
  161.  000731 2  2       while clear^rows [i] <> 0 do
  162.  000735 2  2       begin
  163.  000735 2  3          call term^io (term^num, buf, 2, clear^rows [i], clear^cols [i], write^);
  164.  000752 2  3          i := i+1;
  165.  000754 2  3       end;
  166.  000755 2  2    end; !subproc clear;

CLEAR^COLS                    Variable       INT            Indirect       P+000
CLEAR^ROWS                    Variable       INT            Indirect       P+000
I                             Variable       INT            Direct         S-000

  167.  000760 1  1
  168.  000760 1  1    subproc file^error (error) variable;
  169.  000760 2  1  ! ----------------------------------
  170.  000760 2  1    int error;
  171.  000760 2  1    begin
  172.  000760 2  2        if not $param (error) then
  173.  000762 2  2           call fileinfo (driver^fnum, error);
  174.  001011 2  2        s^buf ':=' ["Line driver I/O error ####", 0];
  175.  001021 2  2        call numout (s^buf [22], error, 10, 3);
  176.  001030 2  2        call term^io (term^num, buf, length (buf), 20, 50, write^);
  177.  001045 2  2        call term^io (term^num, buf, 1, 20, 48, read^);
  178.  001057 2  2    end;

ERROR                         Variable       INT            Direct         S-002

  179.  001077 1  1
  180.  001077 1  1    subproc read^numeric (buffer, l, row, col);
  181.  001077 2  1  ! ----------------------------------------
  182.  001077 2  1    string .buffer; int l, row, col;
  183.  001077 2  1    begin
  184.  001077 2  2       string .fc, .lc;
  185.  001077 2  2       int ok, in^len, i;
  186.  001077 2  2
  187.  001077 2  2       do begin
  188.  001100 2  3          ok := -1;
  189.  001102 2  3          s^buf [l] := 0;
  190.  001105 2  3          call term^io (term^num, buf, l, row, col, read^);
  191.  001117 2  3          scan s^buf while " " -> @fc;
  192.  001123 2  3          rscan s^buf [l-1] while " " -> @lc;
  193.  001131 2  3          in^len := @lc '-' @fc + 1;
  194.  001136 2  3          if in^len > 0 then
  195.  001141 2  3             for i := 0 to in^len -1 do
  196.  001143 2  3                if not $numeric (fc [i]) then ok := 0;
  197.  001160 2  3       end until ok;
  198.  001162 2  2       if in^len > 0 then
  199.  001165 2  2       begin
  200.  001165 2  3          buffer ':=' "0" & buffer for l-1;
  201.  001201 2  3          buffer [l-1] '=:' lc for in^len;
```

```
   202.    001207 2  3        end;
   203.    001207 2  2        s^buf ':=' buffer for l;
   204.    001213 2  2        call term^io (term^num, buf, l, row, col, write^);
   205.    001225 2  2        call update^time;
   206.    001226 2  2     end; ! subproc read^numeric
```

BUFFER	Variable	STRING	Indirect	S-011
COL	Variable	INT	Direct	S-006
FC	Variable	STRING	Indirect	S-004
I	Variable	INT	Direct	S-000
IN^LEN	Variable	INT	Direct	S-001
L	Variable	INT	Direct	S-010
LC	Variable	STRING	Indirect	S-003
OK	Variable	INT	Direct	S-002
ROW	Variable	INT	Direct	S-007

```
   207.    001232 1  1
```

```
209.  001232 1  1    subproc do^it;
210.  001232 2  1  ! --------------
211.  001232 2  1    begin
212.  001232 2  2       int i := -1, j, ok := -1, in^len, error, new^entry := 0;
213.  001232 2  2
214.  001232 2  2       begin
215.  001242 2  3  !------------------------------------------ We dont start in the beginning ---
216.  001242 2  3          goto L^OP;
217.  001243 2  3
218.  001243 2  3  !------------------------------------------------------- Read driver name ----
219.  001243 2  3  L^DR:   call term^io (term^num, driver^tname, 12, 5, 18, read^);
220.  001255 2  3          if driver^tname = [6*["  "]] then
221.  001265 2  3          begin
222.  001265 2  4             call term^io (term^num, driver^fname, 12, 5, 18, write^);
223.  001277 2  4             goto L^RB;
224.  001300 2  4          end;
225.  001300 2  3          driver^fname ':=' driver^tname for 6;
226.  001304 2  3          if driver^fnum > 1 then call close (driver^fnum);
227.  001314 2  3
228.  001314 2  3  !------------------------------------------------------- Open line driver ----
229.  001314 2  3  L^OP:   call term^io (term^num, driver^fname, 12, 5, 18, write^);
230.  001326 2  3          do begin
231.  001326 2  4             s^buf ':=' " " & s^buf for 78;
232.  001341 2  4             call term^io (term^num, buf, 55, 20, 50, write^);
233.  001353 2  4             call open (driver^fname, driver^fnum);
234.  001364 2  4             call fileinfo (driver^fnum, error);
235.  001413 2  4             if error then
236.  001415 2  4             begin
237.  001415 2  5                call file^error( error );
238.  001421 2  5                call term^io (term^num, driver^fname, 12, 5, 18, read^);
239.  001433 2  5             end;
240.  001433 2  4          end until not error;
241.  001435 2  3
242.  001435 2  3  !------------------------------------------- Read Branch/Teller -----------!
243.  001435 2  3  L^RB:   tx^data.branch  ':=' "000000010010";
244.  001447 2  3          tx^data.teller  ':=' "000000010011";
245.  001461 2  3          tx^data.account ':=' "000010000011";
246.  001473 2  3          tx^data.amount  ':=' "000000000628";
247.  001505 2  3
248.  001505 2  3          call read^numeric (tx^data.branch,  12, 6,  18);
249.  001515 2  3          call read^numeric (tx^data.teller,  12, 7,  18);
250.  001525 2  3
251.  001525 2  3  !------------------------------------------ Read Account/Amount ----------!
252.  001525 2  3  L^AA:   call read^numeric (tx^data.account, 12, 11, 18);
253.  001535 2  3          call read^numeric (tx^data.amount,  12, 11, 46);
254.  001545 2  3
255.  001545 2  3  !------------------------------------------- Check if ok to go ahead -----!
256.  001545 2  3          call term^io (term^num, buf, 1, 11, 70, read^);
257.  001557 2  3          if $alpha (s^buf) then s^buf := s^buf iand %337; ! upshift
258.  001565 2  3          if s^buf =  "?" then return;          ! exit program
259.  001572 2  3          if s^buf =  "E" then return;          ! exit program
260.  001600 2  3          if s^buf =  "R" then goto L^DR;       ! reconfigure !;
261.  001604 2  3          if s^buf <> " " and                   !
262.  001604 2  3             s^buf <> "Y" then goto L^AA;       ! read Account & amount again
263.  001613 2  3
264.  001613 2  3  !------------------------------------------- Clear all input fields ------!
265.  001613 2  3          call clear;
```

```
   266.    001614 2  3
   267.    001614 2  3   !---------------------------------------------- Send it off to line driver --!
   268.    001614 2  3           driver^buf ':=' tx^data for $len (tx^data) bytes;
   269.    001620 2  3
   270.    001620 2  3           my^time := Juliantimestamp;
   271.    001630 2  3           call writeread (driver^fnum,
   272.    001630 2  3                           driver^buf,
   273.    001630 2  3                           $len( tx^data ),
   274.    001630 2  3                           $len( tx^resp ));
   275.    001641 2  3           my^time := juliantimestamp - my^time;
   276.    001654 2  3
   277.    001654 2  3           call fileinfo (driver^fnum, error);
   278.    001703 2  3           if error then
   279.    001705 2  3           begin
   280.    001705 2  4              call file^error( error );
   281.    001711 2  4              if error = 1 then return; ! exit, line driver gone
   282.    001720 2  4              goto L^DR;
   283.    001721 2  4           end;
   284.    001721 2  3
   285.    001721 2  3   !---------------------------------------------- Print response stats --------!
   286.    001721 2  3           tx^resp ':=' driver^buf for $len (tx^resp) bytes;
   287.    001725 2  3           call term^io (term^num, tx^resp.term^name, 16, 16, 37, write^);
   288.    001740 2  3           call numout (s^buf, tx^resp.length, 10, 3);
   289.    001746 2  3           call term^io (term^num,                buf, 3, 17, 37, write^);
   290.    001760 2  3           call start^time (s^buf);
   291.    001763 2  3           call term^io (term^num, buf, length (s^buf),   18, 37, write^);
   292.    001777 2  3           tx^resp.xtime [0] := 0d;
   293.    002006 2  3           tx^resp.resp^time := tx^resp.resp^time * 1000f;
   294.    002022 2  3           call response^time (tx^resp.resp^time, s^buf);
   295.    002027 2  3           call term^io (term^num, buf, length (s^buf),   19, 37, write^);
   296.    002043 2  3           call response^time (my^time, s^buf);
   297.    002050 2  3           call term^io (term^num, buf, length (s^buf),   20, 37, write^);
   298.    002064 2  3
   299.    002064 2  3   !---------------------------------------------- Ready for next one ? --------!
   300.    002064 2  3           call update^time;
   301.    002065 2  3           call term^io (term^num, buf, 1, 22, 77, read^);
   302.    002077 2  3           if $alpha (s^buf) then s^buf := s^buf land %337; ! upshift
   303.    002105 2  3           if s^buf =  "?" then return;        ! exit program
   304.    002113 2  3           if s^buf =  "E" then return;        ! exit program
   305.    002120 2  3           if s^buf =  "R" then goto L^DR;     ! reconfigure !;
   306.    002124 2  3
   307.    002124 2  3           goto L^AA;  ! go and read next transaction input;
   308.    002125 2  3
   309.    002125 2  3   !---------------------------------------------- Finnished -----------------!
   310.    002125 2  3        end;
   311.    002125 2  2
   312.    002125 2  2     end; !subproc do^it

ERROR                          Variable      INT           Direct        S-001
I                              Variable      INT           Direct        S-005
IN^LEN                         Variable      INT           Direct        S-002
J                              Variable      INT           Direct        S-004
L^AA                           Label                       %001525
L^DR                           Label                       %001243
L^OP                           Label                       %001314
L^RB                           Label                       %001435
NEW^ENTRY                      Variable      INT           Direct        S-000
```

OK Variable INT Direct S-003

313. 002171 1 1

```
  315.    002171 1  1    begin ! of main's code
  316.    002220 1  2       buf [-1] := 0;                          ! place a stopper for reverse scans
  317.    002223 1  2       buf  ':=' ["$RECEIVE", 8*["  "]];
  318.    002232 1  2       call open (buf, rnum);
  319.    002243 1  2       call read (rnum, buf, 80);
  320.    002253 1  2       driver^fname ':=' buf [21] for 12;
  321.    002260 1  2       call term^io (term^num, buf [9],,,, open^);
  322.    002272 1  2
  323.    002272 1  2       call write^menu;
  324.    002273 1  2       tx^data.f1 :=
  325.    002273 1  2       tx^data.f2 :=
  326.    002273 1  2       tx^data.f3 :=
  327.    002273 1  2       tx^data.f4 := "+";
  328.    002316 1  2
  329.    002316 1  2       call do^it;
  330.    002317 1  2
  331.    002317 1  2       call term^io (term^num,,,,, close^);
  332.    002327 1  2       call stop;
  333.    002334 1  2    end; !of main's code
  334.    002334 1  1 end;
```

BUF	Variable	INT	Indirect	L+001
CLEAR	Subproc		%000714	
DO^IT	Subproc		%001232	
DRIVER^BUF	Variable	INT	Indirect	L+004
DRIVER^FNAME	Variable	INT	Indirect	L+002
DRIVER^FNUM	Variable	INT	Direct	L+007
DRIVER^TNAME	Variable	INT	Indirect	L+003
ERROR	Variable	INT	Direct	L+011
FILE^ERROR	Subproc		%000760	
L	Variable	INT	Direct	L+010
LENGTH	Subproc	INT	%000254	
MY^TIME	Variable	FIXED (0)	Direct	L+012
NEW^TIME	Subproc		%000025	
READ^NUMERIC	Subproc		%001077	
RESPONSE^TIME	Subproc		%000174	
RNUM	Variable	INT	Direct	L+005
START^TIME	Subproc		%000037	
S^BUF	Variable	STRING	Indirect	L+016
TERM^NUM	Variable	INT	Direct	L+006
TX^DATA	Variable,64	STRUCT	Indirect	L+017
1 F1	0,1	STRING	Direct	
1 ACCOUNT[0:11]	1,1	STRING	Direct	
1 F2	15,1	STRING	Direct	
1 TELLER[0:11]	16,1	STRING	Direct	
1 F3	32,1	STRING	Direct	
1 BRANCH[0:11]	33,1	STRING	Direct	
1 F4	47,1	STRING	Direct	
1 AMOUNT[0:11]	50,1	STRING	Direct	
TX^RESP	Variable,52	STRUCT	Indirect	L+020
1 LENGTH	0,2	INT	Direct	
1 TERM^NAME[0:11]	2,2	INT	Direct	
1 START^TIME	32,10	FIXED (0)	Direct	
1 XTIME[0:1]	42,4	INT (32)	Direct	
1 RESP^TIME	42,10	FIXED (0)	Direct	
UPDATE^TIME	Subproc		%000266	
WRITE^MENU	Subproc		%000575	

CLEAR^	Literal	INT	%000002
CLEAR^TO^SPACES	Literal	INT	%000111
CLOSE^	Literal	INT	%000001
COL	Define		$1 + %37
DIM	Literal	INT	%000041
DISABLE^LENGTH	Literal	INT	%000012
ENABLE^LENGTH	Literal	INT	%000105
ESC	Literal	INT	%000033
ET1^DEMO	Proc		%002171
NORMAL	Literal	INT	%000040
OPEN^	Literal	INT	%000000
PROTECT	Literal	INT	%000140
PROTECT^SUBMODE	Literal	INT	%000127
READ^	Literal	INT	%000003
READ^BUFFER	Literal	INT	%000074
READ^STATUS	Literal	INT	%000136
REINITIALIZE	Literal	INT	%000161
REVERSE	Literal	INT	%000044
ROW	Define		$1 + %37
SET^BUFFER	Literal	INT	%000021
SET^MAX^PAGE^NO	Literal	INT	%000160
SET^PAGE^SIZE	Literal	INT	%000164
SM^PAGEMODE	Literal	INT	%000010
START^FIELD	Literal	INT	%000035
TERM^IO	Proc	INT	%000504
UNDERSCORE	Literal	INT	%000060
UNLOCK	Literal	INT	%000142
UNPROTECT	Literal	INT	%000100
WRITE^	Literal	INT	%000004

ENTRY POINT MAP BY NAME

SP	PEP	BASE	LIMIT	ENTRY	ATTRS	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	003	001423	004000	003614	M	ET1^DEMO	18FEB87	19:52	TAL	$BASE.GRAYTGU1.ET1S
00	002	000004	001422	000510	V	TERM^IO	18FEB87	19:52	TAL	$BASE.GRAYTGU1.TALLIB

```
BINDER - OBJECT FILE BINDER - T9621C00 - (15JUL87)   SYSTEM \FOXII
Object file name is $BASE.GRAYTGU1.OBJECT
Object file timestamp is 18FEB87 19:52:32
Number of Binder errors = 0
Number of Binder warnings = 0
Code area size = 3 pages
Resident code size = 0 pages
Data area size = 1 pages
Extended data area size = 0 pages
Top of stack = 0 words
Number of code segments  = 1 segment


The object file will run on a TNS/II, but may not run on a TNS
Number of compiler errors =  0
Number of compiler warnings =  0
Maximum symbol table space used was = 87304 bytes
Number of source lines = 2931
Compile cpu time = 00:00:11
Total Elapsed time = 00:00:40
```

TAL - T9250C00 - (15JUL87) Source language: TAL - Target machine: Tandem NonStop II System
Date - Time : 16FEB87 - 17:56:41 Default options: On (LIST,CODE,MAP,WARN,LMAP) - Off (ICODE,INNERLIST)

```
1.    000000 0  0   ?ERRORS 10, datapages 64, nocode, nowarn, symbols, inspect, columns 74
2.    000000 0  0   ?LMAP(loc,alpha), page "Program TXGENS, modified SLOADGEN with Think Time"
3.    000000 0  0
4.    000000 0  0   ! 3.08 02/16/87 Gerhard Huff. Added support for interactive traffic
5.    000000 0  0   ! 3.07 02/12/87 Nhan Chu. Increase number of seeds to 40 max.
6.    000000 0  0   ! 3.06 02/11/87 Nhan Chu. New param RNSEED. Random number seeds 1-10.
7.    000000 0  0   ! 3.05 02/04/87 Gerhard Huff. Changed THREAD^THINK to force a TASK DISPATCH
8.    000000 0  0   !                              if THINK^TIME = 0, this fixes a problem when
9.    000000 0  0   !                              running RATE MAXIMUM.
10.   000000 0  0   ! 3.05 02/02/87 Gerhard Huff. Changed handling of threads that have to wait
11.   000000 0  0   !                              because of MAX^TX^OUT exceeded.
12.   000000 0  0   ! 3.05 01/29/87 Gerhard Huff. Numerous changes to prevent the load generator
13.   000000 0  0   !                              to 'overrun' the test system and to avoid
14.   000000 0  0   !                              LCB shortage in the RTE.
15.   000000 0  0   !                              added MAXTERM parameter to limit the number
16.   000000 0  0   !                              of threads that are read from CONFIG and started;
17.   000000 0  0   !                              added MAXTX paramter to limit the number
18.   000000 0  0   !                              of transactions send to SIT concurrently;
19.   000000 0  0   !                              this should prevent resource problems
20.   000000 0  0   !                              on both systems;
21.   000000 0  0   !                              fixed the handling of toggle <3> to work as before
22.   000000 0  0   ! 3.04 01/28/87 Gerhard Huff. Inserted logic to ABEND whenever an error
23.   000000 0  0   !                              30 (LCB allocation failure) is returned
24.   000000 0  0   !                              This should prevent that RTE system to get stuck
25.   000000 0  0   ! 3.04 01/28/87 Gerhard Huff. Changed WRITE^LOGFILE to 'ignore' allocation
26.   000000 0  0   !                              failures in GENERAL POOL; it'l now drop a message
27.   000000 0  0   !                              but retry the next time.
28.   000000 0  0   ! 3.04 01/28/87 Gerhard Huff. Changed ALLOCATE/DEALLOCATE^BUFFER to avoid
29.   000000 0  0   !                              recursive calls leading to a stack overflow
30.   000000 0  0   !                              and program abend.
31.   000000 0  0   ! 3.04 01/28/87 Gerhard Huff. Changed read from SCRIPT file from 40 bytes
32.   000000 0  0   !                              to 'buffer^size' bytes to get all the data
33.   000000 0  0   !                              from script (52 bytes at the moment).
34.   000000 0  0   ! 3.03 01/15/87 Nhan Chu. Use 'version' define.
35.   000000 0  0   ! 3.02 01/14/87 Nhan Chu. Flush the last records of TXSTAT file to disc.
36.   000000 0  0   ! 3.01 01/13/87 Gerhard Huff. Changed 'thread think'ing' from SIGNALTIMEOUT
37.   000000 0  0   !                              to AWAITIO with delay time = timeout value
38.   000000 0  0   ! 3.00 01/12/87 Gerhard Huff. Changed TXLOG file to unstrucrured
39.   000000 0  0   ! 2.04 12/10/86 Nhan Chu. Add canceltimeout.
40.   000000 0  0   ! 2.03 12/02/86 Nhan Chu. Add eh after signaltimeout, DELAY param. 3###.
41.   000000 0  0   ! 2.02 10/14/86 Nhan Chu. Add TX100 to control the number of replayed tx's.
42.   000000 0  0   !      Caculate the ext sizes for txlog based on tx100.
43.   000000 0  0   ! 2.01 10/06/86 Nhan Chu. 200 bytes reply and move threads tcb's into U31K.
44.   000000 0  0   ! 2.00 09/13/86 Nhan Chu. Enhancement for fixed arrival rate with Poisson
45.   000000 0  0   !      (random) arrivals or constant arrivals.
46.   000000 0  0   ! 1.00 ???????? Gerhard Huff. Original SLOADGEN.
47.   000000 0  0
```

```
000000 0  0  !-------------------------!
000000 0  0  !    TXGEN Version        !
000000 0  0  !-------------------------!
000000 0  0  define txgen^vs = ["vs 3.08"]#; ! shown in HELP                    3.04
000000 0  0
000000 0  0    define  intaddr (item)       = (@item '>>' 1)#,
000000 0  0            needs^work           = flags.<0>#,
000000 0  0            io^complete          = flags.<1>#,
000000 0  0            io^pending           = flags.<2>#,
000000 0  0            interactive          = flags.<3>#,                      3.08
000000 0  0
000000 0  0            sysmsg^time          = -22#,
000000 0  0            sysmsg^break         = -20#,
000000 0  0            sysmsg^startup       = -1#,
000000 0  0
000000 0  0            setup                = flags.<0>#,
000000 0  0
000000 0  0            guardian^devtype     = <4:9>#,
000000 0  0            guardian^devsubtype  = <10:15>#;
000000 0  0
000000 0  0  ! Guardian device types
000000 0  0
000000 0  0    literal guardian^typeprocess = 0,
000000 0  0            guardian^typereceive = 2,
000000 0  0            guardian^typedisc    = 3,
000000 0  0            guardian^typetape    = 4,
000000 0  0            guardian^typeprinter = 5,
000000 0  0            guardian^typeterm    = 6,
000000 0  0            guardian^typecard    = 8,
000000 0  0            guardian^typex25ptp  = 9,
000000 0  0            guardian^tpyettmp    = 21;
000000 0  0
000000 0  0  ! Open flags definitions
000000 0  0
000000 0  0    literal open^nowait          = %B0000000000000001,
000000 0  0            open^shared          = %B0000000000000000,
000000 0  0            open^exclusive       = %B0000000000010000,              2.00
000000 0  0            open^protected       = %B0000000000110000,              2.00
000000 0  0            open^readwrite       = %B0000000000000000,
000000 0  0            open^readonly        = %B0000010000000000,
000000 0  0            open^writeonly       = %B0000100000000000,
000000 0  0            open^getmsg          = %B0100000000000000;
000000 0  0
000000 0  0
000000 0  0    literal true  = -1,
000000 0  0            false = 0,
000000 0  0            nil   = 0, !should be -1 ?
000000 0  0
000000 0  0            number^busy           = 1,
000000 0  0            net^out^of^order      = 9,
000000 0  0            number^not^obtainable = 13,
000000 0  0
000000 0  0            name^  = 0,
000000 0  0            const^ = 1,
000000 0  0            none^  = 2,
000000 0  0            str^   = 3,                                           2.00
000000 0  0
```

```
106.  000000 0  0            param^default   = 0,
107.  000000 0  0            param^config    = 1,
108.  000000 0  0            param^script    = 2,
109.  000000 0  0            param^log       = 3,
110.  000000 0  0            param^arrival   = 4,                                          2.00
111.  000000 0  0            param^txrate    = 5,                                          2.00
112.  000000 0  0            param^txlog     = 6,                                          2.00
113.  000000 0  0            param^tx100     = 7,                                          2.02
114.  000000 0  0            param^delay     = 8,                                          2.03
115.  000000 0  0            param^maxterm   = 9,                                          3.05
116.  000000 0  0            param^maxtx     = 10,                                         3.05
117.  000000 0  0            param^rnseed    = 11,                                         3.06
118.  000000 0  0            max^keyword     = 11;                                         3.06
119.  000000 0  0
120.  000000 0  0    string param^keywords = 'P' :=
121.  000000 0  0           [6, name^   , "CONFIG   ",
122.  000000 0  0            6, name^   , "SCRIPT   ",
123.  000000 0  0            3, name^   , "LOG      ",
124.  000000 0  0            7, str^    , "ARRIVAL  ",     ! {Constant,Random,Maximum}        2.00
125.  000000 0  0            6, const^  , "TXRATE   ",     ! tx rate in tps * 10              2.00
126.  000000 0  0            5, name^   , "TXLOG    ",     ! tx statistics file              2.00
127.  000000 0  0            5, const^  , "TX100    ",     ! max 100's of tx to be replayed  2.02
128.  000000 0  0            5, const^  , "DELAY    ",     ! startup delay between term in   2.03
129.  000000 0  0                                          ! .01 seconds.                    2.03
130.  000000 0  0            7, const^  , "MAXTERM  ",     ! max number of threads (terminals)3.05
131.  000000 0  0                                          ! to start                        3.05
132.  000000 0  0            5, const^  , "MAXTX    ",     ! limit for the number of TX's    3.05
133.  000000 0  0                                          ! outstanding                     3.05
134.  000000 0  0            6, const^  , "RNSEED   ",     ! Random number seed = 1 to 40    3.06
135.  000000 0  0            0];
136.  000000 0  0
137.  000000 0  0    struct startup^msg^^ (*);
138.  000000 0  0    begin
139.  000000 0  1       int id,
140.  000000 0  1           default [0:7],
141.  000000 0  1           in [0:11],
142.  000000 0  1           out [0:11];
143.  000000 0  1       string
144.  000000 0  1           parm [0:527];
145.  000000 0  1    end;
146.  000000 0  0
147.  000000 0  0    struct startup^param^^ (*);
148.  000000 0  0    begin
149.  000000 0  1       int name [0:11],
150.  000000 0  1           value = name,
151.  000000 0  1           flags;
152.  000000 0  1       string s^name = name;
153.  000000 0  1    end;
154.  000000 0  0
155.  000000 0  0  !-------------------------!
156.  000000 0  0  ! LOGFILE related defines !
157.  000000 0  0  !-------------------------!
158.  000000 0  0
159.  000000 0  0    struct logfile^^ (*);
160.  000000 0  0    begin
161.  000000 0  1       struct msg^q; !queue
162.  000000 0  1       begin
```

```
163.    000000 0  2          int succ;
164.    000000 0  2          int pred;
165.    000000 0  2       end;
166.    000000 0  1       struct dev;       ! same as IO^dev^^
167.    000000 0  1       begin
168.    000000 0  2          int filename   [0:11];
169.    000000 0  2          int filenum;
170.    000000 0  2          int type;
171.    000000 0  2          int error;
172.    000000 0  2          int bufaddr;   ! not used here
173.    000000 0  2       end;
174.    000000 0  1       int active;
175.    000000 0  1       int io^busy;
176.    000000 0  1    end;
177.    000000 0  0
178.    000000 0  0    struct log^msg^^ (*);
179.    000000 0  0    begin
180.    000000 0  1       struct queue;
181.    000000 0  1       begin
182.    000000 0  2          int succ;
183.    000000 0  2          int pred;
184.    000000 0  2       end;
185.    000000 0  1       int length;
186.    000000 0  1       int data;
187.    000000 0  1    end;
188.    000000 0  0
189.    000000 0  0    !-------------------------!
190.    000000 0  0    !  GENERAL POOL defines   !
191.    000000 0  0    !-------------------------!
192.    000000 0  0
193.    000000 0  0    ! define general^pool^start = %100000d#,    starting at 16 kW          2.00
194.    000000 0  0    !        general^pool^size  = %100000d#;    length      16 kW          2.00
195.    000000 0  0
196.    000000 0  0      literal general^pool^size = 2*(08*1024); !08 pages=16384 bytes!      3.01
197.    000000 0  0
198.    000000 0  0      struct general^pool^^ (*);
199.    000000 0  0      begin
200.    000000 0  1         int header [0:19];
201.    000000 0  1         int status;
202.    000000 0  1      end;
203.    000000 0  0
204.    000000 0  0    !-------------------------!
205.    000000 0  0    !  QUEUE related defines  !
206.    000000 0  0    !-------------------------!
207.    000000 0  0
208.    000000 0  0      Define Queue^any (q)   = (q.succ <> @q)#,
209.    000000 0  0             Queue^empty (q) = (q.succ =  @q)#;
210.    000000 0  0
211.    000000 0  0      struct queue^^ (*);
212.    000000 0  0      begin
213.    000000 0  1         int succ;
214.    000000 0  1         int pred;
215.    000000 0  1      end;
216.    000000 0  0
217.    000000 0  0    !-------------------------!
218.    000000 0  0    !  THREAD related defines !
219.    000000 0  0    !-------------------------!
```

```
220.  000000 0  0
221.  000000 0  0     literal max^threads            = 250,
222.  000000 0  0             buffer^size            = 200,  ! bytes                    2.01
223.  000000 0  0
224.  000000 0  0             thread^state^closed    = 0,
225.  000000 0  0             thread^state^ready     = 1,
226.  000000 0  0             thread^state^active    = 2,
227.  000000 0  0             thread^state^stopping  = 3,
228.  000000 0  0             thread^state^think     = 4;                               2.00
229.  000000 0  0
230.  000000 0  0     struct config^record^^ (*); ! config file record lay out
231.  000000 0  0     begin
232.  000000 0  1        string terminal^name [0:15];
233.  000000 0  1        int terminal = terminal^name;
234.  000000 0  1     end;
235.  000000 0  0
236.  000000 0  0     struct tcb^^ (*); ! thread control block
237.  000000 0  0     begin
238.  000000 0  1        struct ready^q; !queue^^
239.  000000 0  1        begin
240.  000000 0  2           int succ;
241.  000000 0  2           int pred;
242.  000000 0  2        end;
243.  000000 0  1        struct terminal;
244.  000000 0  1        begin
245.  000000 0  2           string name^ [0:23];
246.  000000 0  2           int name = name^;
247.  000000 0  2           int fnum;
248.  000000 0  2        end;
249.  000000 0  1        int index;
250.  000000 0  1        int state;
251.  000000 0  1        int substate;
252.  000000 0  1        int flags;
253.  000000 0  1        int last^state;
254.  000000 0  1        int last^substate;
255.  000000 0  1        int last^loc;
256.  000000 0  1        int last^fnum;
257.  000000 0  1        int last^error;
258.  000000 0  1        int last^count;
259.  000000 0  1        fixed send^timestamp;        ! 4-word timestamp of input time
260.  000000 0  1        fixed recv^timestamp;        ! 4-word timestamp of output time
261.  000000 0  1        int(32) think^time;          ! interval until next send (unit=.01s)
262.  000000 0  1        fixed end^time;              ! absolute time in tics when think ends 3.01
263.  000000 0  1        int data^buffer [0: ((buffer^size+1)/2) - 1];                   2.01
264.  000000 0  1     end;
265.  000000 0  0
266.  000000 0  0  !--------------------------!                                         3.08
267.  000000 0  0  !  Interactive definitions  !                                        3.08
268.  000000 0  0  !--------------------------!                                         3.08
269.  000000 0  0
270.  000000 0  0     struct ia^data^^ (*);
271.  000000 0  0     begin
272.  000000 0  1        string f1;
273.  000000 0  1        string account [0:11];
274.  000000 0  1        string f2;
275.  000000 0  1        string teller  [0:11];
276.  000000 0  1        string f3;
```

```
  277.    000000 0  1        string branch  [0:11];
  278.    000000 0  1        string f4;
  279.    000000 0  1        string amount  [0:11];
  280.    000000 0  1     end;
  281.    000000 0  0
  282.    000000 0  0     struct ia^resp^^ (*);
  283.    000000 0  0     begin
  284.    000000 0  1        int length;
  285.    000000 0  1        int term^name [0:11];
  286.    000000 0  1        fixed start^time;
  287.    000000 0  1        int (32) xtime [0:1];
  288.    000000 0  1        fixed resp^time = xtime;
  289.    000000 0  1     end;
  290.    000000 0  0
  291.    000000 0  0   !-------------------------!
  292.    000000 0  0   !  IO device definitions  !
  293.    000000 0  0   !-------------------------!
  294.    000000 0  0
  295.    000000 0  0   ! IO-devices
  296.    000000 0  0
  297.    000000 0  0     literal receive^file^   = 0, config^file^   = 1, log^file^     = 2,
  298.    000000 0  0             trace^file^     = 3, hometerm^file^ = 4, terminal^file^ = 5,
  299.    000000 0  0             server^file^    = 6, tfile^file^    = 7;
  300.    000000 0  0
  301.    000000 0  0   ! IO-operations
  302.    000000 0  0
  303.    000000 0  0     define   iodev^check (f, n, o) =
  304.    000000 0  0              if <> then
  305.    000000 0  0                 call iodev^failure (f, n, o)#;
  306.    000000 0  0
  307.    000000 0  0     literal  open^attempt        = 0,
  308.    000000 0  0              read^attempt        = 1,
  309.    000000 0  0              write^attempt       = 2,
  310.    000000 0  0              wr^attempt          = 3,
  311.    000000 0  0              control^attempt     = 4,
  312.    000000 0  0              awaitio^done        = 5;
  313.    000000 0  0
  314.    000000 0  0     struct iodev^^ (*);
  315.    000000 0  0     begin
  316.    000000 0  1        string name^ [0:23];
  317.    000000 0  1        int filename = name^;
  318.    000000 0  1        int filenum;
  319.    000000 0  1        int type;
  320.    000000 0  1        int error;
  321.    000000 0  1        int bufaddr;
  322.    000000 0  1     end;
  323.    000000 0  0
  324.    000000 0  0   !-------------------------!                                    2.00
  325.    000000 0  0   ! TX STATS related defines!                                    2.00
  326.    000000 0  0   !-------------------------!                                    2.00
  327.    000000 0  0
  328.    000000 0  0   ?SOURCE TXTAL
Source file:  [2]  $TURF.NVCTXGEN.TXTAL  12JAN87 - 09:43:18
    1.    000000 0  0   ! SCHEMA PRODUCED DATE - TIME : 1/12/87 09:43:13
    2.    000000 0  0   ?SECTION TX
```

```
  4.  000000 0  0   ! Record TX created on 01/12/87 at 09:43
  5.  000000 0  0   !---------------------------------------------------------------
  6.  000000 0  0   !  The following record is written by the program TXGEN for each
  7.  000000 0  0   !  ET1 transaction sent and received to/from the SUT system.
  8.  000000 0  0   !  The program TXREP scans these records to produce performance
  9.  000000 0  0   !  statistics about the ET1 benchmark.
 10.  000000 0  0   !
 11.  000000 0  0   !  25Sep86. Nhan Chu.
 12.  000000 0  0   !---------------------------------------------------------------
 13.  000000 0  0   !  18Nov86. Make UNSIGNED binary 8.
 14.  000000 0  0   !  12Jan87. Gerhard Huff: term-count, term-index binary 16, added filler
 15.  000000 0  0   STRUCT    TX^DEF (*);
 16.  000000 0  0      BEGIN
 17.  000000 0  1   !                                  number of terminals ( <= 250 )
 18.  000000 0  1      INT       TERM^COUNT;
 19.  000000 0  1   !                                  terminal id
 20.  000000 0  1      INT       TERM^INDEX;
 21.  000000 0  1   !                                  user-specified tx rate (tps * 10)
 22.  000000 0  1   !                                  =0 max rate, pos=constant, neg=random
 23.  000000 0  1      INT       ARRIVAL^RATE;
 24.  000000 0  1   !                                  4-word timestamp of input time
 25.  000000 0  1      FIXED     SEND^TIMESTAMP;
 26.  000000 0  1   !                                  interval (ms) until the next arrival
 27.  000000 0  1      INT(32)   NEXT^ARRIVAL^TIME;
 28.  000000 0  1   !                                  tx response time in milliseconds
 29.  000000 0  1      INT(32)   RESPONSE^TIME;
 30.  000000 0  1   !                                  filler to bring record up to 32
 31.  000000 0  1      FILLER    10;
 32.  000000 0  1      END;                                                          3.00
329.  000000 0  0                                                                  3.00
330.  000000 0  0     literal txstat^bufsize = 4096,                               3.00
331.  000000 0  0             txstat^numrecs = txstat^bufsize / $len (tx^def); ! must be 128  3.00
332.  000000 0  0                                                                  3.00
```

```
000000 0  0   !-------------------------!
000000 0  0   !     GLOBAL variables    !
000000 0  0   !-------------------------!
000000 0  0
000000 0  0     int     .active^thread (tcb^^);
000000 0  0
000000 0  0     struct  ready^list   (queue^^);
000000 0  0     struct  time^list    (queue^^);                                          3.01
000000 0  0     struct  suspend^list (queue^^);                                          3.05
000000 0  0
000000 0  0   ! struct .threads (tcb^^) [0:max^threads-1];
000000 0  0     int     .threads (tcb^^) := %070000; !needs 34750 w (~35pages, 36 resrvd)3.01
000000 0  0     struct  general^pool (general^pool^^);
000000 0  0     struct .startup^parameter (startup^param^^) [0:max^keyword];
000234 0  0
000234 0  0     struct  receive  (iodev^^);
000234 0  0     struct  hometerm (iodev^^);
000234 0  0     struct  config   (iodev^^);
000234 0  0     struct  script   (iodev^^);
000234 0  0     struct  logfile  (logfile^^);
000234 0  0     struct  txlog    (iodev^^);                                              2.00
000234 0  0     int     .txstat   (tx^def);      ! pointer to current stat log record   3.00
000234 0  0     int      txstat^recs;            ! number of log records in buffer      3.00
000234 0  0
000234 0  0   ! int     .script^buffer [0:2047];                                        2.00
000234 0  0     int      shutdown;
000234 0  0     int      startup;
000234 0  0     int      log^enable;
000234 0  0     int      next^thread := 0;
000234 0  0     int      active^threads;
000234 0  0     int      switch^3;
000234 0  0     fixed    start^time;             ! to compute length of run             2.00
000234 0  0     int      start^delay; !in .01 s! ! time between starting each thread    2.03
000234 0  0
000234 0  0     int      rate := 0;              ! user-specified tx rate (tps*10)      2.00
000234 0  0     int      arrival^type := 0;      ! 0=>think=0; 1=>constant; 2=>random   2.00
000234 0  0
000234 0  0     int(32) mean^per^thread;         ! mean inter-arrival time in ms        2.00
000234 0  0     int      thread^count;           ! =active^threads                      2.00
000234 0  0
000234 0  0     int(32) seed1 :=  82848229D;     ! random number seed                   2.00
000234 0  0     int(32) .seed [0:39] := [        ! these seeds generate numbers         3.06
000354 0  0                                      ! that are 1 million apart             3.06
000354 0  0           78126602D, 1348540613D, 1075807989D, 1418823682D,   25918206D,  3.07
000354 0  0         1709501965D,  524504828D,  796774236D,   74313960D,   87021785D,  3.07
000354 0  0         2099774560D, 1736406066D,  858298972D, 1598162604D, 1855425428D,  3.07
000354 0  0         1352188490D,  351827176D,  622438816D,  617360608D, 1776878068D,  3.07
000354 0  0         1077339412D,  791180181D, 1865168993D, 1905152688D,  746303301D,  3.07
000354 0  0          614446747D,  199010403D, 1413141436D, 1794410409D,  732784251D,  3.07
000354 0  0         1299527802D,  436218352D, 1249141089D,  404021122D, 2084553887D,  3.07
000354 0  0          862847618D,  855703798D,  219155865D, 1762054344D, 1698349082D   3.07
000354 0  0                               ];                                         3.06
000354 0  0
000354 0  0     int(32) txmax,                   ! max number of tx to be replayed      2.02
000354 0  0             txcount := 0d;           ! current number of replayed tx        2.02
000354 0  0
000354 0  0     int      max^tx^out;             ! max number of transaction to be send 3.05
```

```
391.    000354 0  0                                        ! concurrent                          3.05
392.    000354 0  0     int     tx^outstanding := 0;     ! number of TX send to SUT           3.05
393.    000354 0  0
394.    000354 0  0     int     cancel^timer^done := false;                                    2.04
395.    000354 0  0
396.    000354 0  0     int    .receive^buffer [0:127];                                        2.00
397.    000554 0  0     string .general^pool^buffer [0:general^pool^size-1];                   2.00
398.    020554 0  0
399.    020554 0  0     int    inter^active^data := false;                                     3.08
400.    020554 0  0     struct .ia^data (ia^data^^) = receive^buffer;                          3.08
401.    020554 0  0     struct .ia^resp (ia^resp^^) = receive^buffer;                          3.08
402.    020554 0  0
```

```
404.    020554 0  0  ?NOMAP
405.    020554 0  0  !---------- RANDOM NUMBERS ----------------------------------------    2.00
406.    020554 0  0
407.    020554 0  0    proc err (error);                                                   2.00
408.    000000 1  0         int error;                                          forward;   2.00
409.    000000 0  0    real proc randf (seed);                                             2.00
410.    000000 1  0         int(32) .seed;                                      forward;   2.00
411.    000000 0  0    int proc randint (i, j, rand);                                      2.00
412.    000000 1  0         int  i, j; real rand;                               forward;   2.00
413.    000000 0  0    int(32) proc negexp (mean, rand);                                   2.00
414.    000000 1  0         int(32) mean; real rand;                            forward;   2.00
415.    000000 0  0    int(32) proc sample^arrival (mean, arrival^type);                   2.00
416.    000000 1  0         int(32) mean; int arrival^type;                     forward;   2.00
417.    000000 0  0
418.    000000 0  0  !---------- ERROR REPORTING & LOG FILE ------------------------------
419.    000000 0  0    proc log^print (fmt,p1, p2, p3, p4, p5, p6, p7, p8, p9, p10,
420.    000000 1  0                        p11, p12, p13, p14) VARIABLE;
421.    000000 1  0
422.    000000 1  0    string .fmt;                                  !format string
423.    000000 1  0    int     p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,      !parameters
424.    000000 1  0            p11,p12,p13,p14;
425.    000000 1  0                                                                forward;
426.    000000 0  0    proc logfile^iocomplete (error);
427.    000000 1  0         int error;                                             forward;
428.    000000 0  0    proc write^logfile (data, length);
429.    000000 1  0         int .data, length;                                     forward;
430.    000000 0  0    proc init^logfile;                                          forward;
431.    000000 0  0
432.    000000 0  0  !---------- GENERAL POOL ---------------------------------------------
433.    000000 0  0
434.    000000 0  0    int  proc allocate^buffer (size);
435.    000000 1  0         int size;                                              forward;
436.    000000 0  0    proc deallocate^buffer (buffer);
437.    000000 1  0         int .buffer;                                           forward;
438.    000000 0  0    proc init^general^pool;                                     forward;
439.    000000 0  0
440.    000000 0  0  !---------- QUEUE ----------------------------------------------------
441.    000000 0  0
442.    000000 0  0    int proc queue^get (q); int .q (queue^^);                   forward;
443.    000000 0  0    proc queue^put (q, newelem); int .q (queue^^), .newelem (queue^^);
444.    000000 1  0                                                                forward;
445.    000000 0  0    proc queue^init (q); int .q (queue^^);                      forward;
446.    000000 0  0
447.    000000 0  0  !---------- THREAD STATES --------------------------------------------
448.    000000 0  0
449.    000000 0  0    proc thread^closed;                                         forward;
450.    000000 0  0    proc thread^ready;                                          forward;
451.    000000 0  0    proc thread^active;                                         forward;
452.    000000 0  0    proc thread^stopping;                                       forward;
453.    000000 0  0    proc thread^think;                                          forward;   2.00
454.    000000 0  0
455.    000000 0  0  !---------- THREAD ---------------------------------------------------
456.    000000 0  0
457.    000000 0  0    proc assign^thread (tcb, record);
458.    000000 1  0         int .tcb (tcb^^), .record (config^record^^);           forward;
459.    000000 0  0    proc thread^awaitio (next^substate);
460.    000000 1  0         int next^substate;                                     forward;
```

```
461.    000000 0  0     proc activate^thread (tcb);
462.    000000 1  0          int .tcb (tcb^^);                                  forward;
463.    000000 0  0     proc save^thread^state;                                  forward;
464.    000000 0  0     proc select^thread;                                      forward;
465.    000000 0  0     proc suspend^thread;                                     forward;
466.    000000 0  0     proc process^thread;                                     forward;
467.    000000 0  0     proc process^thread^state;                               forward;
468.    000000 0  0     proc init^threads;                                       forward;
469.    000000 0  0
470.    000000 0  0   !---------- IO ----------------------------------------------------
471.    000000 0  0
472.    000000 0  0     INT (32) Proc build^iotag (tcb, file^type) variable;
473.    000000 1  0          int .tcb (tcb^^), file^type;                       forward;
474.    000000 0  0     Proc iodev^failure (f, name, operation) variable;
475.    000000 1  0          int f, .name, operation;                           forward;
476.    000000 0  0     Proc iodev^init (dev,flags, sync, type) variable;
477.    000000 1  0          int .dev (iodev^^), flags, sync, type;             forward;
478.    000000 0  0     proc io^wait;                                            forward;
479.    000000 0  0
480.    000000 0  0   !---------- MAIN --------------------------------------------------
481.    000000 0  0
482.    000000 0  0     proc loadgen^init;                                       forward;
483.    000000 0  0     proc print^help;                                         forward;    2.00
484.    000000 0  0     proc print^times (ts^start, ts^stop);                                2.00
485.    000000 1  0          fixed ts^start, ts^stop;                           forward;    2.00
486.    000000 0  0     proc txgen   main;                                       forward;
487.    000000 0  0   ?NOLIST, columns 80, source $system.system.extdecs (                  2.00
499.    000000 0  0   ?MAP
```

```
   501.    000000 0  0   ?NOLIST
   532.    000000 0  0     proc err (error);                                          2.00
   533.    000000 1  0   ! ----------------
   534.    000000 1  0       int error;
   535.    000000 1  0     begin
   536.    000000 1  1       call debug;
   537.    000001 1  1     end;

ERROR                           Variable       INT             Direct         L-003

   538.    000000 0  0
   539.    000000 0  0     real proc randf ( seed );
   540.    000000 1  0   ! ------------------------                                   2.00
   541.    000000 1  0       int(32)   .seed;                 ! 0 < positive number < 2**31-1
   542.    000000 1  0     begin
   543.    000000 1  1       real        rn;
   544.    000000 1  1
   545.    000000 1  1       fixed(9) subproc RANDOM;
   546.    000000 2  1       begin
   547.    000000 2  2         literal a = 16807F,            ! multiplier = 7**5
   548.    000000 2  2                 m = 2147483647F;       ! modulus = 2**31 - 1
   549.    000000 2  2         fixed(0) z;                    ! 64-bit integer
   550.    000000 2  2
   551.    000000 2  2         z    := a * $DFIX( seed, 0 );
   552.    000010 2  2         seed := $FIXD( z := z - (z/m)*m );!modulo op must be done exactly
   553.    000037 2  2         return  $scale( z, 9 ) / m;    ! giving 9 digits of precision
   554.    000052 2  2       end; !subproc RANDOM!

A                                 Literal        FIXED (0)       %000000 000000 000000 040647
M                                 Literal        FIXED (0)       %000000 000000 077777 177777
Z                                 Variable       FIXED (0)       Direct         S-003

   555.    000053 1  1
   556.    000053 1  1       do rn := $fltr( RANDOM ) / 1.0E+9
   557.    000054 1  1       until rn <> 0.0E0;
   558.    000066 1  1       return  rn;                    ! rn is in the open interval (0,1)
   559.    000070 1  1     end; !real proc randf!

RANDOM                            Subproc        FIXED (9)       %000000
RN                                Variable       REAL (32)       Direct         L+001
SEED                              Variable       INT (32)        Indirect       L-003

   560.    000000 0  0
   561.    000000 0  0     int(32) proc negexp ( mean, rand );                        2.00
   562.    000000 1  0   ! ----------------------------------
   563.    000000 1  0       int(32)       mean;
   564.    000000 1  0       real          rand;            ! random number between 0:1(randf)
   565.    000000 1  0     begin
   566.    000000 1  1       if mean < 0D then call err ( !err^negexp^mean^lt^zero! 0 );
   567.    000007 1  1       ! There is a possibility of overflow in the next expr.
   568.    000007 1  1       ! If so, scale 'mean'.
   569.    000007 1  1       return  $dblr( $eflt( - mean ) * dlog^( $eflt( rand ) ) );
   570.    000023 1  1     end; !proc negexp!

MEAN                              Variable       INT (32)        Direct         L-006
RAND                              Variable       REAL (32)       Direct         L-004
```

```
  571.    000000 0  0
  572.    000000 0  0     int proc randint( i, j, rand );                                    2.00
  573.    000000 1  0   ! -------------------------------
  574.    000000 1  0       int           i,                  ! lower bound
  575.    000000 1  0                     j;                  ! upper bound
  576.    000000 1  0       real          rand;               ! random number between 0:1(randf)
  577.    000000 1  0     begin
  578.    000000 1  1       if j <= i then call err( !err^ub^lt^lb! 0 );
  579.    000007 1  1       return  i + $int( $flt( j - i + 1 ) * rand );
  580.    000022 1  1     end; !int proc randint!
```

I	Variable	INT	Direct	L-006
J	Variable	INT	Direct	L-005
RAND	Variable	REAL (32)	Direct	L-004

```
  582.    000000 0  0
  583.    000000 0  0     int(32) proc sample^arrival (mean, arrival^type);                    2.00
  584.    000000 1  0    !-----------------------------------------------
  585.    000000 1  0       int(32) mean;                 ! inter-arrival time               2.00
  586.    000000 1  0       int     arrival^type;         ! 0=>think time=0,1=>constant,2=>rand  2.00
  587.    000000 1  0     begin
  588.    000000 1  1
  589.    000000 1  1       case arrival^type of
  590.    000002 1  1       begin
  591.    000002 1  2          ! 0 => zero think time
  592.    000002 1  2          return 0d;
  593.    000004 1 CE1         ! 1 => constant arrival with mean 'mean'
  594.    000004 1  2          return mean;
  595.    000006 1 CE2         ! 2 => random arrival, ie exponential with mean 'mean'
  596.    000006 1  2          return negexp (mean, randf (seed1));
  597.    000015 1 CE3       otherwise
  598.    000015 1  2          ! improve ...
  599.    000015 1  2          call debug;
  600.    000016 1  2       end;
  601.    000032 1  1
  602.    000032 1  1     end; !int(32) proc sample^arrival!

ARRIVAL^TYPE                    Variable       INT             Direct         L-003
MEAN                            Variable       INT (32)        Direct         L-005
```

```
604.  000000 0  0  !#########################################################################
605.  000000 0  0  !#                                                                       #
606.  000000 0  0  !#                          FORMATPRINT                                  #
607.  000000 0  0  !#                                                                       #
608.  000000 0  0  !#  FORMAT STRINGS ARE CONTAINED WITHIN A READ/ONLY STRING ARRAY.       #
609.  000000 0  0  !#  EACH LINE OF THE FORMAT IS ENDED WITH A NULL (0) BYTE, AND          #
610.  000000 0  0  !#  THE END OF THE STRING ARRAY IS MARKED BY AN (%377) BYTE.           #
611.  000000 0  0  !#                                                                       #
612.  000000 0  0  !#  FIELD INDICATORS ARE:                                              #
613.  000000 0  0  !#            #  -  signed decimal                                     #
614.  000000 0  0  !#            &  -  unsigned decimal                                   #
615.  000000 0  0  !#            %  -  octal                                              #
616.  000000 0  0  !#            '  -  string (blank-fill to field length)                #
617.  000000 0  0  !#            \  -  string (no blank fill)                             #
618.  000000 0  0  !#            ?  -  escape character                                   #
619.  000000 0  0  !#                                                                     #
620.  000000 0  0  !#  FOR NUMERIC FIELDS, AN INTEGER VALUE SHOULD BE PASSED.            #
621.  000000 0  0  !#  FOR STRING FIELDS, THE ADDRESS OF THE STRING ARRAY SHOULD BE      #
622.  000000 0  0  !#    PASSED.                                                          #
623.  000000 0  0  !#                                                                     #
624.  000000 0  0  !#########################################################################
625.  000000 0  0
626.  000000 0  0    PROC log^print (fmt,p1, p2, p3, p4, p5, p6, p7, p8, p9, p10,
627.  000000 1  0                    p11, p12, p13, p14) VARIABLE;
628.  000000 1  0
629.  000000 1  0    STRING .fmt;                                  !format string
630.  000000 1  0    INT     p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,      !parameters
631.  000000 1  0            p11,p12,p13,p14;
632.  000000 1  0
633.  000000 1  0    BEGIN
634.  000000 1  1
635.  000000 1  1      STRING  buffer[0:132],                      !print buffer
636.  000000 1  1             .bufptr,                             !current position in buffer
637.  000000 1  1             .p;                                  !working ptr
638.  000000 1  1      INT     .parm := @p1,                        !current parameter
639.  000000 1  1              save,                               !saved address
640.  000000 1  1              i;                                  !indexing variable
641.  000000 1  1
642.  000000 1  1
643.  000000 1  1    if startup < 2 and not log^enable then return;
644.  000012 1  1
645.  000012 1  1    DO
646.  000012 1  1      BEGIN
647.  000012 1  2
648.  000012 1  2      STACK @fmt,0; CODE (SBU %660);        !SCAN fmt UNTIL 0 -> @p;
649.  000015 1  2      STORE @p;
650.  000016 1  2
651.  000016 1  2      STACK @buffer, @fmt, @p'-'@fmt+2;  !buffer':='fmt FOR @p'-'@fmt+2
652.  000025 1  2      CODE (MOVB %67);
653.  000026 1  2
654.  000026 1  2      @fmt := @p[1];
655.  000031 1  2      @bufptr := @buffer;
656.  000034 1  2
657.  000034 1  2      WHILE bufptr DO                       !check for format characters!
658.  000036 1  2
659.  000036 1  2        IF bufptr = "#" OR                  !  signed decimal integer!
660.  000036 1  2           bufptr = "&" OR                  !unsigned decimal integer!
```

```
661.    000036 1  2          bufptr = "%" THEN               !unsigned octal   integer!
662.    000047 1  2              BEGIN
663.    000047 1  3                SCAN bufptr WHILE bufptr -> @p;
664.    000053 1  3                IF bufptr = "#" AND parm < 0 THEN
665.    000061 1  3                  BEGIN
666.    000061 1  4                    bufptr  := "-";
667.    000063 1  4                    parm    := -parm;
668.    000066 1  4                    @bufptr := @bufptr[1];
669.    000071 1  4                  END;
670.    000071 1  3                CALL NUMOUT (bufptr, parm,
671.    000071 1  3                             IF bufptr="%" THEN 8 ELSE 10,@p'-'@bufptr);
672.    000106 1  3                @bufptr := @p;
673.    000110 1  3                @parm := @parm[1];
674.    000113 1  3              END
675.    000113 1  2          ELSE IF bufptr = "'" THEN          !character string (blank fill)!
676.    000117 1  2              BEGIN
677.    000117 1  3                @p := parm;
678.    000121 1  3                WHILE bufptr = "'" DO
679.    000124 1  3                  BEGIN
680.    000124 1  4                    IF p THEN
681.    000126 1  4                      BEGIN
682.    000126 1  5                        bufptr := p;
683.    000130 1  5                        @p := @p[1];
684.    000133 1  5                      END
685.    000133 1  4                    ELSE bufptr := " ";
686.    000136 1  4                    @bufptr := @bufptr[1];
687.    000141 1  4                  END;
688.    000142 1  3                @parm := @parm[1];
689.    000145 1  3              END
690.    000145 1  2          ELSE IF bufptr = "\" THEN          !character string (no blank fill)!
691.    000151 1  2              BEGIN
692.    000151 1  3                @p := parm;
693.    000153 1  3                i := -1;
694.    000155 1  3                WHILE bufptr[i:=i+1] = "\" AND p[i] DO
695.    000167 1  3                  bufptr[i] := p[i];
696.    000173 1  3                @bufptr := @bufptr[i];
697.    000176 1  3                IF bufptr = "\" THEN
698.    000201 1  3                  BEGIN
699.    000201 1  4                    SCAN bufptr WHILE "\" -> @p;
700.    000205 1  4                    bufptr ':=' p FOR @buffer[132]'-'@p;
701.    000217 1  4                  END;
702.    000217 1  3                @parm := @parm[1];
703.    000222 1  3              END
704.    000222 1  2          ELSE                               !no replacement field!
705.    000224 1  2              BEGIN
706.    000224 1  3                IF bufptr = "?" THEN       !escape character!
707.    000227 1  3                  bufptr ':=' bufptr[1] FOR @buffer[132]-@bufptr;
708.    000242 1  3                @bufptr := @bufptr[1];
709.    000245 1  3              END;
710.    000246 1  2
711.    000246 1  2        if startup = 2 then
712.    000251 1  2           CALL WRITE (hometerm.filenum, buffer, @bufptr'-'@buffer)
713.    000264 1  2        else
714.    000265 1  2           call write^logfile (buffer, @bufptr '-' @buffer);
715.    000274 1  2        END
716.    000274 1  1      UNTIL bufptr[1] = %377;
717.    000300 1  1
```

```
  718.    000300 1  1     END; ! proc fprint
```

BUFFER	Variable	STRING	Direct	L+001
BUFPTR	Variable	STRING	Indirect	L+104
FMT	Variable	STRING	Indirect	L-022
I	Variable	INT	Direct	L+110
P	Variable	STRING	Indirect	L+105
P1	Variable	INT	Direct	L-021
P10	Variable	INT	Direct	L-010
P11	Variable	INT	Direct	L-007
P12	Variable	INT	Direct	L-006
P13	Variable	INT	Direct	L-005
P14	Variable	INT	Direct	L-004
P2	Variable	INT	Direct	L-020
P3	Variable	INT	Direct	L-017
P4	Variable	INT	Direct	L-016
P5	Variable	INT	Direct	L-015
P6	Variable	INT	Direct	L-014
P7	Variable	INT	Direct	L-013
P8	Variable	INT	Direct	L-012
P9	Variable	INT	Direct	L-011
PARM	Variable	INT	Indirect	L+106
SAVE	Variable	INT	Direct	L+107

```
   720.    000000 0  0    proc write^logfile (data, length);
   721.    000000 1  0  ! ---------------------------------
   722.    000000 1  0    int .data, length;
   723.    000000 1  0    begin
   724.    000000 1  1      int .Q (queue^^),
   725.    000000 1  1          .
   726.    000000 1  1           msg (log^msg^^),
   727.    000000 1  1           wlength, extra;
   728.    000000 1  1
   729.    000000 1  1      begin
   730.    000001 1  2        if not logfile.active then return;     !
   731.    000004 1  2
   732.    000004 1  2        wlength := (length + 1) '>>' 1;      ! size in words
   733.    000010 1  2        extra := $offset (msg.data);
   734.    000012 1  2        if (@msg := allocate^buffer (length + extra)) = nil then
   735.    000022 1  2        begin
   736.    000022 1  3  --         logfile.active := false;                                  3.04
   737.    000022 1  3  --         log^enable := false;                                      3.04
   738.    000022 1  3           return;
   739.    000023 1  3        end;
   740.    000023 1  2
   741.    000023 1  2        msg.data ':=' data for wlength;
   742.    000030 1  2        msg.length := length;
   743.    000033 1  2        msg.queue.succ := msg.queue.pred := nil; !initialize!          2.00
   744.    000037 1  2
   745.    000037 1  2        @q := intaddr (logfile.msg^q);
   746.    000043 1  2        call queue^put (q, msg);
   747.    000047 1  2        if not logfile.io^busy then
   748.    000051 1  2        begin
   749.    000051 1  3          call write (logfile.dev.filenum, msg.data, length,,
   750.    000051 1  3                      build^iotag (, log^file^));
   751.    000065 1  3          logfile.io^busy := true;
   752.    000067 1  3        end;
   753.    000067 1  2      end; !body
   754.    000067 1  1    end; ! of msg^write

DATA                          Variable       INT            Indirect       L-004
EXTRA                         Variable       INT            Direct         L+004
LENGTH                        Variable       INT            Direct         L-003
MSG                           Variable,10    STRUCT-I       Indirect       L+002
Q                             Variable,4     STRUCT-I       Indirect       L+001
WLENGTH                       Variable       INT            Direct         L+003

   755.    000000 0  0
   756.    000000 0  0    proc logfile^iocomplete (error);
   757.    000000 1  0  ! -------------------------------
   758.    000000 1  0    int error;
   759.    000000 1  0    begin
   760.    000000 1  1      int .msg^q (queue^^),
   761.    000000 1  1          .msg (log^msg^^);
   762.    000000 1  1
   763.    000000 1  1      subproc dequeue^msg;
   764.    000000 2  1      begin
   765.    000000 2  2           @msg := queue^get (msg^q);
   766.    000004 2  2           call deallocate^buffer (msg);
   767.    000007 2  2      end;
   768.    000010 1  1
```

```
   769.     000010 1  1        begin
   770.     000011 1  2           @msg^q := intaddr (logfile.msg^q);
   771.     000015 1  2           logfile.io^busy := false;
   772.     000017 1  2           call dequeue^msg;
   773.     000020 1  2           if error then
   774.     000022 1  2           begin
   775.     000022 1  3              logfile.active := false;
   776.     000024 1  3              while queue^any (msg^q) do call dequeue^msg;
   777.     000032 1  3           end else
   778.     000033 1  2              if queue^any (msg^q) then
   779.     000037 1  2              begin
   780.     000037 1  3                 @msg := msg^q.succ;
   781.     000041 1  3                 call write (logfile.dev.filenum,msg.data,msg.length,,
   782.     000041 1  3                            build^iotag (, log^file^));
   783.     000057 1  3                 logfile.io^busy := true;
   784.     000061 1  3              end;
   785.     000061 1  2        end;
   786.     000061 1  1     end; !of msg^iocomplete

DEQUEUE^MSG                      Subproc                        %000000
ERROR                            Variable       INT             Direct        L-003
MSG                              Variable,10    STRUCT-I        Indirect      L+002
MSG^Q                            Variable,4     STRUCT-I        Indirect      L+001

   787.     000000 0  0
   788.     000000 0  0     proc init^logfile;
   789.     000000 1  0   ! -------------------
   790.     000000 1  0     begin
   791.     000000 1  1        int error,
   792.     000000 1  1            .q := intaddr (logfile.msg^q),
   793.     000000 1  1            .device (iodev^^) := intaddr (logfile.dev),
   794.     000000 1  1            .param (startup^param^^) := @startup^parameter [param^log];
   795.     000000 1  1
   796.     000000 1  1        begin
   797.     000015 1  2           if param.setup then
   798.     000021 1  2           begin
   799.     000021 1  3              logfile.dev.filename ':=' param.name for 12;
   800.     000025 1  3              call iodev^init(logfile.dev, open^nowait);
   801.     000036 1  3              call queue^init (q);
   802.     000041 1  3              log^enable := true;
   803.     000043 1  3              logfile.active := true;
   804.     000045 1  3           end else
   805.     000046 1  2           begin
   806.     000046 1  3              log^enable := false;
   807.     000050 1  3              logfile.active := false;
   808.     000052 1  3           end;
   809.     000052 1  2        end;
   810.     000052 1  1     end; ! of init^logfile

DEVICE                           Variable,40    STRUCT-I        Indirect      L+003
ERROR                            Variable       INT             Direct        L+001
PARAM                            Variable,32    STRUCT-I        Indirect      L+004
Q                                Variable       INT             Indirect      L+002
```

```
  812.  000000 0  0    proc init^general^pool;
  813.  000000 1  0  ! ---------------------
  814.  000000 1  0    begin
  815.  000000 1  1       int status;
  816.  000000 1  1       int .ext start := $udbl( @general^pool^buffer );                    2.00
  817.  000000 1  1       string def^err = 'P' :=
  818.  000000 1  1              ["Unable to define general pool, status ?####", 0, %377];
  819.  000027 1  1
  820.  000027 1  1       begin
  821.  000033 1  2          general^pool.status := definepool (general^pool.header,
  822.  000033 1  2                                             start,
  823.  000033 1  2                                             $dbl(general^pool^size));    2.00
  824.  000045 1  2          if general^pool.status then                                      2.00
  825.  000047 1  2          begin
  826.  000047 1  3             call log^print (def^err, general^pool.status);                2.00
  827.  000066 1  3          end;
  828.  000066 1  2       end;
  829.  000066 1  1    end;

DEF^ERR                        Variable       STRING          Direct         P+000
START                          Variable       INT             EXT Pointer    L+002
STATUS                         Variable       INT             Direct         L+001

  830.  000000 0  0
  831.  000000 0  0    int proc allocate^buffer (size);
  832.  000000 1  0  ! -------------------------------
  833.  000000 1  0    int size;
  834.  000000 1  0    begin
  835.  000000 1  1       int (32) buf^size;
  836.  000000 1  1       int .addr, temp;                                                    3.04
  837.  000000 1  1       int .ext xaddr;
  838.  000000 1  1       string alloc^err = 'P' :=
  839.  000000 1  1              ["Unable to allocate a buffer from general pool", 0, %377];
  840.  000030 1  1       begin
  841.  000031 1  2          buf^size := $udbl (size);
  842.  000034 1  2          @xaddr := getpool (general^pool.header, buf^size);
  843.  000043 1  2          if <> then
  844.  000044 1  2          begin
  845.  000044 1  3             temp := startup;                                             3.04
  846.  000046 1  3             startup := 2;   ! print on HOMETERM, instead of LOGFILE      3.04
  847.  000050 1  3             call log^print (alloc^err);
  848.  000067 1  3             startup := temp;                                             3.04
  849.  000071 1  3             return nil;
  850.  000073 1  3          end else
  851.  000074 1  2          begin
  852.  000074 1  3             @addr := $int (@xaddr '>>' 1); ! word address in memory
  853.  000100 1  3             return @addr;
  854.  000102 1  3          end;
  855.  000102 1  2       end;
  856.  000102 1  1    end; !of allocate^buffer

ADDR                           Variable       INT             Indirect       L+003
ALLOC^ERR                      Variable       STRING          Direct         P+000
BUF^SIZE                       Variable       INT (32)        Direct         L+001
SIZE                           Variable       INT             Direct         L-003
TEMP                           Variable       INT             Direct         L+004
XADDR                          Variable       INT             EXT Pointer    L+005
```

```
  857.    000000 0  0
  858.    000000 0  0     proc deallocate^buffer (buffer);
  859.    000000 1  0   ! ------------------------------
  860.    000000 1  0     int .buffer;
  861.    000000 1  0     begin
  862.    000000 1  1        int temp;                                                    3.04
  863.    000000 1  1        int .ext xaddr;
  864.    000000 1  1        string dealloc^err = 'P' :=
  865.    000000 1  1           ["Unable to deallocate a buffer from general pool", 0, %377];
  866.    000031 1  1
  867.    000031 1  1        begin
  868.    000032 1  2           @xaddr := ($udbl (@buffer)) '<<' 1;
  869.    000036 1  2           call putpool (general^pool.header, xaddr);
  870.    000044 1  2           if <> then
  871.    000045 1  2           begin
  872.    000045 1  3              temp := startup;                                      3.04
  873.    000047 1  3              startup := 2;   ! print on HOMETERM, instead of LOGFILE  3.04
  874.    000051 1  3              call log^print (dealloc^err);
  875.    000070 1  3              startup := temp;                                      3.04
  876.    000072 1  3           end ;
  877.    000072 1  2        end;
  878.    000072 1  1     end; !of deallocate^buffer;

BUFFER                          Variable      INT           Indirect      L-003
DEALLOC^ERR                     Variable      STRING        Direct        P+000
TEMP                            Variable      INT           Direct        L+001
XADDR                           Variable      INT           EXT Pointer   L+002
```

```
   880.    000000 0  0     Int Proc Queue^get (q);
   881.    000000 1  0   ! ----------------------
   882.    000000 1  0     Int .q (queue^^);
   883.    000000 1  0     begin
   884.    000000 1  1        int .first  (queue^^),
   885.    000000 1  1            .second (queue^^);
   886.    000000 1  1        begin
   887.    000001 1  2           @first := q.succ;
   888.    000003 1  2           @second := first.succ;
   889.    000005 1  2           q.succ := @second;
   890.    000007 1  2           second.pred := first.pred;
   891.    000012 1  2           first.succ :=
   892.    000012 1  2           first.pred := 0;
   893.    000015 1  2           return @first;
   894.    000017 1  2        end;
   895.    000017 1  1     end;

FIRST                           Variable,4     STRUCT-I        Indirect       L+001
Q                               Variable,4     STRUCT-I        Indirect       L-003
SECOND                          Variable,4     STRUCT-I        Indirect       L+002

   896.    000000 0  0
   897.    000000 0  0     Proc Queue^put (q, newelem);
   898.    000000 1  0   ! --------------------------
   899.    000000 1  0     int .q (queue^^), .newelem (queue^^);
   900.    000000 1  0     begin
   901.    000000 1  1        int .last (queue^^);
   902.    000000 1  1
   903.    000000 1  1        begin
   904.    000001 1  2           if newelem.pred and newelem.succ then ! link fields must be 0     3.05
   905.    000006 1  2              call debug;                        !
   906.    000007 1  2           @last := q.pred;
   907.    000012 1  2           q.pred := @newelem;
   908.    000014 1  2           newelem.pred := @last;
   909.    000016 1  2           newelem.succ := last.succ;
   910.    000020 1  2           last.succ := @newelem;
   911.    000022 1  2        end;
   912.    000022 1  1     end;

LAST                            Variable,4     STRUCT-I        Indirect       L+001
NEWELEM                         Variable,4     STRUCT-I        Indirect       L-003
Q                               Variable,4     STRUCT-I        Indirect       L-004

   913.    000000 0  0
   914.    000000 0  0     Proc Queue^init (q);
   915.    000000 1  0   ! -------------------
   916.    000000 1  0     int .q (queue^^);
   917.    000000 1  0     begin
   918.    000000 1  1        q.succ := q.pred := @q;
   919.    000004 1  1     end;

Q                               Variable,4     STRUCT-I        Indirect       L-003
```

```
  921.    000000 0  0    proc thread^closed;
  922.    000000 1  0  ! ------------------
  923.    000000 1  0    begin
  924.    000000 1  1      int error;
  925.    000000 1  1      int(32) tag;
  926.    000000 1  1      struct .dev (iodev^^);
  927.    000000 1  1
  928.    000000 1  1      begin
  929.    000004 1  2        case active^thread.substate of
  930.    000007 1  2        begin
  931.    000007 1  3          begin ! 0, open terminal
  932.    000007 1 CE0           dev.filename ':=' active^thread.terminal.name for 12;
  933.    000014 1  4            call iodev^init (dev, (open^exclusive + open^nowait), 0,    3.05
  934.    000014 1  4                                 guardian^typex25ptp);                  3.05
  935.    000023 1  4  !         call iodev^init (dev, open^nowait, 0);                     3.05
  936.    000023 1  4            if dev.error then ! abort thread
  937.    000026 1  4            begin
  938.    000026 1  5              call save^thread^state;
  939.    000027 1  5              active^thread.last^fnum  := dev.filenum;
  940.    000033 1  5              active^thread.last^error := dev.error;
  941.    000037 1  5              active^thread.state := thread^state^stopping;
  942.    000042 1  5              active^thread.substate := 0;
  943.    000045 1  5              return;
  944.    000046 1  5            end else
  945.    000047 1  4            begin
  946.    000047 1  5              active^thread.terminal.fnum := dev.filenum;
  947.    000053 1  5              active^thread.state := thread^state^ready;
  948.    000056 1  5              active^thread.substate := 0;
  949.    000061 1  5            end;
  950.    000061 1  4          end;  ! 0
  951.    000062 1 CE1
  952.    000062 1  3          otherwise call debug;
  953.    000063 1  3        end; ! of case substate
  954.    000075 1  2      end; ! of proc body
  955.    000075 1  1    end; !of proc thread^close

DEV                         Variable,40    STRUCT          Indirect       L+004
ERROR                       Variable       INT             Direct         L+001
TAG                         Variable       INT (32)        Direct         L+002
```

```
  957.    000000 0  0     proc thread^ready;
  958.    000000 1  0   ! -----------------
  959.    000000 1  0     begin
  960.    000000 1  1        int(32) tag;
  961.    000000 1  1        int error;
  962.    000000 1  1
  963.    000000 1  1        begin
  964.    000001 1  2           case active^thread.substate of
  965.    000004 1  2           begin
  966.    000004 1  3              begin  ! 0, connect to terminal, (X25 circuit)
  967.    000004 1 CE0                 tag := build^iotag (active^thread, terminal^file^);
  968.    000012 1  4                  call control (active^thread.terminal.fnum, 17,, tag); ! init call
  969.    000022 1  4                  call thread^awaitio (1);                              ! to circuit
  970.    000025 1  4              end;  ! 0
  971.    000026 1 CE1
  972.    000026 1  3              begin ! 1, connected to X25 line
  973.    000026 1  4                 if not (error := active^thread.last^error) then
  974.    000032 1  4                 begin
  975.    000032 1  5   !                active^thread.state := thread^state^active;
  976.    000032 1  5   !                active^thread.substate := 0;
  977.    000032 1  5                    ! the following improvement will brings the SUT          2.00
  978.    000032 1  5                    ! system faster towards a steady-state:                  2.00
  979.    000032 1  5                    ! terminals start  500 ms apart                          2.00
  980.    000032 1  5                    active^thread.think^time := $dbl (active^thread.index)   2.00
  981.    000032 1  5                       * $dbl (start^delay);  ! .5s between each thread      2.00
  982.    000045 1  5                    active^thread.state := thread^state^think;               2.00
  983.    000050 1  5                    active^thread.substate := 0;
  984.    000053 1  5                 end else
  985.    000054 1  4                 begin
  986.    000054 1  5                    call save^thread^state;
  987.    000055 1  5                    active^thread.state := thread^state^stopping;
  988.    000060 1  5                    active^thread.substate := 0;
  989.    000063 1  5                 end;
  990.    000063 1  4              end;   ! 1, connected to terminal
  991.    000064 1 CE2
  992.    000064 1  3              otherwise call debug;
  993.    000065 1  3           end; !of case substate
  994.    000100 1  2        end;
  995.    000100 1  1     end; !of proc thread^ready

ERROR                          Variable       INT            Direct         L+003
TAG                            Variable       INT (32)       Direct         L+001
```

```
 997.   000000 0  0    proc thread^active;
 998.   000000 1  0  ! ------------------
 999.   000000 1  0    begin
1000.   000000 1  1       int(32) tag;
1001.   000000 1  1       int error, count, i;                                          2.04
1002.   000000 1  1       int(32) response^time, next^arrival^time;                     2.00
1003.   000000 1  1
1004.   000000 1  1       int subproc write^stat;                                       2.00
1005.   000000 2  1       !---------------------                                        2.00
1006.   000000 2  1       begin                                                         2.00
1007.   000000 2  2          literal OK = -1, not^OK = 0;                               2.00
1008.   000000 2  2
1009.   000000 2  2          txstat.term^count        := thread^count;                  2.00
1010.   000002 2  2          txstat.term^index        := active^thread.index;           2.00
1011.   000006 2  2          txstat.arrival^rate      := case arrival^type of           2.00
1012.   000006 2  2                                      begin                         2.00
1013.   000006 2  3                                          !0 = maximum !  0;        2.00
1014.   000006 2  3                                          !1 = constant!  rate;     2.00
1015.   000006 2  3                                          !2 = random  ! -rate;     2.00
1016.   000006 2  3                                      end;                          2.00
1017.   000025 2  2          txstat.send^timestamp    := active^thread.send^timestamp;  2.00
1018.   000033 2  2          txstat.next^arrival^time := next^arrival^time;             2.00
1019.   000037 2  2          txstat.response^time     := response^time;                 2.00
1020.   000043 2  2
1021.   000043 2  2          txstat^recs              := txstat^recs + 1;               3.00
1022.   000045 2  2          if txstat^recs < txstat^numrecs then ! space in buffer?   3.00
1023.   000050 2  2          begin                                                     3.00
1024.   000050 2  3             @txstat := @txstat + $len (tx^def)/2;                  3.00
1025.   000052 2  3             return OK;                                             3.00
1026.   000054 2  3          end else                                                  3.00
1027.   000055 2  2          begin  ! buffer full, write it out and reset pointers     3.00
1028.   000055 2  3             txstat^recs := 0;                                      3.00
1029.   000057 2  3             @txstat := txlog.bufaddr;                              3.00
1030.   000061 2  3             ! write WAITED and buffered to txlog file              3.00
1031.   000061 2  3             call write (txlog.filenum, txstat, txstat^bufsize);    3.00
1032.   000071 2  3             if <> then                                             3.00
1033.   000072 2  3             begin                                                  3.00
1034.   000072 2  4                call iodev^failure (txlog.filenum, txlog.filename,  3.00
1035.   000072 2  4                                    write^attempt);                 3.00
1036.   000100 2  4                return not^OK                                       3.00
1037.   000100 2  4             end else                                               3.00
1038.   000103 2  3                return OK;                                          3.00
1039.   000105 2  3          end;                                                      3.00
1040.   000105 2  2       end;                                                         3.00

NOT^OK                        Literal        INT            %000000
OK                            Literal        INT            %177777
```

```
1042.  000105 1  1        subproc wake^all^waiters;                                          3.05
1043.  000105 2  1   !    ------------------------                                           3.05
1044.  000105 2  1        begin ! wake all other waiting tasks to stop themselfs             3.05
1045.  000105 2  2           while queue^any (time^list) do                                  3.05
1046.  000111 2  2              call activate^thread (queue^get (time^list));                3.05
1047.  000117 2  2                                                                          3.05
1048.  000117 2  2           while queue^any (suspend^list) do                               3.05
1049.  000123 2  2              call activate^thread (queue^get (suspend^list));             3.05
1050.  000131 2  2           return;                                                         3.05
1051.  000132 2  2        end;                                                               3.05
1052.  000132 1  1
1053.  000132 1  1
1054.  000132 1  1        begin
1055.  000133 1  2           error := active^thread.last^error;
1056.  000136 1  2           case active^thread.substate of
1057.  000141 1  2           begin
1058.  000141 1  3              begin ! 0, read TX from script file, remember SCRIPT is waited
1059.  000141 1 CE0                 if ($switches.<3> <> switch^3) OR (txcount >= txmax) then  2.02
1060.  000153 1  4                 begin  !Stopping is in order!                              2.04
1061.  000153 1  5                    call save^thread^state;
1062.  000154 1  5                    active^thread.state := thread^state^stopping;
1063.  000157 1  5                    active^thread.substate := 0;
1064.  000162 1  5                    call wake^all^waiters;                                 3.05
1065.  000163 1  5                    return;                                                3.05
1066.  000164 1  5                 end;
1067.  000164 1  4
1068.  000164 1  4                 if inter^active^data then                                  3.08
1069.  000166 1  4                 begin
1070.  000166 1  5                    inter^active^data := false;
1071.  000170 1  5                    active^thread.interactive := true;
1072.  000174 1  5                    active^thread.data^buffer ':='
1073.  000174 1  5                                        ia^data for $len (ia^data) bytes;
1074.  000201 1  5                    active^thread.substate := 1;
1075.  000204 1  5                    return;
1076.  000205 1  5                 end;
1077.  000205 1  4
1078.  000205 1  4                 if tx^outstanding >= max^tx^out then ! SUT flooded already 3.05
1079.  000211 1  4                 begin                                ! wait and try later 3.05
1080.  000211 1  5                    call queue^put (suspend^list, active^thread);          3.05
1081.  000215 1  5                    call thread^awaitio (0);          ! restart in same    3.05
1082.  000220 1  5                    return;                           ! substate           3.05
1083.  000221 1  5                 end;                                                      3.05
1084.  000221 1  4
1085.  000221 1  4                 txcount := txcount + 1d;                                  2.02
1086.  000225 1  4                 call read (script.filenum,                                3.04
1087.  000225 1  4                           active^thread.data^buffer,                      3.04
1088.  000225 1  4                           buffer^size, count);                            3.04
1089.  000236 1  4                 if <> then
1090.  000237 1  4                 begin
1091.  000237 1  5                    call fileinfo (script.filenum, error);
1092.  000266 1  5                    active^thread.last^error := script.error := error;
1093.  000272 1  5                    active^thread.last^fnum  := script.filenum;
1094.  000275 1  5                    call save^thread^state;
1095.  000276 1  5                    active^thread.state := thread^state^stopping;
1096.  000301 1  5                    active^thread.substate := 0;
1097.  000304 1  5                    call wake^all^waiters;                                 3.05
1098.  000305 1  5                 end else
```

```
1099.    000306 1  4                 active^thread.substate := 1;
1100.    000311 1  4              end;  ! 0
1101.    000312 1 CE1
1102.    000312 1  3              begin ! 1, send TX to terminal line (X25 circuit)
1103.    000312 1  4                 tag := build^iotag (active^thread, terminal^file^);
1104.    000320 1  4                 call writeread (active^thread.terminal.fnum,
1105.    000320 1  4                                 active^thread.data^buffer, 100, 200,, tag); 2.01
1106.    000334 1  4                 active^thread.send^timestamp := juliantimestamp;            2.00
1107.    000347 1  4                 tx^outstanding := tx^outstanding + 1;                       3.05
1108.    000351 1  4                 if $switches.<5> then
1109.    000354 1  4                 begin
1110.    000354 1  5                    stack active^thread.terminal.fnum;
1111.    000356 1  5                    code (ssw);
1112.    000357 1  5                 end;
1113.    000357 1  4                 call thread^awaitio (2);
1114.    000362 1  4              end;  ! 1
1115.    000363 1 CE2
1116.    000363 1  3              begin ! 2, received TX reply from terminal line
1117.    000363 1  4                 tx^outstanding := tx^outstanding - 1;                       3.05
1118.    000365 1  4                 if queue^any (suspend^list) then     ! someone suspended ?? 3.05
1119.    000371 1  4                    call activate^thread (queue^get (suspend^list));         3.05
1120.    000376 1  4
1121.    000376 1  4                 if not error then
1122.    000400 1  4                 begin
1123.    000400 1  5
1124.    000400 1  5                    response^time := $dbl( active^thread.recv^timestamp -    2.00
1125.    000400 1  5                        active^thread.send^timestamp ) / 1000d; !in ms!      2.00
1126.    000414 1  5                    next^arrival^time := sample^arrival ( mean^per^thread,   2.00
1127.    000414 1  5                                                          arrival^type );    2.00
1128.    000421 1  5                    if rate = 0 !max rate! then                              2.00
1129.    000424 1  5                      active^thread.think^time := 0d                         2.00
1130.    000424 1  5                    else  ! unit is 0.01s !                                  2.00
1131.    000432 1  5                      active^thread.think^time := ( next^arrival^time -       2.00
1132.    000432 1  5                                                    response^time ) / 10d;   2.00
1133.    000443 1  5                    if write^stat !OK! then                                  2.00
1134.    000445 1  5                    begin                                                    2.00
1135.    000445 1  6                       active^thread.state := thread^state^think;            2.00
1136.    000450 1  6                       active^thread.substate := 0;                          2.00
1137.    000453 1  6                    end else ! write^stat not OK !                           2.00
1138.    000454 1  5                    begin                                                    2.00
1139.    000454 1  6                       call save^thread^state;                               2.00
1140.    000455 1  6                       active^thread.state := thread^state^stopping;         2.00
1141.    000460 1  6                       active^thread.substate := 0;                          2.00
1142.    000463 1  6                    end; ! write^stat !                                      2.00
1143.    000463 1  5
1144.    000463 1  5                    if active^thread.interactive then                        3.08
1145.    000467 1  5                    begin
1146.    000467 1  6                       active^thread.interactive := false;
1147.    000472 1  6                       ia^resp.length     :=  active^thread.last^count;
1148.    000475 1  6                       ia^resp.term^name ':=' active^thread.terminal.name
1149.    000475 1  6                                              for 12;
1150.    000503 1  6                       ia^resp.start^time :=  active^thread.send^timestamp;
1151.    000513 1  6                       ia^resp.xtime [1]  :=  response^time;
1152.    000517 1  6                       call reply (ia^resp, $len (ia^resp));
1153.    000527 1  6                       if <> then call debug;
1154.    000531 1  6                       call build^iotag (, receive^file^);
1155.    000536 1  6                       call readupdate (receive.filenum,
```

```
 1156.    000536 1  6                                        receive^buffer,
 1157.    000536 1  6                                        256,,
 1158.    000536 1  6                                        tag);
 1159.    000546 1  6                        if <> then call debug;
 1160.    000550 1  6                     end;
 1161.    000550 1  5
 1162.    000550 1  5                  end else ! error in receiving TX reply !
 1163.    000551 1  4
 1164.    000551 1  4                     if error = 122 then
 1165.    000554 1  4                        active^thread.substate := 1  ! resend the same TX
 1166.    000554 1  4                     else
 1167.    000560 1  4                     begin
 1168.    000560 1  5                        call save^thread^state;
 1169.    000561 1  5                        active^thread.state := thread^state^stopping;
 1170.    000564 1  5                        active^thread.substate := 0;
 1171.    000567 1  5                     end;
 1172.    000567 1  4               end;  ! 2
 1173.    000570 1 CE3
 1174.    000570 1  3                  otherwise call debug;
 1175.    000571 1  3              end;     ! of case
 1176.    000605 1  2           end;        ! of proc body
 1177.    000605 1  1      end; !of thread^active
```

COUNT	Variable	INT	Direct	L+004
ERROR	Variable	INT	Direct	L+003
I	Variable	INT	Direct	L+005
NEXT^ARRIVAL^TIME	Variable	INT (32)	Direct	L+010
RESPONSE^TIME	Variable	INT (32)	Direct	L+006
TAG	Variable	INT (32)	Direct	L+001
WAKE^ALL^WAITERS	Subproc		%000105	
WRITE^STAT	Subproc	INT	%000000	

```
 1179.    000000 0  0     proc thread^stopping;
 1180.    000000 1  0   ! --------------------
 1181.    000000 1  0     begin
 1182.    000000 1  1        int(32) tag;
 1183.    000000 1  1        string msg = 'P' := ["Thread stopped ",
 1184.    000010 1  1                            18*["\"], " TCB (###, ##, ##, ?%%%%%%%) ",       2.03
 1185.    000037 1  1                            "F ###, E ###", 0, %377],                        2.03
 1186.    000046 1  1              .t^name [0:34];
 1187.    000046 1  1
 1188.    000046 1  1        begin
 1189.    000053 1  2           case active^thread.substate of
 1190.    000056 1  2           begin
 1191.    000056 1  3              begin ! 0,
 1192.    000056 1 CE0                 call aborttransaction;
 1193.    000060 1  4                  t^name [fnamecollapse (active^thread.terminal.name, t^name)]
 1194.    000060 1  4                         := 0;
 1195.    000070 1  4                  call log^print (msg, @t^name,
 1196.    000070 1  4                                      active^thread.index,
 1197.    000070 1  4                                      active^thread.last^state,
 1198.    000070 1  4                                      active^thread.last^substate,
 1199.    000070 1  4                                      active^thread.last^loc,
 1200.    000070 1  4                                      active^thread.last^fnum,
 1201.    000070 1  4                                      active^thread.last^error);
 1202.    000122 1  4                  call close (active^thread.terminal.fnum);
 1203.    000130 1  4
 1204.    000130 1  4                  if (active^threads := active^threads - 1) = 0 then
 1205.    000135 1  4                     shutdown := true;
 1206.    000137 1  4                  call thread^awaitio (0); ! just stop working for that thread
 1207.    000142 1  4              end;  ! 0
 1208.    000143 1 CE1
 1209.    000143 1  3              otherwise call debug;
 1210.    000144 1  3           end; ! of case substate
 1211.    000156 1  2        end; ! of proc body
 1212.    000156 1  1     end; !of thread^stopping
```

MSG	Variable	STRING	Direct	P+000
TAG	Variable	INT (32)	Direct	L+001
T^NAME	Variable	STRING	Indirect	L+003

```
1214.  000000 0  0    proc thread^think;                                                    2.00
1215.  000000 1  0  ! -----------------
1216.  000000 1  0    begin
1217.  000000 1  1       int    value;                                                      2.03
1218.  000000 1  1       int    .tcb (tcb^^), .Q (queue^^);                                 3.01
1219.  000000 1  1       fixed  now, end^time;                                              3.01
1220.  000000 1  1
1221.  000000 1  1       string s^name [0:34],                                              2.03
1222.  000000 1  1              tle^error  = 'P' := ["TLE allocation failure for terminal ",  2.03
1223.  000022 1  1                  34*["\"], ", file no: ?###", 0, %377],                 2.03
1224.  000054 1  1              dlay^error = 'P' := ["Illegal think time value ?##### for ",  2.03
1225.  000076 1  1                  "terminal ", 34*["\"], ", file no: ?###", 0, %377];    2.03
1226.  000134 1  1
1227.  000134 1  1       case active^thread.substate of
1228.  000140 1  1       begin
1229.  000140 1  2          begin ! 0
1230.  000140 1 CE0          if active^thread.think^time > 0d then !delay!                  2.00
1231.  000146 1  3          begin                                                          3.01
1232.  000146 1  4             now := juliantimestamp / 10000f;   ! in tics (0.01 sec)     3.01
1233.  000161 1  4             end^time := active^thread.end^time ! also in tcis           3.01
1234.  000161 1  4                      := now + $DFIX (active^thread.think^time, 0);      3.01
1235.  000202 1  4                                              ! insert active thread in  3.01
1236.  000202 1  4                                              ! list at right spot       3.01
1237.  000202 1  4                                              ! if time list is empty    3.01
1238.  000202 1  4                                              ! insert in timer list     3.01
1239.  000202 1  4             @Q := @time^list;                ! otherwise find right spot 3.01
1240.  000204 1  4                                              !                          3.01
1241.  000204 1  4             if queue^any (time^list) then    ! time list not empty      3.01
1242.  000210 1  4             begin                                                       3.01
1243.  000210 1  5                @tcb := time^list.succ;                                  3.01
1244.  000212 1  5                if end^time < tcb.end^time then ! does it belong in front ? 3.01
1245.  000220 1  5                   @Q := @tcb                   ! yes, use first TCB as Q  3.01
1246.  000220 1  5                else begin                      ! no, walk time list from  3.01
1247.  000223 1  6                   @tcb := @time^list;          ! back                     3.01
1248.  000225 1  6                   do                                                     3.01
1249.  000225 1  6                      @tcb := tcb.ready^q.pred                            3.01
1250.  000225 1  6                   until end^time >= tcb.end^time;                        3.01
1251.  000237 1  6                   @Q := tcb.ready^q.succ;      !                     3.01
1252.  000241 1  6                end;                                                    3.01
1253.  000241 1  5             end;                                                       3.01
1254.  000241 1  4             call queue^put (Q, active^thread); ! insert active thread  3.01
1255.  000245 1  4             call thread^awaitio (1);          ! suspend thread         3.01
1256.  000250 1  4          end else
1257.  000251 1  3          begin                                ! no need to wait        3.01
1258.  000251 1  4             call thread^awaitio (1);          ! set next substate to 1 3.05
1259.  000254 1  4             call activate^thread (active^thread); ! and force a task    3.05
1260.  000257 1  4          end;                                 ! dispatch               3.05
1261.  000257 1  3          end;  ! 0
1262.  000261 1 CE1
1263.  000261 1  2          begin ! 1
1264.  000261 1  3             active^thread.state := thread^state^active;                 2.00
1265.  000264 1  3             active^thread.substate := 0;        ! go do next tx        2.00
1266.  000267 1  3          end;  ! 1
1267.  000270 1 CE2
1268.  000270 1  2          otherwise call debug;
1269.  000271 1  2       end; !case
1270.  000304 1  1    end; !of thread^think
```

DLAY^ERROR	Variable	STRING	Direct	P+000
END^TIME	Variable	FIXED (0)	Direct	L+010
NOW	Variable	FIXED (0)	Direct	L+004
Q	Variable,4	STRUCT-I	Indirect	L+003
S^NAME	Variable	STRING	Direct	L+014
TCB	Variable,426	STRUCT-I	Indirect	L+002
TLE^ERROR	Variable	STRING	Direct	P+000
VALUE	Variable	INT	Direct	L+001

```
  1272.    000000 0  0    proc assign^thread (tcb, record);
  1273.    000000 1  0  ! ------------------------------
  1274.    000000 1  0    int .tcb (tcb^^), .record (config^record^^);
  1275.    000000 1  0    begin
  1276.    000000 1  1       int .buf [0:32];
  1277.    000000 1  1
  1278.    000000 1  1       string .s^buf := @buf '<<' 1, .p;
  1279.    000000 1  1
  1280.    000000 1  1       begin
  1281.    000006 1  2          s^buf ':=' record.terminal^name for 16; !$len(config^record^^.t^name)
  1282.    000013 1  2          call fnameexpand (buf, tcb.terminal.name,
  1283.    000013 1  2                            startup^parameter[param^default].name);
  1284.    000023 1  2          tcb.terminal.fnum := -1;
  1285.    000026 1  2          tcb.state         := thread^state^closed;
  1286.    000031 1  2          tcb.substate      := 0; !initial value;
  1287.    000034 1  2          active^threads    := active^threads + 1;
  1288.    000036 1  2       end;
  1289.    000036 1  1    end; !of assign^thread

BUF                             Variable       INT             Indirect       L+001
P                               Variable       STRING          Indirect       L+003
RECORD                          Variable,20    STRUCT-I        Indirect       L-003
S^BUF                           Variable       STRING          Indirect       L+002
TCB                             Variable,426   STRUCT-I        Indirect       L-004

  1290.    000000 0  0
  1291.    000000 0  0    proc thread^awaitio (next^substate);
  1292.    000000 1  0  ! ---------------------------------
  1293.    000000 1  0    int next^substate;
  1294.    000000 1  0    begin
  1295.    000000 1  1       active^thread.substate := next^substate;
  1296.    000003 1  1       active^thread.needs^work := false;
  1297.    000007 1  1    end; ! of thread^awaitio

NEXT^SUBSTATE                   Variable       INT             Direct         L-003

  1298.    000000 0  0
  1299.    000000 0  0    proc activate^thread (tcb);
  1300.    000000 1  0  ! -------------------------
  1301.    000000 1  0    int .tcb (tcb^^);
  1302.    000000 1  0    begin
  1303.    000000 1  1       call queue^put (ready^list, tcb);
  1304.    000004 1  1       if @tcb = @active^thread then
  1305.    000010 1  1          call suspend^thread;
  1306.    000011 1  1    end;

TCB                             Variable,426   STRUCT-I        Indirect       L-003

  1307.    000000 0  0
  1308.    000000 0  0    proc save^thread^state;
  1309.    000000 1  0  ! ---------------------
  1310.    000000 1  0    begin
  1311.    000000 1  1       int call^location = 'L' - 2;
  1312.    000000 1  1
  1313.    000000 1  1       active^thread.last^state    := active^thread.state;
  1314.    000004 1  1       active^thread.last^substate := active^thread.substate;
  1315.    000010 1  1       active^thread.last^loc      := call^location;
```

```
 1316.    000013 1  1     end;

CALL^LOCATION                    Variable       INT             Direct        L-002

 1317.    000000 0  0
 1318.    000000 0  0     proc select^thread;
 1319.    000000 1  0   ! ------------------
 1320.    000000 1  0     begin
 1321.    000000 1  1        @active^thread := queue^get (ready^list);
 1322.    000004 1  1        active^thread.needs^work := true;
 1323.    000010 1  1     end;
 1324.    000000 0  0
 1325.    000000 0  0     proc suspend^thread;
 1326.    000000 1  0   ! -------------------
 1327.    000000 1  0     begin
 1328.    000000 1  1        @active^thread := nil;
 1329.    000002 1  1     end;
 1330.    000000 0  0
 1331.    000000 0  0     proc process^thread^state;
 1332.    000000 1  0   ! ------------------------
 1333.    000000 1  0     begin
 1334.    000000 1  1        case active^thread.state of
 1335.    000003 1  1        begin
 1336.    000003 1  2           call thread^closed;
 1337.    000005 1 CE1          call thread^ready;
 1338.    000007 1 CE2          call thread^active;
 1339.    000011 1 CE3          call thread^stopping;
 1340.    000013 1 CE4          call thread^think;                              2.00
 1341.    000015 1 CE5          otherwise
 1342.    000015 1  2              call debug; !improve ...
 1343.    000016 1  2        end;
 1344.    000034 1  1     end; ! of process^thread^state;
 1345.    000000 0  0
 1346.    000000 0  0     proc process^thread;
 1347.    000000 1  0   ! -------------------
 1348.    000000 1  0     begin
 1349.    000000 1  1        while active^thread.needs^work do
 1350.    000004 1  1           call process^thread^state;
 1351.    000006 1  1        call suspend^thread;
 1352.    000007 1  1     end;
```

```
1354.    000000 0  0     proc init^threads;
1355.    000000 1  0   ! -----------------
1356.    000000 1  0     begin
1357.    000000 1  1        int name [0:11], error, i,
1358.    000000 1  1            maxterm,                                          3.05
1359.    000000 1  1            .tcb (tcb^^),
1360.    000000 1  1            .param   (startup^param^^),
1361.    000000 1  1            .default (startup^param^^) :=
1362.    000000 1  1                  @startup^parameter [param^default].name;
1363.    000000 1  1        int  filecode,                                        2.00
1364.    000000 1  1             priext, secext;                                  2.02
1365.    000000 1  1
1366.    000000 1  1        struct .record (config^record^^);
1367.    000000 1  1        string  badtxlog = 'P' :=                             2.00
1368.    000000 1  1                 ["Wrong type of TXLOG file", 0, %377];       2.00
1369.    000015 1  1
1370.    000015 1  1        subproc calc^ext^size (priext, secext);               2.02
1371.    000015 2  1   !    ---------------------------------------               2.02
1372.    000015 2  1        int .priext, .secext;                                 2.02
1373.    000015 2  1        begin                                                 2.02
1374.    000015 2  2           int rec^per^block, num^block;                      2.02
1375.    000015 2  2           literal block^size = 4096,                         2.02
1376.    000015 2  2                   page^size  = 2048;                         2.02
1377.    000015 2  2                                                              2.02
1378.    000015 2  2           if startup^parameter[ param^tx100 ].setup then      2.02
1379.    000022 2  2           begin                                              2.02
1380.    000022 2  3              rec^per^block := (block^size - 22) / ($len (tx^def) + 4);  2.02
1381.    000024 2  3              num^block := $int (txmax / $dbl (rec^per^block)) + 1;      2.02
1382.    000034 2  3              priext := num^block * 2; !2 pages per block!               2.02
1383.    000037 2  3              if priext <= 60 then                                       2.02
1384.    000042 2  3              begin                                                      2.02
1385.    000042 2  4                 secext := 1;                                            2.02
1386.    000044 2  4                 return;                                                 2.02
1387.    000046 2  4              end;                                                       2.02
1388.    000046 2  3           end;                                                          2.02
1389.    000046 2  2           priext := secext := 60;     !default = 60 pages!           2.02
1390.    000051 2  2        end;                                                          2.02

BLOCK^SIZE                     Literal        INT            %010000
NUM^BLOCK                      Variable       INT            Direct        S-000
PAGE^SIZE                      Literal        INT            %004000
PRIEXT                         Variable       INT            Indirect      S-004
REC^PER^BLOCK                  Variable       INT            Direct        S-001
SECEXT                         Variable       INT            Indirect      S-003

1391.    000053 1  1
1392.    000053 1  1        subproc next^config^record;  ! read next circuit entry from config file
1393.    000053 2  1   !    --------------------------
1394.    000053 2  1        begin
1395.    000053 2  2           call read (config.filenum, record, $len (config^record^^)); !16 bytes!
1396.    000063 2  2           call fileinfo (config.filenum, error);
1397.    000112 2  2        end;
1398.    000113 1  1
1399.    000113 1  1        begin
1400.    000122 1  2           for i := 0 to max^threads-1 do
1401.    000124 1  2           begin  ! initialize threads !
1402.    000124 1  3              threads [i] ':=' ($len(tcb^^)/2) * [0]; !zero out thread!   2.00
```

```
  1316.     000013 1  1     end;

CALL^LOCATION                    Variable       INT             Direct        L-002

  1317.     000000 0  0
  1318.     000000 0  0     proc select^thread;
  1319.     000000 1  0   ! ------------------
  1320.     000000 1  0     begin
  1321.     000000 1  1        @active^thread := queue^get (ready^list);
  1322.     000004 1  1        active^thread.needs^work := true;
  1323.     000010 1  1     end;
  1324.     000000 0  0
  1325.     000000 0  0     proc suspend^thread;
  1326.     000000 1  0   ! -------------------
  1327.     000000 1  0     begin
  1328.     000000 1  1        @active^thread := nil;
  1329.     000002 1  1     end;
  1330.     000000 0  0
  1331.     000000 0  0     proc process^thread^state;
  1332.     000000 1  0   ! -------------------------
  1333.     000000 1  0     begin
  1334.     000000 1  1        case active^thread.state of
  1335.     000003 1  1        begin
  1336.     000003 1  2           call thread^closed;
  1337.     000005 1 CE1          call thread^ready;
  1338.     000007 1 CE2          call thread^active;
  1339.     000011 1 CE3          call thread^stopping;
  1340.     000013 1 CE4          call thread^think;                              2.00
  1341.     000015 1 CE5          otherwise
  1342.     000015 1  2              call debug; !improve ...
  1343.     000016 1  2        end;
  1344.     000034 1  1     end; ! of process^thread^state;
  1345.     000000 0  0
  1346.     000000 0  0     proc process^thread;
  1347.     000000 1  0   ! -------------------
  1348.     000000 1  0     begin
  1349.     000000 1  1        while active^thread.needs^work do
  1350.     000004 1  1           call process^thread^state;
  1351.     000006 1  1        call suspend^thread;
  1352.     000007 1  1     end;
```

```
1354.    000000 0  0     proc init^threads;
1355.    000000 1  0   ! ----------------
1356.    000000 1  0     begin
1357.    000000 1  1        int name [0:11], error, i,
1358.    000000 1  1            maxterm,                                          3.05
1359.    000000 1  1            .tcb (tcb^^),
1360.    000000 1  1            .param   (startup^param^^),
1361.    000000 1  1            .default (startup^param^^) :=
1362.    000000 1  1                  @startup^parameter [param^default].name;
1363.    000000 1  1        int  filecode,                                        2.00
1364.    000000 1  1             priext, secext;                                  2.02
1365.    000000 1  1
1366.    000000 1  1        struct .record (config^record^^);
1367.    000000 1  1        string  badtxlog = 'P' :=                             2.00
1368.    000000 1  1                ["Wrong type of TXLOG file", 0, %377];        2.00
1369.    000015 1  1
1370.    000015 1  1        subproc calc^ext^size (priext, secext);               2.02
1371.    000015 2  1   !    ------------------------------------                  2.02
1372.    000015 2  1        int .priext, .secext;                                 2.02
1373.    000015 2  1        begin                                                 2.02
1374.    000015 2  2           int rec^per^block, num^block;                      2.02
1375.    000015 2  2           literal block^size = 4096,                         2.02
1376.    000015 2  2                   page^size  = 2048;                         2.02
1377.    000015 2  2                                                              2.02
1378.    000015 2  2           if startup^parameter[ param^tx100 ].setup then      2.02
1379.    000022 2  2           begin                                              2.02
1380.    000022 2  3              rec^per^block := (block^size - 22) / ($len (tx^def) + 4);  2.02
1381.    000024 2  3              num^block := $int (txmax / $dbl (rec^per^block)) + 1;      2.02
1382.    000034 2  3              priext := num^block * 2; !2 pages per block!               2.02
1383.    000037 2  3              if priext <= 60 then                                       2.02
1384.    000042 2  3              begin                                                      2.02
1385.    000042 2  4                 secext := 1;                                            2.02
1386.    000044 2  4                 return;                                                 2.02
1387.    000046 2  4              end;                                                       2.02
1388.    000046 2  3           end;                                                          2.02
1389.    000046 2  2           priext := secext := 60;     !default = 60 pages!           2.02
1390.    000051 2  2        end;                                                          2.02

BLOCK^SIZE                     Literal        INT            %010000
NUM^BLOCK                      Variable       INT            Direct        S-000
PAGE^SIZE                      Literal        INT            %004000
PRIEXT                         Variable       INT            Indirect      S-004
REC^PER^BLOCK                  Variable       INT            Direct        S-001
SECEXT                         Variable       INT            Indirect      S-003

1391.    000053 1  1
1392.    000053 1  1        subproc next^config^record;  ! read next circuit entry from config file
1393.    000053 2  1   !    --------------------------
1394.    000053 2  1        begin
1395.    000053 2  2           call read (config.filenum, record, $len (config^record^^)); !16 bytes!
1396.    000063 2  2           call fileinfo (config.filenum, error);
1397.    000112 2  2        end;
1398.    000113 1  1
1399.    000113 1  1        begin
1400.    000122 1  2           for i := 0 to max^threads-1 do
1401.    000124 1  2           begin  ! initialize threads !
1402.    000124 1  3              threads [i] ':=' ($len(tcb^^)/2) * [0]; !zero out thread!   2.00
```

```
 1316.    000013 1  1     end;

CALL^LOCATION                    Variable       INT             Direct         L-002

 1317.    000000 0  0
 1318.    000000 0  0     proc select^thread;
 1319.    000000 1  0   ! ------------------
 1320.    000000 1  0     begin
 1321.    000000 1  1        @active^thread := queue^get (ready^list);
 1322.    000004 1  1        active^thread.needs^work := true;
 1323.    000010 1  1     end;
 1324.    000000 0  0
 1325.    000000 0  0     proc suspend^thread;
 1326.    000000 1  0   ! -------------------
 1327.    000000 1  0     begin
 1328.    000000 1  1        @active^thread := nil;
 1329.    000002 1  1     end;
 1330.    000000 0  0
 1331.    000000 0  0     proc process^thread^state;
 1332.    000000 1  0   ! -------------------------
 1333.    000000 1  0     begin
 1334.    000000 1  1        case active^thread.state of
 1335.    000003 1  1        begin
 1336.    000003 1  2           call thread^closed;
 1337.    000005 1 CE1          call thread^ready;
 1338.    000007 1 CE2          call thread^active;
 1339.    000011 1 CE3          call thread^stopping;
 1340.    000013 1 CE4          call thread^think;                                2.00
 1341.    000015 1 CE5          otherwise
 1342.    000015 1  2              call debug; !improve ...
 1343.    000016 1  2        end;
 1344.    000034 1  1     end; ! of process^thread^state;
 1345.    000000 0  0
 1346.    000000 0  0     proc process^thread;
 1347.    000000 1  0   ! -------------------
 1348.    000000 1  0     begin
 1349.    000000 1  1        while active^thread.needs^work do
 1350.    000004 1  1           call process^thread^state;
 1351.    000006 1  1        call suspend^thread;
 1352.    000007 1  1     end;
```

```
1354.    000000 0  0     proc init^threads;
1355.    000000 1  0   ! -----------------
1356.    000000 1  0     begin
1357.    000000 1  1        int name [0:11], error, i,
1358.    000000 1  1            maxterm,                                          3.05
1359.    000000 1  1            .tcb (tcb^^),
1360.    000000 1  1            .param   (startup^param^^),
1361.    000000 1  1            .default (startup^param^^) :=
1362.    000000 1  1                  @startup^parameter [param^default].name;
1363.    000000 1  1        int  filecode,                                        2.00
1364.    000000 1  1             priext, secext;                                  2.02
1365.    000000 1  1
1366.    000000 1  1        struct .record (config^record^^);
1367.    000000 1  1        string  badtxlog = 'P' :=                             2.00
1368.    000000 1  1                 ["Wrong type of TXLOG file", 0, %377];       2.00
1369.    000015 1  1
1370.    000015 1  1        subproc calc^ext^size (priext, secext);               2.02
1371.    000015 2  1   !    ------------------------------------                  2.02
1372.    000015 2  1        int .priext, .secext;                                 2.02
1373.    000015 2  1        begin                                                 2.02
1374.    000015 2  2           int rec^per^block, num^block;                      2.02
1375.    000015 2  2           literal block^size = 4096,                         2.02
1376.    000015 2  2                   page^size  = 2048;                         2.02
1377.    000015 2  2                                                              2.02
1378.    000015 2  2           if startup^parameter[ param^tx100 ].setup then      2.02
1379.    000022 2  2           begin                                              2.02
1380.    000022 2  3              rec^per^block := (block^size - 22) / ($len (tx^def) + 4);  2.02
1381.    000024 2  3              num^block := $int (txmax / $dbl (rec^per^block)) + 1;      2.02
1382.    000034 2  3              priext := num^block * 2; !2 pages per block!               2.02
1383.    000037 2  3              if priext <= 60 then                                       2.02
1384.    000042 2  3              begin                                                      2.02
1385.    000042 2  4                 secext := 1;                                            2.02
1386.    000044 2  4                 return;                                                 2.02
1387.    000046 2  4              end;                                                       2.02
1388.    000046 2  3           end;                                                          2.02
1389.    000046 2  2           priext := secext := 60;     !default = 60 pages!           2.02
1390.    000051 2  2        end;                                                          2.02

BLOCK^SIZE                     Literal        INT            %010000
NUM^BLOCK                      Variable       INT            Direct        S-000
PAGE^SIZE                      Literal        INT            %004000
PRIEXT                         Variable       INT            Indirect      S-004
REC^PER^BLOCK                  Variable       INT            Direct        S-001
SECEXT                         Variable       INT            Indirect      S-003

1391.    000053 1  1
1392.    000053 1  1        subproc next^config^record;  ! read next circuit entry from config file
1393.    000053 2  1   !    --------------------------
1394.    000053 2  1        begin
1395.    000053 2  2           call read (config.filenum, record, $len (config^record^^)); !16 bytes!
1396.    000063 2  2           call fileinfo (config.filenum, error);
1397.    000112 2  2        end;
1398.    000113 1  1
1399.    000113 1  1        begin
1400.    000122 1  2           for i := 0 to max^threads-1 do
1401.    000124 1  2           begin  ! initialize threads !
1402.    000124 1  3              threads [i] ':=' ($len(tcb^^)/2) * [0]; !zero out thread!   2.00
```

```
 1316.    000013 1  1     end;

CALL^LOCATION                    Variable       INT            Direct         L-002

 1317.    000000 0  0
 1318.    000000 0  0     proc select^thread;
 1319.    000000 1  0   ! ------------------
 1320.    000000 1  0     begin
 1321.    000000 1  1        @active^thread := queue^get (ready^list);
 1322.    000004 1  1        active^thread.needs^work := true;
 1323.    000010 1  1     end;
 1324.    000000 0  0
 1325.    000000 0  0     proc suspend^thread;
 1326.    000000 1  0   ! -------------------
 1327.    000000 1  0     begin
 1328.    000000 1  1        @active^thread := nil;
 1329.    000002 1  1     end;
 1330.    000000 0  0
 1331.    000000 0  0     proc process^thread^state;
 1332.    000000 1  0   ! -------------------------
 1333.    000000 1  0     begin
 1334.    000000 1  1        case active^thread.state of
 1335.    000003 1  1        begin
 1336.    000003 1  2           call thread^closed;
 1337.    000005 1 CE1          call thread^ready;
 1338.    000007 1 CE2          call thread^active;
 1339.    000011 1 CE3          call thread^stopping;
 1340.    000013 1 CE4          call thread^think;                              2.00
 1341.    000015 1 CE5          otherwise
 1342.    000015 1  2              call debug; !improve ...
 1343.    000016 1  2        end;
 1344.    000034 1  1     end; ! of process^thread^state;
 1345.    000000 0  0
 1346.    000000 0  0     proc process^thread;
 1347.    000000 1  0   ! -------------------
 1348.    000000 1  0     begin
 1349.    000000 1  1        while active^thread.needs^work do
 1350.    000004 1  1           call process^thread^state;
 1351.    000006 1  1        call suspend^thread;
 1352.    000007 1  1     end;
```

```
 1354.    000000 0  0     proc init^threads;
 1355.    000000 1  0   ! -----------------
 1356.    000000 1  0     begin
 1357.    000000 1  1        int name [0:11], error, i,
 1358.    000000 1  1            maxterm,                                          3.05
 1359.    000000 1  1            .tcb (tcb^^),
 1360.    000000 1  1            .param   (startup^param^^),
 1361.    000000 1  1            .default (startup^param^^) :=
 1362.    000000 1  1                 @startup^parameter [param^default].name;
 1363.    000000 1  1        int  filecode,                                        2.00
 1364.    000000 1  1             priext, secext;                                  2.02
 1365.    000000 1  1
 1366.    000000 1  1        struct .record (config^record^^);
 1367.    000000 1  1        string  badtxlog = 'P' :=                             2.00
 1368.    000000 1  1                ["Wrong type of TXLOG file", 0, %377];        2.00
 1369.    000015 1  1
 1370.    000015 1  1        subproc calc^ext^size (priext, secext);               2.02
 1371.    000015 2  1   !    ------------------------------------                  2.02
 1372.    000015 2  1        int .priext, .secext;                                 2.02
 1373.    000015 2  1        begin                                                 2.02
 1374.    000015 2  2           int rec^per^block, num^block;                      2.02
 1375.    000015 2  2           literal block^size = 4096,                         2.02
 1376.    000015 2  2                   page^size  = 2048;                         2.02
 1377.    000015 2  2                                                              2.02
 1378.    000015 2  2           if startup^parameter[ param^tx100 ].setup then      2.02
 1379.    000022 2  2           begin                                              2.02
 1380.    000022 2  3              rec^per^block := (block^size - 22) / ($len (tx^def) + 4);  2.02
 1381.    000024 2  3              num^block := $int (txmax / $dbl (rec^per^block)) + 1;      2.02
 1382.    000034 2  3              priext := num^block * 2; !2 pages per block!               2.02
 1383.    000037 2  3              if priext <= 60 then                                       2.02
 1384.    000042 2  3              begin                                                      2.02
 1385.    000042 2  4                 secext := 1;                                            2.02
 1386.    000044 2  4                 return;                                                 2.02
 1387.    000046 2  4              end;                                                       2.02
 1388.    000046 2  3           end;                                                          2.02
 1389.    000046 2  2           priext := secext := 60;     !default = 60 pages!           2.02
 1390.    000051 2  2        end;                                                          2.02

BLOCK^SIZE                     Literal        INT            %010000
NUM^BLOCK                      Variable       INT            Direct         S-000
PAGE^SIZE                      Literal        INT            %004000
PRIEXT                         Variable       INT            Indirect       S-004
REC^PER^BLOCK                  Variable       INT            Direct         S-001
SECEXT                         Variable       INT            Indirect       S-003

 1391.    000053 1  1
 1392.    000053 1  1        subproc next^config^record;  ! read next circuit entry from config file
 1393.    000053 2  1   !    --------------------------
 1394.    000053 2  1        begin
 1395.    000053 2  2           call read (config.filenum, record, $len (config^record^^)); !16 bytes!
 1396.    000063 2  2           call fileinfo (config.filenum, error);
 1397.    000112 2  2        end;
 1398.    000113 1  1
 1399.    000113 1  1        begin
 1400.    000122 1  2           for i := 0 to max^threads-1 do
 1401.    000124 1  2           begin  ! initialize threads !
 1402.    000124 1  3              threads [i] ':=' ($len(tcb^^)/2) * [0]; !zero out thread!  2.00
```

```
 1316.    000013 1  1     end;

CALL^LOCATION                    Variable       INT            Direct        L-002

 1317.    000000 0  0
 1318.    000000 0  0     proc select^thread;
 1319.    000000 1  0   ! ------------------
 1320.    000000 1  0     begin
 1321.    000000 1  1        @active^thread := queue^get (ready^list);
 1322.    000004 1  1        active^thread.needs^work := true;
 1323.    000010 1  1     end;
 1324.    000000 0  0
 1325.    000000 0  0     proc suspend^thread;
 1326.    000000 1  0   ! -------------------
 1327.    000000 1  0     begin
 1328.    000000 1  1        @active^thread := nil;
 1329.    000002 1  1     end;
 1330.    000000 0  0
 1331.    000000 0  0     proc process^thread^state;
 1332.    000000 1  0   ! -------------------------
 1333.    000000 1  0     begin
 1334.    000000 1  1        case active^thread.state of
 1335.    000003 1  1        begin
 1336.    000003 1  2           call thread^closed;
 1337.    000005 1 CE1          call thread^ready;
 1338.    000007 1 CE2          call thread^active;
 1339.    000011 1 CE3          call thread^stopping;
 1340.    000013 1 CE4          call thread^think;                                2.00
 1341.    000015 1 CE5          otherwise
 1342.    000015 1  2              call debug; !improve ...
 1343.    000016 1  2        end;
 1344.    000034 1  1     end; ! of process^thread^state;
 1345.    000000 0  0
 1346.    000000 0  0     proc process^thread;
 1347.    000000 1  0   ! -------------------
 1348.    000000 1  0     begin
 1349.    000000 1  1        while active^thread.needs^work do
 1350.    000004 1  1           call process^thread^state;
 1351.    000006 1  1        call suspend^thread;
 1352.    000007 1  1     end;
```

```
1354.    000000 0  0     proc init^threads;
1355.    000000 1  0   ! -----------------
1356.    000000 1  0     begin
1357.    000000 1  1        int name [0:11], error, i,
1358.    000000 1  1            maxterm,                                          3.05
1359.    000000 1  1            .tcb (tcb^^),
1360.    000000 1  1            .param   (startup^param^^),
1361.    000000 1  1            .default (startup^param^^) :=
1362.    000000 1  1                  @startup^parameter [param^default].name;
1363.    000000 1  1        int  filecode,                                        2.00
1364.    000000 1  1             priext, secext;                                  2.02
1365.    000000 1  1
1366.    000000 1  1        struct .record (config^record^^);
1367.    000000 1  1        string  badtxlog = 'P' :=                             2.00
1368.    000000 1  1                ["Wrong type of TXLOG file", 0, %377];        2.00
1369.    000015 1  1
1370.    000015 1  1        subproc calc^ext^size (priext, secext);               2.02
1371.    000015 2  1   !    ------------------------------------                  2.02
1372.    000015 2  1        int .priext, .secext;                                 2.02
1373.    000015 2  1        begin                                                 2.02
1374.    000015 2  2           int rec^per^block, num^block;                      2.02
1375.    000015 2  2           literal block^size = 4096,                         2.02
1376.    000015 2  2                   page^size  = 2048;                         2.02
1377.    000015 2  2                                                              2.02
1378.    000015 2  2           if startup^parameter[ param^tx100 ].setup then      2.02
1379.    000022 2  2           begin                                              2.02
1380.    000022 2  3              rec^per^block := (block^size - 22) / ($len (tx^def) + 4);  2.02
1381.    000024 2  3              num^block := $int (txmax / $dbl (rec^per^block)) + 1;      2.02
1382.    000034 2  3              priext := num^block * 2; !2 pages per block!               2.02
1383.    000037 2  3              if priext <= 60 then                                       2.02
1384.    000042 2  3              begin                                                      2.02
1385.    000042 2  4                 secext := 1;                                            2.02
1386.    000044 2  4                 return;                                                 2.02
1387.    000046 2  4              end;                                                       2.02
1388.    000046 2  3           end;                                                          2.02
1389.    000046 2  2           priext := secext := 60;     !default = 60 pages!           2.02
1390.    000051 2  2        end;                                                          2.02

BLOCK^SIZE                    Literal        INT            %010000
NUM^BLOCK                     Variable       INT            Direct         S-000
PAGE^SIZE                     Literal        INT            %004000
PRIEXT                        Variable       INT            Indirect       S-004
REC^PER^BLOCK                 Variable       INT            Direct         S-001
SECEXT                        Variable       INT            Indirect       S-003

1391.    000053 1  1
1392.    000053 1  1        subproc next^config^record;  ! read next circuit entry from config file
1393.    000053 2  1   !    --------------------------
1394.    000053 2  1        begin
1395.    000053 2  2           call read (config.filenum, record, $len (config^record^^)); !16 bytes!
1396.    000063 2  2           call fileinfo (config.filenum, error);
1397.    000112 2  2        end;
1398.    000113 1  1
1399.    000113 1  1        begin
1400.    000122 1  2           for i := 0 to max^threads-1 do
1401.    000124 1  2           begin  ! initialize threads !
1402.    000124 1  3              threads [i] ':=' ($len(tcb^^)/2) * [0]; !zero out thread!  2.00
```

```
1403.  000140 1  3            threads [i].terminal.fnum := -1;
1404.  000150 1  3            threads [i]. index := i;
1405.  000160 1  3            threads [i].data^buffer ':=' (buffer^size/2)*["  "];          2.00
1406.  000175 1  3         end;
1407.  000202 1  2
1408.  000202 1  2         @param := @startup^parameter [param^config];
1409.  000205 1  2         if param.setup then
1410.  000211 1  2            config.filename ':=' param.name for 12
1411.  000211 1  2         else
1412.  000216 1  2         begin
1413.  000216 1  3            name ':=' ["CIRCUITS", 8* ["  "]];  ! config^defaultname
1414.  000225 1  3            call fnameexpand (name, config.filename, default);
1415.  000234 1  3         end;
1416.  000234 1  2         call iodev^init (config, open^readonly, 1, guardian^typedisc);
1417.  000243 1  2
1418.  000243 1  2         @param := @startup^parameter [param^script];
1419.  000246 1  2         if param.setup then
1420.  000252 1  2            script.filename ':=' param.name for 12
1421.  000252 1  2         else
1422.  000257 1  2         begin
1423.  000257 1  3            name ':=' ["SCRIPT  ", 8* ["  "]];  ! script^defaultname
1424.  000266 1  3            call fnameexpand (name, script.filename, default);
1425.  000275 1  3         end;
1426.  000275 1  2
1427.  000275 1  2         !open SCRIPT for SBB, waited                                      2.00
1428.  000275 1  2         call open (script.filename, script.filenum, %20,,,,               2.00
1429.  000275 1  2                   !script^buffer!, 4096);                                 2.00
1430.  000307 1  2         if <> then
1431.  000310 1  2         begin
1432.  000310 1  3            call iodev^failure (script.filenum, script.filename,
1433.  000310 1  3                                open^attempt);
1434.  000316 1  3            call abend;
1435.  000323 1  3         end;
1436.  000323 1  2
1437.  000323 1  2         @param := @startup^parameter [param^txlog];                       2.00
1438.  000326 1  2         if param.setup then                                              2.00
1439.  000332 1  2            txlog.filename ':=' param.name for 12                         2.00
1440.  000332 1  2         else                                                             2.00
1441.  000337 1  2         begin                                                            2.00
1442.  000337 1  3            name ':=' ["TXLOG   ", 8* ["  "]];  ! default name            2.00
1443.  000346 1  3            call fnameexpand (name, txlog.filename, default);             2.00
1444.  000355 1  3         end;                                                             2.00
1445.  000355 1  2         ! open statistic file waited. Create it if one does not exist.   2.00
1446.  000355 1  2         call open (txlog.filename, txlog.filenum, open^exclusive);       2.00
1447.  000367 1  2         if <> then                                                       2.00
1448.  000370 1  2         begin                                                            2.00
1449.  000370 1  3            call fileinfo (-1,error);                                     2.00
1450.  000417 1  3            if error = 11 then                                            2.00
1451.  000422 1  3            begin                                                         2.00
1452.  000422 1  4               call calc^ext^size (priext, secext);                       2.02
1453.  000426 1  4               call create ( txlog.filename  !cap=81120 tx for (60,60)    2.00
1454.  000426 1  4                           , priext     ! pri ext                        2.02
1455.  000426 1  4                           , 6211       ! file-code                      2.00
1456.  000426 1  4                           , secext     ! sec ext                        2.02
1457.  000426 1  4                           , 0          ! unstructured                   3.00
1458.  000426 1  4                           ,            ! rec len                        3.00
1459.  000426 1  4                           ,            ! block size                     3.00
```

```
1460.  000426 1  4                          ,          ! ks param                          3.00
1461.  000426 1  4                          ,          ! ak param                          3.00
1462.  000426 1  4                          ,          ! pa param                          3.00
1463.  000426 1  4                          ,          ! max ext                           3.00
1464.  000426 1  4                          , 4096     ! unst buf size                     3.00
1465.  000426 1  4                          , 0        ! buffered writes enabled           2.00
1466.  000426 1  4                         );                                              2.00
1467.  000445 1  4                if <> then call debug;                                   2.00
1468.  000447 1  4                call open (txlog.filename, txlog.filenum, open^exclusive); 2.00
1469.  000461 1  4                if <> then                                               2.00
1470.  000462 1  4                begin                                                    2.00
1471.  000462 1  5                   call iodev^failure (txlog.filenum, txlog.filename,    2.00
1472.  000462 1  5                                       open^attempt);                    2.00
1473.  000470 1  5                   call abend;                                           2.00
1474.  000475 1  5                end;                                                     2.00
1475.  000475 1  4             end; !error 11!                                             2.00
1476.  000475 1  3          end; ! <> !                                                    2.00
1477.  000475 1  2
1478.  000475 1  2          call fileinfo (,,txlog.filename,,,,,,filecode);                2.00
1479.  000523 1  2          if filecode <> 6211 then                                       2.00
1480.  000527 1  2          begin                                                          2.00
1481.  000527 1  3              call log^print (badtxlog);                                 2.00
1482.  000547 1  3              call abend;                                                2.00
1483.  000554 1  3          end;                                                           2.00
1484.  000554 1  2          call control (txlog.filenum, 20); ! purgedata                  2.00
1485.  000563 1  2
1486.  000563 1  2          txlog.bufaddr := allocate^buffer (txstat^bufsize);             3.00
1487.  000567 1  2          @txstat := txlog.bufaddr;                                      3.00
1488.  000571 1  2          txstat^recs := 0;                                              3.00
1489.  000573 1  2          if txstat^numrecs <> 128 then call debug; ! just in case       3.00
1490.  000576 1  2
1491.  000576 1  2          ! Read configuration file to intialize terminals (threads)     2.00
1492.  000576 1  2          call position (config.filenum, 0d);
1493.  000602 1  2          call next^config^record;
1494.  000603 1  2
1495.  000603 1  2          ! The next enhancement allows to reduce the number of active   3.05
1496.  000603 1  2          ! terminals without changing the CONFIG files; the number of   3.05
1497.  000603 1  2          ! active terminals is limited to MAXTERM even if the CONFIG    3.05
1498.  000603 1  2          ! file contains more entries.                                  3.05
1499.  000603 1  2          ! It also protects us from a CONFIG file that has more than    3.05
1500.  000603 1  2          ! MAX^THREADS entries.                                         3.05
1501.  000603 1  2
1502.  000603 1  2          @param := @startup^parameter [param^maxterm];                   3.05
1503.  000606 1  2          if param.setup and                                             3.05
1504.  000606 1  2             ((maxterm := param.value) <= max^threads) and               3.05
1505.  000606 1  2             (maxterm > 0) then ! ok, maxterm set up correct             3.05
1506.  000621 1  2          else maxterm := max^threads;                                   3.05
1507.  000631 1  2
1508.  000631 1  2          while not error and active^threads < maxterm do                3.05
1509.  000637 1  2          begin
1510.  000637 1  3             @tcb := @threads [next^thread];
1511.  000646 1  3             call assign^thread (tcb, record);
1512.  000652 1  3             call activate^thread (tcb);
1513.  000655 1  3             next^thread := next^thread + 1;
1514.  000657 1  3             call next^config^record;
1515.  000660 1  3          end;  ! while
1516.  000661 1  2
```

```
 1517.   000661 1  2          thread^count := active^threads;                                  2.00
 1518.   000663 1  2
 1519.   000663 1  2          ! Mean inter-arrival time per thread in milliseconds             2.00
 1520.   000663 1  2          ! remember rate is tps scaled up by 10                           2.00
 1521.   000663 1  2          mean^per^thread := if thread^count = 0 or rate = 0 then 0d else  2.00
 1522.   000663 1  2            $dbl( 1e3 / ($flt(rate) / (10e0 * $flt(thread^count))));       2.00
 1523.   000710 1  2
 1524.   000710 1  2          @param := @startup^parameter [param^maxtx];                       3.05
 1525.   000713 1  2          if param.setup and                                               3.05
 1526.   000713 1  2             (max^tx^out := param.value) > 0 then !ok!                     3.05
 1527.   000723 1  2          else                                                             3.05
 1528.   000726 1  2             max^tx^out := max^threads + 1;                                3.05
 1529.   000730 1  2
 1530.   000730 1  2       end;  ! of proc body
 1531.   000730 1  1    end; !of init^threads
```

BADTXLOG	Variable	STRING	Direct	P+000
CALC^EXT^SIZE	Subproc		%000015	
DEFAULT	Variable,32	STRUCT-I	Indirect	L+022
ERROR	Variable	INT	Direct	L+015
FILECODE	Variable	INT	Direct	L+023
I	Variable	INT	Direct	L+016
MAXTERM	Variable	INT	Direct	L+017
NAME	Variable	INT	Direct	L+001
NEXT^CONFIG^RECORD	Subproc		%000053	
PARAM	Variable,32	STRUCT-I	Indirect	L+021
PRIEXT	Variable	INT	Direct	L+024
RECORD	Variable,20	STRUCT	Indirect	L+026
SECEXT	Variable	INT	Direct	L+025
TCB	Variable,426	STRUCT-I	Indirect	L+020

```
1533.    000000 0  0     int (32) proc build^iotag (tcb, file^type) variable;
1534.    000000 1  0   ! -----------------------------------------------
1535.    000000 1  0     int .tcb (tcb^^), file^type;
1536.    000000 1  0     begin
1537.    000000 1  1        int (32) tag := 0d;
1538.    000000 1  1        int high = tag, low = tag+1;
1539.    000000 1  1
1540.    000000 1  1        begin
1541.    000002 1  2           high := nil;
1542.    000004 1  2           if $param (tcb) then
1543.    000007 1  2              high := @tcb;
1544.    000011 1  2           if $param (file^type) then
1545.    000014 1  2              low.<8:15> := file^type;
1546.    000020 1  2        end;
1547.    000020 1  1        return tag;
1548.    000022 1  1     end; !of proc build^iotag;

FILE^TYPE                      Variable       INT             Direct         L-004
HIGH                           Variable       INT             Direct         L+001
LOW                            Variable       INT             Direct         L+002
TAG                            Variable       INT (32)        Direct         L+001
TCB                            Variable,426   STRUCT-I        Indirect       L-005

1549.    000000 0  0
1550.    000000 0  0     Proc iodev^failure (f, name, operation) variable;
1551.    000000 1  0   ! -----------------------------------------------
1552.    000000 1  0     int f, .name, operation;
1553.    000000 1  0     begin
1554.    000000 1  1        int error, convert := true, 1, f^name [0:11];
1555.    000000 1  1        string s^name [0:34],
1556.    000000 1  1               s^oper [0:30],
1557.    000000 1  1               io^failure = 'P' :=
1558.    000000 1  1                   [20*["\"], " with error ?#### on file ", 34*["\"],
1559.    000050 1  1                    ", file no: ?###", 0, %377];
1560.    000061 1  1        begin
1561.    000065 1  2           call fileinfo (f, error, f^name);
1562.    000114 1  2           if f = -1 then
1563.    000117 1  2           begin
1564.    000117 1  3              s^name ':=' "No file name" & 0;
1565.    000137 1  3              convert := false;
1566.    000141 1  3           end;
1567.    000141 1  2           if $param (name) then
1568.    000144 1  2           begin
1569.    000144 1  3              f^name ':=' name for 12;
1570.    000150 1  3              convert := true;
1571.    000152 1  3           end;
1572.    000152 1  2           if convert then s^name [fnamecollapse (f^name, s^name)] := 0;
1573.    000164 1  2           case operation of
1574.    000166 1  2           begin
1575.    000166 1  3              s^oper ':=' ["OPEN failed", 0];
1576.    000200 1 CE1             s^oper ':=' ["READ failed", 0];
1577.    000212 1 CE2             s^oper ':=' ["WRITE failed", 0];
1578.    000224 1 CE3             s^oper ':=' ["CONTROL failed", 0];
1579.    000236 1 CE4             s^oper ':=' ["AWAITIO completion ", 0];
1580.    000250 1 CE5          end; !case
1581.    000256 1  2           call log^print (io^failure, @s^oper, error, @s^name, f);
1582.    000302 1  2        end;
```

```
  1583.    000302 1  1     end;

CONVERT                          Variable       INT            Direct         L+002
ERROR                            Variable       INT            Direct         L+001
F                                Variable       INT            Direct         L-006
F^NAME                           Variable       INT            Direct         L+004
IO^FAILURE                       Variable       STRING         Direct         P+000
L                                Variable       INT            Direct         L+003
NAME                             Variable       INT            Indirect       L-005
OPERATION                        Variable       INT            Direct         L-004
S^NAME                           Variable       STRING         Direct         L+020
S^OPER                           Variable       STRING         Direct         L+042

  1584.    000000 0  0
  1585.    000000 0  0     proc iodev^init (dev,flags, sync, type) variable;
  1586.    000000 1  0   ! -----------------------------------------------
  1587.    000000 1  0     int .dev (iodev^^), flags, sync, type;
  1588.    000000 1  0     begin
  1589.    000000 1  1        string s^name [0:34],
  1590.    000000 1  1               wrong^type = 'P' :=
  1591.    000000 1  1     ["Device type for file ", 34*["\"], " is ?###, should be ?###", 0,%377];
  1592.    000051 1  1        int i;
  1593.    000051 1  1
  1594.    000051 1  1        begin
  1595.    000052 1  2           call open (dev.filename, dev.filenum, flags, sync);
  1596.    000065 1  2           iodev^check (dev.filenum, dev.filename, open^attempt);
  1597.    000075 1  2           call fileinfo (dev.filenum, dev.error,,, dev.type);
  1598.    000133 1  2           if (not dev.error) and $param (type) and
  1599.    000133 1  2              (i := dev.type.guardian^devtype) <> type then
  1600.    000151 1  2           begin
  1601.    000151 1  3              s^name [fnamecollapse (dev.filename, s^name)] := 0;
  1602.    000161 1  3              call log^print (wrong^type, @s^name, i, type);
  1603.    000202 1  3              call close (dev.filenum);
  1604.    000210 1  3              dev.filenum := -1;
  1605.    000213 1  3              dev.error := 16;
  1606.    000216 1  3           end;
  1607.    000216 1  2        end; ! of proc body
  1608.    000216 1  1     end;    ! of proc iodev^init

DEV                              Variable,40    STRUCT-I       Indirect       L-007
FLAGS                            Variable       INT            Direct         L-006
I                                Variable       INT            Direct         L+023
SYNC                             Variable       INT            Direct         L-005
S^NAME                           Variable       STRING         Direct         L+001
TYPE                             Variable       INT            Direct         L-004
WRONG^TYPE                       Variable       STRING         Direct         P+000
```

```
1610.  000000 0  0     proc io^wait;
1611.  000000 1  0   ! -------------
1612.  000000 1  0     begin
1613.  000000 1  1        int (32) tag, delay;
1614.  000000 1  1        int      fnum, error, count, my^filetype, done, buf,
1615.  000000 1  1                 high^tag = tag, low^tag = tag+1,
1616.  000000 1  1                 .tcb (tcb^^) = tag;
1617.  000000 1  1
1618.  000000 1  1        subproc receive^completion;
1619.  000000 2  1   !    ---------------------------
1620.  000000 2  1        begin
1621.  000000 2  2           int (32) tag;
1622.  000000 2  2           int      .tcb (tcb^^),
1623.  000000 2  2                    .buffer := receive.bufaddr;
1624.  000000 2  2
1625.  000000 2  2           begin
1626.  000003 2  3              if error = 6 then ! system msg received
1627.  000006 2  3              begin
1628.  000006 2  4                 if buffer = sysmsg^time then
1629.  000011 2  4                 begin
1630.  000011 2  5                    @buffer := receive.bufaddr;
1631.  000013 2  5                    @tcb := buffer [1];
1632.  000016 2  5                    call activate^thread (tcb);
1633.  000021 2  5                    tag := build^iotag (, receive^file^);
1634.  000026 2  5                    call reply (buffer);
1635.  000034 2  5                    if <> then call debug;
1636.  000036 2  5                    call readupdate (receive.filenum, buffer, 256,, tag);    3.08
1637.  000046 2  5                    if <> then call debug;
1638.  000050 2  5                 end;
1639.  000050 2  4              end else ! user request received, must be interactive traffic
1640.  000051 2  3              begin
1641.  000051 2  4                 if inter^active^data then ! already some request pending ?? 3.08
1642.  000053 2  4                    call debug;
1643.  000054 2  4                 inter^active^data := true;
1644.  000056 2  4
1645.  000056 2  4                 if queue^any (suspend^list) then
1646.  000062 2  4                    call activate^thread (queue^get (suspend^list))
1647.  000067 2  4                 else
1648.  000070 2  4
1649.  000070 2  4                    if queue^any (time^list) then
1650.  000074 2  4                       call activate^thread (queue^get (time^list));
1651.  000101 2  4              end;
1652.  000101 2  3           end; ! of proc body
1653.  000101 2  2        end;    ! of subproc

BUFFER                        Variable       INT             Indirect       S-000
TAG                           Variable       INT (32)        Direct         S-003
TCB                           Variable,426   STRUCT-I        Indirect       S-001

1654.  000103 1  1
1655.  000103 1  1        begin   ! of body of IO^WAIT
1656.  000104 1  2           delay := -1d;
1657.  000106 1  2
1658.  000106 1  2           if queue^any (time^list) then   ! are there any waiting threads   3.01
1659.  000112 1  2           begin                           ! use its wait time as delay      3.01
1660.  000112 1  3              @tcb := time^list.succ;      ! for AWAITIO                     3.01
1661.  000114 1  3              delay := $dbl (tcb.end^time - (juliantimestamp / 10000f));     3.01
```

```
1662.  000136 1  3            if delay <= 0D then          ! time has elapsed already ??     3.01
1663.  000142 1  3            begin                                                          3.01
1664.  000142 1  4               call activate^thread (queue^get (time^list));               3.01
1665.  000147 1  4               return;                   ! continue with that thread       3.01
1666.  000150 1  4            end;                                                           3.01
1667.  000150 1  3         end;                                                              3.01
1668.  000150 1  2
1669.  000150 1  2         if $switches.<4> then
1670.  000153 1  2         begin
1671.  000153 1  3            stack (tx^outstanding);      ! show # tx out in front panel    3.05
1672.  000154 1  3            code (ssw);
1673.  000155 1  3         end;
1674.  000155 1  2
1675.  000155 1  2         fnum := -1;
1676.  000157 1  2         call awaitio (fnum, buf, count, tag, delay);
1677.  000167 1  2         call fileinfo (fnum, error);
1678.  000216 1  2
1679.  000216 1  2         if error = 30 !No LCB! then call abend;                           3.04
1680.  000226 1  2
1681.  000226 1  2         my^filetype := low^tag.<8:15>;
1682.  000231 1  2         if fnum = -1 then                                                 3.01
1683.  000234 1  2            if error <> 40 then          ! not a timeout, very strange     3.01
1684.  000237 1  2               call debug                                                 3.01
1685.  000237 1  2            else                                                          3.01
1686.  000241 1  2            begin                                                         3.01
1687.  000241 1  3               if queue^empty (time^list) then call debug; ! ????         3.01
1688.  000246 1  3               call activate^thread (queue^get (time^list));               3.01
1689.  000253 1  3               return;                                                    3.01
1690.  000255 1  3            end                                                           3.01
1691.  000255 1  2         else
1692.  000256 1  2         begin
1693.  000256 1  3            if @tcb <> nil then
1694.  000261 1  3            begin    ! ---- io for a thread completed
1695.  000261 1  4               tcb.recv^timestamp := juliantimestamp;                     2.00
1696.  000272 1  4               tcb.last^fnum  := fnum;
1697.  000275 1  4               tcb.last^error := error;
1698.  000300 1  4               tcb.last^count := count;
1699.  000303 1  4               call activate^thread (tcb);
1700.  000306 1  4            end else
1701.  000307 1  3            begin    ! ---- some other file completed
1702.  000307 1  4               case my^filetype of
1703.  000311 1  4               begin
1704.  000311 1  5                  begin  ! 0, $receive
1705.  000311 1 CE0                  call receive^completion;
1706.  000312 1  6                  end;
1707.  000313 1 CE1
1708.  000313 1  5                  begin  ! 1, config file
1709.  000313 1  6                  end;
1710.  000313 1 CE2
1711.  000313 1  5                  begin  ! 2, logfile
1712.  000313 1  6                     call logfile^iocomplete (error);
1713.  000316 1  6                  end;
1714.  000317 1 CE3
1715.  000317 1  5                  begin  ! 3, trace file
1716.  000317 1  6                  end;
1717.  000317 1 CE4
1718.  000317 1  5                  begin  ! 4, home terminal
```

```
 1719.    000317 1  6                    end;
 1720.    000317 1 CE5
 1721.    000317 1  5                    begin  ! 5, terminal IO, should never happen here
 1722.    000317 1  6                       call debug;
 1723.    000320 1  6                    end;
 1724.    000321 1 CE6
 1725.    000321 1  5                    begin  ! 6, server IO, should never happen here
 1726.    000321 1  6                       call debug;
 1727.    000322 1  6                    end;
 1728.    000323 1 CE7
 1729.    000323 1  5                    otherwise call debug;
 1730.    000324 1  5                 end; ! of case my^filetype
 1731.    000344 1  4              end;    ! of else other file complete
 1732.    000344 1  3           end;       ! of else for fnum - 1
 1733.    000344 1  2        end;             ! of main body
 1734.    000344 1  1     end;                ! of proc io^wait
```

BUF	Variable	INT	Direct	L+012
COUNT	Variable	INT	Direct	L+007
DELAY	Variable	INT (32)	Direct	L+003
DONE	Variable	INT	Direct	L+011
ERROR	Variable	INT	Direct	L+006
FNUM	Variable	INT	Direct	L+005
HIGH^TAG	Variable	INT	Direct	L+001
LOW^TAG	Variable	INT	Direct	L+002
MY^FILETYPE	Variable	INT	Direct	L+010
RECEIVE^COMPLETION	Subproc		%000000	
TAG	Variable	INT (32)	Direct	L+001
TCB	Variable,426	STRUCT-I	Indirect	L+001

```
1736.    000000 0  0     Proc loadgen^init;
1737.    000000 1  0   ! ----------------
1738.    000000 1  0     begin
1739.    000000 1  1        struct .st^msg (startup^msg^^);
1740.    000000 1  1        int (32) tag;
1741.    000000 1  1        int      fnum,
1742.    000000 1  1                 error,
1743.    000000 1  1                 .buffer;
1744.    000000 1  1
1745.    000000 1  1        string r^error = 'P' :=
1746.    000000 1  1                 ["Unable to read startup message, error ?####", 0, %377],
1747.    000027 1  1               ignore = 'P' :=
1748.    000027 1  1                 ["Parameter ignored: ", 40* ["\"], 0, %377],
1749.    000066 1  1               bad^arrival = 'P' :=
1750.    000066 1  1                 ["Unknown arrival type. Allowed type = ",
1751.    000111 1  1                  "{Maximum,Constant,Random}", 0, %377];
1752.    000126 1  1
1753.    000126 1  1        subproc upshift (buf, l);
1754.    000126 2  1   !    ------------------------
1755.    000126 2  1        string .buf; int l;
1756.    000126 2  1        begin
1757.    000126 2  2           use x;
1758.    000126 2  2           for x := 0 to l-1 do
1759.    000133 2  2              if $alpha (buf [x]) then buf[x] := buf[x] land %337;
1760.    000142 2  2           drop x;
1761.    000142 2  2        end;

BUF                               Variable       STRING          Indirect       S-002
L                                 Variable       INT             Direct         S-001

1762.    000143 1  1
1763.    000143 1  1        subproc scan^startup;
1764.    000143 2  1   !    --------------------
1765.    000143 2  1        begin
1766.    000143 2  2           string .comma, .last, .first, .next, .save, save^char;
1767.    000143 2  2           int i, status, found, command^length, input^length, index,
1768.    000143 2  2               .p (startup^param^^);
1769.    000143 2  2
1770.    000143 2  2           begin
1771.    000144 2  3              for i:= 0 to max^keyword do startup^parameter [i].flags := 0;
1772.    000163 2  3              startup^parameter [param^default].name':='st^msg.default for 8;
1773.    000170 2  3
1774.    000170 2  3              @next := @st^msg.parm;
1775.    000174 2  3              if not next then call print^help; ! then exit !                2.00
1776.    000177 2  3              while next do
1777.    000201 2  3              begin
1778.    000201 2  4                 scan next  until "," -> @comma;
1779.    000205 2  4                 scan next  while " " -> @first;
1780.    000211 2  4                 scan first until " " -> @last;
1781.    000215 2  4                 @save := @first;
1782.    000217 2  4                 @last := $min (@last, @comma);
1783.    000226 2  4                 input^length := @last - @first;
1784.    000232 2  4
1785.    000232 2  4                 call upshift (first, input^length);
1786.    000236 2  4                 i := 0;
1787.    000240 2  4                 found := false;
1788.    000242 2  4
```

```
1789.    000242 2  4              while not found and (command^length := param^keywords[i]) do
1790.    000252 2  4              begin
1791.    000252 2  5                 if first = param^keywords [i+2] for input^length and
1792.    000252 2  5                    input^length = command^length then
1793.    000270 2  5                 begin
1794.    000270 2  6                    found := true;
1795.    000272 2  6                    scan last while " " -> @first;
1796.    000276 2  6                    rscan comma [-1] while " " -> @last;
1797.    000303 2  6                    input^length := $max (0, @last - @first + 1);
1798.    000315 2  6                    @p := @startup^parameter [i/11 + 1];
1799.    000327 2  6
1800.    000327 2  6                    case param^keywords [i+1] of
1801.    000336 2  6                    begin
1802.    000336 2  7                       !0 name^ !
1803.    000336 2  7                       begin  ! a device name must be supplied
1804.    000336 2 CE0                         call upshift (first, input^length);
1805.    000342 2  8                          if first = "#MYTERM" !for 7! then
1806.    000353 2  8                             call myterm (p.name)
1807.    000356 2  8                          else
1808.    000360 2  8                             if input^length <>
1809.    000360 2  8                               fnameexpand (first, p.name, st^msg.default)
1810.    000360 2  8                             then found := false;
1811.    000375 2  8                       end;
1812.    000376 2 CE1                      !1 const^!
1813.    000376 2  7                       begin  ! a numeric parameter must be supplied
1814.    000376 2  8                          if numin (first, p.value, 10, status)
1815.    000376 2  8                             <> @last + 1 or status
1816.    000376 2  8                          then found := false;
1817.    000414 2  8                       end;
1818.    000415 2 CE2                      !2 none^ !
1819.    000415 2  7                       begin  ! no parameter
1820.    000415 2  8                          if first and first <> "," then found := false;
1821.    000424 2  8                       end;
1822.    000426 2 CE3                      !3 str^  !
1823.    000426 2  7                       begin  ! string of 24 char max                    2.00
1824.    000426 2  8                          p.name ':=' 12*["   "];                         2.00
1825.    000435 2  8                          if input^length then                           2.00
1826.    000437 2  8                             p.s^name ':=' first for                     2.00
1827.    000437 2  8                                          $min(input^length, 24)         2.00
1828.    000437 2  8                          else found := false;                           2.00
1829.    000455 2  8                       end;                                              2.00
1830.    000456 2 CE4                      otherwise found := false;
1831.    000460 2  7                    end; !case
1832.    000475 2  6                    if found then p.setup := true;
1833.    000503 2  6                 end; !
1834.    000503 2  5                 i := i + 11;
1835.    000505 2  5              end; !of while not found
1836.    000506 2  4              if not found then
1837.    000510 2  4              begin
1838.    000510 2  5                 save^char := comma;
1839.    000512 2  5                 comma := 0;
1840.    000514 2  5                 call log^print (ignore, @save);
1841.    000534 2  5                 comma := save^char;
1842.    000536 2  5              end;
1843.    000541 2  4              @next := if comma then @comma + 1 else @comma;
1844.    000550 2  4           end; !while next do
1845.    000551 2  3
```

```
  1846.    000551 2  3            !--------------------------!                                          2.00
  1847.    000551 2  3            !  Process Startup Params  !                                          2.00
  1848.    000551 2  3            !--------------------------!                                          2.00
  1849.    000551 2  3            ! initialize ARRIVAL^TYPE                                             2.00
  1850.    000551 2  3            @p := @startup^parameter[ param^arrival ];                             2.00
  1851.    000554 2  3            call upshift (p.s^name, 24);                                          2.00
  1852.    000561 2  3            if p.setup then                                                       2.00
  1853.    000565 2  3               if      p.s^name = "R" then arrival^type := 2  ! Random            2.00
  1854.    000572 2  3               else if p.s^name = "C" then arrival^type := 1  ! Constant          2.00
  1855.    000602 2  3               else if p.s^name = "M" then arrival^type := 0  ! Maximum           2.00
  1856.    000612 2  3               else begin                                                         2.00
  1857.    000615 2  4                  call log^print (bad^arrival);                                   2.00
  1858.    000635 2  4                  call abend;                                                     2.00
  1859.    000642 2  4               end;                                                               2.00
  1860.    000642 2  3            ! initialize with ARRIVAL^RATE                                        2.00
  1861.    000642 2  3            @p := @startup^parameter[ param^txrate ];                              2.00
  1862.    000645 2  3            if p.setup then rate := $abs (p.value);                               2.00
  1863.    000655 2  3            ! ARRIVAL=max takes precedence over TXRATE                            2.00
  1864.    000655 2  3            if not arrival^type then rate := 0; !max rate!                        2.00
  1865.    000661 2  3            ! initialize with TX100 (max number of replayed tx's)                 2.02
  1866.    000661 2  3            @p := @startup^parameter[ param^tx100  ];                              2.02
  1867.    000664 2  3            if p.setup then                                                       2.02
  1868.    000670 2  3               txmax := $dbl ($abs (p.value)) * 100d                              2.02
  1869.    000670 2  3            else                                                                  2.02
  1870.    000704 2  3               txmax := 2000000d;  ! a huge number !                              2.02
  1871.    000707 2  3            ! initialize with startup DELAY between terminals                     2.03
  1872.    000707 2  3            @p := @startup^parameter[ param^delay  ];                              2.03
  1873.    000712 2  3            if p.setup then                                                       2.03
  1874.    000716 2  3               start^delay := $abs (p.value)                                      2.03
  1875.    000716 2  3            else                                                                  2.03
  1876.    000725 2  3               start^delay := 50;  !500 ms!                                       2.03
  1877.    000727 2  3            ! initialize with RN seed                                             3.06
  1878.    000727 2  3            @p := @startup^parameter[ param^rnseed ];                              3.06
  1879.    000732 2  3            if p.setup then                                                       3.06
  1880.    000736 2  3               if p.value > 0 and p.value < 41 then                               3.07
  1881.    000744 2  3                  seed1 := seed [p.value - 1];                                    3.06
  1882.    000751 2  3
  1883.    000751 2  3         end; !of setup body
  1884.    000751 2  2      end; ! of subproc scan^startup;

COMMA                          Variable       STRING         Indirect       S-014
COMMAND^LENGTH                 Variable       INT            Direct         S-003
FIRST                          Variable       STRING         Indirect       S-012
FOUND                          Variable       INT            Direct         S-004
I                              Variable       INT            Direct         S-006
INDEX                          Variable       INT            Direct         S-001
INPUT^LENGTH                   Variable       INT            Direct         S-002
LAST                           Variable       STRING         Indirect       S-013
NEXT                           Variable       STRING         Indirect       S-011
P                              Variable,32    STRUCT-I       Indirect       S-000
SAVE                           Variable       STRING         Indirect       S-010
SAVE^CHAR                      Variable       STRING         Direct         S-007
STATUS                         Variable       INT            Direct         S-005

  1885.    000773 1  1
  1886.    000773 1  1      !--------------------------!                                                2.00
  1887.    000773 1  1      !  Body of LOADGEN^INIT    !                                                2.00
```

```
 1888.    000773 1  1       !--------------------------!                                          2.00
 1889.    000773 1  1       begin !body of proc loadgen^init
 1890.    001001 1  2          receive.filename ':=' "$receive                 ";
 1891.    001010 1  2          call open (receive.filename, receive.filenum, open^nowait, 1);    3.08
 1892.    001022 1  2          call myterm (hometerm.filename);
 1893.    001025 1  2          call open (hometerm.filename, hometerm.filenum);
 1894.    001036 1  2
 1895.    001036 1  2          call read (receive.filenum, st^msg, $len (startup^msg^^));
 1896.    001046 1  2          call awaitio (receive.filenum,,,, 3000d);
 1897.    001055 1  2          call fileinfo (receive.filenum, error);
 1898.    001104 1  2          if error or st^msg.id <> sysmsg^startup then
 1899.    001111 1  2             call log^print (r^error, error);
 1900.    001131 1  2
 1901.    001131 1  2          call scan^startup;
 1902.    001132 1  2
 1903.    001132 1  2          @buffer := receive.bufaddr := @receive^buffer;                     2.00
 1904.    001135 1  2          tag := build^iotag (,receive^file^);
 1905.    001142 1  2          call readupdate (receive.filenum, buffer, 256,, tag);
 1906.    001152 1  2       end;
 1907.    001152 1  1    end;  ! body of proc loadgen^init
```

```
BAD^ARRIVAL                    Variable        STRING          Direct          P+000
BUFFER                         Variable        INT             Indirect        L+006
ERROR                          Variable        INT             Direct          L+005
FNUM                           Variable        INT             Direct          L+004
IGNORE                         Variable        STRING          Direct          P+000
R^ERROR                        Variable        STRING          Direct          P+000
SCAN^STARTUP                   Subproc                         %000143
ST^MSG                         Variable,1122   STRUCT          Indirect        L+001
TAG                            Variable        INT (32)        Direct          L+002
UPSHIFT                        Subproc                         %000126
```

```
1909.  000000 0  0    proc print^help;                                                    2.00
1910.  000000 1  0  ! ----------------
1911.  000000 1  0    begin
1912.  000000 1  1        string  .banner [0:59];                                        3.03
1913.  000000 1  1        string   h1 = 'P' := [60*["\"],0,%377],                        3.03
1914.  000037 1  1                 h2 = 'P' := [" ",0,%377],
1915.  000041 1  1                 h3 = 'P' := ["  ",
1916.  000042 1  1  "TXGEN [LOG      <filename>,]       -- Error File",
1917.  000072 1  1  0,%377],
1918.  000073 1  1                 h4 = 'P' := ["  ",
1919.  000074 1  1  "      [CONFIG   <filename>,]       -- X25 PVC Definition",
1920.  000130 1  1  0,%377],
1921.  000131 1  1                 h5 = 'P' := ["  ",
1922.  000132 1  1  "      [SCRIPT   <filename>,]       -- ET1 transaction script to replay",
1923.  000175 1  1  0,%377],
1924.  000176 1  1                 h6 = 'P' := ["  ",
1925.  000177 1  1  "      [ARRIVAL {Maximum|Constant|Random},] -- Arrival Process",
1926.  000236 1  1  0,%377],
1927.  000237 1  1                 h7 = 'P' := ["  ",
1928.  000240 1  1  "      [TXRATE   <rate*10> ,]       -- 10 * Arrival Rate (in tps)",
1929.  000300 1  1  0,%377],
1930.  000301 1  1                 h8 = 'P' := ["  ",
1931.  000302 1  1  "      [TXLOG    <filename>,]       -- Tx capture file",
1932.  000335 1  1  0,%377],
1933.  000336 1  1                 h9 = 'P' := ["  ",                                   2.02
1934.  000337 1  1  "      [TX100   <count/100>,]       -- Hundreds of Tx to be replayed",  2.02
1935.  000401 1  1  0,%377],                                                            2.02
1936.  000402 1  1                 h10= 'P' := ["  ",                                   2.03
1937.  000403 1  1  "      [DELAY   <unit=.01s>,]       -- Startup delay between terminals", 2.03
1938.  000446 1  1  0,%377],                                                            2.03
1939.  000447 1  1                 h11= 'P' := ["  ",                                   2.03
1940.  000450 1  1  "      [MAXTERM  <n>        ,]      -- Max. number of active terminals", 2.03
1941.  000513 1  1  0,%377],                                                            2.03
1942.  000514 1  1                 h12= 'P' := ["  ",                                   2.03
1943.  000515 1  1  "      [MAXTX    <n>        ,]      -- Max. number of concurrent TX's", 2.03
1944.  000557 1  1  0,%377],                                                            2.03
1945.  000560 1  1                 h13= 'P' := ["  ",                                   3.06
1946.  000561 1  1  "      [RNSEED    [1:40]     ]      -- Random number seed index      ", 3.07
1947.  000623 1  1  0,%377];                                                           3.06
1948.  000624 1  1
1949.  000624 1  1        startup := 2;
1950.  000632 1  1        log^enable := true;
1951.  000634 1  1        banner ':=' "TXGEN - ET1 Load Generator (" &                   3.03
1952.  000634 1  1                    txgen^vs & ") - PAM";                              3.03
1953.  000662 1  1        call log^print (h1, @banner);                                  3.03
1954.  000702 1  1        call log^print (h2);
1955.  000722 1  1        call log^print (h3);
1956.  000742 1  1        call log^print (h4);
1957.  000762 1  1        call log^print (h5);
1958.  001002 1  1        call log^print (h6);
1959.  001022 1  1        call log^print (h7);
1960.  001042 1  1        call log^print (h8);
1961.  001062 1  1        call log^print (h9);
1962.  001113 1  1        call log^print (h10);
1963.  001133 1  1        call log^print (h11);
1964.  001153 1  1        call log^print (h12);
1965.  001173 1  1        call log^print (h13);
```

```
  1966.    001213 1  1          call log^print (h2);
  1967.    001233 1  1
  1968.    001233 1  1          call stop;
  1969.    001240 1  1      end; !proc print^help!
```

BANNER	Variable	STRING	Indirect	L+001
H1	Variable	STRING	Direct	P+000
H10	Variable	STRING	Direct	P+000
H11	Variable	STRING	Direct	P+000
H12	Variable	STRING	Direct	P+000
H13	Variable	STRING	Direct	P+000
H2	Variable	STRING	Direct	P+000
H3	Variable	STRING	Direct	P+000
H4	Variable	STRING	Direct	P+000
H5	Variable	STRING	Direct	P+000
H6	Variable	STRING	Direct	P+000
H7	Variable	STRING	Direct	P+000
H8	Variable	STRING	Direct	P+000
H9	Variable	STRING	Direct	P+000

```
1971.    000000 0  0    proc print^times (ts^start, ts^stop);                                    2.00
1972.    000000 1  0  ! -------------------------------------
1973.    000000 1  0       fixed ts^start, ts^stop;
1974.    000000 1  0    begin
1975.    000000 1  1       string  msg1 = 'P' :=
1976.    000000 1  1       ["  Start Time: ##:##:##    "," Stop Time: ##:##:##    ", 0, %377];
1977.    000033 1  1       string  msg2 = 'P' :=
1978.    000033 1  1       ["Elapsed Time: ##:##:##.###","  Cpu Time: ##:##:##.###", 0, %377];
1979.    000066 1  1
1980.    000066 1  1       fixed   elapsed^time, lct;
1981.    000066 1  1       int(32) jd;
1982.    000066 1  1       int     h1, m1, s1, ms1;
1983.    000066 1  1       int    .t [0:7];
1984.    000066 1  1
1985.    000066 1  1       lct := converttimestamp (ts^start);
1986.    000105 1  1       jd  := interprettimestamp (lct, t);
1987.    000113 1  1       h1  := t [3]; m1 := t [4]; s1 := t [5];
1988.    000124 1  1       lct := converttimestamp (ts^stop );
1989.    000137 1  1       jd  := interprettimestamp (lct, t);
1990.    000145 1  1       call log^print (msg1, h1, m1, s1, t [3], t [4], t [5]);
1991.    000173 1  1
1992.    000173 1  1       elapsed^time := ts^stop - ts^start;
1993.    000202 1  1       call convertprocesstime (elapsed^time, t, t[1], t[2], t[3], t[4]);
1994.    000223 1  1       h1  := t [0]; m1 := t [1]; s1 := t [2]; ms1 := t [3];
1995.    000236 1  1       call convertprocesstime (myprocesstime, t, t[1], t[2], t[3], t[4]);
1996.    000260 1  1       call log^print (msg2, h1, m1, s1, ms1, t, t[1], t[2], t[3]);
1997.    000311 1  1
1998.    000311 1  1    end; !proc print^times!
```

ELAPSED^TIME	Variable	FIXED (0)	Direct	L+001
H1	Variable	INT	Direct	L+013
JD	Variable	INT (32)	Direct	L+011
LCT	Variable	FIXED (0)	Direct	L+005
M1	Variable	INT	Direct	L+014
MS1	Variable	INT	Direct	L+016
MSG1	Variable	STRING	Direct	P+000
MSG2	Variable	STRING	Direct	P+000
S1	Variable	INT	Direct	L+015
T	Variable	INT	Indirect	L+017
TS^START	Variable	FIXED (0)	Direct	L-012
TS^STOP	Variable	FIXED (0)	Direct	L-006

```
1999.    000000 0  0
```

```
2001.  000000 0  0    proc txgen main;
2002.  000000 1  0  ! ----------------
2003.  000000 1  0    begin
2004.  000000 1  1       fixed   ts^start;                                    2.00
2005.  000000 1  1
2006.  000000 1  1       ! start up processing
2007.  000000 1  1       ts^start := juliantimestamp;
2008.  000011 1  1       switch^3 := $switches.<3>;
2009.  000015 1  1       startup := 2;
2010.  000017 1  1       next^thread := active^threads := 0;
2011.  000022 1  1
2012.  000022 1  1       call init^general^pool;
2013.  000023 1  1       call queue^init (ready^list);
2014.  000026 1  1       call queue^init (time^list);                          3.01
2015.  000031 1  1       call queue^init (suspend^list);                       3.01
2016.  000034 1  1                                                             3.05
2017.  000034 1  1       call loadgen^init;
2018.  000035 1  1       call init^logfile;
2019.  000036 1  1       call init^threads;
2020.  000037 1  1       startup := 1;
2021.  000041 1  1
2022.  000041 1  1       while not shutdown do
2023.  000043 1  1       begin
2024.  000043 1  2          while queue^empty (ready^list) do call io^wait;
2025.  000051 1  2          call select^thread;
2026.  000052 1  2          call process^thread;
2027.  000053 1  2       end;
2028.  000054 1  1
2029.  000054 1  1       if txstat^recs > 0 then ! need to flush TXLOG buffer  3.02
2030.  000057 1  1       begin                                                 3.02
2031.  000057 1  2          @txstat := txlog.bufaddr;                          3.02
2032.  000061 1  2          call write (txlog.filenum, txstat,                 3.02
2033.  000061 1  2                      txstat^recs * $len (tx^def));          3.02
2034.  000072 1  2          if <> then                                         3.02
2035.  000073 1  2             call iodev^failure (txlog.filenum, txlog.filename,  3.02
2036.  000073 1  2                                 write^attempt);            3.02
2037.  000101 1  2          call close (txlog.filenum);                        3.02
2038.  000106 1  2       end;                                                  3.02
2039.  000106 1  1
2040.  000106 1  1       call print^times (ts^start, juliantimestamp);
2041.  000121 1  1
2042.  000121 1  1       while logfile.io^busy do call io^wait;
2043.  000125 1  1
2044.  000125 1  1       call stop;
2045.  000132 1  1    end; !of txgen

TS^START                    Variable       FIXED (0)       Direct         L+001
```

```
ABEND                          Proc                          External
ABORTTRANSACTION               Proc            INT           External
ACTIVATE^THREAD                Proc                          %000000
ACTIVE^THREAD                  Variable,426    STRUCT-I      Indirect      #GLOBAL+000
ACTIVE^THREADS                 Variable        INT           Direct        #GLOBAL+210
ALLOCATE^BUFFER                Proc            INT           %000030
ARRIVAL^TYPE                   Variable        INT           Direct        #GLOBAL+220
ASSIGN^THREAD                  Proc                          %000000
AWAITIO                        Proc                          External
AWAITIO^DONE                   Literal         INT           %000005
BUFFER^SIZE                    Literal         INT           %000310
BUILD^IOTAG                    Proc            INT (32)      %000000
CANCELTIMEOUT                  Proc                          External
CANCEL^TIMER^DONE              Variable        INT           Direct        #GLOBAL+235
CLOSE                          Proc                          External
CONFIG                         Variable,40     STRUCT        Direct        #GLOBAL+076
CONFIG^FILE^                   Literal         INT           %000001
CONFIG^RECORD^^                Variable        TEMPLATE ,20
  1 TERMINAL^NAME[0:15]              0,1       STRING        Direct
  1 TERMINAL                         0,2       INT           Direct
CONST^                         Literal         INT           %000001
CONTROL                        Proc                          External
CONTROL^ATTEMPT                Literal         INT           %000004
CONVERTPROCESSTIME             Proc                          External
CONVERTTIMESTAMP               Proc            FIXED (0)     External
CREATE                         Proc                          External
DEALLOCATE^BUFFER              Proc                          %000031
DEBUG                          Proc                          External
DEFINEPOOL                     Proc            INT           External
DLOG^                          Proc            REAL (64)     %000000
ERR                            Proc                          %000000
FALSE                          Literal         INT           %000000
FILEINFO                       Proc                          External
FNAMECOLLAPSE                  Proc            INT           External
FNAMECOMPARE                   Proc            INT           External
FNAMEEXPAND                    Proc            INT           External
GENERAL^POOL                   Variable,52     STRUCT        Direct        #GLOBAL+010
GENERAL^POOL^BUFFER            Variable        STRING        Indirect      #GLOBAL+237
GENERAL^POOL^SIZE              Literal         INT           %040000
GENERAL^POOL^^                 Variable        TEMPLATE ,52
  1 HEADER[0:19]                     0,2       INT           Direct
  1 STATUS                          50,2       INT           Direct
GETPOOL                        Proc            INT (32)      External
GUARDIAN^DEVSUBTYPE            Define                        <10:15>
GUARDIAN^DEVTYPE               Define                        <4:9>
GUARDIAN^TPYETTMP              Literal         INT           %000025
GUARDIAN^TYPECARD              Literal         INT           %000010
GUARDIAN^TYPEDISC              Literal         INT           %000003
GUARDIAN^TYPEPRINTER           Literal         INT           %000005
GUARDIAN^TYPEPROCESS           Literal         INT           %000000
GUARDIAN^TYPERECEIVE           Literal         INT           %000002
GUARDIAN^TYPETAPE              Literal         INT           %000004
GUARDIAN^TYPETERM              Literal         INT           %000006
GUARDIAN^TYPEX25PTP            Literal         INT           %000011
HOMETERM                       Variable,40     STRUCT        Direct        #GLOBAL+056
HOMETERM^FILE^                 Literal         INT           %000004
```

```
IA^DATA                          Variable,64    STRUCT          Indirect      #GLOBAL+236
IA^DATA^^                        Variable       TEMPLATE ,64
  1 F1                                 0,1      STRING          Direct
  1 ACCOUNT[0:11]                      1,1      STRING          Direct
  1 F2                                15,1      STRING          Direct
  1 TELLER[0:11]                      16,1      STRING          Direct
  1 F3                                32,1      STRING          Direct
  1 BRANCH[0:11]                      33,1      STRING          Direct
  1 F4                                47,1      STRING          Direct
  1 AMOUNT[0:11]                      50,1      STRING          Direct
IA^RESP                          Variable,52    STRUCT          Indirect      #GLOBAL+236
IA^RESP^^                        Variable       TEMPLATE ,52
  1 LENGTH                             0,2      INT             Direct
  1 TERM^NAME[0:11]                    2,2      INT             Direct
  1 START^TIME                        32,10     FIXED (0)       Direct
  1 XTIME[0:1]                        42,4      INT (32)        Direct
  1 RESP^TIME                         42,10     FIXED (0)       Direct
INIT^GENERAL^POOL                Proc                         %000027
INIT^LOGFILE                     Proc                         %000000
INIT^THREADS                     Proc                         %000113
INTADDR                          Define                       (@ $1 '>>' 1)
INTERACTIVE                      Define                       FLAGS.<3>
INTERPRETTIMESTAMP               Proc           INT (32)      External
INTER^ACTIVE^DATA                Variable       INT           Direct        #GLOBAL+240
IODEV^CHECK                      Define                       IF <> THEN CALL IODEV^FAILURE ( $1,  $2,  $3)
IODEV^FAILURE                    Proc                         %000061
IODEV^INIT                       Proc                         %000051
IODEV^^                          Variable       TEMPLATE ,40
  1 NAME^[0:23]                        0,1      STRING          Direct
  1 FILENAME                           0,2      INT             Direct
  1 FILENUM                           30,2      INT             Direct
  1 TYPE                              32,2      INT             Direct
  1 ERROR                             34,2      INT             Direct
  1 BUFADDR                           36,2      INT             Direct
IO^COMPLETE                      Define                       FLAGS.<1>
IO^PENDING                       Define                       FLAGS.<2>
IO^WAIT                          Proc                         %000103
JULIANTIMESTAMP                  Proc           FIXED (0)     External
LOADGEN^INIT                     Proc                         %000773
LOGFILE                          Variable,50    STRUCT          Direct        #GLOBAL+136
LOGFILE^IOCOMPLETE               Proc                         %000010
LOGFILE^^                        Variable       TEMPLATE ,50
  1 MSG^Q                              0,4       SUBSTRUCT
    2 SUCC                             0,2      INT             Direct
    2 PRED                             2,2      INT             Direct
  1 DEV                                4,40      SUBSTRUCT
    2 FILENAME[0:11]                   4,2      INT             Direct
    2 FILENUM                         34,2      INT             Direct
    2 TYPE                            36,2      INT             Direct
    2 ERROR                           40,2      INT             Direct
    2 BUFADDR                         42,2      INT             Direct
  1 ACTIVE                            44,2      INT             Direct
  1 IO^BUSY                           46,2      INT             Direct
LOG^ENABLE                       Variable       INT             Direct        #GLOBAL+206
LOG^FILE^                        Literal        INT             %000002
LOG^MSG^^                        Variable       TEMPLATE ,10
  1 QUEUE                              0,4       SUBSTRUCT
```

25

```
    2 SUCC                          0,2     INT             Direct
    2 PRED                          2,2     INT             Direct
  1 LENGTH                          4,2     INT             Direct
  1 DATA                            6,2     INT             Direct
LOG^PRINT                       Proc                        %000000
MAX^KEYWORD                     Literal     INT             %000013
MAX^THREADS                     Literal     INT             %000372
MAX^TX^OUT                      Variable    INT             Direct          #GLOBAL+233
MEAN^PER^THREAD                 Variable    INT (32)        Direct          #GLOBAL+221
MYPROCESSTIME                   Proc        FIXED (0)       External
MYTERM                          Proc                        External
NAME^                           Literal     INT             %000000
NEEDS^WORK                      Define                      FLAGS.<0>
NEGEXP                          Proc        INT (32)        %000000
NET^OUT^OF^ORDER                Literal     INT             %000011
NEXT^THREAD                     Variable    INT             Direct          #GLOBAL+207
NIL                             Literal     INT             %000000
NONE^                           Literal     INT             %000002
NUMBER^BUSY                     Literal     INT             %000001
NUMBER^NOT^OBTAINABLE           Literal     INT             %000015
NUMIN                           Proc        INT             External
NUMOUT                          Proc                        External
OPEN                            Proc                        External
OPEN^ATTEMPT                    Literal     INT             %000000
OPEN^EXCLUSIVE                  Literal     INT             %000020
OPEN^GETMSG                     Literal     INT             %040000
OPEN^NOWAIT                     Literal     INT             %000001
OPEN^PROTECTED                  Literal     INT             %000060
OPEN^READONLY                   Literal     INT             %002000
OPEN^READWRITE                  Literal     INT             %000000
OPEN^SHARED                     Literal     INT             %000000
OPEN^WRITEONLY                  Literal     INT             %004000
PARAM^ARRIVAL                   Literal     INT             %000004
PARAM^CONFIG                    Literal     INT             %000001
PARAM^DEFAULT                   Literal     INT             %000000
PARAM^DELAY                     Literal     INT             %000010
PARAM^KEYWORDS                  Variable    STRING          Direct           P+000
PARAM^LOG                       Literal     INT             %000003
PARAM^MAXTERM                   Literal     INT             %000011
PARAM^MAXTX                     Literal     INT             %000012
PARAM^RNSEED                    Literal     INT             %000013
PARAM^SCRIPT                    Literal     INT             %000002
PARAM^TX100                     Literal     INT             %000007
PARAM^TXLOG                     Literal     INT             %000006
PARAM^TXRATE                    Literal     INT             %000005
POSITION                        Proc                        External
PRINT^HELP                      Proc                        %000624
PRINT^TIMES                     Proc                        %000066
PROCESS^THREAD                  Proc                        %000000
PROCESS^THREAD^STATE            Proc                        %000000
PUTPOOL                         Proc                        External
QUEUE^ANY                       Define                      ( $1.SUCC <> @ $1)
QUEUE^EMPTY                     Define                      ( $1.SUCC = @ $1)
QUEUE^GET                       Proc        INT             %000000
QUEUE^INIT                      Proc                        %000000
QUEUE^PUT                       Proc                        %000000
QUEUE^^                         Variable    TEMPLATE ,4
```

```
  1 SUCC                          0,2     INT             Direct
  1 PRED                          2,2     INT             Direct
RANDF                     Proc            REAL (32)       %000053
RANDINT                   Proc            INT             %000000
RATE                      Variable        INT             Direct      #GLOBAL+217
READ                      Proc                            External
READUPDATE                Proc                            External
READY^LIST                Variable,4      STRUCT          Direct      #GLOBAL+001
READ^ATTEMPT              Literal         INT             %000001
RECEIVE                   Variable,40     STRUCT          Direct      #GLOBAL+036
RECEIVE^BUFFER            Variable        INT             Indirect    #GLOBAL+236
RECEIVE^FILE^             Literal         INT             %000000
REPLY                     Proc                            External
SAMPLE^ARRIVAL            Proc            INT (32)        %000000
SAVE^THREAD^STATE         Proc                            %000000
SCRIPT                    Variable,40     STRUCT          Direct      #GLOBAL+116
SEED                      Variable        INT (32)        Indirect    #GLOBAL+226
SEED1                     Variable        INT (32)        Direct      #GLOBAL+224
SELECT^THREAD             Proc                            %000000
SERVER^FILE^              Literal         INT             %000006
SETUP                     Define                          FLAGS.<0>
SHUTDOWN                  Variable        INT             Direct      #GLOBAL+204
SIGNALTIMEOUT             Proc                            External
STARTUP                   Variable        INT             Direct      #GLOBAL+205
STARTUP^MSG^^             Variable        TEMPLATE ,1122
  1 ID                            0,2     INT             Direct
  1 DEFAULT[0:7]                  2,2     INT             Direct
  1 IN[0:11]                     22,2     INT             Direct
  1 OUT[0:11]                    52,2     INT             Direct
  1 PARM[0:527]                 102,1     STRING          Direct
STARTUP^PARAMETER         Variable,32     STRUCT          Indirect    #GLOBAL+035
STARTUP^PARAM^^           Variable        TEMPLATE ,32
  1 NAME[0:11]                    0,2     INT             Direct
  1 VALUE                         0,2     INT             Direct
  1 FLAGS                        30,2     INT             Direct
  1 S^NAME                        0,1     STRING          Direct
START^DELAY               Variable        INT             Direct      #GLOBAL+216
START^TIME                Variable        FIXED (0)       Direct      #GLOBAL+212
STOP                      Proc                            External
STR^                      Literal         INT             %000003
SUSPEND^LIST              Variable,4      STRUCT          Direct      #GLOBAL+005
SUSPEND^THREAD            Proc                            %000000
SWITCH^3                  Variable        INT             Direct      #GLOBAL+211
SYSMSG^BREAK              Define                          -20
SYSMSG^STARTUP            Define                          -1
SYSMSG^TIME               Define                          -22
TCB^^                     Variable        TEMPLATE ,426
  1 READY^Q                       0,4      SUBSTRUCT
    2 SUCC                        0,2     INT             Direct
    2 PRED                        2,2     INT             Direct
  1 TERMINAL                      4,32     SUBSTRUCT
    2 NAME^[0:23]                 4,1     STRING          Direct
    2 NAME                        4,2     INT             Direct
    2 FNUM                       34,2     INT             Direct
  1 INDEX                        36,2     INT             Direct
  1 STATE                        40,2     INT             Direct
  1 SUBSTATE                     42,2     INT             Direct
```

1 FLAGS	44,2	INT	Direct	
1 LAST^STATE	46,2	INT	Direct	
1 LAST^SUBSTATE	50,2	INT	Direct	
1 LAST^LOC	52,2	INT	Direct	
1 LAST^FNUM	54,2	INT	Direct	
1 LAST^ERROR	56,2	INT	Direct	
1 LAST^COUNT	60,2	INT	Direct	
1 SEND^TIMESTAMP	62,10	FIXED (0)	Direct	
1 RECV^TIMESTAMP	72,10	FIXED (0)	Direct	
1 THINK^TIME	102,4	INT (32)	Direct	
1 END^TIME	106,10	FIXED (0)	Direct	
1 DATA^BUFFER[0:99]	116,2	INT	Direct	
TERMINAL^FILE^	Literal	INT	%000005	
TFILE^FILE^	Literal	INT	%000007	
THREADS	Variable,426	STRUCT-I	Indirect	#GLOBAL+007
THREAD^ACTIVE	Proc		%000132	
THREAD^AWAITIO	Proc		%000000	
THREAD^CLOSED	Proc		%000000	
THREAD^COUNT	Variable	INT	Direct	#GLOBAL+223
THREAD^READY	Proc		%000000	
THREAD^STATE^ACTIVE	Literal	INT	%000002	
THREAD^STATE^CLOSED	Literal	INT	%000000	
THREAD^STATE^READY	Literal	INT	%000001	
THREAD^STATE^STOPPING	Literal	INT	%000003	
THREAD^STATE^THINK	Literal	INT	%000004	
THREAD^STOPPING	Proc		%000046	
THREAD^THINK	Proc		%000134	
TIME^LIST	Variable,4	STRUCT	Direct	#GLOBAL+003
TRACE^FILE^	Literal	INT	%000003	
TRUE	Literal	INT	%177777	
TXCOUNT	Variable	INT (32)	Direct	#GLOBAL+231
TXGEN	Proc		%000000	
TXGEN^VS	Define		["vs 3.08"]	
TXLOG	Variable,40	STRUCT	Direct	#GLOBAL+162
TXMAX	Variable	INT (32)	Direct	#GLOBAL+227
TXSTAT	Variable,40	STRUCT-I	Indirect	#GLOBAL+202
TXSTAT^BUFSIZE	Literal	INT	%010000	
TXSTAT^NUMRECS	Literal	INT	%000200	
TXSTAT^RECS	Variable	INT	Direct	#GLOBAL+203
TX^DEF	Variable	TEMPLATE ,40		
1 TERM^COUNT	0,2	INT	Direct	
1 TERM^INDEX	2,2	INT	Direct	
1 ARRIVAL^RATE	4,2	INT	Direct	
1 SEND^TIMESTAMP	6,10	FIXED (0)	Direct	
1 NEXT^ARRIVAL^TIME	16,4	INT (32)	Direct	
1 RESPONSE^TIME	22,4	INT (32)	Direct	
1 FILLER	26,1			
TX^OUTSTANDING	Variable	INT	Direct	#GLOBAL+234
WRITE	Proc		External	
WRITEREAD	Proc		External	
WRITE^ATTEMPT	Literal	INT	%000002	
WRITE^LOGFILE	Proc		%000000	
WR^ATTEMPT	Literal	INT	%000003	

ENTRY POINT MAP BY NAME

SP	PEP	BASE	LIMIT	ENTRY	ATTRS	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	031	003441	003452	003441		ACTIVATE^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	015	001433	001534	001463		ALLOCATE^BUFFER	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	027	003372	003430	003372		ASSIGN^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	040	005125	005146	005125	V	BUILD^IOTAG	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	016	001535	001627	001566		DEALLOCATE^BUFFER	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	002	000145	000416	000145		DLOG^	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	003	000417	000420	000417		ERR	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	014	001344	001432	001373		INIT^GENERAL^POOL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	013	001271	001343	001271		INIT^LOGFILE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	037	003550	005124	003663		INIT^THREADS	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	041	005147	005526	005230	V	IODEV^FAILURE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	042	005527	005745	005600	V	IODEV^INIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	043	005746	006313	006051		IO^WAIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	044	006314	007507	007307		LOADGEN^INIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	012	001207	001270	001217		LOGFILE^IOCOMPLETE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	010	000614	001116	000614	V	LOG^PRINT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	005	000513	000535	000513		NEGEXP	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	045	007510	011004	010334		PRINT^HELP	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	046	011005	011320	011073		PRINT^TIMES	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	036	003540	003547	003540		PROCESS^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	035	003503	003537	003503		PROCESS^THREAD^STATE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	017	001630	001646	001630		QUEUE^GET	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	021	001672	001676	001672		QUEUE^INIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	020	001647	001671	001647		QUEUE^PUT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	004	000421	000512	000474		RANDF	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	006	000536	000557	000536		RANDINT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	007	000560	000613	000560		SAMPLE^ARRIVAL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	032	003453	003466	003453		SAVE^THREAD^STATE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	033	003467	003477	003467		SELECT^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	034	003500	003502	003500		SUSPEND^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	024	002076	002704	002230		THREAD^ACTIVE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	030	003431	003440	003431		THREAD^AWAITIO	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	022	001677	001774	001677		THREAD^CLOSED	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	023	001775	002075	001775		THREAD^READY	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	025	002705	003064	002753		THREAD^STOPPING	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	026	003065	003371	003221		THREAD^THINK	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	047	011321	011455	011321	M	TXGEN	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	011	001117	001206	001117		WRITE^LOGFILE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S

READ-ONLY DATA BLOCK MAP BY NAME

SP	BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	C000050	000144	COMMON	STRING	PARAM^KEYWORDS	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S

DATA BLOCK MAP BY NAME

BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
000000	000240	COMMON	WORD	#GLOBAL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
000241	021014	COMMON	WORD	.#GLOBAL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S

ENTRY POINT MAP BY LOCATION
CODE SEGMENT 00

SP	PEP	BASE	LIMIT	ENTRY	ATTRS	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	002	000145	000416	000145		DLOG^	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	003	000417	000420	000417		ERR	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	004	000421	000512	000474		RANDF	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	005	000513	000535	000513		NEGEXP	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	006	000536	000557	000536		RANDINT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	007	000560	000613	000560		SAMPLE^ARRIVAL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	010	000614	001116	000614	V	LOG^PRINT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	011	001117	001206	001117		WRITE^LOGFILE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	012	001207	001270	001217		LOGFILE^IOCOMPLETE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	013	001271	001343	001271		INIT^LOGFILE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	014	001344	001432	001373		INIT^GENERAL^POOL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	015	001433	001534	001463		ALLOCATE^BUFFER	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	016	001535	001627	001566		DEALLOCATE^BUFFER	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	017	001630	001646	001630		QUEUE^GET	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	020	001647	001671	001647		QUEUE^PUT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	021	001672	001676	001672		QUEUE^INIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	022	001677	001774	001677		THREAD^CLOSED	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	023	001775	002075	001775		THREAD^READY	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	024	002076	002704	002230		THREAD^ACTIVE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	025	002705	003064	002753		THREAD^STOPPING	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	026	003065	003371	003221		THREAD^THINK	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	027	003372	003430	003372		ASSIGN^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	030	003431	003440	003431		THREAD^AWAITIO	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	031	003441	003452	003441		ACTIVATE^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	032	003453	003466	003453		SAVE^THREAD^STATE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	033	003467	003477	003467		SELECT^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	034	003500	003502	003500		SUSPEND^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	035	003503	003537	003503		PROCESS^THREAD^STATE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	036	003540	003547	003540		PROCESS^THREAD	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	037	003550	005124	003663		INIT^THREADS	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	040	005125	005146	005125	V	BUILD^IOTAG	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	041	005147	005526	005230	V	IODEV^FAILURE	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	042	005527	005745	005600	V	IODEV^INIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	043	005746	006313	006051		IO^WAIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	044	006314	007507	007307		LOADGEN^INIT	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	045	007510	011004	010334		PRINT^HELP	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	046	011005	011320	011073		PRINT^TIMES	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
00	047	011321	011455	011321	M	TXGEN	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S

Code size = 4910 words
  PEP = 40 words
  Global P-relative arrays = 61 words
  Procedures = 4809 words
  Gap at 32K = 0 words
  XEP = 29 words
Code area size = 5 pages
Resident code size = 0 pages

READ-ONLY DATA BLOCK MAP BY LOCATION
CODE SEGMENT 00

SP	BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	C000050	000144	COMMON	STRING	PARAM^KEYWORDS	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S

DATA BLOCK MAP BY LOCATION

BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
000000	000240	COMMON	WORD	#GLOBAL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S
000241	021014	COMMON	WORD	.#GLOBAL	16FEB87	17:56	TAL	$BASE.GRAYTGUN.TXGEN38S

```
BINDER - OBJECT FILE BINDER - T9621C00 - (15JUL87)   SYSTEM \FOXII
Object file name is $BASE.GRAYTGUN.OBJECT
Object file timestamp is 16FEB87 17:56:41
Number of Binder errors = 0
Number of Binder warnings = 0
Primary data = 161 words
Secondary data = 8556 words
Code area size = 5 pages
Resident code size = 0 pages
Data area size = 64 pages
Extended data area size = 0 pages
Top of stack = 8717 words
Number of code segments  = 1 segment


The object file will run on a TNS/II, but may not run on a TNS
Number of compiler errors =  0
Number of compiler warnings =  33
Maximum symbol table space used was = 43354 bytes
Number of source lines = 4129
Compile cpu time = 00:00:35
Total Elapsed time = 00:01:50
```

TAL - T9250C00 - (15JUL87) Source language: TAL - Target machine: Tandem NonStop II System
Date - Time : 17FEB87 - 22:05:48 Default options: On (LIST,CODE,MAP,WARN,LMAP) - Off (ICODE,INNERLIST)

```
    1.  000000 0  0  ?ERRORS 10, nowarn, !datapages 32,! nocode, symbols, inspect, columns 74
    3.  000000 0  0  !---------------------------------------------------------------------
    4.  000000 0  0  ! 16Feb87. 2.03. Nhan Chu. Inter-arrival time distribution plotting of
    5.  000000 0  0  !   theoretical vs actual per auditor's request. New params: ATPLOT,
    6.  000000 0  0  !   PCELLA, CCELLA, TERMS.
    7.  000000 0  0  !   Fix bug in plotting the last cell.
    8.  000000 0  0  ! 19Jan87. 2.02. Nhan Chu. Record deblocking for TXLOG.
    9.  000000 0  0  ! 15Jan87. 2.01. Nhan Chu. Use 'version' define.
   10.  000000 0  0  ! 14Jan87. 2.00. Gerhard Huff. TXLOG file is unstructured.
   11.  000000 0  0  ! 10Dec86. 1.02. Nhan Chu. Clarify labels for input params in report
   12.  000000 0  0  ! 05Dec86. 1.01. Nhan Chu. Error handler to window cmd.
   13.  000000 0  0  ! 20Oct86. 1.00. Nhan Chu. Add POFF 8 to output.
   14.  000000 0  0  ! 14Oct86. 0.00. Nhan Chu. Add Tx inter-arrival time in output.
   15.  000000 0  0  !---------------------------------------------------------------------
   16.  000000 0  0
   17.  000000 0  0  !-------------------------!
   18.  000000 0  0  !    TXREP Version        !
   19.  000000 0  0  !-------------------------!
   20.  000000 0  0  define txrep^vs = ["vs 2.03"]#; ! shown in HELP and output report        2.01
   21.  000000 0  0
   22.  000000 0  0  !-------------------------!
   23.  000000 0  0  !       IO devices        !
   24.  000000 0  0  !-------------------------!
   25.  000000 0  0  define eh = if <> then call debug#;
   26.  000000 0  0
   27.  000000 0  0  ?NOLIST,noprintsym,source $system.system.gpldefs
   29.  000000 0  0
   30.  000000 0  0  int    .cfcb [0:fcbsize-1] := 0;  ! common fcb
   31.  000074 0  0  int    .out  [0:fcbsize-1];       ! out file
   32.  000170 0  0  int    .out^buffer [0:2047];      ! 4096 bytes block size
   33.  004170 0  0
   34.  004170 0  0  struct iodev^^ (*);
   35.  004170 0  0  begin
   36.  004170 0  1      string name^ [0:23];
   37.  004170 0  1      int filename = name^;
   38.  004170 0  1      int filenum;
   39.  004170 0  1      int type;
   40.  004170 0  1      int error;
   41.  004170 0  1      int bufaddr;
   42.  004170 0  1  end;
   43.  004170 0  0
   44.  004170 0  0  struct .hometerm (iodev^^);
   45.  004210 0  0  struct .in (iodev^^);             ! input file= tx stats log
   46.  004230 0  0
   47.  004230 0  0  !-------------------------!
   48.  004230 0  0  !  RT distribution table  !
   49.  004230 0  0  !-------------------------!
   50.  004230 0  0  literal cellcount = 1000;
   51.  004230 0  0  struct  rt^table^^ (*);
   52.  004230 0  0  begin
   53.  004230 0  1     string  name^ [0:23];
   54.  004230 0  1     int     name = name^;
   55.  004230 0  1     int(32) count;
```

```
    int(32) minimum;
    int(32) maximum;
    real    sum;
    real    sum^2;
    int(32) lobound;
    int(32) upbound;
    int     cellsize;
    int     numcell;
    int     freq [0:cellcount];
end;
struct  .rt (rt^table^^);          ! response time stats
!-------------------------!                                          2.03
!  Inter-Arrival Times    !                                          2.03
!-------------------------!                                          2.03
struct  .at (rt^table^^);          ! inter-arrival time stats       2.03
struct  .at^stats ;                                                 2.03
begin                                                               2.03
   real     avg;                   ! sec                            2.03
   real     std;                   ! sec                            2.03
   real     min;                   ! sec                            2.03
   real     max;                   ! sec                            2.03
end;                                                                2.03

!-------------------------!
!  DELAY and THINK stats  !
!-------------------------!
struct   time^stats^^ (*);
begin
   int(32)  count;
   real     avg;
   real     std;
   int(32)  minimum;
   int(32)  maximum;
   real     sum;
   real     sum^2;
end;
struct  .think (time^stats^^);     ! Think time stats (constant,random)
struct  .dilay (time^stats^^);     ! Delay time stats (constant,random)

!-------------------------!
!  ET1 tx statistics      !
!-------------------------!
struct   .et1;                     ! reported et1 transaction statistics
begin
    real    thruput;               ! tx/sec
    real    rt^avg;                ! sec
    real    rt^std;                ! sec
    real    rt^min;                ! sec
    real    rt^max;                ! sec
    real    rt^dist [0:9];         ! resp time distribution in 10% increment
    real    rt^pct  [0:9];         ! actual percentile values (%)
end;

!-------------------------!
!  Run Env Parameters     !
!-------------------------!
```

```
113.    010370 0  0   literal   maximum = 0, constant = 1, random = 2;
114.    010370 0  0   struct    .test;                    ! et1 test run parameters
115.    010370 0  0   begin
116.    010370 0  1       int     terminals;             ! #
117.    010370 0  1       int     arrival^rate;          ! (tx/sec) * 10
118.    010370 0  1       int     arrival^type;          ! 0=max, 1=constant, 2=random
119.    010370 0  1       fixed   ts^window^begin;       ! timestamp when window begins
120.    010370 0  1       fixed   ts^window^end;         ! timestamp when window ends
121.    010370 0  1       int(32) len^window;            ! ts^window^end - ts^window^begin (sec)
122.    010370 0  1       fixed   ts^first^rec;          ! timestamp of the first record
123.    010370 0  1       fixed   ts^last^rec;           ! timestamp of the last  record
124.    010370 0  1       int(32) len^test;              ! ts^last^rec - ts^first^rec (sec)
125.    010370 0  1   end;
126.    010417 0  0
127.    010417 0  0   !-------------------------!
128.    010417 0  0   !    Startup Parameters   !
129.    010417 0  0   !-------------------------!
130.    010417 0  0   literal   name^  = 0,
131.    010417 0  0             const^ = 1,
132.    010417 0  0             none^  = 2,
133.    010417 0  0             str^   = 3,
134.    010417 0  0
135.    010417 0  0             param^default   = 0,
136.    010417 0  0             param^window    = 1,
137.    010417 0  0             param^title     = 2,
138.    010417 0  0             param^rtplot    = 3,
139.    010417 0  0             param^rtdump    = 4,
140.    010417 0  0             param^txlog     = 5,
141.    010417 0  0             param^pcell     = 6,
142.    010417 0  0             param^ccell     = 7,
143.    010417 0  0             param^atplot    = 8,                                    2.03
144.    010417 0  0             param^pcella    = 9,                                    2.03
145.    010417 0  0             param^ccella    = 10,                                   2.03
146.    010417 0  0             param^terms     = 11,                                   2.03
147.    010417 0  0             max^keyword     = 11;
148.    010417 0  0
149.    010417 0  0   string    param^keywords = 'P' :=
150.    010417 0  0             [6, str^   , "WINDOW   ",     ! hh:mm:ss/hh:mm:ss
151.    010417 0  0              5, str^   , "TITLE    ",     ! 24 char max
152.    010417 0  0              6, none^  , "RTPLOT   ",     ! response time plot
153.    010417 0  0              6, none^  , "RTDUMP   ",     ! response time dump
154.    010417 0  0              5, name^  , "TXLOG    ",     ! Tx input file from TXGEN
155.    010417 0  0              5, const^ , "PCELL    ",     ! Plot cell size    (Def=100ms)
156.    010417 0  0              5, const^ , "CCELL    ",     ! Collect cell size (Def= 50ms)
157.    010417 0  0              6, none^  , "ATPLOT   ",     ! inter-arrival time plot          2.03
158.    010417 0  0              6, const^ , "PCELLA   ",     ! Plot cell size    (Def= 1000ms) 2.03
159.    010417 0  0              6, const^ , "CCELLA   ",     ! Collect cell size (Def=  500ms) 2.03
160.    010417 0  0              5, const^ , "TERMS    ",     ! Num of terms with merged txlog  2.03
161.    010417 0  0              0];
162.    010417 0  0
163.    010417 0  0   define    setup = flags.<0>#;
164.    010417 0  0   struct    startup^param^^ (*);
165.    010417 0  0   begin
166.    010417 0  1       int name [0:11],
167.    010417 0  1           value = name,
168.    010417 0  1           flags;
169.    010417 0  1       string s^name = name;
```

```
  170.    010417 0  1   end;
  171.    010417 0  0   struct   .startup^parameter (startup^param^^) [0:max^keyword];
  172.    010653 0  0
  173.    010653 0  0   !-------------------------!                                  2.02
  174.    010653 0  0   !  TXLOG Rec Definition   !                                  2.02
  175.    010653 0  0   !-------------------------!                                  2.02
  176.    010653 0  0   ?SOURCE TXTAL                                               2.02
Source file:  [3]  $TURF.NVCTXGEN.TXTAL  12JAN87 - 09:43:18
    1.    010653 0  0   ! SCHEMA PRODUCED DATE - TIME : 1/12/87 09:43:13
    2.    010653 0  0   ?SECTION TX
```

```
 4.    010653 0  0   ! Record TX created on 01/12/87 at 09:43
 5.    010653 0  0   !-------------------------------------------------------------
 6.    010653 0  0   !  The following record is written by the program TXGEN for each
 7.    010653 0  0   !  ET1 transaction sent and received to/from the SUT system.
 8.    010653 0  0   !  The program TXREP scans these records to produce performance
 9.    010653 0  0   !  statistics about the ET1 benchmark.
10.    010653 0  0   !
11.    010653 0  0   !  25Sep86. Nhan Chu.
12.    010653 0  0   !-------------------------------------------------------------
13.    010653 0  0   !  18Nov86. Make UNSIGNED binary 8.
14.    010653 0  0   !  12Jan87. Gerhard Huff: term-count, term-index binary 16, added filler
15.    010653 0  0   STRUCT    TX^DEF (*);
16.    010653 0  0      BEGIN
17.    010653 0  1   !                                  number of terminals ( <= 250 )
18.    010653 0  1      INT       TERM^COUNT;
19.    010653 0  1   !                                  terminal id
20.    010653 0  1      INT       TERM^INDEX;
21.    010653 0  1   !                                  user-specified tx rate (tps * 10)
22.    010653 0  1   !                                  =0 max rate, pos=constant, neg=random
23.    010653 0  1      INT       ARRIVAL^RATE;
24.    010653 0  1   !                                  4-word timestamp of input time
25.    010653 0  1      FIXED     SEND^TIMESTAMP;
26.    010653 0  1   !                                  interval (ms) until the next arrival
27.    010653 0  1      INT(32)   NEXT^ARRIVAL^TIME;
28.    010653 0  1   !                                  tx response time in milliseconds
29.    010653 0  1      INT(32)   RESPONSE^TIME;
30.    010653 0  1   !                                  filler to bring record up to 32
31.    010653 0  1      FILLER    10;
32.    010653 0  1      END;
177.   010653 0  0
178.   010653 0  0   !------------------------!
179.   010653 0  0   !  Other global variables !
180.   010653 0  0   !------------------------!
181.   010653 0  0
182.   010653 0  0   define    guardian^devtype     = <4:9>#,
183.   010653 0  0             guardian^devsubtype  = <10:15>#;
184.   010653 0  0
185.   010653 0  0   ! Guardian device types
186.   010653 0  0
187.   010653 0  0   literal   guardian^typeprocess = 0,
188.   010653 0  0             guardian^typereceive = 2,
189.   010653 0  0             guardian^typedisc    = 3,
190.   010653 0  0             guardian^typetape    = 4,
191.   010653 0  0             guardian^typeprinter = 5,
192.   010653 0  0             guardian^typeterm    = 6,
193.   010653 0  0             guardian^typecard    = 8,
194.   010653 0  0             guardian^typex25ptp  = 9,
195.   010653 0  0             guardian^typettmp    = 21;
196.   010653 0  0
197.   010653 0  0   ! Open flags definitions
198.   010653 0  0
199.   010653 0  0   literal   open^nowait          = %B0000000000000001,
200.   010653 0  0             open^shared          = %B0000000000000000,
201.   010653 0  0             open^exclusive       = %B0000000000010000,
202.   010653 0  0             open^protected       = %B0000000000110000,
203.   010653 0  0             open^readwrite       = %B0000000000000000,
204.   010653 0  0             open^readonly        = %B0000010000000000,
```

```
205.  010653 0  0            open^writeonly       = %B0000100000000000,
206.  010653 0  0            open^getmsg          = %B0100000000000000;
207.  010653 0  0
208.  010653 0  0  literal    open^attempt         = 0,
209.  010653 0  0            read^attempt         = 1,
210.  010653 0  0            write^attempt        = 2,
211.  010653 0  0            wr^attempt           = 3,
212.  010653 0  0            control^attempt      = 4,
213.  010653 0  0            awaitio^done         = 5;
214.  010653 0  0
215.  010653 0  0  literal    true  = -1,
216.  010653 0  0            false = 0,
217.  010653 0  0            nil   = 0; !should be -1 ?
218.  010653 0  0
219.  010653 0  0  string     .p;                        ! str utility pointer
220.  010653 0  0  int(32)    throw^count;               ! records outside of window
221.  010653 0  0
222.  010653 0  0  literal    txlog^bufsize = 4096;                                    2.02
223.  010653 0  0  int       .txlog^buf [0:(txlog^bufsize/2)-1],                      2.02
224.  014653 0  0            .txlog^bufptr,                                           2.02
225.  014653 0  0            txlog^maxrec := 0,         ! max recs in block           2.02
226.  014653 0  0            txlog^rec := 0,                                          2.02
227.  014653 0  0            first^block^read := true;                                2.02
228.  014653 0  0
229.  014653 0  0  ?NOLIST,columns 80,source $system.system.extdecs (
241.  000000 0  0  ?NOMAP
```

```
000000 0  0   !----------- ERROR REPORTING & NUMBER CONVERSION ---------------------
000000 0  0
000000 0  0     proc fprint (outfnum,fmt,p1, p2, p3, p4, p5, p6, p7, p8, p9, p10,
000000 1  0                                p11, p12, p13, p14) VARIABLE;
000000 1  0
000000 1  0     int     outfnum;                              !output file number
000000 1  0     string .fmt;                                  !format string
000000 1  0     int     p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,       !parameters
000000 1  0             p11,p12,p13,p14;
000000 1  0                                                                forward;
000000 0  0     int proc dascii (dv,base,stg);
000000 1  0          int(32)    dv;                           ! value to convert
000000 1  0          int        base;                         ! -10 if right justified
000000 1  0          string     .stg;                         ! target string
000000 1  0                                                                forward;
000000 0  0     real proc sqrt (r);
000000 1  0          real .r;                                              forward;
000000 0  0
000000 0  0   !----------- IO ------------------------------------------------------
000000 0  0
000000 0  0     Proc iodev^failure (f, name, operation) variable;
000000 1  0          int f, .name, operation;                              forward;
000000 0  0     Proc iodev^init (dev,flags, sync, type) variable;
000000 1  0          int .dev (iodev^^), flags, sync, type;                forward;
000000 0  0
000000 0  0   !----------- OTHERS --------------------------------------------------
000000 0  0
000000 0  0     proc txrep^init;                                           forward;
000000 0  0     proc calc^window^ts;                                       forward;
000000 0  0     proc read^rec (rec, error);                                              2.02
000000 1  0          struct .rec (tx^def); int .error;                     forward;     2.02
000000 0  0     proc scan^file;                                            forward;
000000 0  0     proc etl^stats;                                            forward;
000000 0  0     proc report;                                               forward;
000000 0  0     proc print^help;                                           forward;
000000 0  0     proc print^times (ts^start, ts^stop);
000000 1  0          fixed ts^start, ts^stop;                              forward;
000000 0  0
000000 0  0   ! proc txrep main;                                          forward;
```

```
283.  000000 0  0  !#########################################################################
284.  000000 0  0  !#                                                                       #
285.  000000 0  0  !#                          FORMATPRINT                                  #
286.  000000 0  0  !#                                                                       #
287.  000000 0  0  !#  FORMAT STRINGS ARE CONTAINED WITHIN A READ/ONLY STRING ARRAY.       #
288.  000000 0  0  !#  EACH LINE OF THE FORMAT IS ENDED WITH A NULL (0) BYTE, AND          #
289.  000000 0  0  !#  THE END OF THE STRING ARRAY IS MARKED BY AN (%377) BYTE.           #
290.  000000 0  0  !#                                                                       #
291.  000000 0  0  !#  FIELD INDICATORS ARE:                                              #
292.  000000 0  0  !#            #  -  signed decimal                                     #
293.  000000 0  0  !#            &  -  unsigned decimal                                   #
294.  000000 0  0  !#            %  -  octal                                              #
295.  000000 0  0  !#            '  -  string (blank-fill to field length)               #
296.  000000 0  0  !#            \  -  string (no blank fill)                            #
297.  000000 0  0  !#            ?  -  escape character                                   #
298.  000000 0  0  !#                                                                     #
299.  000000 0  0  !#  FOR NUMERIC FIELDS, AN INTEGER VALUE SHOULD BE PASSED.            #
300.  000000 0  0  !#  FOR STRING FIELDS, THE ADDRESS OF THE STRING ARRAY SHOULD BE      #
301.  000000 0  0  !#    PASSED.                                                          #
302.  000000 0  0  !#                                                                     #
303.  000000 0  0  !#########################################################################
304.  000000 0  0
305.  000000 0  0  PROC fprint (outfnum,fmt,p1, p2, p3, p4, p5, p6, p7, p8, p9, p10,
306.  000000 1  0                            p11, p12, p13, p14) VARIABLE;
307.  000000 1  0
308.  000000 1  0    INT     outfnum;                              !output file number
309.  000000 1  0    STRING .fmt;                                  !format string
310.  000000 1  0    INT     p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,       !parameters
311.  000000 1  0            p11,p12,p13,p14;
312.  000000 1  0
313.  000000 1  0  BEGIN
314.  000000 1  1
315.  000000 1  1    STRING  buffer[0:132],                        !print buffer
316.  000000 1  1           .bufptr,                               !current position in buffer
317.  000000 1  1           .p;                                    !working ptr
318.  000000 1  1    INT    .parm := @p1,                          !current parameter
319.  000000 1  1            save,                                 !saved address
320.  000000 1  1            i;                                    !indexing variable
321.  000000 1  1
322.  000000 1  1  DO
323.  000004 1  1    BEGIN
324.  000004 1  2
325.  000004 1  2    STACK @fmt,0; CODE (SBU %660);        !SCAN fmt UNTIL 0 -> @p;
326.  000007 1  2    STORE @p;
327.  000010 1  2
328.  000010 1  2    STACK @buffer, @fmt, @p'-'@fmt+2;  !buffer':='fmt FOR @p'-'@fmt+2
329.  000017 1  2    CODE (MOVB %67);
330.  000020 1  2
331.  000020 1  2    @fmt := @p[1];
332.  000023 1  2    @bufptr := @buffer;
333.  000026 1  2
334.  000026 1  2    WHILE bufptr DO                        !check for format characters!
335.  000030 1  2
336.  000030 1  2      IF bufptr = "#" OR                   !  signed decimal integer!
337.  000030 1  2         bufptr = "&" OR                   !unsigned decimal integer!
338.  000030 1  2         bufptr = "%" THEN                 !unsigned octal   integer!
339.  000041 1  2           BEGIN
```

```
340.    000041 1  3              SCAN bufptr WHILE bufptr -> @p;
341.    000045 1  3              IF bufptr = "#" AND parm < 0 THEN
342.    000053 1  3                BEGIN
343.    000053 1  4                  bufptr  := "-";
344.    000055 1  4                  parm    := -parm;
345.    000060 1  4                  @bufptr := @bufptr[1];
346.    000063 1  4                END;
347.    000063 1  3              CALL NUMOUT (bufptr, parm,
348.    000063 1  3                           IF bufptr="%" THEN 8 ELSE 10,@p'-'@bufptr);
349.    000100 1  3              @bufptr := @p;
350.    000102 1  3              @parm := @parm[1];
351.    000105 1  3            END
352.    000105 1  2        ELSE IF bufptr = "'" THEN          !character string (blank fill)!
353.    000111 1  2             BEGIN
354.    000111 1  3               @p := parm;
355.    000113 1  3               WHILE bufptr = "'" DO
356.    000116 1  3                 BEGIN
357.    000116 1  4                   IF p THEN
358.    000120 1  4                     BEGIN
359.    000120 1  5                       bufptr := p;
360.    000122 1  5                       @p := @p[1];
361.    000125 1  5                     END
362.    000125 1  4                   ELSE bufptr := " ";
363.    000130 1  4                   @bufptr := @bufptr[1];
364.    000133 1  4                 END;
365.    000134 1  3               @parm := @parm[1];
366.    000137 1  3             END
367.    000137 1  2        ELSE IF bufptr = "\" THEN          !character string (no blank fill)!
368.    000143 1  2             BEGIN
369.    000143 1  3               @p := parm;
370.    000145 1  3               i := -1;
371.    000147 1  3               WHILE bufptr[i:=i+1] = "\" AND p[i] DO
372.    000161 1  3                 bufptr[i] := p[i];
373.    000165 1  3               @bufptr := @bufptr[i];
374.    000170 1  3               IF bufptr = "\" THEN
375.    000173 1  3                 BEGIN
376.    000173 1  4                   SCAN bufptr WHILE "\" -> @p;
377.    000177 1  4                   bufptr ':=' p FOR @buffer[132]'-'@p;
378.    000211 1  4                 END;
379.    000211 1  3               @parm := @parm[1];
380.    000214 1  3             END
381.    000214 1  2        ELSE                               !no replacement field!
382.    000216 1  2             BEGIN
383.    000216 1  3               IF bufptr = "?" THEN       !escape character!
384.    000221 1  3                 bufptr ':=' bufptr[1] FOR @buffer[132]-@bufptr;
385.    000234 1  3               @bufptr := @bufptr[1];
386.    000237 1  3             END;
387.    000240 1  2
388.    000240 1  2        CALL WRITE (outfnum,buffer,@bufptr'-'@buffer);
389.    000253 1  2
390.    000253 1  2        END
391.    000253 1  1      UNTIL bufptr[1] = %377;
392.    000257 1  1
393.    000257 1  1    END; !proc fprint!
```

```
395.  000000 0  0  INT PROC DASCII(DV,BASE,STG); ! PER GUARDIAN FUP
396.  000000 1  0  !--------------------------!
397.  000000 1  0     !
398.  000000 1  0     ! Usage:  -10 => right justified, 10 => left justified
399.  000000 1  0     !
400.  000000 1  0     ! 4567890b  [44:51] 7   userid
401.  000000 1  0     ! err := dascii( $DBL( owner.<0:7> ), -10, line^[46] );
402.  000000 1  0     ! line^[47] := ",";
403.  000000 1  0     ! err := dascii( $DBL( owner.<8:15> ), 10, line^[48] );
404.  000000 1  0     !
405.  000000 1  0
406.  000000 1  0     INT(32)    DV;              ! VALUE TO CONVERT
407.  000000 1  0     INT        BASE;            ! -10 IF RIGHT JUSTIFIED
408.  000000 1  0     STRING     .STG;            ! TARGET STRING
409.  000000 1  0     BEGIN
410.  000000 1  1       INT(32)  DVT;
411.  000000 1  1       INT      RT:=0,
412.  000000 1  1                SGN:=0,
413.  000000 1  1                K:=11,
414.  000000 1  1                N,
415.  000000 1  1                REM,
416.  000000 1  1                DVL=DVT,
417.  000000 1  1                DVR=DVT+1;
418.  000000 1  1       STRING   B[0:11]:=[11*[" "],"0"];
419.  000006 1  1
420.  000006 1  1           IF BASE<0 THEN RT := 1;       ! RIGHT JUSTIFY
421.  000025 1  1           IF DV<0D THEN
422.  000031 1  1            BEGIN               ! NEGATIVE
423.  000031 1  2              SGN := 1;
424.  000033 1  2              DV := $ABS(DV);
425.  000037 1  2            END;
426.  000037 1  1
427.  000037 1  1           WHILE DV<>0D DO
428.  000043 1  1            BEGIN
429.  000043 1  2              STACK 0; STACK $HIGH(DV); STACK 10; CODE(LDIV);
430.  000047 1  2  !           R0=REMAINDER, R1=QUOTIENT (NEW $HIGH(DV))
431.  000047 1  2              STORE DVL;
432.  000050 1  2              STACK $INT(DV); STACK 10; CODE(LDIV);
433.  000054 1  2  !           R0=DV MOD 10 (NEXT DIGIT), R1=NEW $LOW(DV)
434.  000054 1  2              STORE DVR; STORE REM; DV := DVT;
435.  000060 1  2              B[K] := REM+"0";
436.  000064 1  2              K := K-1;
437.  000066 1  2            END;
438.  000067 1  1           IF SGN THEN
439.  000071 1  1            BEGIN               ! INSERT SIGN
440.  000071 1  2              B[K] := "-";
441.  000074 1  2              K := K-1;
442.  000076 1  2            END;
443.  000076 1  1
444.  000076 1  1           IF NOT (N:=11-K) THEN N := 1;
445.  000105 1  1           IF RT THEN
446.  000107 1  1            STG ':=' B[11] FOR N
447.  000107 1  1           ELSE
448.  000120 1  1            STG ':=' B[12-N] FOR N;
449.  000132 1  1           RETURN N;
450.  000134 1  1     END !DASCII! ;
451.  000000 0  0  ?NOLIST
```

01

```
  512.    000000 0  0     Proc iodev^failure (f, name, operation) variable;
  513.    000000 1  0   ! ---------------------------------------------
  514.    000000 1  0     int f, .name, operation;
  515.    000000 1  0     begin
  516.    000000 1  1        int error, convert := true, 1, f^name [0:11];
  517.    000000 1  1        string s^name [0:34],
  518.    000000 1  1               s^oper [0:30],
  519.    000000 1  1               io^failure = 'P' :=
  520.    000000 1  1                   [20*["\"], " with error ?#### on file ", 34*["\"],
  521.    000050 1  1                    ", file no: ?###", 0, %377];
  522.    000061 1  1        begin
  523.    000065 1  2           call fileinfo (f, error, f^name);
  524.    000114 1  2           if f = -1 then
  525.    000117 1  2           begin
  526.    000117 1  3              s^name ':=' "No file name" & 0;
  527.    000137 1  3              convert := false;
  528.    000141 1  3           end;
  529.    000141 1  2           if $param (name) then
  530.    000144 1  2           begin
  531.    000144 1  3              f^name ':=' name for 12;
  532.    000150 1  3              convert := true;
  533.    000152 1  3           end;
  534.    000152 1  2           if convert then s^name [fnamecollapse (f^name, s^name)] := 0;
  535.    000164 1  2           case operation of
  536.    000166 1  2           begin
  537.    000166 1  3              s^oper ':=' ["OPEN failed", 0];
  538.    000200 1 CE1             s^oper ':=' ["READ failed", 0];
  539.    000212 1 CE2             s^oper ':=' ["WRITE failed", 0];
  540.    000224 1 CE3             s^oper ':=' ["CONTROL failed", 0];
  541.    000236 1 CE4             s^oper ':=' ["AWAITIO completion ", 0];
  542.    000250 1 CE5          end; !case
  543.    000256 1  2           call fprint (hometerm.filenum,io^failure, @s^oper, error,
  544.    000256 1  2                        @s^name, f);
  545.    000304 1  2        end;
  546.    000304 1  1     end;
```

CONVERT	Variable	INT	Direct	L+002
ERROR	Variable	INT	Direct	L+001
F	Variable	INT	Direct	L-006
F^NAME	Variable	INT	Direct	L+004
IO^FAILURE	Variable	STRING	Direct	P+000
L	Variable	INT	Direct	L+003
NAME	Variable	INT	Indirect	L-005
OPERATION	Variable	INT	Direct	L-004
S^NAME	Variable	STRING	Direct	L+020
S^OPER	Variable	STRING	Direct	L+042

```
  548.    000000 0  0     proc iodev^init (dev,flags, sync, type) variable;
  549.    000000 1  0   ! ---------------------------------------------
  550.    000000 1  0     int .dev (iodev^^), flags, sync, type;
  551.    000000 1  0     begin
  552.    000000 1  1        string s^name [0:34],
  553.    000000 1  1               wrong^type = 'P' :=
  554.    000000 1  1     ["Device type for file ", 34*["\"], " is ?###, should be ?###", 0,%377];
  555.    000051 1  1        int i;
  556.    000051 1  1
  557.    000051 1  1        begin
  558.    000052 1  2           call open (dev.filename, dev.filenum, flags, sync);
  559.    000065 1  2           if <> then
  560.    000066 1  2              call iodev^failure (dev.filenum, dev.filename, open^attempt);
  561.    000075 1  2           call fileinfo (dev.filenum, dev.error,,, dev.type);
  562.    000133 1  2           if (not dev.error) and $param (type) and
  563.    000133 1  2              (i := dev.type.guardian^devtype) <> type then
  564.    000151 1  2           begin
  565.    000151 1  3              s^name [fnamecollapse (dev.filename, s^name)] := 0;
  566.    000161 1  3              call fprint (hometerm.filenum,wrong^type, @s^name, i, type);
  567.    000206 1  3              call close (dev.filenum);
  568.    000214 1  3              dev.filenum := -1;
  569.    000217 1  3              dev.error := 16;
  570.    000222 1  3           end;
  571.    000222 1  2        end; ! of proc body
  572.    000222 1  1     end;    ! of proc iodev^init
```

DEV	Variable,40	STRUCT-I	Indirect	L-007
FLAGS	Variable	INT	Direct	L-006
I	Variable	INT	Direct	L+023
SYNC	Variable	INT	Direct	L-005
S^NAME	Variable	STRING	Direct	L+001
TYPE	Variable	INT	Direct	L-004
WRONG^TYPE	Variable	STRING	Direct	P+000

```
574.    000000 0  0    proc TXREP^INIT;
575.    000000 1  0   !----------------
576.    000000 1  0     begin
577.    000000 1  1       struct startup^msg^^ (*);
578.    000000 1  1       begin
579.    000000 1  2           int id,
580.    000000 1  2               default [0:7],
581.    000000 1  2               in [0:11],
582.    000000 1  2               out [0:11];
583.    000000 1  2           string parm [0:527];
584.    000000 1  2       end;
585.    000000 1  1       struct   .st^msg (startup^msg^^);
586.    000000 1  1       struct   .receive (iodev^^);
587.    000000 1  1       int      .receive^buffer [0:127];
588.    000000 1  1       int (32) tag;
589.    000000 1  1       int      fnum,
590.    000000 1  1                error,
591.    000000 1  1                .buffer;
592.    000000 1  1       int      filecode; !of in file!
593.    000000 1  1
594.    000000 1  1       string r^error = 'P' :=
595.    000000 1  1                ["Unable to read startup message, error ?####", 0, %377],
596.    000027 1  1              ignore = 'P' :=
597.    000027 1  1                ["Parameter ignored: ", 40* ["\"], 0, %377],
598.    000066 1  1              badinfile = 'P' :=
599.    000066 1  1                ["Wrong type of TXLOG file", 0, %377];
600.    000103 1  1
601.    000103 1  1       subproc upshift (buf, l);
602.    000103 2  1   !   ------------------------
603.    000103 2  1       string .buf; int l;
604.    000103 2  1       begin
605.    000103 2  2         use x;
606.    000103 2  2         for x := 0 to l-1 do
607.    000110 2  2           if $alpha (buf [x]) then buf[x] := buf[x] land %337;
608.    000117 2  2         drop x;
609.    000117 2  2       end;

BUF                         Variable       STRING          Indirect       S-002
L                           Variable       INT             Direct         S-001

610.    000120 1  1
611.    000120 1  1       subproc scan^startup;
612.    000120 2  1   !   --------------------
613.    000120 2  1       begin
614.    000120 2  2         string .comma, .last, .first, .next, .save, save^char;
615.    000120 2  2         int i, status, found, command^length, input^length, index,
616.    000120 2  2             .p (startup^param^^);
617.    000120 2  2
618.    000120 2  2         begin
619.    000121 2  3           for i:= 0 to max^keyword do startup^parameter [i].flags := 0;
620.    000140 2  3           startup^parameter [param^default].name':='st^msg.default for 8;
621.    000145 2  3
622.    000145 2  3           @next := @st^msg.parm;
623.    000151 2  3           if not next then call print^help; ! then exit !
624.    000154 2  3           while next do
625.    000156 2  3           begin
626.    000156 2  4             scan next  until "," -> @comma;
```

```
627.    000162 2  4              scan next  while " " -> @first;
628.    000166 2  4              scan first until " " -> @last;
629.    000172 2  4              @save := @first;
630.    000174 2  4              @last := $min (@last, @comma);
631.    000203 2  4              input^length := @last - @first;
632.    000207 2  4
633.    000207 2  4              call upshift (first, input^length);
634.    000213 2  4              i := 0;
635.    000215 2  4              found := false;
636.    000217 2  4
637.    000217 2  4              while not found and (command^length := param^keywords[i]) do
638.    000227 2  4              begin
639.    000227 2  5                 if first = param^keywords [i+2] for input^length and
640.    000227 2  5                    input^length = command^length then
641.    000245 2  5                 begin
642.    000245 2  6                    found := true;
643.    000247 2  6                    scan last while " " -> @first;
644.    000253 2  6                    rscan comma [-1] while " " -> @last;
645.    000260 2  6                    input^length := $max (0, @last - @first + 1);
646.    000272 2  6                    @p := @startup^parameter [i/11 + 1];
647.    000304 2  6
648.    000304 2  6                    case param^keywords [i+1] of
649.    000313 2  6                    begin
650.    000313 2  7                       ! 0  name^ !
651.    000313 2  7                       begin  ! a device name must be supplied
652.    000313 2 CE0                         call upshift (first, input^length);
653.    000317 2  8                          if first = "#MYTERM" !for 7! then
654.    000330 2  8                             call myterm (p.name)
655.    000333 2  8                          else
656.    000335 2  8                             if input^length <>
657.    000335 2  8                               fnameexpand (first, p.name, st^msg.default)
658.    000335 2  8                             then found := false;
659.    000352 2  8                       end;
660.    000353 2 CE1                      ! 1  const^ !
661.    000353 2  7                       begin  ! a numeric parameter must be supplied
662.    000353 2  8                          if numin (first, p.value, 10, status)
663.    000353 2  8                             <> @last + 1 or status
664.    000353 2  8                          then found := false;
665.    000371 2  8                       end;
666.    000372 2 CE2                      ! 2  none^  !
667.    000372 2  7                       begin  ! no parameter
668.    000372 2  8                          if first and first <> "," then found := false;
669.    000401 2  8                       end;
670.    000403 2 CE3                      ! 3  str^   !
671.    000403 2  7                       begin  ! string of 24 char max
672.    000403 2  8                          p.name ':=' 12*["  "];
673.    000412 2  8                          if input^length then
674.    000414 2  8                             p.s^name ':=' first for
675.    000414 2  8                                           $min(input^length, 24)
676.    000414 2  8                          else found := false;
677.    000432 2  8                       end;
678.    000433 2 CE4                      otherwise found := false;
679.    000435 2  7                    end; !case
680.    000452 2  6                    if found then p.setup := true;
681.    000460 2  6                 end; !
682.    000460 2  5                 i := i + 11;
683.    000462 2  5              end; !of while not found
```

```
   684.    000463 2  4                if not found then
   685.    000465 2  4                begin
   686.    000465 2  5                   save^char := comma;
   687.    000467 2  5                   comma := 0;
   688.    000471 2  5                   call fprint (hometerm.filenum, ignore, @save);
   689.    000520 2  5                   comma := save^char;
   690.    000522 2  5                end;
   691.    000522 2  4                @next := if comma then @comma + 1 else @comma;
   692.    000531 2  4             end; !while next do
   693.    000532 2  3          end; !of setup body
   694.    000532 2  2       end; ! of subproc scan^startup;

COMMA                             Variable       STRING          Indirect       S-014
COMMAND^LENGTH                    Variable       INT             Direct         S-003
FIRST                             Variable       STRING          Indirect       S-012
FOUND                             Variable       INT             Direct         S-004
I                                 Variable       INT             Direct         S-006
INDEX                             Variable       INT             Direct         S-001
INPUT^LENGTH                      Variable       INT             Direct         S-002
LAST                              Variable       STRING          Indirect       S-013
NEXT                              Variable       STRING          Indirect       S-011
P                                 Variable,32    STRUCT-I        Indirect       S-000
SAVE                              Variable       STRING          Indirect       S-010
SAVE^CHAR                         Variable       STRING          Direct         S-007
STATUS                            Variable       INT             Direct         S-005

   695.    000556 1  1
   696.    000556 1  1       !-------------------------!
   697.    000556 1  1       !   Body of TXREP^INIT    !
   698.    000556 1  1       !-------------------------!
   699.    000556 1  1       begin
   700.    000574 1  2          call myterm (hometerm.filename);
   701.    000577 1  2          call open (hometerm.filename, hometerm.filenum); eh;
   702.    000613 1  2
   703.    000613 1  2          receive.filename ':=' "$receive                  ";
   704.    000622 1  2          call open (receive.filename, receive.filenum);
   705.    000634 1  2          call read (receive.filenum, st^msg, $len (startup^msg^^));
   706.    000645 1  2          if <> then
   707.    000646 1  2          begin
   708.    000646 1  3              call fileinfo (receive.filenum, error);
   709.    000676 1  3              if error or st^msg.id <> -1 then
   710.    000703 1  3                  call fprint (hometerm.filenum, r^error, error);
   711.    000727 1  3              call abend;
   712.    000734 1  3          end;
   713.    000734 1  2          call close (receive.filenum);
   714.    000742 1  2
   715.    000742 1  2          call scan^startup;
   716.    000743 1  2
   717.    000743 1  2          ! open the OUT file using SIO
   718.    000743 1  2          call set^file (out, init^filefcb);
   719.    000752 1  2          call set^file (out, assign^filename, @st^msg.out);
   720.    000771 1  2          call set^file (out, assign^openaccess, write^access);
   721.    001001 1  2          call set^file (out, assign^primaryextentsize, 4);
   722.    001011 1  2          call set^file (out, assign^secondaryextentsize, 4);
   723.    001021 1  2          call open^file (cfcb, out, out^buffer, 4096);
   724.    001036 1  2
   725.    001036 1  2          in.filename ':=' startup^parameter [param^txlog].name for 12;
```

```
   726.    001043 1  2          call fileinfo (,,in.filename,,,,,,,filecode);
   727.    001074 1  2          if filecode <> 6211 then
   728.    001100 1  2          begin
   729.    001100 1  3              call fprint (hometerm.filenum, badinfile);
   730.    001122 1  3              call abend;
   731.    001127 1  3          end;
   732.    001127 1  2          ! open the IN file using SBB
   733.    001127 1  2   --     call open (in.filename, in.filenum, open^exclusive,,,,, 4096);
   734.    001127 1  2          call open (in.filename, in.filenum, open^exclusive);              3.01
   735.    001150 1  2          if <> then
   736.    001151 1  2          begin
   737.    001151 1  3             call iodev^failure (in.filenum, in.filename, open^attempt);
   738.    001160 1  3             call abend;
   739.    001165 1  3          end;
   740.    001165 1  2
   741.    001165 1  2       end;
   742.    001165 1  1    end;  !proc txrep^init!

BADINFILE                      Variable       STRING          Direct         P+000
BUFFER                         Variable       INT             Indirect       L+010
ERROR                          Variable       INT             Direct         L+007
FILECODE                       Variable       INT             Direct         L+011
FNUM                           Variable       INT             Direct         L+006
IGNORE                         Variable       STRING          Direct         P+000
RECEIVE                        Variable,40    STRUCT          Indirect       L+002
RECEIVE^BUFFER                 Variable       INT             Indirect       L+003
R^ERROR                        Variable       STRING          Direct         P+000
SCAN^STARTUP                   Subproc                        %000120
STARTUP^MSG^^                  Variable       TEMPLATE ,1122
  1 ID                               0,2      INT             Direct
  1 DEFAULT[0:7]                     2,2      INT             Direct
  1 IN[0:11]                        22,2      INT             Direct
  1 OUT[0:11]                       52,2      INT             Direct
  1 PARM[0:527]                    102,1      STRING          Direct
ST^MSG                         Variable,1122  STRUCT          Indirect       L+001
TAG                            Variable       INT (32)        Direct         L+004
UPSHIFT                        Subproc                        %000103
```

```
744.    000000 0  0   proc calc^window^ts;
745.    000000 1  0   !--------------------
746.    000000 1  0   begin
747.    000000 1  1       string   badwindow = 'p' := ["Window end time is less than",
748.    000016 1  1                  " window begin time",0,%377],
749.    000030 1  1                badsyntax = 'p' := ["Illegal Window value, correct format",  1.01
750.    000052 1  1                  " is 'hh:mm:ss/hh:mm:ss'",0,%377];
751.    000067 1  1       int     .p (startup^param^^);
752.    000067 1  1       string  .s;
753.    000067 1  1       int      error;
754.    000067 1  1       int      h1,h2,m1,m2,s1,s2;
755.    000067 1  1       define   eh = if error then call debug#;
756.    000067 1  1       fixed    ts, lct;
757.    000067 1  1       int(32)  jday;
758.    000067 1  1       int     .date^time [0:7];
759.    000067 1  1       literal  lct^to^gmt = 2;
760.    000067 1  1
761.    000067 1  1       define check^error =                                          1.01
762.    000067 1  1          if error <> 0 then                                        1.01
763.    000067 1  1          begin                                                     1.01
764.    000067 1  1             call fprint (hometerm.filenum, badsyntax);             1.01
765.    000067 1  1             call abend;                                            1.01
766.    000067 1  1          end;#;                                                    1.01
767.    000067 1  1
768.    000067 1  1       @p := @startup^parameter [param^window];
769.    000076 1  1       if p.setup then
770.    000102 1  1       begin
771.    000102 1  2          @s := @p.s^name;
772.    000105 1  2          call numin (s, h1, 10, error); check^error;
773.    000145 1  2          @s := @s '+' 3;
774.    000150 1  2          call numin (s, m1, 10, error); check^error;
775.    000210 1  2          @s := @s '+' 3;
776.    000213 1  2          call numin (s, s1, 10, error); check^error;
777.    000253 1  2          @s := @s '+' 3;
778.    000256 1  2          call numin (s, h2, 10, error); check^error;
779.    000322 1  2          @s := @s '+' 3;
780.    000325 1  2          call numin (s, m2, 10, error); check^error;
781.    000366 1  2          @s := @s '+' 3;
782.    000371 1  2          call numin (s, s2, 10, error); check^error;
783.    000432 1  2
784.    000432 1  2          ! assume that the first record has been read from tx file
785.    000432 1  2          lct  := converttimestamp (test.ts^first^rec);
786.    000446 1  2          jday := interprettimestamp (lct, date^time);
787.    000454 1  2          date^time [3] := h1;
788.    000457 1  2          date^time [4] := m1;
789.    000462 1  2          date^time [5] := s1;
790.    000465 1  2          lct  := computetimestamp (date^time, error); check^error;
791.    000527 1  2          test.ts^window^begin := converttimestamp (lct, lct^to^gmt);
792.    000547 1  2
793.    000547 1  2          ! assume that midnight is NOT in the window given, otherwise
794.    000547 1  2          ! one needs to account for it by advancing the day.
795.    000547 1  2          date^time [3] := h2;
796.    000551 1  2          date^time [4] := m2;
797.    000554 1  2          date^time [5] := s2;
798.    000557 1  2          lct  := computetimestamp (date^time, error); check^error;
799.    000621 1  2          test.ts^window^end   := converttimestamp (lct, lct^to^gmt);
800.    000635 1  2          test.len^window := $dbl((test.ts^window^end - test.ts^window^begin)
```

```
  801.    000635 1  2                                  / 1000000F); ! seconds
  802.    000653 1  2          if test.len^window < 0d then
  803.    000660 1  2          begin
  804.    000660 1  3              call fprint (hometerm.filenum, badwindow);
  805.    000703 1  3              call abend;
  806.    000710 1  3          end;
  807.    000710 1  2
  808.    000710 1  2       end else ! defaulting to all records in tx log file !
  809.    000711 1  1       begin
  810.    000711 1  2
  811.    000711 1  2          test.ts^window^begin := test.ts^first^rec;
  812.    000717 1  2          test.ts^window^end   := juliantimestamp; !now
  813.    000730 1  2
  814.    000730 1  2       end;
  815.    000730 1  1   end;  !proc calc^window^ts!
```

```
BADSYNTAX                      Variable       STRING          Direct          P+000
BADWINDOW                      Variable       STRING          Direct          P+000
CHECK^ERROR                    Define                         IF ERROR <> 0 THEN BEGIN CALL FPRINT (HOMETERM.FILENUM, BADS
DATE^TIME                      Variable       INT             Indirect        L+024
EH                             Define                         IF ERROR THEN CALL DEBUG
ERROR                          Variable       INT             Direct          L+003
H1                             Variable       INT             Direct          L+004
H2                             Variable       INT             Direct          L+005
JDAY                           Variable       INT (32)        Direct          L+022
LCT                            Variable       FIXED (0)       Direct          L+016
LCT^TO^GMT                     Literal        INT             %000002
M1                             Variable       INT             Direct          L+006
M2                             Variable       INT             Direct          L+007
P                              Variable,32    STRUCT-I        Indirect        L+001
S                              Variable       STRING          Indirect        L+002
S1                             Variable       INT             Direct          L+010
S2                             Variable       INT             Direct          L+011
TS                             Variable       FIXED (0)       Direct          L+012
```

```
  817.    000000 0  0   proc read^rec (rec, error);                                    2.02
  818.    000000 1  0   !-------------------------                                     2.02
  819.    000000 1  0   struct .rec (tx^def);                                         2.02
  820.    000000 1  0   int .error;                                                   2.02
  821.    000000 1  0   begin
  822.    000000 1  1       int count^read;
  823.    000000 1  1
  824.    000000 1  1       subproc read^block;
  825.    000000 2  1       !-----------------
  826.    000000 2  1       begin
  827.    000000 2  2           call read (in.filenum, txlog^buf, txlog^bufsize, count^read);
  828.    000011 2  2           if < then
  829.    000012 2  2           begin
  830.    000012 2  3              call iodev^failure (in.filenum, in.filename, read^attempt);
  831.    000021 2  3              call abend;
  832.    000026 2  3           end else if > then        ! EOF
  833.    000030 2  2           if first^block^read then  ! when reading 1st block is fatal
  834.    000032 2  2           begin
  835.    000032 2  3              call iodev^failure (in.filenum, in.filename, read^attempt);
  836.    000041 2  3              call abend;
  837.    000046 2  3           end else                  ! not the first block
  838.    000047 2  2           begin                     ! returns EOF error code to caller
  839.    000047 2  3              error := 1;
  840.    000051 2  3              return;
  841.    000052 2  3           end;
  842.    000052 2  2           ! block just read in is OK, reset everything.
  843.    000052 2  2           first^block^read := false;
  844.    000054 2  2           txlog^rec := 1;
  845.    000056 2  2           @txlog^bufptr := @txlog^buf;
  846.    000060 2  2           txlog^maxrec := count^read / $len (tx^def);
  847.    000064 2  2       end; !subproc read^block!
  848.    000065 1  1
  849.    000065 1  1
  850.    000065 1  1       txlog^rec := txlog^rec + 1;
  851.    000070 1  1       if txlog^rec > txlog^maxrec then
  852.    000074 1  1           call read^block;
  853.    000075 1  1       if error then return; ! EOF encountered !
  854.    000100 1  1       rec ':=' txlog^bufptr for $len (tx^def)/2;
  855.    000104 1  1       @txlog^bufptr := @txlog^bufptr '+' $len (tx^def)/2;
  856.    000107 1  1       error := 0;
  857.    000111 1  1
  858.    000111 1  1   end; !proc read^rec!

COUNT^READ                  Variable       INT            Direct         L+001
ERROR                       Variable       INT            Indirect       L-003
READ^BLOCK                  Subproc                       %000000
REC                         Variable,40    STRUCT-I       Indirect       L-004
```

```
860.    000000 0  0    proc scan^file;
861.    000000 1  0   !---------------
862.    000000 1  0   begin
863.    000000 1  1
864.    000000 1  1   ! ?source TXTAL                                                    2.02
865.    000000 1  1      struct .txrec (tx^def);       ! tx stats record
866.    000000 1  1
867.    000000 1  1      int      i, error := 0;                                        2.02
868.    000000 1  1      int(32)  delta;
869.    000000 1  1      real     delta^r;
870.    000000 1  1      real     resp^time;
871.    000000 1  1      int     .p (startup^param^^);
872.    000000 1  1      int(32)  atime; ! inter-arrival time !                         2.03
873.    000000 1  1      real     atime^r;                                              2.03
874.    000000 1  1
875.    000000 1  1      !--------------------!
876.    000000 1  1      !    Init RT table   !
877.    000000 1  1      !--------------------!
878.    000000 1  1      rt ':=' ($len (rt)/2) * [0];
879.    000015 1  1      rt.minimum :=  99999999d;
880.    000021 1  1      rt.maximum := -99999999d;
881.    000025 1  1      @p := @startup^parameter [param^ccell];
882.    000030 1  1      rt.cellsize := if p.setup then $abs (p.value) else 50; ! milliseconds
883.    000043 1  1      rt.numcell :=  cellcount; ! 1000
884.    000046 1  1      rt.lobound :=          0d;
885.    000051 1  1      rt.upbound := rt.lobound + $dbl (rt.numcell) * $dbl (rt.cellsize);
886.    000073 1  1
887.    000073 1  1      !--------------------!                                         2.03
888.    000073 1  1      !    Init AT table   !                                         2.03
889.    000073 1  1      !--------------------!                                         2.03
890.    000073 1  1      at ':=' ($len (at)/2) * [0];                                  2.03
891.    000102 1  1      at.minimum :=  99999999d;                                     2.03
892.    000106 1  1      at.maximum := -99999999d;                                     2.03
893.    000112 1  1      @p := @startup^parameter [param^ccella];                       2.03
894.    000115 1  1      at.cellsize := if p.setup then $abs (p.value) else 500;! milliseconds 2.03
895.    000130 1  1      at.numcell :=  cellcount; ! 1000                              2.03
896.    000133 1  1      at.lobound :=          0d;                                    2.03
897.    000136 1  1      at.upbound := at.lobound + $dbl (at.numcell) * $dbl (at.cellsize);  2.03
898.    000160 1  1
899.    000160 1  1      !--------------------!
900.    000160 1  1      !  Read First Record !
901.    000160 1  1      !--------------------!
902.    000160 1  1      call read^rec (txrec, error);                                 2.02
903.    000164 1  1   !  call read (in.filenum, txrec, $len (tx^def));
904.    000164 1  1   !  if <> then
905.    000164 1  1   !  begin
906.    000164 1  1   !     call iodev^failure (in.filenum, in.filename, read^attempt);
907.    000164 1  1   !     call abend;
908.    000164 1  1   !  end;
909.    000164 1  1
910.    000164 1  1      !--------------------!
911.    000164 1  1      !    Global Info     !
912.    000164 1  1      !--------------------!
913.    000164 1  1      test.terminals := txrec.term^count;
914.    000166 1  1      test.arrival^rate := $abs( txrec.arrival^rate );
915.    000174 1  1      if test.arrival^rate then ! pos=constant, neg=random !
916.    000176 1  1         test.arrival^type := if txrec.arrival^rate > 0 then
```

```
917.    000176 1  1                                  constant
918.    000176 1  1                                else random
919.    000176 1  1      else ! maximum rate !
920.    000215 1  1         test.arrival^type := maximum;
921.    000220 1  1      test.ts^first^rec := txrec.send^timestamp;
922.    000226 1  1
923.    000226 1  1      !--------------------!
924.    000226 1  1      ! Init THINK & DELAY !
925.    000226 1  1      !--------------------!
926.    000226 1  1      if test.arrival^type <> maximum then
927.    000232 1  1      begin
928.    000232 1  2          think.minimum := dilay.minimum :=  99999999d;
929.    000237 1  2          think.maximum := dilay.maximum := -99999999d;
930.    000245 1  2      end;
931.    000245 1  1
932.    000245 1  1      !--------------------!
933.    000245 1  1      ! Window Timestamps  !
934.    000245 1  1      !--------------------!
935.    000245 1  1      call calc^window^ts;           ! to compute ts^window^begin/end
936.    000246 1  1
937.    000246 1  1      !--------------------!
938.    000246 1  1      ! Loop thru all recs !
939.    000246 1  1      !--------------------!
940.    000246 1  1      error := 0;
941.    000250 1  1      while not error do
942.    000252 1  1      begin
943.    000252 1  2         if txrec.send^timestamp < test.ts^window^begin OR
944.    000252 1  2            txrec.send^timestamp > test.ts^window^end
945.    000252 1  2         then
946.    000271 1  2            throw^count := throw^count + 1d
947.    000271 1  2         else begin
948.    000276 1  3            !--------------------!
949.    000276 1  3            !   Process Record   !
950.    000276 1  3            !--------------------!
951.    000276 1  3            rt.count := rt.count + 1d;
952.    000303 1  3
953.    000303 1  3            !--------------------!
954.    000303 1  3            ! RESPONSE TIME stats!
955.    000303 1  3            !--------------------!
956.    000303 1  3            ! Frequency dist: if a value falls on the boundary of two
957.    000303 1  3            ! cells, it is recorded in the cell below.
958.    000303 1  3            if txrec.response^time <= rt.lobound then
959.    000312 1  3               i := 0                  ! first cell
960.    000312 1  3            else if txrec.response^time > rt.upbound then
961.    000324 1  3               i := rt.numcell         ! last cell
962.    000324 1  3            else
963.    000330 1  3               i := (txrec.response^time + $dbl (rt.cellsize)
964.    000330 1  3                     - 1d - rt.lobound) '/' rt.cellsize;
965.    000351 1  3
966.    000351 1  3            rt.freq [i] := rt.freq [i] + 1;  ! tabulate in cell
967.    000356 1  3            resp^time := $flt (txrec.response^time);
968.    000363 1  3            rt.sum    := rt.sum + resp^time;
969.    000367 1  3            rt.sum^2  := rt.sum^2 + resp^time * resp^time;
970.    000403 1  3            if txrec.response^time < rt.minimum then
971.    000411 1  3               rt.minimum := txrec.response^time;
972.    000414 1  3            if txrec.response^time > rt.maximum then
973.    000423 1  3               rt.maximum := txrec.response^time;
```

```
 974.    000426 1  3
 975.    000426 1  3          !--------------------------!                                  2.03
 976.    000426 1  3          ! INTER-ARRIVAL TIME stats !                                  2.03
 977.    000426 1  3          !--------------------------!                                  2.03
 978.    000426 1  3          if txrec.response^time > txrec.next^arrival^time then         2.03
 979.    000436 1  3             atime := txrec.response^time                               2.03
 980.    000436 1  3          else                                                          2.03
 981.    000443 1  3             atime := txrec.next^arrival^time;                          2.03
 982.    000447 1  3          ! Frequency dist: if a value falls on the boundary of two     2.03
 983.    000447 1  3          ! cells, it is recorded in the cell below.                    2.03
 984.    000447 1  3          if atime <= at.lobound then                                   2.03
 985.    000454 1  3             i := 0                ! first cell                         2.03
 986.    000454 1  3          else if atime > at.upbound then                               2.03
 987.    000465 1  3             i := at.numcell       ! last cell                          2.03
 988.    000465 1  3          else                                                          2.03
 989.    000471 1  3             i := (atime + $dbl (at.cellsize)                           2.03
 990.    000471 1  3                   - 1d - at.lobound) '/' at.cellsize;                  2.03
 991.    000510 1  3                                                                        2.03
 992.    000510 1  3          at.freq [i] := at.freq [i] + 1;  ! tabulate in cell           2.03
 993.    000515 1  3          atime^r   := $flt (atime);                                    2.03
 994.    000520 1  3          at.sum    := at.sum + atime^r;                                2.03
 995.    000525 1  3          at.sum^2  := at.sum^2 + atime^r * atime^r;                    2.03
 996.    000541 1  3          if atime < at.minimum then                                    2.03
 997.    000546 1  3             at.minimum := atime;                                       2.03
 998.    000550 1  3          if atime > at.maximum then                                    2.03
 999.    000555 1  3             at.maximum := atime;                                       2.03
1000.    000557 1  3
1001.    000557 1  3          !--------------------------!
1002.    000557 1  3          ! DELAY or THINK^TIME stats !
1003.    000557 1  3          !--------------------------!
1004.    000557 1  3          if test.arrival^type <> maximum then
1005.    000563 1  3          begin
1006.    000563 1  4             delta := txrec.next^arrival^time - txrec.response^time;
1007.    000573 1  4             delta^r := $flt (delta);
1008.    000576 1  4             if delta > 0d then
1009.    000602 1  4             begin
1010.    000602 1  5                think.count := think.count + 1d;
1011.    000610 1  5                think.sum   := think.sum + delta^r;
1012.    000615 1  5                think.sum^2 := think.sum^2 + delta^r * delta^r;
1013.    000631 1  5                if delta < think.minimum then think.minimum := delta;
1014.    000640 1  5                if delta > think.maximum then think.maximum := delta;
1015.    000647 1  5             end else
1016.    000650 1  4             begin
1017.    000650 1  5                delta       := $abs (delta);
1018.    000654 1  5                delta^r     := $abs (delta^r);
1019.    000660 1  5                dilay.count := dilay.count + 1d;
1020.    000666 1  5                dilay.sum   := dilay.sum + delta^r;
1021.    000673 1  5                dilay.sum^2 := dilay.sum^2 + delta^r * delta^r;
1022.    000702 1  5                if delta < dilay.minimum then dilay.minimum := delta;
1023.    000711 1  5                if delta > dilay.maximum then dilay.maximum := delta;
1024.    000720 1  5             end;
1025.    000720 1  4          end;
1026.    000720 1  3
1027.    000720 1  3       end; !process record!
1028.    000720 1  2
1029.    000720 1  2       !--------------------!
1030.    000720 1  2       !  Read next record  !
```

```
1031.  000720 1  2        !--------------------!
1032.  000720 1  2        call read^rec (txrec, error);                                  2.02
1033.  000724 1  2  !     call read (in.filenum, txrec, $len (tx^def));
1034.  000724 1  2  !     call fileinfo (in.filenum, error);
1035.  000724 1  2
1036.  000724 1  2     end; !while!
1037.  000725 1  1
1038.  000725 1  1     if error <> 1 then
1039.  000730 1  1     begin
1040.  000730 1  2         call iodev^failure (in.filenum, in.filename, read^attempt);
1041.  000737 1  2         call abend;
1042.  000744 1  2     end;
1043.  000744 1  1
1044.  000744 1  1     test.ts^last^rec := txrec.send^timestamp;
1045.  000752 1  1     test.len^test := $dbl ((test.ts^last^rec - test.ts^first^rec)
1046.  000752 1  1                              / 1000000f); ! seconds
1047.  000770 1  1     if    not startup^parameter [param^window].setup !no window passed!
1048.  000770 1  1        OR (test.ts^window^begin <= test.ts^first^rec AND
1049.  000770 1  1            test.ts^window^end   >= test.ts^last^rec) !test is in window
1050.  000770 1  1     then
1051.  001014 1  1        test.len^window := test.len^test;
1052.  001022 1  1
1053.  001022 1  1  end; !proc scan^file!
```

ATIME	Variable	INT (32)	Direct	L+013
ATIME^R	Variable	REAL (32)	Direct	L+015
DELTA	Variable	INT (32)	Direct	L+004
DELTA^R	Variable	REAL (32)	Direct	L+006
ERROR	Variable	INT	Direct	L+003
I	Variable	INT	Direct	L+002
P	Variable,32	STRUCT-I	Indirect	L+012
RESP^TIME	Variable	REAL (32)	Direct	L+010
TXREC	Variable,40	STRUCT	Indirect	L+001

```
1055.    000000 0  0    proc et1^stats;
1056.    000000 1  0   !--------------
1057.    000000 1  0   begin
1058.    000000 1  1       int       i, j;               ! loop indices
1059.    000000 1  1       int(32)   total;
1060.    000000 1  1       real      freq^pct^cum;
1061.    000000 1  1       real      sigma^2;            ! variance of response time
1062.    000000 1  1       real      intlen;             ! width of a single cell in sec
1063.    000000 1  1       int      .done [0:9];         ! flags
1064.    000000 1  1
1065.    000000 1  1       et1.thruput := 0.0e0;
1066.    000007 1  1       if test.len^window > 1d then
1067.    000015 1  1           et1.thruput := $fltr (rt.count) / $fltr (test.len^window); ! tx/sec
1068.    000026 1  1
1069.    000026 1  1       !-------------------!
1070.    000026 1  1       !    Response Time   !
1071.    000026 1  1       !-------------------!
1072.    000026 1  1       et1.rt^avg := et1.rt^std := sigma^2 := 0.0e0;
1073.    000036 1  1       if rt.count > 1d then
1074.    000043 1  1       begin
1075.    000043 1  2           et1.rt^avg := (rt.sum / $fltr (rt.count)) / 1.0e3; ! sec
1076.    000054 1  2           sigma^2 := (rt.sum^2 - (rt.sum * rt.sum / $fltr (rt.count))) /
1077.    000054 1  2                      $fltr (rt.count - 1d);
1078.    000105 1  2           et1.rt^std := if $abs (sigma^2) < 1.0e-6 or sigma^2 < 0.0e0
1079.    000105 1  2                      then 0.0e0
1080.    000105 1  2                      else sqrt (sigma^2) / 1.0e3; ! sec
1081.    000132 1  2       end;
1082.    000132 1  1
1083.    000132 1  1       et1.rt^min := $fltr (rt.minimum) / 1.0e3; ! sec
1084.    000142 1  1       et1.rt^max := $fltr (rt.maximum) / 1.0e3; ! sec
1085.    000152 1  1
1086.    000152 1  1       freq^pct^cum := 0.0e0;
1087.    000154 1  1       j := 1;
1088.    000156 1  1       i := -1;
1089.    000160 1  1       total := 0d;
1090.    000162 1  1       done ':=' 10 * [0];
1091.    000171 1  1       intlen := $flt (rt.cellsize) / 1.0e3;  ! cell size in sec
1092.    000200 1  1       if rt.count > 1d then
1093.    000205 1  1
1094.    000205 1  1       do begin
1095.    000205 1  2           i := i + 1;
1096.    000207 1  2           freq^pct^cum := freq^pct^cum + (1.0e2 *
1097.    000207 1  2                           $flt (rt.freq [i]) / $flt (rt.count));
1098.    000226 1  2           j := $int (freq^pct^cum / 10e0);
1099.    000234 1  2           if j > 0 then
1100.    000237 1  2           begin
1101.    000237 1  3               if j < 9  and (freq^pct^cum >= (10.0e0 * $flt (j))  and
1102.    000237 1  3                              freq^pct^cum <  (10.0e0 * $flt (j+1)))
1103.    000237 1  3                      and (not done [j]) then
1104.    000274 1  3               begin
1105.    000274 1  4                   et1.rt^dist [j] := intlen * $flt (i);      ! resp time in sec
1106.    000306 1  4                   et1.rt^pct  [j] := freq^pct^cum;           ! actual %
1107.    000315 1  4                   done [j] := true;                          ! first value only
1108.    000317 1  4               end;
1109.    000317 1  3               if j = 9  and (freq^pct^cum >= 90.0e0  and
1110.    000317 1  3                              freq^pct^cum <  95.0e0)
1111.    000317 1  3                      and (not done [9]) then
```

```
1112.  000337 1  3              begin
1113.  000337 1  4                  et1.rt^dist [j] := intlen * $flt (i);      ! resp time in sec
1114.  000351 1  4                  et1.rt^pct  [j] := freq^pct^cum;           ! actual %
1115.  000360 1  4                  done [9] := true;                          ! 90 percentile
1116.  000362 1  4              end;
1117.  000362 1  3              if j >= 9  and freq^pct^cum >= 95.0e0
1118.  000362 1  3                       and (not done [0]) then
1119.  000374 1  3              begin
1120.  000374 1  4                  et1.rt^dist [0] := intlen * $flt (i);      ! resp time in sec
1121.  000404 1  4                  et1.rt^pct  [0] := freq^pct^cum;           ! actual %
1122.  000407 1  4                  done [0] := true;                          ! 95 percentile
1123.  000411 1  4              end;
1124.  000411 1  3          end; !if j > 0!
1125.  000411 1  2
1126.  000411 1  2          total := total + $dbl (rt.freq [i]);
1127.  000422 1  2      end until total >= rt.count;
1128.  000427 1  1
1129.  000427 1  1      !--------------------!                                                  2.03
1130.  000427 1  1      !    InterArv Time   !                                                  2.03
1131.  000427 1  1      !--------------------!                                                  2.03
1132.  000427 1  1      at^stats.avg := at^stats.std := sigma^2 := 0.0e0;                       2.03
1133.  000437 1  1      if rt.count > 1d then                                                   2.03
1134.  000444 1  1      begin                                                                  2.03
1135.  000444 1  2          at^stats.avg := (at.sum / $fltr (rt.count)) / 1.0e3; ! sec         2.03
1136.  000456 1  2          sigma^2 := (at.sum^2 - (at.sum * at.sum / $fltr (rt.count))) /     2.03
1137.  000456 1  2                       $fltr (rt.count - 1d);                             2.03
1138.  000507 1  2          at^stats.std := if $abs (sigma^2) < 1.0e-6 or sigma^2 < 0.0e0      2.03
1139.  000507 1  2                       then 0.0e0                                         2.03
1140.  000507 1  2                       else sqrt (sigma^2) / 1.0e3; ! sec                 2.03
1141.  000534 1  2      end;                                                                   2.03
1142.  000534 1  1                                                                             2.03
1143.  000534 1  1      at^stats.min := $fltr (at.minimum) / 1.0e3; ! sec                      2.03
1144.  000544 1  1      at^stats.max := $fltr (at.maximum) / 1.0e3; ! sec                      2.03
1145.  000554 1  1
1146.  000554 1  1      !--------------------!
1147.  000554 1  1      !      Think Time    !
1148.  000554 1  1      !--------------------!
1149.  000554 1  1      think.avg := think.std := sigma^2 := 0.0e0;
1150.  000563 1  1      if think.count > 1d then
1151.  000570 1  1      begin
1152.  000570 1  2          think.avg := (think.sum / $fltr (think.count)) / 1.0e3; ! sec
1153.  000602 1  2          sigma^2   := (think.sum^2 -
1154.  000602 1  2                        (think.sum * think.sum / $fltr (think.count))) /
1155.  000602 1  2                        $fltr (think.count - 1d);
1156.  000642 1  2          think.std := if $abs (sigma^2) < 1.0e-6 or sigma^2 < 0.0e0
1157.  000642 1  2                       then 0.0e0
1158.  000642 1  2                       else sqrt (sigma^2) / 1.0e3; ! sec
1159.  000667 1  2      end;
1160.  000667 1  1
1161.  000667 1  1      !--------------------!
1162.  000667 1  1      !      Delay Time    !
1163.  000667 1  1      !--------------------!
1164.  000667 1  1      dilay.avg := dilay.std := sigma^2 := 0.0e0;
1165.  000677 1  1      if dilay.count > 1d then
1166.  000704 1  1      begin
1167.  000704 1  2          dilay.avg := (dilay.sum / $fltr (dilay.count)) / 1.0e3; ! sec
1168.  000716 1  2          sigma^2   := (dilay.sum^2 -
```

25

```
 1169.    000716 1  2                          (dilay.sum * dilay.sum / $fltr (dilay.count))) /
 1170.    000716 1  2                          $fltr (dilay.count - 1d);
 1171.    000751 1  2           dilay.std := if $abs (sigma^2) < 1.0e-6 or sigma^2 < 0.0e0
 1172.    000751 1  2                        then 0.0e0
 1173.    000751 1  2                        else sqrt (sigma^2) / 1.0e3; ! sec
 1174.    000776 1  2       end;
 1175.    000776 1  1
 1176.    000776 1  1   end; !proc et1^stats!
```

DONE	Variable	INT	Indirect	L+013
FREQ^PCT^CUM	Variable	REAL (32)	Direct	L+005
I	Variable	INT	Direct	L+001
INTLEN	Variable	REAL (32)	Direct	L+011
J	Variable	INT	Direct	L+002
SIGMA^2	Variable	REAL (32)	Direct	L+007
TOTAL	Variable	INT (32)	Direct	L+003

```
1178.    000000 0  0   ?IF 15
1179.    000000 0  0   ------------------------------------------------------------------------
1180.    000000 0  0   *   ET1 Report (RTE System) - Performance Analysis and Measurement   *
1181.    000000 0  0   ------------------------------------------------------------------------
1182.    000000 0  0
1183.    000000 0  0   ----- PARAMETERS -----------------------------------------------------
1184.    000000 0  0
1185.    000000 0  0            1         2         3         4         5         6
1186.    000000 0  0   123456789-123456789-123456789-123456789-123456789-123456789-123456789-
1187.    000000 0  0   Test Title                =   B30 - Standard
1188.    000000 0  0   Test Date                 =   86-09-29
1189.    000000 0  0   Test Length               =   13:12:00 - 13:30:35  (2390 sec)
1190.    000000 0  0   Window Length             =   13:14:30 - 13:25:00  (1987 sec)
1191.    000000 0  0   Tx Log File               =   \VIEW.$SPOOL.ET1TXLOG.L2R
1192.    000000 0  0
1193.    000000 0  0   Terminals/X25 Line        =       30 #
1194.    000000 0  0   Tx Arrival Process        =  MAXIMUM          {CONSTANT, RANDOM}
1195.    000000 0  0   Tx Arrival Rate/X25 Line  =   15.500 tps
1196.    000000 0  0   Tx Arrival Rate/Terminal  =    0.234 tps
1197.    000000 0  0   Tx Inter-Arrival Time     =    6.000 sec
1198.    000000 0  0
1199.    000000 0  0   ----- MEASUREMENT RESULTS (vs 1.02) ---------------------------------
1200.    000000 0  0
1201.    000000 0  0   Tx Throughput             =   15.500 tps
1202.    000000 0  0   Tx Response Time Average  =    0.780 sec
1203.    000000 0  0                   Std Dev   =    0.234 sec
1204.    000000 0  0                   Minimum   =    0.345 sec
1205.    000000 0  0                   Maximum   =    3.789 sec
1206.    000000 0  0                   Count     =    12345 #
1207.    000000 0  0
1208.    000000 0  0            1         2         3         4         5         6
1209.    000000 0  0   123456789-123456789-123456789-123456789-123456789-123456789-123456789-
1210.    000000 0  0   Tx Response Time 10%      =    0.050 sec  ( 10.4%)
1211.    000000 0  0                   20%      =    0.100 sec  ( 21.6%)
1212.    000000 0  0                   30%      =    0.250 sec  ( 35.8%)
1213.    000000 0  0                   40%      =    0.300 sec  ( 43.3%)
1214.    000000 0  0                   50%      =    0.450 sec  ( 54.8%)
1215.    000000 0  0                   60%      =    0.650 sec  ( 67.9%)
1216.    000000 0  0                   70%      =    0.750 sec  ( 78.5%)
1217.    000000 0  0                   80%      =    1.100 sec  ( 88.1%)
1218.    000000 0  0                   90%      =    1.200 sec  ( 91.5%)
1219.    000000 0  0                   95%      =    1.245 sec  ( 94.5%)
1220.    000000 0  0
1221.    000000 0  0          Tx Think Average  =    0.098 sec
1222.    000000 0  0                   Std Dev   =    0.008 sec
1223.    000000 0  0                   Minimum   =    0.345 sec
1224.    000000 0  0                   Maximum   =    3.789 sec
1225.    000000 0  0                   Count     =    23456 #    ( 89.0%)
1226.    000000 0  0
1227.    000000 0  0          Tx Delay Average  =    0.098 sec
1228.    000000 0  0                   Std Dev   =    0.008 sec
1229.    000000 0  0                   Minimum   =    0.345 sec
1230.    000000 0  0                   Maximum   =    3.789 sec
1231.    000000 0  0                   Count     =    23456 #    ( 11.0%)
1232.    000000 0  0
1233.    000000 0  0   ------------------------------------------------------------------------
1234.    000000 0  0   ?ENDIF 15
```

```
1236.  000000 0  0   proc report;
1237.  000000 1  0   !-----------
1238.  000000 1  0   begin
1239.  000000 1  1
1240.  000000 1  1       string    header = 'P' := ["*   ET1 Report (RTE System) - ",
1241.  000017 1  1                 "Performance Analysis and Measurement   *"];
1242.  000043 1  1
1243.  000043 1  1       define    smear (string^array, character, length) =
1244.  000043 1  1                 begin
1245.  000043 1  1                    string^array := character;
1246.  000043 1  1                    string^array[1] ':=' string^array for (length-1);
1247.  000043 1  1                 end#;
1248.  000043 1  1       define    blank (str, strlen) = smear (str, " ", strlen)#;
1249.  000043 1  1       define    blank^line  = begin line^ := " ";
1250.  000043 1  1                               call write^file (out, line, 1); end#;
1251.  000043 1  1       define    hyphen^line = begin smear (line^,"-",70);
1252.  000043 1  1                               write^it; end#;                          1.00
1253.  000043 1  1       define    write^it    = begin                                  1.00
1254.  000043 1  1                                 line^ [77] '=:' line^ [69] for 70;   1.00
1255.  000043 1  1                                 line^ ':=' "        ";               1.00
1256.  000043 1  1                                 call write^file (out, line, 78);     1.00
1257.  000043 1  1                               end#;                                  1.00
1258.  000043 1  1
1259.  000043 1  1       string   .p;
1260.  000043 1  1       int       linelen;
1261.  000043 1  1       int      .line [0:66];
1262.  000043 1  1       string   .line^ := @line '<<' 1;
1263.  000043 1  1       fixed     lct;                    ! local civil time
1264.  000043 1  1       int(32)   jday;                   ! dummy
1265.  000043 1  1       int      .date^time[0:7];         ! for use with interprettimestamp
1266.  000043 1  1       int       err;
1267.  000043 1  1       string   .eformat [0:11];         !  "M<ZZZ9.999> " or "M<ZZ9.9>    "
1268.  000043 1  1       string   .iformat [0:199];
1269.  000043 1  1       string   .num^str[0:7];
1270.  000043 1  1       int      .num^wrd := @num^str'>>'1;
1271.  000043 1  1       real      real^temp;
1272.  000043 1  1       int      .netname [0:11];
1273.  000043 1  1       real      tput^per^terminal,                                    2.03
1274.  000043 1  1                 t1, t2,                                               2.03
1275.  000043 1  1                 terms,                                                2.03
1276.  000043 1  1                 prob;                                                 2.03
1277.  000043 1  1
1278.  000043 1  1       int subproc real^to^ascii (real^num, real^str, flag) variable;
1279.  000043 2  1   !   ------------------------------------------------------------
1280.  000043 2  1           real    real^num;
1281.  000043 2  1           string .real^str;
1282.  000043 2  1           int     flag;
1283.  000043 2  1       begin
1284.  000043 2  2           int     len, err,
1285.  000043 2  2                   s := 0,
1286.  000043 2  2                  .format;
1287.  000043 2  2           literal real32 = 8;       ! data type for real
1288.  000043 2  2           struct  vle^def (*);      ! 4 word form of variable list entry
1289.  000043 2  2           begin
1290.  000043 2  3               int    elm^ptr;
1291.  000043 2  3               string elm^scale, elm^type;
1292.  000043 2  3               int    elm^len;
```

```
  1293.    000043 2  3              int    elm^occurs;
  1294.    000043 2  3          end;
  1295.    000043 2  2          struct vle (vle^def);
  1296.    000043 2  2
  1297.    000043 2  2          if not $param (flag) then
  1298.    000052 2  2            eformat ':=' "M<ZZZ9.999> " ! time in seconds
  1299.    000052 2  2          else
  1300.    000063 2  2            eformat ':=' "M<ZZ9.9>    ";! percent
  1301.    000073 2  2
  1302.    000073 2  2          len := formatconvert (iformat,200,eformat,12,s,s,1);
  1303.    000105 2  2          if len <= 0 then call debug;
  1304.    000111 2  2          @format := @iformat '>>' 1;   ! word addess of array !
  1305.    000114 2  2
  1306.    000114 2  2          vle ':=' [0, 0 '<<' 8 '+' real32];
  1307.    000123 2  2          vle.elm^ptr := @real^num;
  1308.    000125 2  2          vle.elm^len := 4; !bytes long!
  1309.    000127 2  2          vle.elm^occurs := 1;
  1310.    000131 2  2          err := formatdata (real^str, 8!chars!, 1, len, format, vle, 1, 0);
  1311.    000144 2  2          if err then call debug;
  1312.    000147 2  2          return len;
  1313.    000152 2  2      end; !subproc real^to^ascii!

ERR                            Variable       INT             Direct         S-006
FLAG                           Variable       INT             Direct         S-012
FORMAT                         Variable       INT             Indirect       S-004
LEN                            Variable       INT             Direct         S-007
REAL32                         Literal        INT             %000010
REAL^NUM                       Variable       REAL (32)       Direct         S-015
REAL^STR                       Variable       STRING          Indirect       S-013
S                              Variable       INT             Direct         S-005
VLE                            Variable,10    STRUCT          Direct         S-003
VLE^DEF                        Variable       TEMPLATE ,10
  1 ELM^PTR                          0,2      INT             Direct
  1 ELM^SCALE                        2,1      STRING          Direct
  1 ELM^TYPE                         3,1      STRING          Direct
  1 ELM^LEN                          4,2      INT             Direct
  1 ELM^OCCURS                       6,2      INT             Direct

  1314.    000170 1  1
  1315.    000170 1  1      subproc plot^line (rtime, count, pct, pct^cum, maxfreq, j);
  1316.    000170 2  1  !   ------------------------------------------------------------
  1317.    000170 2  1          real    rtime, pct, pct^cum;
  1318.    000170 2  1          int(32) count, maxfreq;
  1319.    000170 2  1          int     j;
  1320.    000170 2  1      begin
  1321.    000170 2  2          int(32) maxpoints := 42d;
  1322.    000170 2  2          int     num^star;
  1323.    000170 2  2
  1324.    000170 2  2          blank (line^, 70);
  1325.    000203 2  2          call real^to^ascii (rtime / 1e3, p); ! in seconds
  1326.    000215 2  2          call dascii (count, -1, p [13]);
  1327.    000224 2  2          call real^to^ascii (pct    , p [15], 0);
  1328.    000234 2  2          call real^to^ascii (pct^cum, p [21], 0);
  1329.    000244 2  2          p [27] := "|";
  1330.    000247 2  2
  1331.    000247 2  2          num^star := $intr ( count * maxpoints / maxfreq );
  1332.    000256 2  2          if num^star then
```

```
 1333.  000260 2  2              p[28] ':=' "*" & p [28] for num^star - 1;
 1334.  000275 2  2
 1335.  000275 2  2          if j >= rt.numcell then  ! overflow interval !                    2.03
 1336.  000302 2  2          begin                                                             2.03
 1337.  000302 2  3              line^ [77] ':=' line^ [69] for 70;                            2.03
 1338.  000310 2  3              line^ ':=' "    >   "; ! to indicate overflow cell            2.03
 1339.  000320 2  3              call write^file (out, line, 78);                              2.03
 1340.  000334 2  3              return;                                                       2.03
 1341.  000337 2  3          end;                                                              2.03
 1342.  000337 2  2
 1343.  000337 2  2          write^it;
 1344.  000370 2  2      end; !subproc plot^line!

COUNT                           Variable       INT (32)        Direct          S-014
J                               Variable       INT             Direct          S-004
MAXFREQ                         Variable       INT (32)        Direct          S-006
MAXPOINTS                       Variable       INT (32)        Direct          S-002
NUM^STAR                        Variable       INT             Direct          S-000
PCT                             Variable       REAL (32)       Direct          S-012
PCT^CUM                         Variable       REAL (32)       Direct          S-010
RTIME                           Variable       REAL (32)       Direct          S-016

 1345.  000403 1  1
 1346.  000403 1  1      subproc rtplot;
 1347.  000403 2  1  !   ---------------
 1348.  000403 2  1      begin
 1349.  000403 2  2          int     i, j,
 1350.  000403 2  2                 .s (startup^param^^),
 1351.  000403 2  2                  pcellsize,
 1352.  000403 2  2                  n,
 1353.  000403 2  2                  count;
 1354.  000403 2  2
 1355.  000403 2  2          int(32) rtime,
 1356.  000403 2  2                  maxfreq,
 1357.  000403 2  2                  total,
 1358.  000403 2  2                  stopcount;
 1359.  000403 2  2
 1360.  000403 2  2          real    pct,
 1361.  000403 2  2                  pct^cum;
 1362.  000403 2  2
 1363.  000403 2  2  ?IF 15
 1364.  000403 2  2
 1365.  000403 2  2                    * ET1 Response Time Distribution *
 1366.  000403 2  2
 1367.  000403 2  2  ------------------------------------------------------------------------
 1368.  000403 2  2     Mean       Std        Min        Max        Count     Thruput
 1369.  000403 2  2  xxx1.259 s xxx0.552 s xxx0.329 s xxx6.856 s     8998    xx23.679 tx/s
 1370.  000403 2  2  ------------------------------------------------------------------------
 1371.  000403 2  2            1         2         3         4         5         6
 1372.  000403 2  2  0123456789-123456789-123456789-123456789-123456789-123456789-123456789
 1373.  000403 2  2
 1374.  000403 2  2  RespTime Count  Pct  CumPct
 1375.  000403 2  2  -------- ----- ----- ------
 1376.  000403 2  2  xx23.750 xx345 x12.5 x99.8 |*****************  42 points max
 1377.  000403 2  2
 1378.  000403 2  2  ?ENDIF 15
 1379.  000403 2  2
```

```
1380.    000403 2  2          !------------------!
1381.    000403 2  2          !   Print Header   !
1382.    000403 2  2          !------------------!
1383.    000403 2  2          @p := @line^;
1384.    000406 2  2          blank (line^, 70);
1385.    000415 2  2          line^ ':=' 26*[" "] & "*  ET1 Response Time Distribution  *";     1.00
1386.    000434 2  2          call write^file (out, line, 70,, 0); ! with top of form
1387.    000447 2  2          blank^line; hyphen^line;
1388.    000525 2  2          blank (line^, 70);
1389.    000534 2  2          p[ 3] ':=' "Mean"; p[14] ':=' "Std";   p[25] ':=' "Min";
1390.    000567 2  2          p[36] ':=' "Max" ; p[47] ':=' "Count"; p[57] ':=' "Thruput";
1391.    000622 2  2          write^it;
1392.    000654 2  2          blank (line^, 70);
1393.    000663 2  2          p[ 9] := "s"; p[20] := "s"; p[31] := "s"; p[42] := "s";
1394.    000677 2  2          p[65] ':=' "tx/s";
1395.    000710 2  2          call real^to^ascii (et1.rt^avg, p [ 0]);
1396.    000720 2  2          call real^to^ascii (et1.rt^std, p [11]);
1397.    000731 2  2          call real^to^ascii (et1.rt^min, p [22]);
1398.    000742 2  2          call real^to^ascii (et1.rt^max, p [33]);
1399.    000753 2  2          call dascii      (rt.count, -1, p [51]);
1400.    000763 2  2          call real^to^ascii (et1.thruput,p [56]);
1401.    000774 2  2          write^it;
1402.    001025 2  2          hyphen^line; blank^line;
1403.    001103 2  2          blank (line^, 70);
1404.    001112 2  2          p ':=' "RespTime Count  Pct  CumPct"; write^it;
1405.    001156 2  2          blank (line^, 70);
1406.    001165 2  2          p ':=' "-------- ----- ----- ------"; write^it;
1407.    001227 2  2
1408.    001227 2  2          !------------------!
1409.    001227 2  2          !    Plot Lines    !
1410.    001227 2  2          !------------------!
1411.    001227 2  2          @s := @startup^parameter [param^pcell];
1412.    001232 2  2          pcellsize := if s.setup then $abs (s.value) else 100; ! ms
1413.    001244 2  2          n  := $max (0, (pcellsize / rt.cellsize) - 1);
1414.    001257 2  2
1415.    001257 2  2          stopcount := rt.count - $dbl (rt.freq [rt.numcell]); !all but last cell
1416.    001272 2  2          maxfreq := total := $dbl (rt.freq [0]); ! first cell stays as is.
1417.    001301 2  2          i := 1;   ! aggregate starting with second cell
1418.    001303 2  2          do begin  ! loop thru up to 998 data cells to find MAX freq
1419.    001303 2  3              count := 0;
1420.    001305 2  3              for j := i to i + n do
1421.    001307 2  3                  if j < rt.numcell then
1422.    001314 2  3                  begin
1423.    001314 2  4                      count := count + rt.freq [j];
1424.    001321 2  4                  end;
1425.    001331 2  3              total := total + $dbl (count);
1426.    001337 2  3              if $dbl (count) > maxfreq then maxfreq := $dbl (count);
1427.    001351 2  3              i := i + n + 1;
1428.    001356 2  3          end until total >= stopcount;
1429.    001362 2  2          if i >= rt.numcell then   ! check with last cell count
1430.    001367 2  2              if $dbl (rt.freq [rt.numcell]) > maxfreq then
1431.    001400 2  2                  maxfreq := $dbl (rt.freq [rt.numcell]);
1432.    001407 2  2
1433.    001407 2  2          pct := pct^cum := 0.0e0;
1434.    001413 2  2          total := $dbl (rt.freq [0]); ! first cell stays as is.
1435.    001420 2  2          call plot^line (0e0, $dbl (rt.freq[0]), pct, pct^cum, maxfreq, 0);
1436.    001433 2  2
```

```
 1437.   001433 2  2          i := 1;   ! aggregate starting with second cell
 1438.   001435 2  2          rtime := 0e0;
 1439.   001437 2  2          do begin  ! loop thru up to 998 data cells !
 1440.   001437 2  3              count := 0; pct := 0e0;
 1441.   001443 2  3              for j := i to i + n do
 1442.   001445 2  3                  if j < rt.numcell then
 1443.   001452 2  3                  begin
 1444.   001452 2  4                      rtime := rtime + $dbl (rt.cellsize);
 1445.   001461 2  4                      count := count + rt.freq [j];
 1446.   001466 2  4                      pct   := pct + 1.0e2 *
 1447.   001466 2  4                               ($flt (rt.freq [j]) / $flt (rt.count));
 1448.   001514 2  4                  end else ! keep j constant so that plot^line does not     2.03
 1449.   001515 2  3                           ! mistaken it for the overflow interval          2.03
 1450.   001515 2  3                  begin                                                     2.03
 1451.   001515 2  4                      j := rt.numcell - 1; ! the last valid index !         2.03
 1452.   001521 2  4                      goto rt^out^loop;                                     2.03
 1453.   001522 2  4                  end;                                                      2.03
 1454.   001532 2  3              rt^out^loop:                                                  2.03
 1455.   001532 2  3              pct^cum := pct^cum + pct;
 1456.   001536 2  3              call plot^line ($flt (rtime), $dbl (count), pct, pct^cum,
 1457.   001536 2  3                              maxfreq, j);
 1458.   001552 2  3              total := total + $dbl (count);
 1459.   001560 2  3              i := i + n + 1;
 1460.   001565 2  3          end until total >= stopcount;
 1461.   001571 2  2
 1462.   001571 2  2          if i >= rt.numcell then   ! plot last cell if needed
 1463.   001576 2  2          begin
 1464.   001576 2  3              rtime := rtime + $dbl (rt.cellsize);
 1465.   001605 2  3              count := rt.freq [rt.numcell];                                2.03
 1466.   001612 2  3              pct   := 1.0e2 *                                              2.03
 1467.   001612 2  3                       ($flt (rt.freq [rt.numcell]) / $flt (rt.count));
 1468.   001627 2  3              pct^cum := pct^cum + pct;
 1469.   001636 2  3              call plot^line ($flt (rtime), $dbl (count), pct, pct^cum,
 1470.   001636 2  3                              maxfreq, rt.numcell);
 1471.   001653 2  3          end;
 1472.   001653 2  2
 1473.   001653 2  2      end; !subproc rtplot!

COUNT                          Variable      INT             Direct        S-014
I                              Variable      INT             Direct        S-021
J                              Variable      INT             Direct        S-020
MAXFREQ                        Variable      INT (32)        Direct        S-011
N                              Variable      INT             Direct        S-015
PCELLSIZE                      Variable      INT             Direct        S-016
PCT                            Variable      REAL (32)       Direct        S-003
PCT^CUM                        Variable      REAL (32)       Direct        S-001
RTIME                          Variable      INT (32)        Direct        S-013
RT^OUT^LOOP                    Label                         %001532
S                              Variable,32   STRUCT-I        Indirect      S-017
STOPCOUNT                      Variable      INT (32)        Direct        S-005
TOTAL                          Variable      INT (32)        Direct        S-007

 1474.   002021 1  1
 1475.   002021 1  1      subproc plot^line^a (rtime, count, pct, prob, maxfreq, j);          2.03
 1476.   002021 2  1  !   ------------------------------------------------------------
 1477.   002021 2  1          real    rtime, pct, prob;
 1478.   002021 2  1          int(32) count, maxfreq;
```

```
  1479.    002021 2  1          int     j;
  1480.    002021 2  1      begin
  1481.    002021 2  2          int(32) maxpoints := 42d;
  1482.    002021 2  2          int     num^star;
  1483.    002021 2  2          real    maxpct;                                          2.03
  1483.1   002021 2  2          int     offset;                                          2.03
  1484.    002021 2  2
  1485.    002021 2  2          blank (line^, 70);
  1486.    002034 2  2          call real^to^ascii (rtime / 1e3, p); ! in seconds
  1487.    002046 2  2          call dascii (count, -1, p [13]);
  1488.    002055 2  2          call real^to^ascii (pct    , p [15], 0);
  1489.    002065 2  2          call real^to^ascii (prob   , p [21], 0);
  1490.    002075 2  2          p [27] := "|";
  1491.    002100 2  2
  1492.    002100 2  2          num^star := $intr ( count * maxpoints / maxfreq );
  1493.    002107 2  2          if num^star then
  1494.    002111 2  2              p[28] ':=' "*" & p [28] for num^star - 1;
  1495.    002126 2  2
  1496.    002126 2  2          maxpct := 1e2 * $flt (maxfreq) / $flt (rt.count); !in percent!   2.03
  1497.    002140 2  2          if prob >= maxpct then                                   2.03
  1498.    002144 2  2              offset := $int (maxpoints)                           2.03
  1499.    002144 2  2          else                                                     2.03
  1500.    002150 2  2              offset := $intr ($flt (maxpoints) * (prob / maxpct)); 2.03
  1500.1   002160 2  2          if offset then p [27 + offset] := "T";                   2.03
  1501.    002167 2  2
  1502.    002167 2  2          if j >= at.numcell then  ! overflow interval !           2.03
  1503.    002174 2  2          begin                                                    2.03
  1504.    002174 2  3              line^ [77] '=:' line^ [69] for 70;                   2.03
  1505.    002202 2  3              line^ ':=' "    >   "; ! to indicate overflow cell   2.03
  1506.    002212 2  3              call write^file (out, line, 78);                     2.03
  1507.    002226 2  3              return;                                              2.03
  1508.    002233 2  3          end;                                                     2.03
  1509.    002233 2  2
  1510.    002233 2  2          write^it;
  1511.    002264 2  2      end; !subproc plot^line!

COUNT                          Variable      INT (32)       Direct        S-017
J                              Variable      INT            Direct        S-007
MAXFREQ                        Variable      INT (32)       Direct        S-011
MAXPCT                         Variable      REAL (32)      Direct        S-002
MAXPOINTS                      Variable      INT (32)       Direct        S-005
NUM^STAR                       Variable      INT            Direct        S-003
OFFSET                         Variable      INT            Direct        S-000
PCT                            Variable      REAL (32)      Direct        S-015
PROB                           Variable      REAL (32)      Direct        S-013
RTIME                          Variable      REAL (32)      Direct        S-021

  1512.    002277 1  1
  1513.    002277 1  1
  1514.    002277 1  1      subproc atplot;                                              2.03
  1515.    002277 2  1  !   ---------------                                              2.03
  1516.    002277 2  1      begin
  1517.    002277 2  2          int     i, j,
  1518.    002277 2  2                 .s (startup^param^^),
  1519.    002277 2  2                  pcellsize,
  1520.    002277 2  2                  n,
  1521.    002277 2  2                  count;
```

```
1522.    002277 2  2
1523.    002277 2  2          int(32) atime,                                          2.03
1524.    002277 2  2                  maxfreq,
1525.    002277 2  2                  total,
1526.    002277 2  2                  stopcount;
1527.    002277 2  2
1528.    002277 2  2          real    pct,
1529.    002277 2  2                  pct^cum;
1530.    002277 2  2   ?IF 15
1531.    002277 2  2
1532.    002277 2  2                 * ET1 Inter-Arrival Time Distribution *
1533.    002277 2  2
1534.    002277 2  2   ------------------------------------------------------------------
1535.    002277 2  2      Mean       Std        Min        Max        Count     Thruput
1536.    002277 2  2   xxx1.259 s xxx0.552 s xxx0.329 s xxx6.856 s     8998    xx23.679 tx/s
1537.    002277 2  2   ------------------------------------------------------------------
1538.    002277 2  2             1         2         3         4         5         6
1539.    002277 2  2   0123456789-123456789-123456789-123456789-123456789-123456789-123456789
1540.    002277 2  2
1541.    002277 2  2   ArrvTime Count  Pct  Theory
1542.    002277 2  2   -------- ----- ----- ------
1543.    002277 2  2   xx23.750 xx345 x12.5 x99.8 |*****************  42 points max
1544.    002277 2  2
1545.    002277 2  2   ?ENDIF 15
1546.    002277 2  2
1547.    002277 2  2          @s := @startup^parameter [param^terms];                  2.03
1548.    002303 2  2          terms := if s.setup then ! in case of merged txlog files !  2.03
1549.    002303 2  2                      $flt ($abs (s.value))                       2.03
1550.    002303 2  2                   else           ! this is correct with unmerged txlog!  2.03
1551.    002303 2  2                      $flt (test.terminals);                      2.03
1552.    002317 2  2          tput^per^terminal := et1.thruput / terms;               2.03
1553.    002324 2  2          t2 := 1e2; !=exp(0) when t=0!  !in percent=100%!        2.03
1554.    002327 2  2
1555.    002327 2  2          !-----------------!
1556.    002327 2  2          !   Print Header  !
1557.    002327 2  2          !-----------------!
1558.    002327 2  2          @p := @line^;
1559.    002331 2  2          blank (line^, 70);
1560.    002340 2  2          line^ ':='23*[" "] & "*  ET1 Inter-Arrival Time Distribution  *"; 2.03
1561.    002357 2  2          call write^file (out, line, 70,, 0); ! with top of form
1562.    002372 2  2          blank^line; hyphen^line;
1563.    002450 2  2          blank (line^, 70);
1564.    002457 2  2          p[ 3] ':=' "Mean"; p[14] ':=' "Std";   p[25] ':=' "Min";
1565.    002512 2  2          p[36] ':=' "Max" ; p[47] ':=' "Count"; p[57] ':=' "Thruput";
1566.    002545 2  2          write^it;
1567.    002577 2  2          blank (line^, 70);
1568.    002606 2  2          p[ 9] := "s"; p[20] := "s"; p[31] := "s"; p[42] := "s";
1569.    002622 2  2          p[65] ':=' "tx/s";
1570.    002633 2  2          call real^to^ascii (at^stats.avg, p [ 0]);
1571.    002643 2  2          call real^to^ascii (at^stats.std, p [11]);
1572.    002654 2  2          call real^to^ascii (at^stats.min, p [22]);
1573.    002665 2  2          call real^to^ascii (at^stats.max, p [33]);
1574.    002676 2  2          call dascii        (rt.count, -1, p [51]);
1575.    002706 2  2          call real^to^ascii (et1.thruput,p [56]);
1576.    002717 2  2          write^it;
1577.    002750 2  2          hyphen^line; blank^line;
1578.    003127 2  2          blank (line^, 70);
```

```
1579.  003136 2  2          p ':=' "ArrvTime Count  Pct  Theory"; write^it;                    2.03
1580.  003200 2  2          blank (line^, 70);
1581.  003207 2  2          p ':=' "-------- ----- ----- ------"; write^it;
1582.  003251 2  2
1583.  003251 2  2          !-----------------!
1584.  003251 2  2          !    Plot Lines   !
1585.  003251 2  2          !-----------------!
1586.  003251 2  2          @s := @startup^parameter [param^pcella];                          2.03
1587.  003254 2  2          pcellsize := if s.setup then $abs (s.value) else 1000; !1sec!    2.03
1588.  003266 2  2          n  := $max (0, (pcellsize / at.cellsize) - 1);
1589.  003301 2  2
1590.  003301 2  2          stopcount := rt.count - $dbl (at.freq [at.numcell]); !all but last cell
1591.  003314 2  2          maxfreq := total := $dbl (at.freq [0]); ! first cell stays as is.
1592.  003323 2  2          i := 1;   ! aggregate starting with second cell
1593.  003325 2  2          do begin  ! loop thru up to 998 data cells to find MAX freq
1594.  003325 2  3              count := 0;
1595.  003327 2  3              for j := i to i + n do
1596.  003331 2  3                  if j < at.numcell then
1597.  003336 2  3                  begin
1598.  003336 2  4                      count := count + at.freq [j];
1599.  003343 2  4                  end;
1600.  003353 2  3              total := total + $dbl (count);
1601.  003361 2  3              if $dbl (count) > maxfreq then maxfreq := $dbl (count);
1602.  003373 2  3              i := i + n + 1;
1603.  003400 2  3          end until total >= stopcount;
1604.  003404 2  2          if i >= at.numcell then   ! check with last cell count
1605.  003411 2  2              if $dbl (at.freq [at.numcell]) > maxfreq then
1606.  003422 2  2                  maxfreq := $dbl (at.freq [at.numcell]);
1607.  003431 2  2
1608.  003431 2  2          pct := pct^cum := 0.0e0;
1609.  003435 2  2          total := $dbl (at.freq [0]); ! first cell stays as is.
1610.  003442 2  2          call plot^line (0e0, $dbl (at.freq[0]), pct, pct^cum, maxfreq, 0);
1611.  003455 2  2
1612.  003455 2  2          i := 1;   ! aggregate starting with second cell
1613.  003457 2  2          atime := 0e0;
1614.  003461 2  2          do begin  ! loop thru up to 998 data cells !
1615.  003463 2  3              count := 0; pct := 0e0;
1616.  003467 2  3              for j := i to i + n do
1617.  003471 2  3                  if j < at.numcell then
1618.  003476 2  3                  begin
1619.  003476 2  4                      atime := atime + $dbl (at.cellsize);
1620.  003505 2  4                      count := count + at.freq [j];
1621.  003512 2  4                      pct   := pct + 1.0e2 *
1622.  003512 2  4                               ($flt (at.freq [j]) / $flt (rt.count));
1623.  003540 2  4                  end else ! keep j constant so that plot^line does not    2.03
1624.  003541 2  3                           ! mistaken it for the overflow interval         2.03
1625.  003541 2  3                  begin                                                    2.03
1626.  003541 2  4                      j := at.numcell - 1; ! the last valid index !        2.03
1627.  003545 2  4                      goto at^out^loop;                                    2.03
1628.  003546 2  4                  end;                                                     2.03
1629.  003556 2  3              at^out^loop:                                                 2.03
1630.  003556 2  3              pct^cum := pct^cum + pct;
1631.  003562 2  3      --                                                                   2.03
1632.  003562 2  3      -- To compare 'pct' versus the theoretical value for an exponential  2.03
1633.  003562 2  3      -- distribution, we use the folowing formula:                        2.03
1634.  003562 2  3      --                                                                   2.03
1635.  003562 2  3      --                        -(rate * a)     -(rate * b)                2.03
```

```
  1636.  003562 2  3   --   Pr {a <= x <= b} = e             - e                                    2.03
  1637.  003562 2  3   --                                                                           2.03
  1638.  003562 2  3   -- where rate is the throughput rate and a,b are points in time.            2.03
  1639.  003562 2  3   --                                                                           2.03
  1640.  003562 2  3              t1 := t2;                                                         2.03
  1641.  003564 2  3              t2 := 1e2 * exp^ (- tput^per^terminal * $flt (atime) / 1e3);      2.03
  1642.  003604 2  3              prob := if $abs (t1 - t2) < 1e-6                                  2.03
  1643.  003604 2  3                      then 0e0                                                  2.03
  1644.  003604 2  3                      else (t1 - t2);                                           2.03
  1645.  003627 2  3
  1646.  003627 2  3              call plot^line^a ($flt (atime), $dbl (count), pct, prob,          2.03
  1647.  003627 2  3                               maxfreq, j);
  1648.  003643 2  3              total := total + $dbl (count);
  1649.  003651 2  3              i := i + n + 1;
  1650.  003660 2  3          end until total >= stopcount;
  1651.  003664 2  2
  1652.  003664 2  2          if i >= at.numcell then   ! plot last cell if needed
  1653.  003671 2  2          begin
  1654.  003671 2  3              atime := atime + $dbl (at.cellsize);
  1655.  003700 2  3              count := at.freq [at.numcell];                                    2.03
  1656.  003705 2  3              pct   := 1.0e2 *                                                  2.03
  1657.  003705 2  3                       ($flt (at.freq [at.numcell]) / $flt (rt.count));         2.03
  1658.  003722 2  3              pct^cum := pct^cum + pct;
  1659.  003726 2  3              ! Prob {x >= a} = 1 - Prob {0 <= x <= a}                          2.03
  1660.  003726 2  3              !                   - (rate * a)                                  2.03
  1661.  003726 2  3              !                 = e                                             2.03
  1662.  003726 2  3              prob := 1e2 * exp^ (- tput^per^terminal * $flt (atime) / 1e3);    2.03
  1663.  003746 2  3              prob := if $abs (prob) < 1e-6                                     2.03
  1664.  003746 2  3                      then 0e0                                                  2.03
  1665.  003746 2  3                      else prob;
  1666.  003761 2  3              call plot^line^a ($flt (atime), $dbl (count), pct, prob,          2.03
  1667.  003761 2  3                               maxfreq, at.numcell);
  1668.  003776 2  3          end;
  1669.  003776 2  2
  1670.  003776 2  2      end; !subproc atplot!                                                     2.03

ATIME                          Variable       INT (32)        Direct         S-013
AT^OUT^LOOP                    Label                          %003556
COUNT                          Variable       INT             Direct         S-014
I                              Variable       INT             Direct         S-021
J                              Variable       INT             Direct         S-020
MAXFREQ                        Variable       INT (32)        Direct         S-011
N                              Variable       INT             Direct         S-015
PCELLSIZE                      Variable       INT             Direct         S-016
PCT                            Variable       REAL (32)       Direct         S-003
PCT^CUM                        Variable       REAL (32)       Direct         S-001
S                              Variable,32    STRUCT-I        Indirect       S-017
STOPCOUNT                      Variable       INT (32)        Direct         S-005
TOTAL                          Variable       INT (32)        Direct         S-007

  1671.  004052 1  1
  1672.  004052 1  1
  1673.  004052 1  1
  1674.  004052 1  1      !-----------------!
  1675.  004052 1  1      !      Header     !
  1676.  004052 1  1      !-----------------!
  1677.  004052 1  1      hyphen^line;
```

```
1678.   004146 1  1       line^ ':=' header for 70;
1679.   004155 1  1       write^it;
1680.   004206 1  1       hyphen^line;
1681.   004247 1  1       blank^line;
1682.   004264 1  1
1683.   004264 1  1       !-----------------!
1684.   004264 1  1       !    Parameters   !
1685.   004264 1  1       !-----------------!
1686.   004264 1  1       line^ ':=' "----- PARAMETERS " -> @p;
1687.   004275 1  1       p ':=' "-" & p  for 70 - (@p '-' @line^);
1688.   004314 1  1       write^it;
1689.   004345 1  1       blank^line;
1690.   004364 1  1
1691.   004364 1  1       !--------------------!
1692.   004364 1  1       !  Test Information  !
1693.   004364 1  1       !--------------------!
1694.   004364 1  1       blank (line^, 70); line^ ':=' "Test Title"; line^[26] := "=";
1695.   004406 1  1       if startup^parameter[param^title].setup then
1696.   004412 1  1           line^[30] ':=' startup^parameter[param^title].s^name for 24;
1697.   004421 1  1       write^it;
1698.   004452 1  1
1699.   004452 1  1       blank (line^, 70); line^ ':=' "Test Date"; line^[26] := "=";
1700.   004474 1  1       lct := converttimestamp (test.ts^first^rec);
1701.   004510 1  1       jday := interprettimestamp (lct, date^time);
1702.   004516 1  1       line^[30] ':=' "yy-mm-dd";
1703.   004527 1  1       call numout (line^[30], ($udbl (date^time[0])'\'100), 10, 2); ! year
1704.   004541 1  1       call numout (line^[33], date^time[1], 10, 2); ! month
1705.   004551 1  1       call numout (line^[36], date^time[2], 10, 2); ! day
1706.   004561 1  1       write^it;
1707.   004612 1  1
1708.   004612 1  1       blank (line^, 70); line^ ':=' "Test Length"; line^[26] := "=";
1709.   004634 1  1       line^[30] ':=' "hh:mm:ss - hh:mm:ss  (" -> @p;
1710.   004646 1  1       call numout (line^[30], date^time[3], 10, 2); ! hour
1711.   004655 1  1       call numout (line^[33], date^time[4], 10, 2); ! minute
1712.   004665 1  1       call numout (line^[36], date^time[5], 10, 2); ! second
1713.   004675 1  1       lct := converttimestamp (test.ts^last^rec);
1714.   004711 1  1       jday := interprettimestamp (lct, date^time);
1715.   004717 1  1       call numout (line^[41], date^time[3], 10, 2); ! hour
1716.   004727 1  1       call numout (line^[44], date^time[4], 10, 2); ! minute
1717.   004737 1  1       call numout (line^[47], date^time[5], 10, 2); ! second
1718.   004747 1  1       err := dascii (test.len^test, 10, p);
1719.   004757 1  1       @p := @p '+' err;
1720.   004763 1  1       p ':=' " sec)";
1721.   004773 1  1       write^it;
1722.   005024 1  1
1723.   005024 1  1       if startup^parameter [param^window].setup then
1724.   005030 1  1       begin ! window parameter was passed in startup message
1725.   005030 1  2           blank (line^, 70); line^ ':=' "Window Length"; line^[26] := "=";
1726.   005052 1  2           line^[30] ':=' "hh:mm:ss - hh:mm:ss  (" -> @p;
1727.   005064 1  2           lct := converttimestamp (test.ts^window^begin);
1728.   005100 1  2           jday := interprettimestamp (lct, date^time);
1729.   005106 1  2           call numout (line^[30], date^time[3], 10, 2); ! hour
1730.   005116 1  2           call numout (line^[33], date^time[4], 10, 2); ! minute
1731.   005126 1  2           call numout (line^[36], date^time[5], 10, 2); ! second
1732.   005136 1  2           lct := converttimestamp (test.ts^window^end);
1733.   005152 1  2           jday := interprettimestamp (lct, date^time);
1734.   005160 1  2           call numout (line^[41], date^time[3], 10, 2); ! hour
```

```
1735.  005170 1  2          call numout (line^[44], date^time[4], 10, 2); ! minute
1736.  005200 1  2          call numout (line^[47], date^time[5], 10, 2); ! second
1737.  005210 1  2          err := dascii (test.len^window, 10, p);
1738.  005220 1  2          @p := @p '+' err;
1739.  005224 1  2          p ':=' " sec)";
1740.  005236 1  2          write^it;
1741.  005267 1  2      end else
1742.  005270 1  1      begin                                                          1.00
1743.  005270 1  2          line^[8]':=' "Window Length"; !default to all recs in file 1.00
1744.  005301 1  2          call write^file (out, line, 78);                           1.00
1745.  005314 1  2      end;
1746.  005314 1  1
1747.  005314 1  1      blank (line^, 70); line^ ':=' "Tx Log File"; line^[26] := "=";
1748.  005336 1  1      @p := @netname '<<' 1;
1749.  005341 1  1      p     := "\";
1750.  005343 1  1      p [1] := mysystemnumber;  ! make network form
1751.  005346 1  1      p [2] ':=' in.name^ [1] for 6;
1752.  005355 1  1      netname [4] ':=' in.filename [4] for 8;
1753.  005362 1  1      call fnamecollapse (netname, line^[30]);
1754.  005370 1  1      write^it;
1755.  005421 1  1
1756.  005421 1  1      blank^line;
1757.  005436 1  1
1758.  005436 1  1      !---------------------!
1759.  005436 1  1      !  ARRIVAL Statistics !
1760.  005436 1  1      !---------------------!
1761.  005436 1  1      blank (line^, 70);
1762.  005445 1  1      line^ ':=' "Terminals/X25 Line"; line^[26] := "=";             1.02
1763.  005460 1  1      err := dascii ($dbl(test.terminals), -10, line^[35]);
1764.  005471 1  1      line^[36] ':=' " #";
1765.  005502 1  1      write^it;
1766.  005534 1  1
1767.  005534 1  1      blank (line^, 70); line^ ':=' "Tx Arrival Process"; line^[26] := "=";
1768.  005556 1  1      case test.arrival^type of
1769.  005561 1  1      begin
1770.  005561 1  2          line^[28] ':=' " MAXIMUM";
1771.  005573 1 CE1          line^[28] ':=' "CONSTANT";
1772.  005605 1 CE2          line^[28] ':=' "  RANDOM";
1773.  005617 1 CE3      end;
1774.  005623 1  1      write^it;
1775.  005654 1  1
1776.  005654 1  1      blank (line^, 70);
1777.  005663 1  1      line^ ':=' "Tx Arrival Rate/X25 Line"; line^[26] := "=";       1.02
1778.  005676 1  1      real^temp := $flt (test.arrival^rate) / 10.0e0;
1779.  005704 1  1      @p := @line^[28] '+' real^to^ascii (real^temp, line^[28]);
1780.  005720 1  1      p ':=' " tx/sec";
1781.  005730 1  1      write^it;
1782.  005761 1  1
1783.  005761 1  1      blank (line^, 70); line^ ':=' "Tx Arrival Rate/Terminal";
1784.  006000 1  1      line^[26] := "=";
1785.  006003 1  1      real^temp := ($flt (test.arrival^rate) / 10.0e0) /
1786.  006003 1  1                   $flt (test.terminals);
1787.  006014 1  1      @p := @line^[28] '+' real^to^ascii (real^temp, line^[28]);
1788.  006030 1  1      p ':=' " tx/sec";
1789.  006040 1  1      write^it;
1790.  006071 1  1
1791.  006071 1  1      blank (line^, 70); line^ ':=' "Tx Inter-Arrival Time";
```

```
1792.  006113 1  1      line^[26] := "=";
1793.  006116 1  1      real^temp := if real^temp > 0e0 then 1e0 / real^temp else 0e0;
1794.  006131 1  1      @p := @line^[28] '+' real^to^ascii (real^temp, line^[28]);
1795.  006144 1  1      line^[36] ':=' " sec";
1796.  006155 1  1      write^it;
1797.  006207 1  1
1798.  006207 1  1      blank^line;
1799.  006224 1  1
1800.  006224 1  1      !---------------------!
1801.  006224 1  1      ! Measurement Results !
1802.  006224 1  1      !---------------------!
1803.  006224 1  1      line^ ':=' "----- MEASUREMENT RESULTS  (" & txrep^vs & ") " -> @p;    2.01
1804.  006253 1  1      p ':=' "-" & p  for 70 - (@p '-' @line^);
1805.  006272 1  1      write^it;
1806.  006323 1  1      blank^line;
1807.  006340 1  1
1808.  006340 1  1      !---------------------!
1809.  006340 1  1      !    ET1 Statistics   !
1810.  006340 1  1      !---------------------!
1811.  006340 1  1      blank (line^, 70); line^ ':=' "Tx Throughput"; line^[26] := "=";
1812.  006362 1  1      @p := @line^[28] '+' real^to^ascii (et1.thruput, line^[28]);
1813.  006377 1  1      p ':=' " tx/sec";
1814.  006407 1  1      write^it;
1815.  007030 1  1
1816.  007030 1  1      blank (line^, 70); line^ ':=' "Tx Response Time Average";
1817.  007047 1  1      line^[26] := "=";
1818.  007052 1  1      @p := @line^[28] '+' real^to^ascii (et1.rt^avg, line^[28]);
1819.  007066 1  1      p ':=' " sec    ";
1820.  007076 1  1      write^it;
1821.  007127 1  1
1822.  007127 1  1      blank (line^, 70); line^[17] ':=' "Std Dev"; line^[26] := "=";
1823.  007152 1  1      @p := @line^[28] '+' real^to^ascii (et1.rt^std, line^[28]);
1824.  007167 1  1      p ':=' " sec    ";
1825.  007177 1  1      write^it;
1826.  007230 1  1
1827.  007230 1  1      blank (line^, 70); line^[17] ':=' "Minimum"; line^[26] := "=";
1828.  007253 1  1      @p := @line^[28] '+' real^to^ascii (et1.rt^min, line^[28]);
1829.  007272 1  1      p ':=' " sec    ";
1830.  007302 1  1      write^it;
1831.  007333 1  1
1832.  007333 1  1      blank (line^, 70); line^[17] ':=' "Maximum"; line^[26] := "=";
1833.  007356 1  1      @p := @line^[28] '+' real^to^ascii (et1.rt^max, line^[28]);
1834.  007373 1  1      p ':=' " sec    ";
1835.  007403 1  1      write^it;
1836.  007434 1  1
1837.  007434 1  1      blank (line^, 70); line^[17] ':=' "Count"; line^[26] := "=";
1838.  007457 1  1      err := dascii (rt.count, -10, line^[35]);
1839.  007467 1  1      line^[36] ':=' " #      ";
1840.  007500 1  1      write^it;
1841.  007532 1  1
1842.  007532 1  1      blank^line;
1843.  007547 1  1
1844.  007547 1  1      !---------------------!
1845.  007547 1  1      !  RT Distribution    !
1846.  007547 1  1      !---------------------!
1847.  007547 1  1      blank (line^, 70); line^ ':=' "Tx Response Time 10%";
1848.  007566 1  1      line^[26] := "=";
```

```
1849.    007571 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [1], line^[28]);
1850.    007606 1  1       p ':=' " sec  (" -> @p;
1851.    007617 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [1], p, err);
1852.    007632 1  1       p ':=' "%)";
1853.    007642 1  1       write^it;
1854.    007673 1  1
1855.    007673 1  1       blank (line^, 70); line^[17] ':=' "20%"; line^[26] := "=";
1856.    007716 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [2], line^[28]);
1857.    007733 1  1       p ':=' " sec  (" -> @p;
1858.    007744 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [2], p, err);
1859.    007757 1  1       p ':=' "%)";
1860.    007767 1  1       write^it;
1861.    010022 1  1
1862.    010022 1  1       blank (line^, 70); line^[17] ':=' "30%"; line^[26] := "=";
1863.    010045 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [3], line^[28]);
1864.    010062 1  1       p ':=' " sec  (" -> @p;
1865.    010073 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [3], p, err);
1866.    010106 1  1       p ':=' "%)";
1867.    010116 1  1       write^it;
1868.    010147 1  1
1869.    010147 1  1       blank (line^, 70); line^[17] ':=' "40%"; line^[26] := "=";
1870.    010172 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [4], line^[28]);
1871.    010207 1  1       p ':=' " sec  (" -> @p;
1872.    010220 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [4], p, err);
1873.    010233 1  1       p ':=' "%)";
1874.    010243 1  1       write^it;
1875.    010274 1  1
1876.    010274 1  1       blank (line^, 70); line^[17] ':=' "50%"; line^[26] := "=";
1877.    010317 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [5], line^[28]);
1878.    010334 1  1       p ':=' " sec  (" -> @p;
1879.    010345 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [5], p, err);
1880.    010360 1  1       p ':=' "%)";
1881.    010370 1  1       write^it;
1882.    010423 1  1
1883.    010423 1  1       blank (line^, 70); line^[17] ':=' "60%"; line^[26] := "=";
1884.    010446 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [6], line^[28]);
1885.    010463 1  1       p ':=' " sec  (" -> @p;
1886.    010474 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [6], p, err);
1887.    010507 1  1       p ':=' "%)";
1888.    010517 1  1       write^it;
1889.    010550 1  1
1890.    010550 1  1       blank (line^, 70); line^[17] ':=' "70%"; line^[26] := "=";
1891.    010573 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [7], line^[28]);
1892.    010610 1  1       p ':=' " sec  (" -> @p;
1893.    010621 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [7], p, err);
1894.    010634 1  1       p ':=' "%)";
1895.    010644 1  1       write^it;
1896.    010675 1  1
1897.    010675 1  1       blank (line^, 70); line^[17] ':=' "80%"; line^[26] := "=";
1898.    010720 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [8], line^[28]);
1899.    010735 1  1       p ':=' " sec  (" -> @p;
1900.    010746 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [8], p, err);
1901.    010761 1  1       p ':=' "%)";
1902.    010771 1  1       write^it;
1903.    011024 1  1
1904.    011024 1  1       blank (line^, 70); line^[17] ':=' "90%"; line^[26] := "=";
1905.    011047 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [9], line^[28]);
```

```
1906.    011064 1  1       p ':=' " sec  (" -> @p;
1907.    011075 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [9], p, err);
1908.    011110 1  1       p ':=' "%)";
1909.    011120 1  1       write^it;
1910.    011431 1  1
1911.    011431 1  1       blank (line^, 70); line^[17] ':=' "95%"; line^[26] := "=";
1912.    011454 1  1       @p := @line^[28] '+' real^to^ascii (et1.rt^dist [0], line^[28]);
1913.    011471 1  1       p ':=' " sec  (" -> @p;
1914.    011502 1  1       @p := @p '+' real^to^ascii (et1.rt^pct [0], p, err);
1915.    011515 1  1       p ':=' "%)";
1916.    011525 1  1       write^it;
1917.    011556 1  1
1918.    011556 1  1       if test.arrival^type <> maximum then
1919.    011562 1  1       begin
1920.    011562 1  2       blank^line;
1921.    011577 1  2
1922.    011577 1  2       !--------------------!
1923.    011577 1  2       !  THINK Statistics  !
1924.    011577 1  2       !--------------------!
1925.    011577 1  2       blank (line^, 70); line^ ':=' "        Tx Think Average";
1926.    011616 1  2       line^[26] := "=";
1927.    011621 1  2       @p := @line^[28] '+' real^to^ascii (think.avg, line^[28]);
1928.    011635 1  2       p ':=' " sec   ";
1929.    011645 1  2       write^it;
1930.    011700 1  2
1931.    011700 1  2       blank (line^, 70); line^[17] ':=' "Std Dev"; line^[26] := "=";
1932.    011723 1  2       @p := @line^[28] '+' real^to^ascii (think.std, line^[28]);
1933.    011740 1  2       p ':=' " sec   ";
1934.    011750 1  2       write^it;
1935.    012003 1  2
1936.    012003 1  2       blank (line^, 70); line^[17] ':=' "Minimum"; line^[26] := "=";
1937.    012026 1  2       real^temp := if think.count > 0d then
1938.    012026 1  2                    $flt (think.minimum) / 1e3 else 0e0;
1939.    012044 1  2       @p := @line^[28] '+' real^to^ascii (real^temp, line^[28]);
1940.    012057 1  2       p ':=' " sec   ";
1941.    012067 1  2       write^it;
1942.    012120 1  2
1943.    012120 1  2       blank (line^, 70); line^[17] ':=' "Maximum"; line^[26] := "=";
1944.    012143 1  2       real^temp := if think.count > 0d then
1945.    012143 1  2                    $flt (think.maximum) / 1e3 else 0e0;
1946.    012162 1  2       @p := @line^[28] '+' real^to^ascii (real^temp, line^[28]);
1947.    012175 1  2       p ':=' " sec   ";
1948.    012205 1  2       write^it;
1949.    012236 1  2
1950.    012236 1  2       blank (line^, 70); line^[17] ':=' "Count"; line^[26] := "=";
1951.    012261 1  2       err := dascii (think.count, -10, line^[35]);
1952.    012271 1  2       line^[36] ':=' " #    (" -> @p;
1953.    012303 1  2       real^temp := 0.0e0;
1954.    012305 1  2       if rt.count > 1d then
1955.    012312 1  2           real^temp := ($flt (think.count) / $flt (rt.count)) * 1.0e2;
1956.    012324 1  2       @p := @p '+' real^to^ascii (real^temp, p, err);
1957.    012336 1  2       p ':=' "%)";
1958.    012346 1  2       write^it;
1959.    012402 1  2
1960.    012402 1  2       blank^line;
1961.    012417 1  2
1962.    012417 1  2       !--------------------!
```

```
1963.   012417 1  2       !  DELAY Statistics   !
1964.   012417 1  2       !--------------------!
1965.   012417 1  2       blank (line^, 70); line^ ':=' "        Tx Delay Average";
1966.   012436 1  2       line^[26] := "=";
1967.   012441 1  2       @p := @line^[28] '+' real^to^ascii (dilay.avg, line^[28]);
1968.   012455 1  2       p ':=' " sec   ";
1969.   012465 1  2       write^it;
1970.   012516 1  2
1971.   012516 1  2       blank (line^, 70); line^[17] ':=' "Std Dev"; line^[26] := "=";
1972.   012541 1  2       @p := @line^[28] '+' real^to^ascii (dilay.std, line^[28]);
1973.   012556 1  2       p ':=' " sec   ";
1974.   012566 1  2       write^it;
1975.   012617 1  2
1976.   012617 1  2       blank (line^, 70); line^[17] ':=' "Minimum"; line^[26] := "=";
1977.   012642 1  2       real^temp := if dilay.count > 0d then
1978.   012642 1  2                       $flt (dilay.minimum) / 1e3 else 0e0;
1979.   012660 1  2       @p := @line^[28] '+' real^to^ascii (real^temp, line^[28]);
1980.   012673 1  2       p ':=' " sec   ";
1981.   012703 1  2       write^it;
1982.   012734 1  2
1983.   012734 1  2       blank (line^, 70); line^[17] ':=' "Maximum"; line^[26] := "=";
1984.   012757 1  2       real^temp := if dilay.count > 0d then
1985.   012757 1  2                       $flt (dilay.maximum) / 1e3 else 0e0;
1986.   012776 1  2       @p := @line^[28] '+' real^to^ascii (real^temp, line^[28]);
1987.   013011 1  2       p ':=' " sec   ";
1988.   013021 1  2       write^it;
1989.   013052 1  2
1990.   013052 1  2       blank (line^, 70); line^[17] ':=' "Count"; line^[26] := "=";
1991.   013077 1  2       err := dascii (dilay.count, -10, line^[35]);
1992.   013107 1  2       line^[36] ':=' " #    (" -> @p;
1993.   013121 1  2       real^temp := 0.0e0;
1994.   013123 1  2       if rt.count > 1d then
1995.   013130 1  2           real^temp := ($flt (dilay.count) / $flt (rt.count)) * 1.0e2;
1996.   013142 1  2       @p := @p '+' real^to^ascii (real^temp, p, err);
1997.   013154 1  2       p ':=' "%)";
1998.   013164 1  2       write^it;
1999.   013215 1  2
2000.   013215 1  2       end; !arrival^type <> maximum!
2001.   013215 1  1
2002.   013215 1  1       blank^line; hyphen^line;
2003.   013273 1  1
2004.   013273 1  1       !--------------------!
2005.   013273 1  1       !       RT Plot      !
2006.   013273 1  1       !--------------------!
2007.   013273 1  1       if startup^parameter [param^rtplot].setup then
2008.   013277 1  1           call rtplot;
2009.   013300 1  1
2010.   013300 1  1       !--------------------!                                    2.03
2011.   013300 1  1       !       AT Plot      !                                    2.03
2012.   013300 1  1       !--------------------!                                    2.03
2013.   013300 1  1       if startup^parameter [param^atplot].setup then            2.03
2014.   013304 1  1           call atplot;                                         2.03
2015.   013305 1  1
2016.   013305 1  1       !--------------------!
2017.   013305 1  1       !       RT Dump      !
2018.   013305 1  1       !--------------------!
2019.   013305 1  1   !   if startup^parameter [param^rtdump].setup then
```

```
  2020.    013305 1  1   !       call rtdump;
  2021.    013305 1  1
  2022.    013305 1  1   end; !proc REPORT!
```

ATPLOT	Subproc		%002277	
BLANK	Define		SMEAR ($1, " ", $2)	
BLANK^LINE	Define		BEGIN LINE^ := " "; CALL WRITE^FILE (OUT, LINE, 1); END	
DATE^TIME	Variable	INT	Indirect	L+013
EFORMAT	Variable	STRING	Indirect	L+015
ERR	Variable	INT	Direct	L+014
HEADER	Variable	STRING	Direct	P+000
HYPHEN^LINE	Define		BEGIN SMEAR (LINE^,"-",70); WRITE^IT; END	
IFORMAT	Variable	STRING	Indirect	L+016
JDAY	Variable	INT (32)	Direct	L+011
LCT	Variable	FIXED (0)	Direct	L+005
LINE	Variable	INT	Indirect	L+003
LINELEN	Variable	INT	Direct	L+002
LINE^	Variable	STRING	Indirect	L+004
NETNAME	Variable	INT	Indirect	L+023
NUM^STR	Variable	STRING	Indirect	L+017
NUM^WRD	Variable	INT	Indirect	L+020
P	Variable	STRING	Indirect	L+001
PLOT^LINE	Subproc		%000170	
PLOT^LINE^A	Subproc		%002021	
PROB	Variable	REAL (32)	Direct	L+034
REAL^TEMP	Variable	REAL (32)	Direct	L+021
REAL^TO^ASCII	Subproc	INT	%000043	
RTPLOT	Subproc		%000403	
SMEAR	Define		BEGIN $1 := $2; $1[1] ':=' $1 FOR ($3-1); END	
T1	Variable	REAL (32)	Direct	L+026
T2	Variable	REAL (32)	Direct	L+030
TERMS	Variable	REAL (32)	Direct	L+032
TPUT^PER^TERMINAL	Variable	REAL (32)	Direct	L+024
WRITE^IT	Define		BEGIN LINE^ [77] ':=' LINE^ [69] FOR 70; LINE^ ':=' "	

```
2024.  000000 0  0  ?NOWARN
2025.  000000 0  0    proc print^help;
2026.  000000 1  0  ! ----------------
2027.  000000 1  0    begin
2028.  000000 1  1        string  .banner [0:59];                                    2.01
2029.  000000 1  1        string   h1 = 'P' := [60*["\"],0,%377],                    2.01
2030.  000037 1  1                 h2 = 'P' := [" ",0,%377],
2031.  000041 1  1                 h3 = 'P' := ["  ",
2032.  000042 1  1  "TXREP /out <report>/ ",0,%377],
2033.  000056 1  1                 h4 = 'P' := ["      ",
2034.  000061 1  1  "  [TXLOG  <filename>,]         -- Capture file created by TXGEN",
2035.  000121 1  1  0,%377],
2036.  000122 1  1                 h5 = 'P' := ["      ",
2037.  000125 1  1  "  [TITLE  <24-char-string>,]   -- Report title",
2038.  000154 1  1  0,%377],
2039.  000155 1  1                 h6 = 'P' := ["      ",
2040.  000160 1  1  "  [WINDOW  h1:m1:s1/h2:m2:s2,] -- Delimits <txlog> for analysis",
2041.  000220 1  1  0,%377],
2042.  000221 1  1                 h7 = 'P' := ["      ",
2043.  000224 1  1  "  [RTPLOT,]                    -- Generate plot of response times",
2044.  000265 1  1  0,%377],
2045.  000266 1  1                 h8 = 'P' := ["      ",
2046.  000271 1  1  "  [ATPLOT]                     -- Generate plot of inter-arrival times", 2.03
2047.  000334 1  1  0,%377];                                                         2.03
2048.  000335 1  1  !                h9 = 'P' := ["      ",
2049.  000335 1  1  ! "  [RTDUMP ]                    -- Generate table of response times",
2050.  000335 1  1  ! 0,%377];
2051.  000335 1  1
2052.  000335 1  1        banner ':=' "TXREP - ET1 Performance Report (" &             2.01
2053.  000341 1  1                    txrep^vs & ") - PAM";                          2.01
2054.  000367 1  1        call fprint (hometerm.filenum, h1, @banner);               2.01
2055.  000413 1  1        call fprint (hometerm.filenum, h2);
2056.  000435 1  1        call fprint (hometerm.filenum, h3);
2057.  000457 1  1        call fprint (hometerm.filenum, h4);
2058.  000501 1  1        call fprint (hometerm.filenum, h5);
2059.  000523 1  1        call fprint (hometerm.filenum, h6);
2060.  000545 1  1        call fprint (hometerm.filenum, h7);
2061.  000567 1  1        call fprint (hometerm.filenum, h8);
2062.  000621 1  1  !     call fprint (hometerm.filenum, h9);
2063.  000621 1  1        call fprint (hometerm.filenum, h2);
2064.  000643 1  1
2065.  000643 1  1        call stop;
2066.  000650 1  1    end; !proc print^help!
```

BANNER	Variable	STRING	Indirect	L+001
H1	Variable	STRING	Direct	P+000
H2	Variable	STRING	Direct	P+000
H3	Variable	STRING	Direct	P+000
H4	Variable	STRING	Direct	P+000
H5	Variable	STRING	Direct	P+000
H6	Variable	STRING	Direct	P+000
H7	Variable	STRING	Direct	P+000
H8	Variable	STRING	Direct	P+000

```
  2068.    000000 0  0    proc print^times (ts^start, ts^stop);
  2069.    000000 1  0  ! -----------------------------------
  2070.    000000 1  0       fixed ts^start, ts^stop;
  2071.    000000 1  0    begin
  2072.    000000 1  1       string  msg1 = 'P' :=
  2073.    000000 1  1       ["  Start Time: ##:##:##    "," Stop Time: ##:##:##    ", 0, %377];
  2074.    000033 1  1       string  msg2 = 'P' :=
  2075.    000033 1  1       ["Elapsed Time: ##:##:##.###","   Cpu Time: ##:##:##.###", 0, %377];
  2076.    000066 1  1
  2077.    000066 1  1       fixed   elapsed^time, lct;
  2078.    000066 1  1       int(32) jd;
  2079.    000066 1  1       int     h1, m1, s1, ms1;
  2080.    000066 1  1       int    .t [0:7];
  2081.    000066 1  1
  2082.    000066 1  1       lct := converttimestamp (ts^start);
  2083.    000105 1  1       jd  := interprettimestamp (lct, t);
  2084.    000113 1  1       h1  := t [3]; m1 := t [4]; s1 := t [5];
  2085.    000124 1  1       lct := converttimestamp (ts^stop );
  2086.    000137 1  1       jd  := interprettimestamp (lct, t);
  2087.    000145 1  1       call fprint (hometerm.filenum, msg1, h1, m1, s1, t[3], t[4], t[5]);
  2088.    000176 1  1
  2089.    000176 1  1       elapsed^time := ts^stop - ts^start;
  2090.    000205 1  1       call convertprocesstime (elapsed^time, t, t[1], t[2], t[3], t[4]);
  2091.    000226 1  1       h1  := t [0]; m1 := t [1]; s1 := t [2]; ms1 := t [3];
  2092.    000241 1  1       call convertprocesstime (myprocesstime, t, t[1], t[2], t[3], t[4]);
  2093.    000263 1  1       call fprint (hometerm.filenum,msg2,h1,m1,s1,ms1,t,t[1],t[2],t[3]);
  2094.    000316 1  1
  2095.    000316 1  1    end; !proc print^times!
```

ELAPSED^TIME	Variable	FIXED (0)	Direct	L+001
H1	Variable	INT	Direct	L+013
JD	Variable	INT (32)	Direct	L+011
LCT	Variable	FIXED (0)	Direct	L+005
M1	Variable	INT	Direct	L+014
MS1	Variable	INT	Direct	L+016
MSG1	Variable	STRING	Direct	P+000
MSG2	Variable	STRING	Direct	P+000
S1	Variable	INT	Direct	L+015
T	Variable	INT	Indirect	L+017
TS^START	Variable	FIXED (0)	Direct	L-012
TS^STOP	Variable	FIXED (0)	Direct	L-006

```
  2096.    000000 0  0  ?WARN
```

```
 2098.    000000 0  0   proc TXREP main;
 2099.    000000 1  0   !---------------
 2100.    000000 1  0   begin
 2101.    000000 1  1       fixed ts^start;
 2102.    000000 1  1
 2103.    000000 1  1       ts^start := juliantimestamp;
 2104.    000011 1  1       call txrep^init;
 2105.    000012 1  1       call scan^file;
 2106.    000013 1  1       call et1^stats;
 2107.    000014 1  1       call report;
 2108.    000015 1  1       call close^file (out);
 2109.    000023 1  1       call print^times (ts^start, juliantimestamp);
 2110.    000036 1  1   end;
```

```
TS^START                    Variable      FIXED (0)      Direct         L+001
```

ABEND	Proc		External	
AT	Variable,4012	STRUCT	Indirect	#GLOBAL+006
AT^STATS	Variable,20	STRUCT	Indirect	#GLOBAL+007
1 AVG	0,4	REAL (32)	Direct	
1 STD	4,4	REAL (32)	Direct	
1 MIN	10,4	REAL (32)	Direct	
1 MAX	14,4	REAL (32)	Direct	
AWAITIO^DONE	Literal	INT	%000005	
CALC^WINDOW^TS	Proc		%000067	
CELLCOUNT	Literal	INT	%001750	
CFCB	Variable	INT	Indirect	#GLOBAL+000
CLOSE	Proc		External	
CLOSE^FILE	Proc	INT	External	
COMPUTETIMESTAMP	Proc	FIXED (0)	External	
CONSTANT	Literal	INT	%000001	
CONST^	Literal	INT	%000001	
CONTROL^ATTEMPT	Literal	INT	%000004	
CONVERTPROCESSTIME	Proc		External	
CONVERTTIMESTAMP	Proc	FIXED (0)	External	
DASCII	Proc	INT	%000006	
DEBUG	Proc		External	
DILAY	Variable,34	STRUCT	Indirect	#GLOBAL+011
EH	Define		IF <> THEN CALL DEBUG	
ET1	Variable,144	STRUCT	Indirect	#GLOBAL+012
1 THRUPUT	0,4	REAL (32)	Direct	
1 RT^AVG	4,4	REAL (32)	Direct	
1 RT^STD	10,4	REAL (32)	Direct	
1 RT^MIN	14,4	REAL (32)	Direct	
1 RT^MAX	20,4	REAL (32)	Direct	
1 RT^DIST[0:9]	24,4	REAL (32)	Direct	
1 RT^PCT[0:9]	74,4	REAL (32)	Direct	
ET1^STATS	Proc		%000000	
EXP^	Proc	REAL (32)	%000000	
FALSE	Literal	INT	%000000	
FILEINFO	Proc		External	
FIRST^BLOCK^READ	Variable	INT	Direct	#GLOBAL+024
FNAMECOLLAPSE	Proc	INT	External	
FNAMEEXPAND	Proc	INT	External	
FORMATCONVERT	Proc	INT	External	
FORMATDATA	Proc	INT	External	
FPRINT	Proc		%000000	
GUARDIAN^DEVSUBTYPE	Define		<10:15>	
GUARDIAN^DEVTYPE	Define		<4:9>	
GUARDIAN^TYPECARD	Literal	INT	%000010	
GUARDIAN^TYPEDISC	Literal	INT	%000003	
GUARDIAN^TYPEPRINTER	Literal	INT	%000005	
GUARDIAN^TYPEPROCESS	Literal	INT	%000000	
GUARDIAN^TYPERECEIVE	Literal	INT	%000002	
GUARDIAN^TYPETAPE	Literal	INT	%000004	
GUARDIAN^TYPETERM	Literal	INT	%000006	
GUARDIAN^TYPETTMP	Literal	INT	%000025	
GUARDIAN^TYPEX25PTP	Literal	INT	%000011	
HOMETERM	Variable,40	STRUCT	Indirect	#GLOBAL+003
IN	Variable,40	STRUCT	Indirect	#GLOBAL+004
INTERPRETTIMESTAMP	Proc	INT (32)	External	
IODEV^FAILURE	Proc		%000061	

```
IODEV^INIT                      Proc                            %000051
IODEV^^                         Variable        TEMPLATE ,40
  1 NAME^[0:23]                       0,1       STRING          Direct
  1 FILENAME                          0,2       INT             Direct
  1 FILENUM                          30,2       INT             Direct
  1 TYPE                             32,2       INT             Direct
  1 ERROR                            34,2       INT             Direct
  1 BUFADDR                          36,2       INT             Direct
JULIANTIMESTAMP                 Proc            FIXED (0)       External
MAXIMUM                         Literal         INT             %000000
MAX^KEYWORD                     Literal         INT             %000013
MYPROCESSTIME                   Proc            FIXED (0)       External
MYSYSTEMNUMBER                  Proc            INT             External
MYTERM                          Proc                            External
NAME^                           Literal         INT             %000000
NIL                             Literal         INT             %000000
NONE^                           Literal         INT             %000002
NUMIN                           Proc            INT             External
NUMOUT                          Proc                            External
OPEN                            Proc                            External
OPEN^ATTEMPT                    Literal         INT             %000000
OPEN^EXCLUSIVE                  Literal         INT             %000020
OPEN^FILE                       Proc            INT             External
OPEN^GETMSG                     Literal         INT             %040000
OPEN^NOWAIT                     Literal         INT             %000001
OPEN^PROTECTED                  Literal         INT             %000060
OPEN^READONLY                   Literal         INT             %002000
OPEN^READWRITE                  Literal         INT             %000000
OPEN^SHARED                     Literal         INT             %000000
OPEN^WRITEONLY                  Literal         INT             %004000
OUT                             Variable        INT             Indirect      #GLOBAL+001
OUT^BUFFER                      Variable        INT             Indirect      #GLOBAL+002
P                               Variable        STRING          Indirect      #GLOBAL+015
PARAM^ATPLOT                    Literal         INT             %000010
PARAM^CCELL                     Literal         INT             %000007
PARAM^CCELLA                    Literal         INT             %000012
PARAM^DEFAULT                   Literal         INT             %000000
PARAM^KEYWORDS                  Variable        STRING          Direct         P+000
PARAM^PCELL                     Literal         INT             %000006
PARAM^PCELLA                    Literal         INT             %000011
PARAM^RTDUMP                    Literal         INT             %000004
PARAM^RTPLOT                    Literal         INT             %000003
PARAM^TERMS                     Literal         INT             %000013
PARAM^TITLE                     Literal         INT             %000002
PARAM^TXLOG                     Literal         INT             %000005
PARAM^WINDOW                    Literal         INT             %000001
PRINT^HELP                      Proc                            %000335
PRINT^TIMES                     Proc                            %000066
RANDOM                          Literal         INT             %000002
READ                            Proc                            External
READ^ATTEMPT                    Literal         INT             %000001
READ^REC                        Proc                            %000065
REPORT                          Proc                            %004052
RT                              Variable,4012   STRUCT          Indirect      #GLOBAL+005
RT^TABLE^^                      Variable        TEMPLATE ,4012
  1 NAME^[0:23]                       0,1       STRING          Direct
  1 NAME                              0,2       INT             Direct
```

4

```
  1 COUNT                          30,4     INT (32)        Direct
  1 MINIMUM                        34,4     INT (32)        Direct
  1 MAXIMUM                        40,4     INT (32)        Direct
  1 SUM                            44,4     REAL (32)       Direct
  1 SUM^2                          50,4     REAL (32)       Direct
  1 LOBOUND                        54,4     INT (32)        Direct
  1 UPBOUND                        60,4     INT (32)        Direct
  1 CELLSIZE                       64,2     INT             Direct
  1 NUMCELL                        66,2     INT             Direct
  1 FREQ[0:1000]                   70,2     INT             Direct
SCAN^FILE                  Proc                             %000000
SETUP                      Define                           FLAGS.<0>
SET^FILE                   Proc             INT             External
SQRT                       Proc             REAL (32)       %000000
STARTUP^PARAMETER          Variable,32      STRUCT          Indirect      #GLOBAL+014
STARTUP^PARAM^^            Variable         TEMPLATE ,32
  1 NAME[0:11]                      0,2     INT             Direct
  1 VALUE                           0,2     INT             Direct
  1 FLAGS                          30,2     INT             Direct
  1 S^NAME                          0,1     STRING          Direct
STOP                       Proc                             External
STR^                       Literal          INT             %000003
TEST                       Variable,56      STRUCT          Indirect      #GLOBAL+013
  1 TERMINALS                       0,2     INT             Direct
  1 ARRIVAL^RATE                    2,2     INT             Direct
  1 ARRIVAL^TYPE                    4,2     INT             Direct
  1 TS^WINDOW^BEGIN                 6,10    FIXED (0)       Direct
  1 TS^WINDOW^END                  16,10    FIXED (0)       Direct
  1 LEN^WINDOW                     26,4     INT (32)        Direct
  1 TS^FIRST^REC                   32,10    FIXED (0)       Direct
  1 TS^LAST^REC                    42,10    FIXED (0)       Direct
  1 LEN^TEST                       52,4     INT (32)        Direct
THINK                      Variable,34      STRUCT          Indirect      #GLOBAL+010
THROW^COUNT                Variable         INT (32)        Direct        #GLOBAL+016
TIME^STATS^^               Variable         TEMPLATE ,34
  1 COUNT                           0,4     INT (32)        Direct
  1 AVG                             4,4     REAL (32)       Direct
  1 STD                            10,4     REAL (32)       Direct
  1 MINIMUM                        14,4     INT (32)        Direct
  1 MAXIMUM                        20,4     INT (32)        Direct
  1 SUM                            24,4     REAL (32)       Direct
  1 SUM^2                          30,4     REAL (32)       Direct
TRUE                       Literal          INT             %177777
TXLOG^BUF                  Variable         INT             Indirect      #GLOBAL+020
TXLOG^BUFPTR               Variable         INT             Indirect      #GLOBAL+021
TXLOG^BUFSIZE              Literal          INT             %010000
TXLOG^MAXREC               Variable         INT             Direct        #GLOBAL+022
TXLOG^REC                  Variable         INT             Direct        #GLOBAL+023
TXREP                      Proc                             %000000
TXREP^INIT                 Proc                             %000556
TXREP^VS                   Define                           ["vs 2.03"]
TX^DEF                     Variable         TEMPLATE ,40
  1 TERM^COUNT                      0,2     INT             Direct
  1 TERM^INDEX                      2,2     INT             Direct
  1 ARRIVAL^RATE                    4,2     INT             Direct
  1 SEND^TIMESTAMP                  6,10    FIXED (0)       Direct
  1 NEXT^ARRIVAL^TIME              16,4     INT (32)        Direct
```

```
  1 RESPONSE^TIME                 22,4     INT (32)        Direct
  1 FILLER                        26,1
WRITE                         Proc                        External
WRITE^ATTEMPT                 Literal      INT            %000002
WRITE^FILE                    Proc         INT            External
WR^ATTEMPT                    Literal      INT            %000003
```

ENTRY POINT MAP BY NAME

SP	PEP	BASE	LIMIT	ENTRY	ATTRS	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	011	002761	003714	003050		CALC^WINDOW^TS	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	003	000400	000533	000406		DASCII	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	014	011066	012076	011066		ET1^STATS	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	005	000620	000751	000620		EXP^	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	002	000116	000377	000116	V	FPRINT	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	006	000752	001333	001033	V	IODEV^FAILURE	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	007	001334	001556	001405	V	IODEV^INIT	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	016	025644	026546	026201		PRINT^HELP	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	017	026547	027066	026635		PRINT^TIMES	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	012	003715	004026	004002		READ^REC	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	015	012077	025643	016151		REPORT	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	013	004027	011065	004027		SCAN^FILE	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	004	000534	000617	000534		SQRT	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	020	027067	027127	027067	M	TXREP	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
00	010	001557	002760	002335		TXREP^INIT	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S

READ-ONLY DATA BLOCK MAP BY NAME

SP	BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	C000021	000115	COMMON	STRING	PARAM^KEYWORDS	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S

DATA BLOCK MAP BY NAME

BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
000000	000024	COMMON	WORD	#GLOBAL	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S
000025	014677	COMMON	WORD	.#GLOBAL	17FEB87	22:05	TAL	$TURF.NVCTXGEN.TXREP23S

```
BINDER - OBJECT FILE BINDER - T9621C00 - (15JUL87)   SYSTEM \FOXII
Object file name is $TURF.NVCTXGEN.txrep23
Object file timestamp is 17FEB87 22:05:48
Number of Binder errors = 0
Number of Binder warnings = 0
Primary data = 21 words
Secondary data = 6571 words
Code area size = 12 pages
Resident code size = 0 pages
Data area size = 8 pages
Extended data area size = 0 pages
Top of stack = 6592 words
Number of code segments  = 1 segment


The object file will run on a TNS/II, but may not run on a TNS
Number of compiler errors =  0
Number of compiler warnings =  25
Maximum symbol table space used was = 40938 bytes
Number of source lines = 4467
Compile cpu time = 00:00:32
Total Elapsed time = 00:01:16
```

```
1010 rem   ================================================================
1020 rem   =                                                              =
1030 rem   =  ET1 Script Generator                                        =
1040 rem   =                                                              =
1050 rem   ================================================================

1060       on error goto 2540

1070 rem   ================================================================
1080 rem   =  Print Blurb                                                 =
1090 rem   ================================================================

1100       print "This program generates ET1 according to the standard ";    &
                 "definition."
1110       print "    "
1120       print "A random teller, within the scripts teller range is picked."
1130       print "The branch the teller is connected to is determined, this is"
1140       print "done by dividing by 10. "
1150       print "    "
1160       print "In 85% of the cases, a random account is picked within ";  &
                 "the branches"
1170       print "account range of 10,000. In the other 15% of the cases ";  &
                 "a random"
1180       print "account is picked allowing any branch but the 'home' branch."
1190       print "    "
1200       print "The statistics at the end show the truly generated ";      &
                 "distribution."
1210       print "    "

1220 rem   tgal command &
```

```
1230 rem   ============================================================
1240 rem   =  Get Information                                          =
1250 rem   ============================================================

1260       input "How many tellers in total set?                ", fullsize
1270       input "How many transactions for this node?          ", txnums
1280       lowtell = 0
1290       hightell = 0
1300       input "Lowest valid teller number for this node?     ", lowtell
1310       input "Highest valid teller number for this node?    ", hightell
1320       input "Number of partitions on this node             ", parts
1330       input "Number of script files for this node          ", scripts

1340       print "Ramping information:"
1350       input "Projected TPS for this system:                ", tps
1360       input "Desired rampup time (minutes):                ", ramp
1370       rampkt = int(ramp * 60 *tps)
1380       input "Ramp identifier:                              ", rampid
1390       input "Local identifier:                             ", locid
1400       input "Remote identifier:                            ", remid

1410 rem   tgal command &
```

```
1420 rem   ================================================================
1430 rem   =  Open Script Files                                            =
1440 rem   ================================================================

1450       sc$ = "SCRIPT"
1460       for x = 1 to scripts
1470       name$ = sc$+string$(1,(x+48))
1480       print "opening",name$

1490 rem   open           name$ for output as #x,       &
           organization   edit,                         &
           access         append,                       &
           primextent     (txnums/48/16/scripts),       &
           secextent      (txnums/48/16/scripts)

1500       open           name$ for output as #x,       &
           organization   edit,                         &
           access         append,                       &
           primextent     (txnums/48/16/scripts),       &
           secextent      (50)

1510       next x

1520 rem   tgal command &
```

```
1530 rem   ================================================================
1540 rem   =  Perform Initial Calculations                                 =
1550 rem   ================================================================

1560       tnums         = hightell-lowtell+1
1570       local         = 0
1580       remote        = 0
1590       homebranch    = 0
1600       samedisc      = 0
1610       psize         = tnums / parts
1620       print "tellers in script:",tnums,"partition size:",psize

1630 rem   tgal command &
```

```
1640 rem   ============================================================
1650 rem   =  Generate Scripts                                         =
1660 rem   ============================================================

1670       for x = 0 to (txnums-1)

1680 rem   ---------------------------------------
1690 rem   |  Pick a random terminal from range  |
1700 rem   ---------------------------------------
1710       teller = int(rnd*tnums)+lowtell

1720 rem   ------------------------------------------------------
1730 rem   |  Find branch #, add trailing zero for record number  |
1740 rem   ------------------------------------------------------
1750       bra = int (teller/10) * 10

1760 rem   ------------------------------------------------------
1770 rem   |  In 15% of all cases generate a non home transaction  |
1780 rem   ------------------------------------------------------
1790       if rnd*100000 >= 85000  then goto 2100

1800 rem   ----------------------------------------
1810 rem   |  Generate a "Home Branch Transaction"  |
1820 rem   ----------------------------------------

1830 rem   ------------------------------------------------
1840 rem   |  Pick an account out of 10,000 at this branch  |
1850 rem   ------------------------------------------------
1860       acc = int(rnd*10000)+ bra*1000

1870 rem   ---------------------------------------------------------------------
1880 rem   |  Set credit value to $1, indicating transaction at home branch  |
1890 rem   |  Insert correct credit (ramp or local)                           |
1900 rem   ---------------------------------------------------------------------
1910       if rampkt <> 0                  &
             then                          &
                  credit = rampid          &
 \                 rampkt = rampkt - 1     &
             else                          &
                  credit = locid
1770 rem   endif

1930 rem   ----------------------------------------
1940 rem   |  all records on same node as terminal  |
1950 rem   ----------------------------------------
1960       local = local +1
1970 rem   ------------------------------------------------------
1980 rem   |  all records (except history) on same disc volume  |
1990 rem   ------------------------------------------------------
2000       samedisc = samedisc +1

2010 rem   tgal command &
```

```
2020 rem   ------------------------------------------
2030 rem   |  all records same branch (should be 85%)  |
2040 rem   ------------------------------------------
2050       homebranch = homebranch +1
2060       goto 2310

2070 rem   --------------------------------------
2080 rem   |  generate a "Non Home Transaction"  |
2090 rem   --------------------------------------
2100       acc = int(rnd*(fullsize*1000))
2110       if int(acc/10000) = (bra/10) then goto 2100

2120 rem   ------------------------------------------------------------------
2130 rem   |  set credit value to 000 indicating transaction at home branch  |
2140 rem   ------------------------------------------------------------------

2150 rem   ------------------------------------------
2160 rem   |  insert correct credit (ramp or remote)  |
2170 rem   ------------------------------------------
2180       if rampkt <> 0                  &
            then                           &
                 credit = rampid           &
                 rampkt = rampkt -1        &
            else                           &
                 credit = remid
2190 rem   endif

2200 rem   ------------------------------------------
2210 rem   |  all records on same node as terminal ?  |
2220 rem   ------------------------------------------
2230       if int(acc/1000) <= hightell and int(acc/1000) >= lowtell   &
           then local = local +1                                       &
           else remote = remote +1

2240 rem   ------------------------------------------------------
2250 rem   |  all records (except history) on same disc volume ?  |
2260 rem   ------------------------------------------------------
2270       pnumt = int((teller-1)/psize)
2280       pnuma = int((acc-1)/(psize*1000))
2290       if pnuma = pnumt then samedisc = samedisc +1

2300 rem   ----------------------------------------
2310 rem   |  format data to have leading zeroes.  |
2320 rem   ----------------------------------------
2330       acc    = acc    * 0.000000000001
2340       teller = teller * 0.000000000001
2350       bra    = bra    * 0.000000000001
2360       credit = credit * 0.000000000001
2370 rem   tgal command &
```

```
2380 rem   ---------------------------------
2390 rem   |  write record to script file  |
2400 rem   ---------------------------------
2410       print    #(1+mod(x,scripts))   &
           using    ".############",       &
           acc,teller,bra,credit

2420       next x

2430 rem   tgal command &
```

```
2440 rem    ============================================================
2450 rem    =  Report Statistics                                       =
2460 rem    ============================================================

2470        print "*************************************************"
2480        print "Home Branch TX        :", homebranch
2490        print "Remote TX             :", remote
2500        print "Same Partition TX     :", samedisc
2510        print "Home Branch Frequency:", homebranch/(remote+local)*100
2520        print "Remote Frequency      :", remote/(remote+local)*100
2530        if err = 0 goto 2550
2540        print "Error: ";err
2550        end
```

The following is a fragment of one of the driver input scripts. The input consists of the account number, branch number, teller number and delta (debit or credit). For this benchmark the delta is always .01$ (if the transaction is local or 1000$ if the transaction is remote. The auditor will specify the exact amounts at the time of the audit. Notice that the teller number is simply branch number plus the teller number within the branch.

ACCOUNT	TELLER	BRANCH	DELTA
.000005558826	.000000005553	.000000005550	.000000000001
.000000394111	.000000000390	.000000000390	.000000000001
.000000256934	.000000000254	.000000000250	.000000000001
.000022222747	.000000005943	.000000005940	.000000100000
.000005450198	.000000005456	.000000005450	.000000000001
.000005993139	.000000005994	.000000005990	.000000000001
.000001958619	.000000001957	.000000001950	.000000000001
.000000547888	.000000000545	.000000000540	.000000000001
.000002552195	.000000002559	.000000002550	.000000000001
.000002392792	.000000002397	.000000002390	.000000000001
.000000860928	.000000000860	.000000000860	.000000000001
.000005147853	.000000005149	.000000005140	.000000000001
.000012526702	.000000003262	.000000003260	.000000100000
.000002403071	.000000002400	.000000002400	.000000000001
.000003363863	.000000003363	.000000003360	.000000000001
.000000558444	.000000000551	.000000000550	.000000000001
.000001168065	.000000001165	.000000001160	.000000000001
.000006003974	.000000006004	.000000006000	.000000000001
.000003067422	.000000003066	.000000003060	.000000000001
.000005959660	.000000005955	.000000005950	.000000000001
.000002661937	.000000002668	.000000002660	.000000000001
.000000405502	.000000000406	.000000000400	.000000000001
.000006371607	.000000006370	.000000006370	.000000000001
.000003903411	.000000002158	.000000002150	.000000100000
.000001376434	.000000001371	.000000001370	.000000000001

and much more.....

```
------------------------------------------------------------------------
*   ET1 Report (RTE System) - Performance Analysis and Measurement    *
------------------------------------------------------------------------

----- PARAMETERS -------------------------------------------------------

Test Title                 =   Test 200; 220  TPS; SQL
Test Date                  =   87-02-15
Test Length                =   02:19:38 - 02:44:59  (1521 sec)
Window Length              =   02:32:00 - 02:38:00  (360 sec)
Tx Log File                =   \DRIVER.$DM01.TXGEN.TXLOGSUM

Terminals/X25 Line         =       80 #
Tx Arrival Process         =   RANDOM
Tx Arrival Rate/X25 Line   =    7.500 tx/sec
Tx Arrival Rate/Terminal   =    0.094 tx/sec
Tx Inter-Arrival Time      =   10.667 sec

----- MEASUREMENT RESULTS  (vs 2.02) -----------------------------------

Tx Throughput              =  218.597 tx/sec
Tx Response Time Average   =    1.695 sec
                 Std Dev   =    0.721 sec
                 Minimum   =    0.464 sec
                 Maximum   =   48.168 sec
                 Count     =    78695 #

Tx Response Time 10%       =    1.050 sec   ( 12.2%)
                 20%       =    1.200 sec   ( 23.3%)
                 30%       =    1.300 sec   ( 31.3%)
                 40%       =    1.450 sec   ( 43.4%)
                 50%       =    1.550 sec   ( 50.8%)
                 60%       =    1.700 sec   ( 60.4%)
                 70%       =    1.900 sec   ( 70.2%)
                 80%       =    2.200 sec   ( 81.0%)
                 90%       =    2.600 sec   ( 90.1%)
                 95%       =    3.000 sec   ( 95.3%)

        Tx Think Average   =   10.659 sec
                 Std Dev   =   10.700 sec
                 Minimum   =    0.001 sec
                 Maximum   =  121.863 sec
                 Count     =    67170 #     ( 85.4%)

        Tx Delay Average   =    0.994 sec
                 Std Dev   =    0.834 sec
                 Minimum   =    0.000 sec
                 Maximum   =   39.744 sec
                 Count     =    11525 #     ( 14.6%)

------------------------------------------------------------------------
                  *  ET1 Response Time Distribution  *

------------------------------------------------------------------------
   Mean        Std         Min         Max         Count      Thruput
   1.695 s     0.721 s     0.464 s    48.168 s     78695      218.597 tx/s
------------------------------------------------------------------------

RespTime Count  Pct  CumPct
-------- ------ ----- ------
```

0.000	0	0.0	0.0	|
0.100	0	0.0	0.0	|
0.200	0	0.0	0.0	|
0.300	0	0.0	0.0	|
0.400	0	0.0	0.0	|
0.500	7	0.0	0.0	|
0.600	69	0.1	0.1	|
0.700	401	0.5	0.6	|**
0.800	1072	1.4	2.0	|******
0.900	2203	2.8	4.8	|**********
1.000	3544	4.5	9.3	|**********
1.100	4949	6.3	15.6	|**********
1.200	6100	7.8	23.3	|**********
1.300	6313	8.0	31.3	|**********
1.400	6477	8.2	39.6	|**********
1.500	6039	7.7	47.2	|**********
1.600	5433	6.9	54.1	|**********
1.700	4933	6.3	60.4	|**********
1.800	4109	5.2	65.6	|**********
1.900	3587	4.6	70.2	|**********
2.000	3267	4.2	74.3	|**********
2.100	2783	3.5	77.9	|**********
2.200	2445	3.1	81.0	|**********
2.300	2177	2.8	83.8	|**********
2.400	1898	2.4	86.2	|**********
2.500	1648	2.1	88.3	|**********
2.600	1416	1.8	90.1	|*********
2.700	1264	1.6	91.7	|********
2.800	1132	1.4	93.1	|*******
2.900	944	1.2	94.3	|******
3.000	761	1.0	95.3	|****
3.100	653	0.8	96.1	|****
3.200	555	0.7	96.8	|***
3.300	432	0.5	97.4	|**
3.400	410	0.5	97.9	|**
3.500	297	0.4	98.3	|*
3.600	268	0.3	98.6	|*
3.700	186	0.2	98.8	|*
3.800	155	0.2	99.0	|*
3.900	144	0.2	99.2	|
4.000	90	0.1	99.3	|
4.100	87	0.1	99.4	|
4.200	78	0.1	99.5	|
4.300	64	0.1	99.6	|
4.400	49	0.1	99.7	|
4.500	41	0.1	99.7	|
4.600	37	0.0	99.8	|
4.700	30	0.0	99.8	|
4.800	20	0.0	99.8	|
4.900	10	0.0	99.9	|
5.000	25	0.0	99.9	|
5.100	15	0.0	99.9	|
5.200	10	0.0	99.9	|
5.300	13	0.0	99.9	|
5.400	9	0.0	99.9	|
5.500	9	0.0	100.0	|
5.600	3	0.0	100.0	|
5.700	4	0.0	100.0	|
5.800	1	0.0	100.0	|
5.900	2	0.0	100.0	|

6.000	3	0.0	100.0
6.100	3	0.0	100.0
6.200	1	0.0	100.0
6.300	4	0.0	100.0
6.400	0	0.0	100.0
6.500	1	0.0	100.0
6.600	5	0.0	100.0
6.700	0	0.0	100.0
6.800	0	0.0	100.0
6.900	0	0.0	100.0
7.000	3	0.0	100.0
7.100	1	0.0	100.0
7.200	1	0.0	100.0
7.300	0	0.0	100.0
7.400	0	0.0	100.0
7.500	0	0.0	100.0
7.600	0	0.0	100.0
7.700	0	0.0	100.0
7.800	0	0.0	100.0
7.900	0	0.0	100.0
8.000	0	0.0	100.0
8.100	0	0.0	100.0
8.200	0	0.0	100.0
8.300	0	0.0	100.0
8.400	0	0.0	100.0
8.500	0	0.0	100.0
8.600	0	0.0	100.0
8.700	0	0.0	100.0
8.800	0	0.0	100.0
8.900	0	0.0	100.0
9.000	0	0.0	100.0
9.100	0	0.0	100.0
9.200	0	0.0	100.0
9.300	0	0.0	100.0
9.400	0	0.0	100.0
9.500	0	0.0	100.0
9.600	0	0.0	100.0
9.700	0	0.0	100.0
9.800	0	0.0	100.0
9.900	0	0.0	100.0
10.000	0	0.0	100.0
10.100	0	0.0	100.0
10.200	0	0.0	100.0
10.300	0	0.0	100.0
10.400	0	0.0	100.0
10.500	0	0.0	100.0
10.600	0	0.0	100.0
10.700	0	0.0	100.0
10.800	0	0.0	100.0
10.900	0	0.0	100.0
11.000	0	0.0	100.0
11.100	0	0.0	100.0
11.200	0	0.0	100.0
11.300	0	0.0	100.0
11.400	0	0.0	100.0
11.500	0	0.0	100.0
11.600	0	0.0	100.0
11.700	0	0.0	100.0
11.800	0	0.0	100.0
11.900	0	0.0	100.0

12.000	0	0.0	100.0
12.100	0	0.0	100.0
12.200	0	0.0	100.0
12.300	0	0.0	100.0
12.400	0	0.0	100.0
12.500	0	0.0	100.0
12.600	0	0.0	100.0
12.700	0	0.0	100.0
12.800	0	0.0	100.0
12.900	0	0.0	100.0
13.000	0	0.0	100.0
13.100	0	0.0	100.0
13.200	0	0.0	100.0
13.300	0	0.0	100.0
13.400	0	0.0	100.0
13.500	0	0.0	100.0
13.600	0	0.0	100.0
13.700	0	0.0	100.0
13.800	0	0.0	100.0
13.900	0	0.0	100.0
14.000	0	0.0	100.0
14.100	0	0.0	100.0
14.200	0	0.0	100.0
14.300	0	0.0	100.0
14.400	0	0.0	100.0
14.500	0	0.0	100.0
14.600	0	0.0	100.0
14.700	0	0.0	100.0
14.800	0	0.0	100.0
14.900	0	0.0	100.0
15.000	0	0.0	100.0
15.100	0	0.0	100.0
15.200	0	0.0	100.0
15.300	0	0.0	100.0
15.400	0	0.0	100.0
15.500	0	0.0	100.0
15.600	0	0.0	100.0
15.700	0	0.0	100.0
15.800	0	0.0	100.0
15.900	0	0.0	100.0
16.000	0	0.0	100.0
16.100	0	0.0	100.0
16.200	0	0.0	100.0
16.300	0	0.0	100.0
16.400	0	0.0	100.0
16.500	0	0.0	100.0
16.600	0	0.0	100.0
16.700	0	0.0	100.0
16.800	0	0.0	100.0
16.900	0	0.0	100.0
17.000	0	0.0	100.0
17.100	0	0.0	100.0
17.200	0	0.0	100.0
17.300	0	0.0	100.0
17.400	0	0.0	100.0
17.500	0	0.0	100.0
17.600	0	0.0	100.0
17.700	0	0.0	100.0
17.800	0	0.0	100.0
17.900	0	0.0	100.0

18.000	0	0.0	100.0
18.100	0	0.0	100.0
18.200	0	0.0	100.0
18.300	0	0.0	100.0
18.400	0	0.0	100.0
18.500	0	0.0	100.0
18.600	0	0.0	100.0
18.700	0	0.0	100.0
18.800	0	0.0	100.0
18.900	0	0.0	100.0
19.000	0	0.0	100.0
19.100	0	0.0	100.0
19.200	0	0.0	100.0
19.300	0	0.0	100.0
19.400	0	0.0	100.0
19.500	0	0.0	100.0
19.600	0	0.0	100.0
19.700	0	0.0	100.0
19.800	0	0.0	100.0
19.900	0	0.0	100.0
20.000	0	0.0	100.0
20.100	0	0.0	100.0
20.200	0	0.0	100.0
20.300	0	0.0	100.0
20.400	0	0.0	100.0
20.500	0	0.0	100.0
20.600	0	0.0	100.0
20.700	0	0.0	100.0
20.800	0	0.0	100.0
20.900	0	0.0	100.0
21.000	0	0.0	100.0
21.100	0	0.0	100.0
21.200	0	0.0	100.0
21.300	0	0.0	100.0
21.400	0	0.0	100.0
21.500	0	0.0	100.0
21.600	0	0.0	100.0
21.700	0	0.0	100.0
21.800	0	0.0	100.0
21.900	0	0.0	100.0
22.000	0	0.0	100.0
22.100	0	0.0	100.0
22.200	0	0.0	100.0
22.300	0	0.0	100.0
22.400	0	0.0	100.0
22.500	0	0.0	100.0
22.600	0	0.0	100.0
22.700	0	0.0	100.0
22.800	0	0.0	100.0
22.900	0	0.0	100.0
23.000	0	0.0	100.0
23.100	0	0.0	100.0
23.200	0	0.0	100.0
23.300	0	0.0	100.0
23.400	0	0.0	100.0
23.500	0	0.0	100.0
23.600	0	0.0	100.0
23.700	0	0.0	100.0
23.800	0	0.0	100.0
23.900	0	0.0	100.0

24.000	0	0.0	100.0
24.100	0	0.0	100.0
24.200	0	0.0	100.0
24.300	0	0.0	100.0
24.400	0	0.0	100.0
24.500	0	0.0	100.0
24.600	0	0.0	100.0
24.700	0	0.0	100.0
24.800	0	0.0	100.0
24.900	0	0.0	100.0
25.000	0	0.0	100.0
25.100	0	0.0	100.0
25.200	0	0.0	100.0
25.300	0	0.0	100.0
25.400	0	0.0	100.0
25.500	0	0.0	100.0
25.600	0	0.0	100.0
25.700	0	0.0	100.0
25.800	0	0.0	100.0
25.900	0	0.0	100.0
26.000	0	0.0	100.0
26.100	0	0.0	100.0
26.200	0	0.0	100.0
26.300	0	0.0	100.0
26.400	0	0.0	100.0
26.500	0	0.0	100.0
26.600	0	0.0	100.0
26.700	0	0.0	100.0
26.800	0	0.0	100.0
26.900	0	0.0	100.0
27.000	0	0.0	100.0
27.100	0	0.0	100.0
27.200	0	0.0	100.0
27.300	0	0.0	100.0
27.400	0	0.0	100.0
27.500	0	0.0	100.0
27.600	0	0.0	100.0
27.700	0	0.0	100.0
27.800	0	0.0	100.0
27.900	0	0.0	100.0
28.000	0	0.0	100.0
28.100	0	0.0	100.0
28.200	0	0.0	100.0
28.300	0	0.0	100.0
28.400	0	0.0	100.0
28.500	0	0.0	100.0
28.600	0	0.0	100.0
28.700	0	0.0	100.0
28.800	0	0.0	100.0
28.900	0	0.0	100.0
29.000	0	0.0	100.0
29.100	0	0.0	100.0
29.200	0	0.0	100.0
29.300	0	0.0	100.0
29.400	0	0.0	100.0
29.500	0	0.0	100.0
29.600	0	0.0	100.0
29.700	0	0.0	100.0
29.800	0	0.0	100.0
29.900	0	0.0	100.0

30.000	0	0.0	100.0
30.100	0	0.0	100.0
30.200	0	0.0	100.0
30.300	0	0.0	100.0
30.400	0	0.0	100.0
30.500	0	0.0	100.0
30.600	0	0.0	100.0
30.700	0	0.0	100.0
30.800	0	0.0	100.0
30.900	0	0.0	100.0
31.000	0	0.0	100.0
31.100	0	0.0	100.0
31.200	0	0.0	100.0
31.300	0	0.0	100.0
31.400	0	0.0	100.0
31.500	0	0.0	100.0
31.600	1	0.0	100.0
31.700	0	0.0	100.0
31.800	0	0.0	100.0
31.900	0	0.0	100.0
32.000	0	0.0	100.0
32.100	0	0.0	100.0
32.200	0	0.0	100.0
32.300	0	0.0	100.0
32.400	0	0.0	100.0
32.500	0	0.0	100.0
32.600	0	0.0	100.0
32.700	0	0.0	100.0
32.800	0	0.0	100.0
32.900	0	0.0	100.0
33.000	0	0.0	100.0
33.100	0	0.0	100.0
33.200	0	0.0	100.0
33.300	0	0.0	100.0
33.400	0	0.0	100.0
33.500	0	0.0	100.0
33.600	0	0.0	100.0
33.700	0	0.0	100.0
33.800	0	0.0	100.0
33.900	0	0.0	100.0
34.000	0	0.0	100.0
34.100	0	0.0	100.0
34.200	0	0.0	100.0
34.300	0	0.0	100.0
34.400	0	0.0	100.0
34.500	0	0.0	100.0
34.600	0	0.0	100.0
34.700	0	0.0	100.0
34.800	0	0.0	100.0
34.900	0	0.0	100.0
35.000	0	0.0	100.0
35.100	0	0.0	100.0
35.200	0	0.0	100.0
35.300	0	0.0	100.0
35.400	0	0.0	100.0
35.500	0	0.0	100.0
35.600	0	0.0	100.0
35.700	0	0.0	100.0
35.800	0	0.0	100.0
35.900	0	0.0	100.0

36.000	0	0.0	100.0	
36.100	0	0.0	100.0	
36.200	1	0.0	100.0	
36.300	1	0.0	100.0	
36.400	0	0.0	100.0	
36.500	0	0.0	100.0	
36.600	0	0.0	100.0	
36.700	0	0.0	100.0	
36.800	0	0.0	100.0	
36.900	0	0.0	100.0	
37.000	0	0.0	100.0	
37.100	0	0.0	100.0	
37.200	0	0.0	100.0	
37.300	0	0.0	100.0	
37.400	0	0.0	100.0	
37.500	0	0.0	100.0	
37.600	0	0.0	100.0	
37.700	0	0.0	100.0	
37.800	0	0.0	100.0	
37.900	0	0.0	100.0	
38.000	0	0.0	100.0	
38.100	0	0.0	100.0	
38.200	0	0.0	100.0	
38.300	0	0.0	100.0	
38.400	0	0.0	100.0	
38.500	0	0.0	100.0	
38.600	0	0.0	100.0	
38.700	0	0.0	100.0	
38.800	0	0.0	100.0	
38.900	0	0.0	100.0	
39.000	0	0.0	100.0	
39.100	0	0.0	100.0	
39.200	0	0.0	100.0	
39.300	0	0.0	100.0	
39.400	1	0.0	100.0	
39.500	0	0.0	100.0	
39.600	0	0.0	100.0	
39.700	0	0.0	100.0	
39.800	0	0.0	100.0	
39.900	0	0.0	100.0	
40.000	0	0.0	100.0	
40.100	0	0.0	100.0	
40.200	0	0.0	100.0	
40.300	0	0.0	100.0	
40.400	0	0.0	100.0	
40.500	0	0.0	100.0	
40.600	0	0.0	100.0	
40.700	0	0.0	100.0	
40.800	0	0.0	100.0	
40.900	0	0.0	100.0	
41.000	0	0.0	100.0	
41.100	0	0.0	100.0	
41.200	0	0.0	100.0	
41.300	0	0.0	100.0	
41.400	0	0.0	100.0	
41.500	0	0.0	100.0	
41.600	0	0.0	100.0	
41.700	0	0.0	100.0	
41.800	0	0.0	100.0	
41.900	0	0.0	100.0	

42.000	0	0.0	100.0
42.100	0	0.0	100.0
42.200	0	0.0	100.0
42.300	0	0.0	100.0
42.400	0	0.0	100.0
42.500	0	0.0	100.0
42.600	0	0.0	100.0
42.700	0	0.0	100.0
42.800	0	0.0	100.0
42.900	0	0.0	100.0
43.000	0	0.0	100.0
43.100	0	0.0	100.0
43.200	0	0.0	100.0
43.300	0	0.0	100.0
43.400	0	0.0	100.0
43.500	0	0.0	100.0
43.600	0	0.0	100.0
43.700	0	0.0	100.0
43.800	0	0.0	100.0
43.900	0	0.0	100.0
44.000	0	0.0	100.0
44.100	0	0.0	100.0
44.200	0	0.0	100.0
44.300	0	0.0	100.0
44.400	0	0.0	100.0
44.500	0	0.0	100.0
44.600	0	0.0	100.0
44.700	0	0.0	100.0
44.800	0	0.0	100.0
44.900	0	0.0	100.0
45.000	0	0.0	100.0
45.100	0	0.0	100.0
45.200	0	0.0	100.0
45.300	0	0.0	100.0
45.400	0	0.0	100.0
45.500	0	0.0	100.0
45.600	0	0.0	100.0
45.700	0	0.0	100.0
45.800	0	0.0	100.0
45.900	0	0.0	100.0
46.000	0	0.0	100.0
46.100	0	0.0	100.0
46.200	0	0.0	100.0
46.300	0	0.0	100.0
46.400	0	0.0	100.0
46.500	0	0.0	100.0
46.600	0	0.0	100.0
46.700	0	0.0	100.0
46.800	0	0.0	100.0
46.900	0	0.0	100.0
47.000	0	0.0	100.0
47.100	0	0.0	100.0
47.200	0	0.0	100.0
47.300	0	0.0	100.0
47.400	0	0.0	100.0
47.500	0	0.0	100.0
47.600	0	0.0	100.0
47.700	0	0.0	100.0
47.800	0	0.0	100.0
47.900	0	0.0	100.0

48.000	0	0.0	100.0	|
48.100	0	0.0	100.0	|
48.200	1	0.0	100.0	|

PATHWAY SCREEN COBOL - T9153C00 - (15JUL87) SOURCE LANGUAGE: SCOBOL TARGET MACHINE: TCP/INTERP
COMPILED: 16 FEB 87 16:42:52 OPTIONS: ON - (LIST,WARN) OFF - (MAP,SYMBOLS,CROSSREF)

```
1       ?symbols
2        identification division.
3         program-id. ids-drva-short.
4         author.                          tlw.
5        installation.                     tandem   hprc.
6        date-written.                     28-jan-87.
7        DATE-COMPILED.  87/02/16 - 16:43:04.
9        environment division.
10        configuration section.
11        source-computer.          t-16.
12        object-computer.          t-16,
13           terminal is intelligent.
14      *
15      *  this requestor is to be used in the topgun benchmark.
16      *
17      *
18      *  terminal (x.25)           requestor           server
19      *
20      *  100 bytes        ----->
21      *
22      *                            52 bytes   ----->
23      *
24      *  = = = = = = = = = = = = = = = = = = = = = =  (no error) = = = =
25      *
26      *                                               functioncode := 0
27      *                                         <-----    100 bytes
28      *
29      *                   <-----    200 bytes
30      *
31      *  = = = = = = = = = = = = = = = = = = = = = =  (no error) = = = =
32      *  = = = = = = = = = = = = = = = = = = = = = =  (error) = = = = =
33      *
34      *                                               functioncode := non-zero
35      *
36      *                                         <-----    100 bytes
37      *
38      *                   <-----     52 bytes
39      *
40      *  = = = = = = = = = = = = = = = = = = = = = =  (error) = = = = =
```

```
41       /
42        data division.
43         working-storage section.
44         01 x25in.
45            03 tx-data                    pic x(52).
46            03 tx-misc                    pic x(48).
47
48         01 x25out-ok                     pic x(200).
49         01 x25out-error                  pic x(52).
50
51         01 server-reply.
52            03 sv-header                  pic 9(4) comp.
53            03 sv-data                    pic x(100).
54
55         01 server-error.
56            03 sv-header                  pic 9(4) comp.
57            03 sv-data-bad                pic x(100).
58
59         01 exit-flag                     pic 9(01) comp value zero.
60            88 exit-program                            value 1.
61            88 line-drop                               value 2.
```

respects 18 Feb 87
Tom Sawyer

```
62        /
63        ?heading "top level procedure div"
64         procedure division.
65
66         begin-program.
67             send message
68                  reply yields x25in
69                  on error go to reconnect-line.
70             perform loop until line-drop.
71
72         reconnect-line.
73             reconnect modem.
74             move 0 to exit-flag.
75             go to begin-program.
76
77         do-abort.
78             abort-transaction.
79             move tx-data to x25out-error.
80             send message x25out-error
81                  reply yields x25in
82                  on error perform set-line-drop.
83             move 1000 to termination-status.
84
85         loop.
86             begin-transaction.
87             send tx-data to "eta"
88                reply code  0 yields server-reply
89                      code  1 yields server-error
90                      code  2 yields server-error
91                      code  3 yields server-error
92                      code  4 yields server-error
93                      code  5 yields server-error
94                      code  6 yields server-error
95                      code  7 yields server-error
96                      code  8 yields server-error
97                      code  9 yields server-error
98                on error perform do-abort.
99
100            if   termination-status = 1
101                 end-transaction
102                 move sv-data to x25out-ok
103                 send message x25out-ok
104                      reply yields x25in
105                      on error perform set-line-drop
106            else if termination-status < 10 perform do-abort.
```

inspected 19 Feb 82
by Tim Sawyer

```
107         /
108          set-line-drop.
109              move 2 to exit-flag.
```

```
OBJECT FILE NAME IS $BASE.GRAYTGUN.POBJ
PROGRAM NAME IS IDS-DRVA-SHORT
PROGRAM VERSION IS 1
NO. ERRORS=0;     NO. WARNINGS=0
CODE SIZE=258
RUN UNIT SIZE=964
DATA SIZE=598
NUMBER OF SOURCE LINES READ=108
MAXIMUM SYMBOL TABLE SIZE=2220  WORDS
ELAPSED TIME -   0:00:29
```

```
!-------------------------------------------------------------------!
!                                                                   !
! create file for TOPGUN benchmark.                                 !
!                                                                   !
! this is version   3                                               !
!                                                                   !
!-------------------------------------------------------------------!
! change log                                                        !
!-------------------------------------------------------------------!
!                                                                   !
! tlw  01 - initial version:  general cleanup; make db layouts      !
!           consistent with spec.                                   !
!                                                                   !
! tlw  02 - change "rest" parts of records to make ten fields per   !
!           record.                                                 !
!                                                                   !
! tlw  03 - change "rest" fields to pic x                           !
!           change balance fields to pic s9(12)                     !
!           change all other fields to pic x                        !
!           change "history_delta" to "history_amount" for clarity  !
!           change first record to 0 (to be consistent with         !
!           "syskey" value)                                         !
!                                                                   !
!-------------------------------------------------------------------!

!-------------------------------------------------------------------!
! set log file                                                      !
!-------------------------------------------------------------------!
LOG $S.#SQL.CREATE;
?section create_catalog
!-------------------------------------------------------------------!
! create catalogs                                                   !
!-------------------------------------------------------------------!
create catalog \a.$system.CATET1;
create catalog \b.$system.CATET1;
create catalog \c.$system.CATET1;
create catalog \d.$system.CATET1;
create catalog \e.$system.CATET1;

?section create_accounts
!-------------------------------------------------------------------!
! create accounts file  -  record length 100                        !
!-------------------------------------------------------------------!
!                                                                   !
!  field:  len: function:                                           !
!  ------  ---- ---------                                          !
!    1      12  account number                                     !
!    2      12  account balance                                    !
!    3      12  account branch                                     !
!    4       4  filler                                             !
!    5      10  filler                                             !
!    6      10  filler                                             !
!    7      10  filler                                             !
!    8      10  filler                                             !
!    9      10  filler                                             !
!   10      10  filler                                             !
!                                                                   !
!  partitions:                                                      !
!  -----------                                                      !
!    primary    : \a.$system                                       !
```

```
!                                                                    !
!    secondaries:                                                    !
!                                                                    !
!    \a.$da02   \b.$system  \c.$system  \d.$system  \e.$system        !
!    \a.$da04   \b.$db02    \c.$dc02    \d.$dd02                     !
!    \a.$da06   \b.$db04    \c.$dc04    \d.$dd04                     !
!    \a.$da08   \b.$db06    \c.$dc06    \d.$dd06                     !
!    \a.$da10   \b.$db08    \c.$dc08    \d.$dd08                     !
!    \a.$da12   \b.$db10    \c.$dc10    \d.$dd10                     !
!    \a.$da14   \b.$db12    \c.$dc12    \d.$dd12                     !
!               \b.$db14    \c.$dc14    \d.$dd14                     !
!                                                                    !
!--------------------------------------------------------------------!
create table \a.$system.sqlet1.accounts (

       account_number  PICTURE x(12),
       account_balance PICTURE s9(12),
       account_branch  PICTURE x(12),
       account_rest_1  PICTURE x(4),
       account_rest_2  PICTURE x(10),
       account_rest_3  PICTURE x(10),
       account_rest_4  PICTURE x(10),
       account_rest_5  PICTURE x(10),
       account_rest_6  PICTURE x(10),
       account_rest_7  PICTURE x(10)
      )
      ORGANIZATION Key sequenced
      KEY account_number
      NO AUDIT
      BUFFERED
      CATALOG \a.$system.catet1
      BLOCKSIZE 4096
      MAXEXTENTS 16
      EXTENT ( 2600,2600 )
      PARTITION (

\a.$DA02.SQLET1.ACCOUNTS    CATALOG \a.$system.catet1 EXTENT(2600,2600) START KEY "000000800000",
\a.$DA04.SQLET1.ACCOUNTS    CATALOG \a.$system.catet1 EXTENT(2600,2600) START KEY "000001600000",
\a.$DA06.SQLET1.ACCOUNTS    CATALOG \a.$system.catet1 EXTENT(2600,2600) START KEY "000002400000",
\a.$DA08.SQLET1.ACCOUNTS    CATALOG \a.$system.catet1 EXTENT(2600,2600) START KEY "000003200000",
\a.$DA10.SQLET1.ACCOUNTS    CATALOG \a.$system.catet1 EXTENT(2600,2600) START KEY "000004000000",
\a.$Da12.SQLET1.ACCOUNTS    CATALOG \a.$system.catet1 EXTENT(2600,2600) START KEY "000004800000",
\a.$Da14.SQLET1.ACCOUNTS    CATALOG \a.$system.catet1 EXTENT(2600,2600) START KEY "000005600000",

\b.$system.SQLET1.ACCOUNTS CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000006400000",
\b.$Db02.SQLET1.ACCOUNTS    CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000007200000",
\b.$Db04.SQLET1.ACCOUNTS    CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000008000000",
\b.$Db06.SQLET1.ACCOUNTS    CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000008800000",
\b.$Db08.SQLET1.ACCOUNTS    CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000009600000",
\b.$Db10.SQLET1.ACCOUNTS    CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000010400000",
\b.$Db12.SQLET1.ACCOUNTS    CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000011200000",
\b.$Db14.SQLET1.ACCOUNTS    CATALOG \b.$system.catet1 EXTENT(2600,2600) START KEY "000012000000",

\c.$system.SQLET1.ACCOUNTS CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000012800000",
\c.$Dc02.SQLET1.ACCOUNTS    CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000013600000",
\c.$Dc04.SQLET1.ACCOUNTS    CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000014400000",
\c.$Dc06.SQLET1.ACCOUNTS    CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000015200000",
\c.$Dc08.SQLET1.ACCOUNTS    CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000016000000",
\c.$Dc10.SQLET1.ACCOUNTS    CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000016800000",
\c.$Dc12.SQLET1.ACCOUNTS    CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000017600000",
```

```
\c.$Dc14.SQLET1.ACCOUNTS    CATALOG \c.$system.catet1 EXTENT(2600,2600) START KEY "000018400000",

\d.$system.SQLET1.ACCOUNTS CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000019200000",
\d.$Dd02.SQLET1.ACCOUNTS    CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000020000000",
\d.$Dd04.SQLET1.ACCOUNTS    CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000020800000",
\d.$Dd06.SQLET1.ACCOUNTS    CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000021600000",
\d.$Dd08.SQLET1.ACCOUNTS    CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000022400000",
\d.$Dd10.SQLET1.ACCOUNTS    CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000023200000",
\d.$Dd12.SQLET1.ACCOUNTS    CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000024000000",
\d.$Dd14.SQLET1.ACCOUNTS    CATALOG \d.$system.catet1 EXTENT(2600,2600) START KEY "000024800000",

\e.$system.SQLET1.ACCOUNTS CATALOG \e.$system.catet1 EXTENT(1300,1300) START KEY "000025600000"

);

?section create_branches
!-------------------------------------------------------------------!
! create branches file  -  record length 100                        !
!-------------------------------------------------------------------!
!                                                                   !
!  field:  len: function:                                           !
!  ------  ---- ---------                                           !
!    1      12  branch number                                       !
!    2      12  branch balance                                      !
!    3       6  filler                                              !
!    4      10  filler                                              !
!    5      10  filler                                              !
!    6      10  filler                                              !
!    7      10  filler                                              !
!    8      10  filler                                              !
!    9      10  filler                                              !
!   10      10  filler                                              !
!                                                                   !
!  partitions:                                                      !
!  -----------                                                      !
!    primary    : \a.$system                                        !
!                                                                   !
!    secondaries:                                                   !
!                                                                   !
!    \a.$da02   \b.$system  \c.$system  \d.$system  \e.$system       !
!    \a.$da04   \b.$db02    \c.$dc02    \d.$dd02                    !
!    \a.$da06   \b.$db04    \c.$dc04    \d.$dd04                    !
!    \a.$da08   \b.$db06    \c.$dc06    \d.$dd06                    !
!    \a.$da10   \b.$db08    \c.$dc08    \d.$dd08                    !
!    \a.$da12   \b.$db10    \c.$dc10    \d.$dd10                    !
!    \a.$da14   \b.$db12    \c.$dc12    \d.$dd12                    !
!               \b.$db14    \c.$dc14    \d.$dd14                    !
!                                                                   !
!-------------------------------------------------------------------!
CREATE TABLE \a.$SYSTEM.SQLET1.BRANCHES (
       branch_number   PICTURE x(12),
       branch_balance  PICTURE s9(12),
       branch_rest_1   PICTURE x(12),
       branch_rest_2   PICTURE x(4),
       branch_rest_3   PICTURE x(10),
       branch_rest_4   PICTURE x(10),
       branch_rest_5   PICTURE x(10),
       branch_rest_6   PICTURE x(10),
       branch_rest_7   PICTURE x(10),
       branch_rest_8   PICTURE x(10)
```

```
)
ORGANIZATION Relative
NO AUDIT
BUFFERED
CATALOG \A.$SYSTEM.CATET1
BLOCKSIZE 512
MAXEXTENTS 16
EXTENT ( 21,2 )
PARTITION (

\a.$DA02.sqlet1.branches   CATALOG \A.$SYSTEM.CATET1 EXTENT( 20,2 ),
\a.$DA04.sqlet1.branches   CATALOG \A.$SYSTEM.CATET1 EXTENT( 20,2 ),
\a.$DA06.sqlet1.branches   CATALOG \A.$SYSTEM.CATET1 EXTENT( 20,2 ),
\a.$DA08.sqlet1.branches   CATALOG \A.$SYSTEM.CATET1 EXTENT( 21,2 ),
\a.$DA10.sqlet1.branches   CATALOG \A.$SYSTEM.CATET1 EXTENT( 20,2 ),
\a.$DA12.sqlet1.branches   CATALOG \A.$SYSTEM.CATET1 EXTENT( 20,2 ),
\a.$DA14.sqlet1.branches   CATALOG \A.$SYSTEM.CATET1 EXTENT( 22,2 ),

\b.$system.sqlet1.branches CATALOG \b.$SYSTEM.CATET1 EXTENT( 21,2 ),
\b.$Db02.sqlet1.branches   CATALOG \b.$SYSTEM.CATET1 EXTENT( 20,2 ),
\b.$Db04.sqlet1.branches   CATALOG \b.$SYSTEM.CATET1 EXTENT( 20,2 ),
\b.$Db06.sqlet1.branches   CATALOG \b.$SYSTEM.CATET1 EXTENT( 20,2 ),
\b.$Db08.sqlet1.branches   CATALOG \b.$SYSTEM.CATET1 EXTENT( 21,2 ),
\b.$Db10.sqlet1.branches   CATALOG \b.$SYSTEM.CATET1 EXTENT( 20,2 ),
\b.$Db12.sqlet1.branches   CATALOG \b.$SYSTEM.CATET1 EXTENT( 20,2 ),
\b.$Db14.sqlet1.branches   CATALOG \b.$SYSTEM.CATET1 EXTENT( 22,2 ),

\c.$system.sqlet1.branches CATALOG \c.$SYSTEM.CATET1 EXTENT( 21,2 ),
\c.$Dc02.sqlet1.branches   CATALOG \c.$SYSTEM.CATET1 EXTENT( 20,2 ),
\c.$Dc04.sqlet1.branches   CATALOG \c.$SYSTEM.CATET1 EXTENT( 20,2 ),
\c.$Dc06.sqlet1.branches   CATALOG \c.$SYSTEM.CATET1 EXTENT( 20,2 ),
\c.$Dc08.sqlet1.branches   CATALOG \c.$SYSTEM.CATET1 EXTENT( 21,2 ),
\c.$Dc10.sqlet1.branches   CATALOG \c.$SYSTEM.CATET1 EXTENT( 20,2 ),
\c.$Dc12.sqlet1.branches   CATALOG \c.$SYSTEM.CATET1 EXTENT( 20,2 ),
\c.$Dc14.sqlet1.branches   CATALOG \c.$SYSTEM.CATET1 EXTENT( 22,2 ),

\d.$system.sqlet1.branches CATALOG \d.$SYSTEM.CATET1 EXTENT( 21,2 ),
\d.$Dd02.sqlet1.branches   CATALOG \d.$SYSTEM.CATET1 EXTENT( 20,2 ),
\d.$Dd04.sqlet1.branches   CATALOG \d.$SYSTEM.CATET1 EXTENT( 20,2 ),
\d.$Dd06.sqlet1.branches   CATALOG \d.$SYSTEM.CATET1 EXTENT( 20,2 ),
\d.$Dd08.sqlet1.branches   CATALOG \d.$SYSTEM.CATET1 EXTENT( 21,2 ),
\d.$Dd10.sqlet1.branches   CATALOG \d.$SYSTEM.CATET1 EXTENT( 20,2 ),
\d.$Dd12.sqlet1.branches   CATALOG \d.$SYSTEM.CATET1 EXTENT( 20,2 ),
\d.$Dd14.sqlet1.branches   CATALOG \d.$SYSTEM.CATET1 EXTENT( 22,2 ),

\e.$system.sqlet1.branches CATALOG \e.$SYSTEM.CATET1 EXTENT( 20,2 )
);

?section create_tellers
!-------------------------------------------------------------------!
! create tellers file  -  record length 100                         !
!-------------------------------------------------------------------!
!                                                                   !
!  field:  len: function:                                           !
!  ------  ---- ---------                                           !
!    1      12  teller number                                       !
!    2      12  teller balance                                      !
!    3       6  filler                                              !
!    4      10  filler                                              !
!    5      10  filler                                              !
```

```
!    6      10  filler                                                    !
!    7      10  filler                                                    !
!    8      10  filler                                                    !
!    9      10  filler                                                    !
!   10      10  filler                                                    !
!                                                                         !
!  partitions:                                                            !
!  -----------                                                            !
!    primary    : \a.$system                                             !
!                                                                         !
!    secondaries:                                                         !
!                                                                         !
!    \a.$da02   \b.$system  \c.$system  \d.$system  \e.$system            !
!    \a.$da04   \b.$db02    \c.$dc02    \d.$dd02                          !
!    \a.$da06   \b.$db04    \c.$dc04    \d.$dd04                          !
!    \a.$da08   \b.$db06    \c.$dc06    \d.$dd06                          !
!    \a.$da10   \b.$db08    \c.$dc08    \d.$dd08                          !
!    \a.$da12   \b.$db10    \c.$dc10    \d.$dd10                          !
!    \a.$da14   \b.$db12    \c.$dc12    \d.$dd12                          !
!               \b.$db14    \c.$dc14    \d.$dd14                          !
!                                                                         !
!-------------------------------------------------------------------------!
CREATE TABLE \a.$SYSTEM.SQLET1.TELLERS  (
       teller_number   PICTURE x(12),
       teller_balance  PICTURE s9(12),
       teller_rest_1   PICTURE x(12),
       teller_rest_2   PICTURE x(4),
       teller_rest_3   PICTURE x(10),
       teller_rest_4   PICTURE x(10),
       teller_rest_5   PICTURE x(10),
       teller_rest_6   PICTURE x(10),
       teller_rest_7   PICTURE x(10),
       teller_rest_8   PICTURE x(10)
      )
      ORGANIZATION Relative
      NO AUDIT
      BUFFERED
      CATALOG \A.$system.catet1
      BLOCKSIZE 2048
      MAXEXTENTS 16
      EXTENT (13,2 )
      PARTITION (

        \a.$DA02.sqlet1.tellers    CATALOG \A.$SYSTEM.CATET1 EXTENT(13,2 ),
        \a.$DA04.sqlet1.tellers    CATALOG \A.$SYSTEM.CATET1 EXTENT(13,2 ),
        \a.$DA06.sqlet1.tellers    CATALOG \A.$SYSTEM.CATET1 EXTENT(13,2 ),
        \a.$DA08.sqlet1.tellers    CATALOG \A.$SYSTEM.CATET1 EXTENT(13,2 ),
        \a.$DA10.sqlet1.tellers    CATALOG \a.$SYSTEM.CATET1 EXTENT(13,2 ),
        \a.$DA12.sqlet1.tellers    CATALOG \a.$SYSTEM.CATET1 EXTENT(13,2 ),
        \a.$da14.sqlet1.tellers    CATALOG \a.$SYSTEM.CATET1 EXTENT(14,2 ),

        \b.$system.sqlet1.tellers  CATALOG \b.$SYSTEM.CATET1 EXTENT(13,2 ),
        \b.$Db02.sqlet1.tellers    CATALOG \b.$SYSTEM.CATET1 EXTENT(13,2 ),
        \b.$Db04.sqlet1.tellers    CATALOG \b.$SYSTEM.CATET1 EXTENT(13,2 ),
        \b.$Db06.sqlet1.tellers    CATALOG \b.$SYSTEM.CATET1 EXTENT(13,2 ),
        \b.$Db08.sqlet1.tellers    CATALOG \b.$SYSTEM.CATET1 EXTENT(13,2 ),
        \b.$Db10.sqlet1.tellers    CATALOG \b.$SYSTEM.CATET1 EXTENT(13,2 ),
        \b.$Db12.sqlet1.tellers    CATALOG \b.$SYSTEM.CATET1 EXTENT(13,2 ),
        \b.$db14.sqlet1.tellers    CATALOG \b.$SYSTEM.CATET1 EXTENT(14,2 ),
```

```
        \c.$system.sqlet1.tellers  CATALOG \c.$SYSTEM.CATET1 EXTENT(13,2 ),
        \c.$Dc02.sqlet1.tellers    CATALOG \c.$SYSTEM.CATET1 EXTENT(13,2 ),
        \c.$Dc04.sqlet1.tellers    CATALOG \c.$SYSTEM.CATET1 EXTENT(13,2 ),
        \c.$Dc06.sqlet1.tellers    CATALOG \c.$SYSTEM.CATET1 EXTENT(13,2 ),
        \c.$Dc08.sqlet1.tellers    CATALOG \c.$SYSTEM.CATET1 EXTENT(13,2 ),
        \c.$Dc10.sqlet1.tellers    CATALOG \c.$SYSTEM.CATET1 EXTENT(13,2 ),
        \c.$Dc12.sqlet1.tellers    CATALOG \c.$SYSTEM.CATET1 EXTENT(13,2 ),
        \c.$dc14.sqlet1.tellers    CATALOG \c.$SYSTEM.CATET1 EXTENT(14,2 ),

        \d.$system.sqlet1.tellers  CATALOG \d.$SYSTEM.CATET1 EXTENT(13,2 ),
        \d.$Dd02.sqlet1.tellers    CATALOG \d.$SYSTEM.CATET1 EXTENT(13,2 ),
        \d.$Dd04.sqlet1.tellers    CATALOG \d.$SYSTEM.CATET1 EXTENT(13,2 ),
        \d.$Dd06.sqlet1.tellers    CATALOG \d.$SYSTEM.CATET1 EXTENT(13,2 ),
        \d.$Dd08.sqlet1.tellers    CATALOG \d.$SYSTEM.CATET1 EXTENT(13,2 ),
        \d.$Dd10.sqlet1.tellers    CATALOG \d.$SYSTEM.CATET1 EXTENT(13,2 ),
        \d.$Dd12.sqlet1.tellers    CATALOG \d.$SYSTEM.CATET1 EXTENT(13,2 ),
        \d.$dd14.sqlet1.tellers    CATALOG \d.$SYSTEM.CATET1 EXTENT(14,2 ),

        \e.$system.sqlet1.tellers  CATALOG \e.$SYSTEM.CATET1 EXTENT(14,2 )
      );

?section create_history
!-------------------------------------------------------------------!
! create history file  -  record length 50                          !
!-------------------------------------------------------------------!
!                                                                   !
!  field:  len: function:                                           !
!  ------  ---- ---------                                           !
!    1      12  account number                                     !
!    2      12  teller number                                      !
!    3      12  amount tendered                                    !
!    4       6  date                                               !
!    5       8  time                                               !
!                                                                   !
!  partitions:                                                      !
!  -----------                                                      !
!                                                                   !
!    no secondary partitions.                                       !
!                                                                   !
!-------------------------------------------------------------------!
CREATE TABLE \A.$B41VLX.SQLET1.HISTORY  (
       history_account_number   PICTURE x(12),
       history_teller_number    PICTURE x(12),
       history_amount           PICTURE s9(12),
       history_yymmdd           PICTURE x(6),
       history_hhmmsscc         PICTURE x(8)
       )
       ORGANIZATION Entry sequenced
       AUDIT
       CATALOG \A.$SYSTEM.CATET1
       BLOCKSIZE 4096
       EXTENT (1000,1000) ;

CREATE TABLE \b.$db01.SQLET1.HISTORY  (
       history_account_number   PICTURE x(12),
       history_teller_number    PICTURE x(12),
       history_amount           PICTURE s9(12),
       history_yymmdd           PICTURE x(6),
       history_hhmmsscc         PICTURE x(8)
       )
```

```
        ORGANIZATION Entry sequenced
        AUDIT
        CATALOG \b.$SYSTEM.CATET1
        BLOCKSIZE 4096
        EXTENT (1000,1000) ;

CREATE TABLE \c.$dc01.SQLET1.HISTORY  (
        history_account_number    PICTURE x(12),
        history_teller_number     PICTURE x(12),
        history_amount            PICTURE s9(12),
        history_yymmdd            PICTURE x(6),
        history_hhmmsscc          PICTURE x(8)
        )
        ORGANIZATION Entry sequenced
        AUDIT
        CATALOG \c.$SYSTEM.CATET1
        BLOCKSIZE 4096
        EXTENT (1000,1000) ;

CREATE TABLE \d.$dd01.SQLET1.HISTORY  (
        history_account_number    PICTURE x(12),
        history_teller_number     PICTURE x(12),
        history_amount            PICTURE s9(12),
        history_yymmdd            PICTURE x(6),
        history_hhmmsscc          PICTURE x(8)
        )
        ORGANIZATION Entry sequenced
        AUDIT
        CATALOG \d.$SYSTEM.CATET1
        BLOCKSIZE 4096
        EXTENT (1000,1000) ;

CREATE TABLE \e.$system.SQLET1.HISTORY  (
        history_account_number    PICTURE x(12),
        history_teller_number     PICTURE x(12),
        history_amount            PICTURE s9(12),
        history_yymmdd            PICTURE x(6),
        history_hhmmsscc          PICTURE x(8)
        )
        ORGANIZATION Entry sequenced
        AUDIT
        CATALOG \e.$SYSTEM.CATET1
        BLOCKSIZE 4096
        EXTENT (500,500) ;
```

```
< Top command file for creating ENSCRIBE database for the TPC/DJN benchmark

< accounts file
allow 100 errors
set type k
set ext ( 2600 pages, 2600 pages )
set maxextents 99
set rec 100
set block  4096
set iblock 4096
set keylen 12
set keyoff 0
set part ( 1,  \a.$da03,   2600, 2600, "000000800000" )
set part ( 2,  \a.$da05,   2600, 2600, "000001600000" )
set part ( 3,  \a.$da07,   2600, 2600, "000002400000" )
set part ( 4,  \a.$da09,   2600, 2600, "000003200000" )
set part ( 5,  \a.$da11,   2600, 2600, "000004000000" )
set part ( 6,  \a.$da13,   2600, 2600, "000004800000" )
set part ( 7,  \a.$da15,   2600, 2600, "000005600000" )
set part ( 8,  \b.$db01,   2600, 2600, "000006400000" )
set part ( 9,  \b.$db03,   2600, 2600, "000007200000" )
set part (10,  \b.$db05,   2600, 2600, "000008000000" )
set part (11,  \b.$db07,   2600, 2600, "000008800000" )
set part (12,  \b.$db09,   2600, 2600, "000009600000" )
set part (13,  \b.$db11,   2600, 2600, "000010400000" )
set part (14,  \b.$db13,   2600, 2600, "000011200000" )
set part (15,  \b.$db15,   2600, 2600, "000012000000" )

create $b41vlx.accounts

< history

reset
set type e
set rec 100
set block 4096
set ext (1000, 1000)
create \a.$system.history
create \b.$system.history

< tellers
reset
set type r
set block 2048
set rec 100
set ext (13,2 )
set part ( 1, \a.$da03,13,2 )
set part ( 2, \a.$da05,13,2 )
set part ( 3, \a.$da07,13,2 )
set part ( 4, \a.$da09,13,2 )
set part ( 5, \a.$da11,13,2 )
set part ( 6, \a.$da13,13,2 )
set part ( 7, \a.$da15,14,2 )
set part ( 8, \b.$db01,13,2 )
set part ( 9, \b.$db03,13,2 )
set part (10, \b.$db05,13,2 )
set part (11, \b.$db07,13,2 )
set part (12, \b.$db09,13,2 )
set part (13, \b.$db11,13,2 )
set part (14, \b.$db13,13,2 )
```

```
set part (15, \b.$db15,14,2 )

create $b41vlx.tellers

< branches
reset
set type r
set block 512
set rec 100
set ext ( 21,2 )
set part ( 1, \a.$da03,20,2 )
set part ( 2, \a.$da05,20,2 )
set part ( 3, \a.$da07,20,2 )
set part ( 4, \a.$da09,21,2 )
set part ( 5, \a.$da11,20,2 )
set part ( 6, \a.$da13,20,2 )
set part ( 7, \a.$da15,20,2 )
set part ( 8, \b.$db01,21,2 )
set part ( 9, \b.$db03,20,2 )
set part (10, \b.$db05,20,2 )
set part (11, \b.$db07,20,2 )
set part (12, \b.$db09,21,2 )
set part (13, \b.$db11,20,2 )
set part (14, \b.$db13,20,2 )
set part (15, \b.$db15,20,2 )

create $b41vlx.branches
```

```
?list
?SAVE PARAM
?HEADING "COBSQL for DebitCredit Transaction (ET1)"
?SYMBOLS
?INSPECT

*****************************************************************************
*  TOPGUN Server                                                            *
*****************************************************************************
*                                                                           *
*  This server is designed to be run in a PATHWAY environment to test the   *
*  performance of Tandem's SQL in the ET1 debit/credit transaction.         *
*                                                                           *
*****************************************************************************

*****************************************************************************
*  General flow:                                                            *
*****************************************************************************
*                                                                           *
*    BEGINTRANSACTION                                                       *
*      ACCEPT 100 bytes from Requestor.                                     *
*      UPDATE Accounts File (100 bytes)                                     *
*      UPDATE Tellers  File (100 bytes)                                     *
*      UPDATE Branches File (100 bytes)                                     *
*      WRITE  History  File (50 bytes)                                      *
*      REPLY  102 bytes to Requestor.                                       *
*    ENDTRANSACTION                                                         *
*                                                                           *
*****************************************************************************

*****************************************************************************
*  Data Descriptions                                                        *
*****************************************************************************
*                                                                           *
*    Message from Requestor:                                                *
*                                                                           *
*       Filler                1 Byte                                       *
*       Account Number       12 Bytes                                      *
*       Filler                1 Byte                                       *
*       Teller Number        12 Bytes                                      *
*       Filler                1 Byte                                       *
*       Branch Number        12 Bytes                                      *
*       Filler                1 Byte                                       *
*       Amount Tendered      12 Bytes                                      *
*                                                                           *
*    Reply to Requestor:                                                    *
*                                                                           *
*       Reply Code            2 Bytes  (Non-zero if error else zero.)      *
*       Filler              100 Bytes                                       *
*                                                                           *
*    (For data descriptions of files, please see the appropriate SQLCI      *
*    command file(s).)                                                      *
*                                                                           *
*****************************************************************************

*****************************************************************************
* Change history:                                                           *
*****************************************************************************
*                                                                           *
*   tlw fill4 and delta added to message-in-record.                         *
```

edit file

```
*       this value will be used to keep track of remote                        *
*       vs. local transactions.                                                *
*                                                                              *
*   tlw 02 modify data formats to agree with sql, et al.                       *
*          ALL numeric fields changed to pic 9(x); i.e.,                       *
*          there are NO more computational fields.                             *
*          (the reply code to the requestor remains comp, however.)            *
*                                                                              *
*   tlw 02 delete blank substitution (with "0")                                *
*          (script generator was modified for this.)                           *
*                                                                              *
*   tlw 02 commented out leading "1" in account number.                        *
*          (database loading routines must be modified.)                       *
*                                                                              *
*   tlw 02 added WHENEVER NOT FOUND ...                                        *
*                                                                              *
*   tlw 02 deleted branch number and filler from write to                      *
*          history file.  (these fields are now all pic 9;                     *
*          a history file record thus contains fifty bytes                     *
*          --account and teller numbers, amount tendered,                      *
*          and date/time group)                                                *
*                                                                              *
*   tlw 02 commented out call to DEBUG  (INSPECT does not                      *
*          recognize the "s" command issued by $HOME.)                         *
*                                                                              *
*   tlw 02 DTG changed to pic 9.                                               *
*                                                                              *
*   tlw 03 all data types, with the exception of "balance" fields, changed     *
*          to pic x.                                                           *
*                                                                              *
*   tlw 03 added teller-key-hv and branch-key-hv (both pic 9(12) comp) in      *
*          order to be compatible with search by SYSKEY (32-bit value)         *
*                                                                              *
*   tlw 04 corrections found by Tom Sawyer:                                    *
*          . loop counter initialization                                       *
*          . pic x changed to pic 9 for branch and teller keys                 *
*          . WHERE clause changed to test for account balance limit            *
*                                                                              *
*   tlw 04 changed amount-tendered to signed                                   *
*                                                                              *
*   tlw 04 changed syskey comparison to teller-key-hv and branch-key-hv        *
*          instead of teller-number-hv and branch-number-hv                    *
*                                                                              *
********************************************************************************

 IDENTIFICATION DIVISION.
   PROGRAM-ID.    ET1-COBSQL-server.
   Author.        Version 4.
   DATE-WRITTEN.
   SECURITY.
 ENVIRONMENT DIVISION.
   CONFIGURATION SECTION.
      SOURCE-COMPUTER.  TANDEM/16.
      OBJECT-COMPUTER.  TANDEM/16.

   INPUT-OUTPUT SECTION.
        FILE-CONTROL.
          SELECT message-in
            ASSIGN TO $RECEIVE
            ORGANIZATION IS SEQUENTIAL
```

```
           ACCESS MODE IS SEQUENTIAL
           FILE STATUS IS file-stat.

         SELECT message-out
           ASSIGN TO $RECEIVE
           ORGANIZATION IS SEQUENTIAL
           ACCESS MODE IS SEQUENTIAL
           FILE STATUS IS file-stat.

       RECEIVE-CONTROL.
           TABLE OCCURS 2 TIMES
           SYNCDEPTH LIMIT IS 2
           REPLY CONTAINS message-out RECORD.

DATA DIVISION.

    FILE SECTION.

     FD message-in
        LABEL RECORDS ARE OMITTED.
     01 message-in-record.
        02 filler-1              PIC X.
        02 account-number        PIC X(12).
        02 filler-2              PIC X.
        02 teller-number         PIC 9(12).
        02 filler-3              PIC X.
        02 branch-number         PIC 9(12).
        02 filler-4              PIC X.
        02 amount-tendered       PIC s9(12).

     FD message-out
        LABEL RECORDS ARE OMITTED
        RECORD CONTAINS 102 CHARACTERS.
     01 message-out-record.
        02 reply-header          PIC 9(4) COMP.
        02 out-buffer            PIC X(100).

  WORKING-STORAGE SECTION.

  01 loop-record              PIC 9.
        88 not-all-done          VALUE 0.
        88 all-done              VALUE 1.

  01 file-stat                  PIC XX.

*************************************************************************
*   Working storage section for the ET1 SQL code - HOST VARIABLES       *
*************************************************************************

    EXEC SQL BEGIN DECLARE SECTION  END-EXEC.

      01 account-number-hv        PIC X(12).
      01 teller-number-hv         PIC X(12).
      01 teller-key-hv            PIC 9(12) comp.
      01 branch-number-hv         PIC X(12).
      01 branch-key-hv            PIC 9(12) comp.
      01 amount-tendered-hv       PIC S9(12).
      01 yymmdd-hv                PIC X(6).
      01 hhmmsscc-hv              PIC X(8).
```

```
   EXEC SQL END DECLARE SECTION END-EXEC.

/
*****************************************************************
*  Main routine                                                 *
*****************************************************************
 PROCEDURE DIVISION.

 MAIN SECTION.

 BEGIN-ET1-COBOL-server.

*****************************************************************
*  Initialize error routines                                    *
*****************************************************************
    EXEC SQL WHENEVER SQLERROR    PERFORM :SHOWIT END-EXEC.
    EXEC SQL WHENEVER SQLWARNING  PERFORM :SHOWIT END-EXEC.
    EXEC SQL WHENEVER NOT FOUND   PERFORM :SHOWIT END-EXEC.

*****************************************************************
*  Enter main loop                                              *
*****************************************************************
    PERFORM Init-files.
    PERFORM do-transaction UNTIL all-done.
    PERFORM shut-files.
    STOP    RUN.
/
*****************************************************************
*  Initialization, open the server $RECEIVE                     *
*****************************************************************
 Init-files.
    OPEN INPUT message-in   SYNCDEPTH 1.
    OPEN OUTPUT message-out.
    MOVE 0 TO reply-header.
    MOVE 0 TO loop-record.
/
*****************************************************************
*  Shut down - close $RECEIVE                                   *
*****************************************************************
 Shut-files.
   CLOSE message-in.
   CLOSE message-out.
/
 do-transaction.
*****************************************************************
*   ACCEPT message from Requestor                               *
*****************************************************************

    MOVE 0 TO reply-header.
    READ message-in.
    IF   file-stat NOT EQUAL zero MOVE 1 TO reply-header.

*****************************************************************
*   Get current timestamp for history file                      *
*****************************************************************
    ACCEPT yymmdd-hv   FROM DATE.
    ACCEPT hhmmsscc-hv FROM TIME.

*****************************************************************
*   Move the numbers to SQL host variables.                     *
```

4

```
*********************************************************************
    MOVE account-number    TO account-number-hv
    MOVE teller-number     TO teller-number-hv.
    MOVE teller-number     TO teller-key-hv.
    MOVE branch-number     TO branch-number-hv.
    MOVE branch-number     TO branch-key-hv.
    MOVE amount-tendered   TO amount-tendered-hv.

*********************************************************************
*  UPDATE Account File                                              *
*********************************************************************
    EXEC SQL UPDATE  =accounts
             SET     account_balance =  account_balance + :amount-tendered-hv
             WHERE   account_number  =  :account-number-hv
             AND     (account_balance + :amount-tendered-hv) >= 0
    END-EXEC.

*********************************************************************
*  UPDATE Teller File                                               *
*********************************************************************
    EXEC SQL UPDATE =tellers
             SET    teller_balance = teller_balance  + :amount-tendered-hv
             WHERE  syskey         = :teller-key-hv
    END-EXEC.

*********************************************************************
*  UPDATE Branch File                                               *
*********************************************************************
    EXEC SQL UPDATE =branches
             SET    branch_balance = branch_balance + :amount-tendered-hv
             WHERE  syskey         = :branch-key-hv
    END-EXEC.

*********************************************************************
*  WRITE History File                                               *
*********************************************************************
    EXEC SQL INSERT INTO  =HISTORY
                      VALUES ( :account-number-hv
                             , :teller-number-hv
                             , :amount-tendered-hv
                             , :yymmdd-hv
                             , :hhmmsscc-hv
                             )
    END-EXEC.

*********************************************************************
*  REPLY to Requestor                                               *
*********************************************************************
    WRITE message-out-record   END-WRITE.

*********************************************************************
* Error Handling Routine                                            *
*********************************************************************
  SHOWIT.
    ENTER TAL SQLCADISPLAY USING SQLCA, -1.
*   ENTER TAL DEBUG.
    MOVE 1 TO reply-header.
```

ANSI 1985 COBOL - T9257B40 - (30 OCT 86) Source language: COBOL-1985 Target machine: Tandem NonStop System
Compiled: 87/02/18 - 18:57:13 Default options: On (LIST,LMAP,SHOWCOPY,WARN) Off (CODE,ICODE,MAP)

cobol listing

```
1       ?list
2       ?SAVE PARAM
3       ?HEADING "COBSQL for DebitCredit Transaction (ET1)"
4       ?SYMBOLS
5       ?INSPECT
6
7       ******************************************************************************
8       *  TOPGUN Server                                                             *
9       ******************************************************************************
10      *                                                                            *
11      *  This server is designed to be run in a PATHWAY environment to test the    *
12      *  performance of Tandem's SQL in the ET1 debit/credit transaction.          *
13      *                                                                            *
14      ******************************************************************************
15
16      ******************************************************************************
17      *  General flow:                                                             *
18      ******************************************************************************
19      *                                                                            *
20      *    BEGINTRANSACTION                                                        *
21      *      ACCEPT 100 bytes from Requestor.                                      *
22      *      UPDATE Accounts File (100 bytes)                                      *
23      *      UPDATE Tellers  File (100 bytes)                                      *
24      *      UPDATE Branches File (100 bytes)                                      *
25      *      WRITE  History  File (50 bytes)                                       *
26      *      REPLY  102 bytes to Requestor.                                        *
27      *    ENDTRANSACTION                                                          *
28      *                                                                            *
29      ******************************************************************************
30
31      ******************************************************************************
32      *  Data Descriptions                                                         *
33      ******************************************************************************
34      *                                                                            *
35      *    Message from Requestor:                                                 *
36      *                                                                            *
37      *       Filler               1 Byte                                         *
38      *       Account Number      12 Bytes                                        *
39      *       Filler               1 Byte                                         *
40      *       Teller Number       12 Bytes                                        *
41      *       Filler               1 Byte                                         *
42      *       Branch Number       12 Bytes                                        *
43      *       Filler               1 Byte                                         *
44      *       Amount Tendered     12 Bytes                                        *
45      *                                                                            *
46      *    Reply to Requestor:                                                     *
47      *                                                                            *
48      *       Reply Code           2 Bytes  (Non-zero if error else zero.)        *
49      *       Filler             100 Bytes                                        *
50      *                                                                            *
51      *    (For data descriptions of files, please see the appropriate SQLCI       *
52      *    command file(s).)                                                       *
53      *                                                                            *
```

9

```
      ******************************************************************************
      ******************************************************************************
      * Change history:                                                            *
      ******************************************************************************
      *                                                                            *
      *   tlw fill4 and delta added to message-in-record.                          *
      *       this value will be used to keep track of remote                      *
      *       vs. local transactions.                                              *
      *                                                                            *
      *   tlw 02 modify data formats to agree with sql, et al.                     *
      *          ALL numeric fields changed to pic 9(x); i.e.,                     *
      *          there are NO more computational fields.                           *
      *          (the reply code to the requestor remains comp, however.)          *
      *                                                                            *
      *   tlw 02 delete blank substitution (with "0")                              *
      *          (script generator was modified for this.)                         *
      *                                                                            *
      *   tlw 02 commented out leading "1" in account number.                      *
      *          (database loading routines must be modified.)                     *
      *                                                                            *
      *   tlw 02 added WHENEVER NOT FOUND ...                                      *
      *                                                                            *
      *   tlw 02 deleted branch number and filler from write to                    *
      *          history file.  (these fields are now all pic 9;                   *
      *          a history file record thus contains fifty bytes                   *
      *          --account and teller numbers, amount tendered,                    *
      *          and date/time group)                                              *
      *                                                                            *
      *   tlw 02 commented out call to DEBUG  (INSPECT does not                    *
      *          recognize the "s" command issued by $HOME.)                       *
      *                                                                            *
      *   tlw 02 DTG changed to pic 9.                                             *
      *                                                                            *
      *   tlw 03 all data types, with the exception of "balance" fields, changed   *
      *          to pic x.                                                         *
      *                                                                            *
      *   tlw 03 added teller-key-hv and branch-key-hv (both pic 9(12) comp) in    *
      *          order to be compatible with search by SYSKEY (32-bit value)       *
      *                                                                            *
      *   tlw 04 corrections found by Tom Sawyer:                                  *
      *          . loop counter initialization                                     *
      *          . pic x changed to pic 9 for branch and teller keys               *
      *          . WHERE clause changed to test for account balance limit          *
      *                                                                            *
      *   tlw 04 changed amount-tendered to signed                                 *
      *                                                                            *
      *   tlw 04 changed syskey comparison to teller-key-hv and branch-key-hv      *
      *          instead of teller-number-hv and branch-number-hv                  *
      *                                                                            *
      ******************************************************************************

       IDENTIFICATION DIVISION.
         PROGRAM-ID.   ET1-COBSQL-server.
         Author.       Version 4.
         DATE-WRITTEN.
         SECURITY.
```

```
111         ENVIRONMENT DIVISION.
112           CONFIGURATION SECTION.
113              SOURCE-COMPUTER.  TANDEM/16.
114              OBJECT-COMPUTER.  TANDEM/16.
115
116           INPUT-OUTPUT SECTION.
117                FILE-CONTROL.
118                  SELECT message-in
119                    ASSIGN TO $RECEIVE
120                    ORGANIZATION IS SEQUENTIAL
121                    ACCESS MODE IS SEQUENTIAL
122                    FILE STATUS IS file-stat.
123
124                  SELECT message-out
125                    ASSIGN TO $RECEIVE
126                    ORGANIZATION IS SEQUENTIAL
127                    ACCESS MODE IS SEQUENTIAL
128                    FILE STATUS IS file-stat.
129
130                RECEIVE-CONTROL.
131                    TABLE OCCURS 2 TIMES
132                    SYNCDEPTH LIMIT IS 2
133                    REPLY CONTAINS message-out RECORD.
134
135         DATA DIVISION.
136
137             FILE SECTION.
138
139              FD message-in
140                 LABEL RECORDS ARE OMITTED.
141              01 message-in-record.
142                 02 filler-1             PIC X.
143                 02 account-number       PIC X(12).
144                 02 filler-2             PIC X.
145                 02 teller-number        PIC 9(12).
146                 02 filler-3             PIC X.
147                 02 branch-number        PIC 9(12).
148                 02 filler-4             PIC X.
149                 02 amount-tendered      PIC s9(12).
150
151              FD message-out
152                 LABEL RECORDS ARE OMITTED
153                 RECORD CONTAINS 102 CHARACTERS.
154              01 message-out-record.
155                 02 reply-header         PIC 9(4) COMP.
156                 02 out-buffer           PIC X(100).
157
158           WORKING-STORAGE SECTION.
158.001     ?NOLIST
159
160           01 loop-record              PIC 9.
161                 88 not-all-done          VALUE 0.
162                 88 all-done              VALUE 1.
163
164           01 file-stat                PIC XX.
165
166         ********************************************************************
```

```
167       *   Working storage section for the ET1 SQL code - HOST VARIABLES        *
168       ******************************************************************************
169
170       *   EXEC SQL BEGIN DECLARE SECTION  END-EXEC.
171
172             01 account-number-hv        PIC X(12).
173             01 teller-number-hv         PIC X(12).
174             01 teller-key-hv            PIC 9(12) comp.
175             01 branch-number-hv         PIC X(12).
176             01 branch-key-hv            PIC 9(12) comp.
177             01 amount-tendered-hv       PIC S9(12).
178             01 yymmdd-hv                PIC X(6).
179             01 hhmmsscc-hv              PIC X(8).
180
181       *  EXEC SQL END DECLARE SECTION END-EXEC.
182
```

```
183       /
184       ******************************************************************
185       *  Main routine                                                  *
186       ******************************************************************
187        PROCEDURE DIVISION.
187.001               SQL-INIT-PAR.
187.002               PERFORM SQL-INIT.
188
189        MAIN SECTION.
190
191        BEGIN-ET1-COBOL-server.
192
193       ******************************************************************
194       *  Initialize error routines                                     *
195       ******************************************************************
196       *    EXEC SQL WHENEVER SQLERROR    PERFORM :SHOWIT END-EXEC.
197       *    EXEC SQL WHENEVER SQLWARNING  PERFORM :SHOWIT END-EXEC.
198       *    EXEC SQL WHENEVER NOT FOUND   PERFORM :SHOWIT END-EXEC.
199
200       ******************************************************************
201       *  Enter main loop                                               *
202       ******************************************************************
203           PERFORM Init-files.
204           PERFORM do-transaction UNTIL all-done.
205           PERFORM shut-files.
206           STOP    RUN.
```

```
207        /
208        ******************************************************************
209        *  Initialization, open the server $RECEIVE                      *
210        ******************************************************************
211         Init-files.
212            OPEN INPUT message-in   SYNCDEPTH 1.
213            OPEN OUTPUT message-out.
214            MOVE 0 TO reply-header.
215            MOVE 0 TO loop-record.
```

```
216       /
217       ******************************************************************
218       *  Shut down - close $RECEIVE                                    *
219       ******************************************************************
220        Shut-files.
221          CLOSE message-in.
222          CLOSE message-out.
```

```
223        /
224         do-transaction.
225        ******************************************************************
226        *   ACCEPT message from Requestor                                *
227        ******************************************************************
228
229            MOVE 0 TO reply-header.
230            READ message-in.
231            IF   file-stat NOT EQUAL zero MOVE 1 TO reply-header.
232
233        ******************************************************************
234        *   Get current timestamp for history file                       *
235        ******************************************************************
236            ACCEPT yymmdd-hv   FROM DATE.
237            ACCEPT hhmmsscc-hv FROM TIME.
238
239        ******************************************************************
240        *   Move the numbers to SQL host variables.                      *
241        ******************************************************************
242            MOVE account-number    TO account-number-hv
243            MOVE teller-number     TO teller-number-hv.
244            MOVE teller-number     TO teller-key-hv.
245            MOVE branch-number     TO branch-number-hv.
246            MOVE branch-number     TO branch-key-hv.
247            MOVE amount-tendered   TO amount-tendered-hv.
248
249        ******************************************************************
250        *  UPDATE Account File                                           *
251        ******************************************************************
252        *   EXEC SQL UPDATE  =accounts
253        *            SET     account_balance =  account_balance + :amount-tendered-hv
254        *            WHERE   account_number  =  :account-number-hv
255        *            AND     (account_balance + :amount-tendered-hv) >= 0
256        *   END-EXEC.
256.001             PERFORM SQLDO-0.
257
258        ******************************************************************
259        *  UPDATE Teller File                                            *
260        ******************************************************************
261        *   EXEC SQL UPDATE =tellers
262        *            SET    teller_balance = teller_balance  + :amount-tendered-hv
263        *            WHERE  syskey         = :teller-key-hv
264        *   END-EXEC.
264.001             PERFORM SQLDO-1.
265
266        ******************************************************************
267        *  UPDATE Branch File                                            *
268        ******************************************************************
269        *   EXEC SQL UPDATE =branches
270        *            SET    branch_balance = branch_balance + :amount-tendered-hv
271        *            WHERE  syskey         = :branch-key-hv
272        *   END-EXEC.
272.001             PERFORM SQLDO-2.
273
274        ******************************************************************
275        *  WRITE History File                                            *
276        ******************************************************************
```

```
277       *   EXEC SQL INSERT INTO  =HISTORY
278       *                    VALUES ( :account-number-hv
279       *                           , :teller-number-hv
280       *                           , :amount-tendered-hv
281       *                           , :yymmdd-hv
282       *                           , :hhmmsscc-hv
283       *                           )
284       *   END-EXEC.
284.001             PERFORM SQLDO-3.
285
286       ******************************************************************
287       *  REPLY to Requestor                                            *
288       ******************************************************************
289           WRITE message-out-record   END-WRITE.
290
291       ******************************************************************
292       * Error Handling Routine                                         *
293       ******************************************************************
294         SHOWIT.
295           ENTER TAL SQLCADISPLAY USING SQLCA, -1.
296       *   ENTER TAL DEBUG.
297           MOVE 1 TO reply-header.
297.001   ?NOLIST
297.082           SQLDO-0 SECTION.
297.083               ENTER TAL EXECSQL USING SQLIN0 , SQLCA.
297.084               IF (SQLCODE OF SQLCA =  100 )
297.085               PERFORM  SHOWIT  .
297.086               IF (SQLCODE OF SQLCA < 0)
297.087               PERFORM  SHOWIT  .
297.088               IF (SQLCODE OF SQLCA > 0) AND
297.089               (SQLCODE OF SQLCA NOT EQUAL 100 )
297.09                PERFORM  SHOWIT  .
297.091           SQLDO-1 SECTION.
297.092               ENTER TAL EXECSQL USING SQLIN1 , SQLCA.
297.093               IF (SQLCODE OF SQLCA =  100 )
297.094               PERFORM  SHOWIT  .
297.095               IF (SQLCODE OF SQLCA < 0)
297.096               PERFORM  SHOWIT  .
297.097               IF (SQLCODE OF SQLCA > 0) AND
297.098               (SQLCODE OF SQLCA NOT EQUAL 100 )
297.099               PERFORM  SHOWIT  .
297.1             SQLDO-2 SECTION.
297.101               ENTER TAL EXECSQL USING SQLIN2 , SQLCA.
297.102               IF (SQLCODE OF SQLCA =  100 )
297.103               PERFORM  SHOWIT  .
297.104               IF (SQLCODE OF SQLCA < 0)
297.105               PERFORM  SHOWIT  .
297.106               IF (SQLCODE OF SQLCA > 0) AND
297.107               (SQLCODE OF SQLCA NOT EQUAL 100 )
297.108               PERFORM  SHOWIT  .
297.109           SQLDO-3 SECTION.
297.11                ENTER TAL EXECSQL USING SQLIN3 , SQLCA.
297.111               IF (SQLCODE OF SQLCA =  100 )
297.112               PERFORM  SHOWIT  .
297.113               IF (SQLCODE OF SQLCA < 0)
297.114               PERFORM  SHOWIT  .
297.115               IF (SQLCODE OF SQLCA > 0) AND
```

```
297.116              (SQLCODE OF SQLCA NOT EQUAL 100 )
297.117              PERFORM  SHOWIT  .
298.117  ?TANDEM
299.117  ****************************************************************
299.118  *BINDER - OBJECT FILE BINDER - T9621B41 - (05JAN87)   SYSTEM \TESS
299.119  *Object file name is $SAUR.JGRAY.MCOBSQL
299.12   *Object file timestamp is 18FEB87 18:43:03
299.121  *Number of Binder errors = 0
299.122  *Number of Binder warnings = 0
299.123  *Code area size = 0 pages
299.124  *Resident code size = 0 pages
299.125  *Data area size = 1 pages
299.126  *Extended data area size = 0 pages
299.127  *Top of stack = 0 words
299.128  *Number of code segments  = 1 segment
299.129  ?SEARCH $SAUR.JGRAY.MCOBSQL
299.13   *SQL ****************************************************
299.131  * SQL - COBOL SQL PREPROCESSOR - T9192B41
299.132  *    SQL -    input  source =  $SAUR.JGRAY.COBSQL
299.133  *    SQL -    output source =  $SAUR.JGRAY.CCOBSQL
299.134  *    SQL -    object file   =  $SAUR.JGRAY.MCOBSQL
299.135  *    SQL -    date          = 1987/02/18 - 18:43:28
299.136  *SQL ****************************************************
299.137  * SQL - SUMMARY OF SQL PREPROCESSING
299.138  *    SQL - number of sql statements = 4
299.139  *    SQL - number of errors         = 0
299.14   *    SQL - number of warnings       = 0
299.141  *    SQL - preprocess cpu time      = 00:00:02
299.142  *    SQL - total elapse time        = 00:00:26
299.143  *SQL ****************************************************
```

ENTRY POINT MAP BY NAME

SP	PEP	BASE	LIMIT	ENTRY	ATTRS	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	002	000003	002046	000441	M	ET1-COBSQL-SERVER	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL

DATA BLOCK MAP BY NAME

BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
000020	000052	SPECIAL	WORD	#ERUG	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL
000000	000017	SPECIAL	WORD	#GO	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL
100341	104301	SPECIAL	WORD	#HIGHBUF	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL
100341		SPECIAL	WORD	#RQTB	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL
100000	100114	SPECIAL	WORD	#RUCB	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL
000053	001065	OWN	STRING	ET1-COBSQL-SERVER	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL
100115	100340	SPECIAL	WORD	ET1-COBSQL-SERVER#	18FEB87	18:57	COBOL85	$SAUR.JGRAY.CCOBSQL

RUN TIME DATA UNIT MAP BY NAME

TYPE	LENGTH	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
SQL_SRC	001250	ET1-COBSQL-SERVER	18FEB87	18:43	RTDU	$SAUR.JGRAY.COBSQL

```
BINDER - OBJECT FILE BINDER - T9621B41 - (05JAN87)   SYSTEM \TESS
Object file name is $SAUR.JGRAY.RUNUNIT
Object file timestamp is 18FEB87 18:57:13
Number of Binder errors = 0
Number of Binder warnings = 0
Primary data = 16 words
Secondary data = 550 words
Code area size = 2 pages
Resident code size = 0 pages
Data area size = 35 pages
Extended data area size = 0 pages
Top of stack = 566 words
Number of code segments  = 1 segment


COBOL85 - T9257B40 - (30 OCT 86)
The object file is executable on a Tandem NonStop System
Number of compiler errors = 0
Number of compiler warnings = 0
Maximum symbol table size = 17034 bytes
Elapsed time - 00:01:09
```

Revision Level: E00XM

Interface: Parallel/ DP

Line Ending: CR,LF

Data Encoding: ASCII/7

Fonts Available:

XCP14-L	8958 Bytes	Rev.	8
Titan10iso-P	22092 Bytes	Rev.	6
20149A-P	12536 Bytes	Rev.	1
20149B-P	12600 Bytes	Rev.	1
20149C-P	12700 Bytes	Rev.	1
Spokesman10-P	15930 Bytes	Rev.	5

Job Status:

Page	3:22
Page	4:22
Page	5:22
Page	6:22
Page	7:22
Page	8:22
Page	9:22
Page	10:22
Page	11:22
Page	15:26

ANSI 1985 COBOL - T9257C00 - (15 JUL 87) Source language: COBOL-1985 Target machine: Tandem NonStop System
Compiled: 87/02/18 - 18:40:10 Default options: On (LIST,LMAP,SHOWCOPY,WARN) Off (CODE,ICODE,MAP)

```
1       ?HEADING "COBSQL for DebitCredit Transaction (ET1)"
2       ?SYMBOLS
3       ?INSPECT
4
5        IDENTIFICATION DIVISION.
6
7          PROGRAM-ID.  ET1-COBOL-server.
8          Author.  Praful  Shah
9
10         DATE-WRITTEN. December 1986.
11
12         SECURITY.
13
14                This server is designed to measure the performance of
15                the ENSCRIBE database.  It runs under the PATHWAY
16                environment.  The requestor does the BEGIN/END
17                transaction.
18
19                This server accepts a message from the the REQUESTOR,
20                time stamps the transaction.  It the updates the
21                account, branch and teller records. At the end
22                of the transaction it writes a memo record to the
23                history file.
24
25       ENVIRONMENT DIVISION.
26         CONFIGURATION SECTION.
27            SOURCE-COMPUTER.  TANDEM/16.
28            OBJECT-COMPUTER.  TANDEM/16.
29
30         INPUT-OUTPUT SECTION.
31              FILE-CONTROL.
32                SELECT message-in
33                  ASSIGN TO $RECEIVE
34                  ORGANIZATION IS SEQUENTIAL
35                  ACCESS MODE IS SEQUENTIAL
36                  FILE STATUS IS file-stat.
37
38                SELECT message-out
39                  ASSIGN TO $RECEIVE
40                  ORGANIZATION IS SEQUENTIAL
41                  ACCESS MODE IS SEQUENTIAL
42                  FILE STATUS IS file-stat.
43
44                SELECT account
45                  ASSIGN TO account
46                  ORGANIZATION IS INDEXED
47                  ACCESS  MODE IS RANDOM
48                  RECORD KEY IS account-number of account-record
49                  FILE STATUS IS file-stat.
50
51                SELECT branch
52                  ASSIGN TO branch
53                  ORGANIZATION IS RELATIVE
```

```
54                ACCESS IS  RANDOM  RELATIVE KEY IS branch-number-rel
55                FILE STATUS IS file-stat.
56
57              SELECT teller
58                ASSIGN TO teller
59                ORGANIZATION IS RELATIVE
60                ACCESS IS  RANDOM  RELATIVE KEY IS teller-number-rel
61                FILE STATUS IS file-stat.
62
63              SELECT history
64                ASSIGN TO history
65                ORGANIZATION IS SEQUENTIAL
66                ACCESS IS  SEQUENTIAL
67                FILE STATUS IS file-stat.
68
69
70      ************************************************************************
71      *    In the RECEIVE CONTROL paragraph the TABLE OCCURS is  set         *
72      *    to 2 since the MAXLINKS to the server in PATHWAY has been         *
73      *    set to 2.  This should always match.                              *
74      ************************************************************************
75
76            RECEIVE-CONTROL.
77                TABLE OCCURS 1 TIMES
78                SYNCDEPTH LIMIT is 1
79                REPLY CONTAINS message-out record.
80
81       DATA DIVISION.
82
83           FILE SECTION.
84
85      ************************************************************************
86      *    The message coming from the requestor looks like the following    *
87      *    "*  12 chars  *  12 chars  *   12  chars"                          *
88      *    "*123456789012*456789012345*789012345678"012345678901"           *
89      *    "* Account-num* teller-num * branch-num * descriptor "            *
90      *                                                                      *
91      *tlw: descriptor = network/local transaction.                          *
92      *                                                                      *
93      ************************************************************************
94
95
96      *tlw: fill4 and in-field4 added to message-in-record.
97      *tlw: this value will be used to keep track of remote
98      *tlw: vs. local transactions.
99
100           FD message-in
101              LABEL RECORDS ARE OMITTED.
102           01 message-in-record.
103              02 fill1                PIC X(1).
104              02 in-field1            PIC X(12).
105              02 fill2                PIC X(1).
106              02 in-field2            PIC X(12).
107              02 fill3                PIC X(1).
108              02 in-field3            PIC X(12).
109              02 fill4                PIC X(1).
110              02 in-field4            PIC X(12).
```

```
111
112             FD message-out
113                LABEL RECORDS ARE OMITTED
114                RECORD CONTAINS 102 CHARACTERS.
115             01 message-out-record.
116                02 reply-header          PIC 9(4) COMP.
117                02 outjunk               PIC X(100).
118
119          *tlw: account-balance field switched with account-branch field.
120
121             FD account
122                LABEL RECORDS ARE OMITTED
123                RECORD CONTAINS 100 CHARACTERS.
124             01 account-record.
125                02 account-number        PIC  X(12).
126                02 account-balance       PIC  S9(10)V9(2).
127                02 account-branch        PIC  9(12).
128                02 account-junk4         PIC  X(12).
129                02 account-junk5         PIC  S9(4) COMP.
130                02 account-junk6         PIC  X(10).
131                02 account-junk7         PIC  X(10).
132                02 account-junk8         PIC  X(10).
133                02 account-junk9         PIC  X(10).
134                02 account-junk10.
135                   03  account-junk11   PIC S9(4) COMP.
136                   03  account-junk12   PIC X(8).
137
138             FD branch
139                LABEL RECORDS ARE OMITTED
140                RECORD CONTAINS 100 CHARACTERS.
141             01 branch-record.
142                02 branch-number         PIC  X(12).
143                02 branch-balance        PIC  S9(10)V9(2).
144                02 branch-junk3          PIC  9(12).
145                02 branch-junk4          PIC  X(12).
146                02 branch-junk5          PIC  S9(4) COMP.
147                02 branch-junk6          PIC  X(10).
148                02 branch-junk7          PIC  X(10).
149                02 branch-junk8          PIC  X(10).
150                02 branch-junk9          PIC  X(10).
151                02 branch-junk10.
152                   03  branch-junk11    PIC S9(4) COMP.
153                   03  branch-junk12    PIC X(8).
154
155             FD teller
156                LABEL RECORDS ARE OMITTED
157                RECORD CONTAINS 100 CHARACTERS.
158             01 teller-record.
159                02 teller-number         PIC  X(12).
160                02 teller-balance        PIC  S9(10)V9(2).
161                02 teller-junk3          PIC  9(12).
162                02 teller-junk4          PIC  X(12).
163                02 teller-junk5          PIC  S9(4) COMP.
164                02 teller-junk6          PIC  X(10).
165                02 teller-junk7          PIC  X(10).
166                02 teller-junk8          PIC  X(10).
167                02 teller-junk9          PIC  X(10).
```

```
168                02 teller-junk10.
169                   03  teller-junk11  PIC S9(4) COMP.
170                   03  teller-junk12  PIC X(8).
171
172          *tlw: changed history-delta to history-descriptor.
173
174                FD history
175                   LABEL RECORDS ARE OMITTED
176                   RECORD CONTAINS 50 CHARACTERS.
177                01 history-record.
178                   02 history-account-number     PIC  X(12).
179                   02 history-teller-number      PIC  X(5).
180                   02 history-branch-number      PIC  X(5).
181                   02 history-descriptor         PIC  S9(10)V9(2).
182                   02 history-yymmdd             PIC  X(6).
183                   02 history-hhmmsscc           PIC  X(9).
184                   02 history-junk               PIC  X(1).
185
186             WORKING-STORAGE SECTION.
187
188             01 file-stat                 PIC X(2).
189             01 teller-number-rel         PIC 9(12).
190             01 branch-number-rel         PIC 9(12).
191             01 loop-record               PIC 9.
192                   88 not-all-done           VALUE 0.
193                   88 all-done               VALUE 1.
194
195          *tlw: fill4-ws, descriptor-ws, and descriptor-rd added to message-in-record.
196          *tlw: this value is used to keep track of remote vs. local transactions.
197
198            01 message-in-ws.
199                   02 fill1-ws             PIC X(1).
200                   02 account-number-ws    PIC X(12).
201                   02 fill2-ws             PIC X(1).
202                   02 teller-number-ws     PIC X(12).
203                   02 teller-number-rd  REDEFINES teller-number-ws  PIC 9(12).
204                   02 fill3-ws             PIC X(1).
205                   02 branch-number-ws     PIC X(12).
206                   02 branch-number-rd  REDEFINES branch-number-ws  PIC 9(12).
207                   02 fill4-ws             PIC X(1).
208                   02 descriptor-ws        PIC X(12).
209                   02 descriptor-rd     REDEFINES descriptor-ws  PIC 9(12).
```

```
210       /
211       ******************************************************************
212       *  Main routine                                                  *
213       ******************************************************************
214        PROCEDURE DIVISION.
215
216        MAIN SECTION.
217
218        BEGIN-ET1-COBOL-server.
219
220           PERFORM Init-files.
221
222           PERFORM do-transaction UNTIL all-done.
223
224           PERFORM shut-files.
225
226           STOP     RUN.
```

```
227       /
228       ******************************************************************
229       *  Initialization, open $RECEIVE and all disc files              *
230       ******************************************************************
231        Init-files.
232           OPEN INPUT message-in   SYNCDEPTH 1.
233           OPEN OUTPUT message-out.
234           OPEN I-O account SHARED.
235           OPEN I-O teller  SHARED.
236           OPEN I-O branch  SHARED.
237           OPEN extend history SHARED.
238       *tlw-: MOVE 00 TO reply-header.
239
```

```
240       /
241       ******************************************************************
242       *  Shut down - close $RECEIVE and all disc files                 *
243       ******************************************************************
244
245        Shut-files.
246          CLOSE message-in.
247          CLOSE message-out.
248          CLOSE account.
249          CLOSE teller.
250          CLOSE branch.
251          CLOSE history.
```

```
252       /
253        do-transaction.
254       ******************************************************************
255       *  read from $receive and get account-num, branch-num and teller-num.*
256       *  Time stamp the transaction.                                     *
257       ******************************************************************
258
259       *tlw
260          MOVE 00 TO reply-header.
261
262          READ   message-in.
263          if     file-stat not equal to zero move 1 to reply-header.
264
265          MOVE   message-in-record TO message-in-ws.
266
267          ACCEPT history-yymmdd FROM DATE.
268          ACCEPT history-hhmmsscc FROM TIME.
269
270          MOVE account-number-ws TO account-number.
271          MOVE teller-number-rd TO teller-number-rel.
272          MOVE branch-number-ws TO branch-number-rel.
273
274       ******************************************************************
275       *  Read the account record and update the record.                  *
276       ******************************************************************
277
278          READ account RECORD
279               WITH LOCK
280               KEY IS account-number
281          END-READ.
282
283          if   file-stat not equal to zero move 2 to reply-header.
284
285          ADD descriptor-rd to account-balance.
286
287          REWRITE account-record END-REWRITE.
288          if file-stat not equal to zero move 3 to reply-header.
289
290       ************************************************************
291       *  Read the teller record and update the record.           *
292       ************************************************************
293
294          READ teller  RECORD WITH LOCK END-READ.
295          if file-stat not equal to zero move 4 to reply-header.
296
297          ADD descriptor-rd TO teller-balance.
298
299          REWRITE teller-record END-REWRITE.
300          if file-stat not equal to zero move 5 to reply-header.
301
302       ************************************************
303       *  read the branch record and update the record. *
304       ************************************************
305
306          READ branch  RECORD WITH LOCK END-READ.
307          if file-stat not equal to zero move 6 to reply-header.
308
```

```
309          ADD descriptor-rd to branch-balance.
310
311          REWRITE branch-record END-REWRITE.
312          if file-stat not equal to zero move 7 to reply-header.
313
314       ****************************************
315       *  Memo post to history file.          *
316       ****************************************
317
318          MOVE account-number   TO history-account-number.
319          MOVE teller-number-ws TO history-teller-number.
320          MOVE branch-number-ws TO history-branch-number.
321          MOVE descriptor-rd    TO history-descriptor.
322
323          WRITE  history-record  END-WRITE.
324          if file-stat not equal to zero move 8 to reply-header.
325
326
327       ********************************************************************
328       *  Reply back to the requestor                                     *
329       ********************************************************************
330
331          WRITE message-out-record   END-WRITE.
332
333       ********************************************************************
334       *   the transaction is finished                                    *
335       ********************************************************************
336
337        SHOWIT.
338           ENTER TAL DEBUG.
```

ENTRY POINT MAP BY NAME

SP	PEP	BASE	LIMIT	ENTRY	ATTRS	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	002	000003	001460	000023	M	ET1-COBOL-SERVER	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL

DATA BLOCK MAP BY NAME

BASE	LIMIT	TYPE	MODE	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
000020	000052	SPECIAL	WORD	#ERUG	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL
000000	000017	SPECIAL	WORD	#GO	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL
100643	104355	SPECIAL	WORD	#HIGHBUF	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL
100643		SPECIAL	WORD	#RQTB	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL
100000	100114	SPECIAL	WORD	#RUCB	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL
000053	000514	OWN	STRING	ET1-COBOL-SERVER	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL
100115	100642	SPECIAL	WORD	ET1-COBOL-SERVER#	18FEB87	18:40	COBOL85	$BASE.GRAYTGU1.COBOL

```
BINDER - OBJECT FILE BINDER - T9621C00 - (15JUL87)   SYSTEM \FOXII
Object file name is $BASE.GRAYTGU1.RUNUNIT
Object file timestamp is 18FEB87 18:40:10
Number of Binder errors = 0
Number of Binder warnings = 0
Primary data = 16 words
Secondary data = 317 words
Code area size = 1 pages
Resident code size = 0 pages
Data area size = 35 pages
Extended data area size = 0 pages
Top of stack = 333 words
Number of code segments  = 1 segment


COBOL85 - T9257C00 - (15 JUL 87)
The object file is executable on a Tandem NonStop System
Number of compiler errors = 0
Number of compiler warnings = 0
Maximum symbol table size = 12280 bytes
Elapsed time - 00:00:27
```

READER COMMENT CARD

Tandem welcomes your comments on the quality and usefulness of its software documentation. Does this manual serve your needs? If not, how could we improve it? Your comments will be forwarded to the writer for review and action, as appropriate.

If your answer to any of the questions below is "no," please supply detailed information, including page numbers, under Comments. Use additional sheets if necessary.

- ▶ Is this manual technically accurate? Yes ☐ No ☐
- ▶ Does it provide the information you need? Yes ☐ No ☐
- ▶ Are the organization and content clear? Yes ☐ No ☐
- ▶ Are the format and packaging convenient? Yes ☐ No ☐

Comments

Name Date

Company

Address

City/State Zip

NonStop SQL Benchmark Workbook

NonStop™ Systems

First Edition: Part No. 84160

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

B U S I N E S S R E P L Y M A I L

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

POSTAGE WILL BE PAID BY ADDRESSEE

Tandem Computers Incorporated
Attn: Manager—Software Publications
Location 01, Department 6350
19333 Vallco Parkway
Cupertino CA 95014-9990

TAPE TAPE

Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, CA 95014-2599