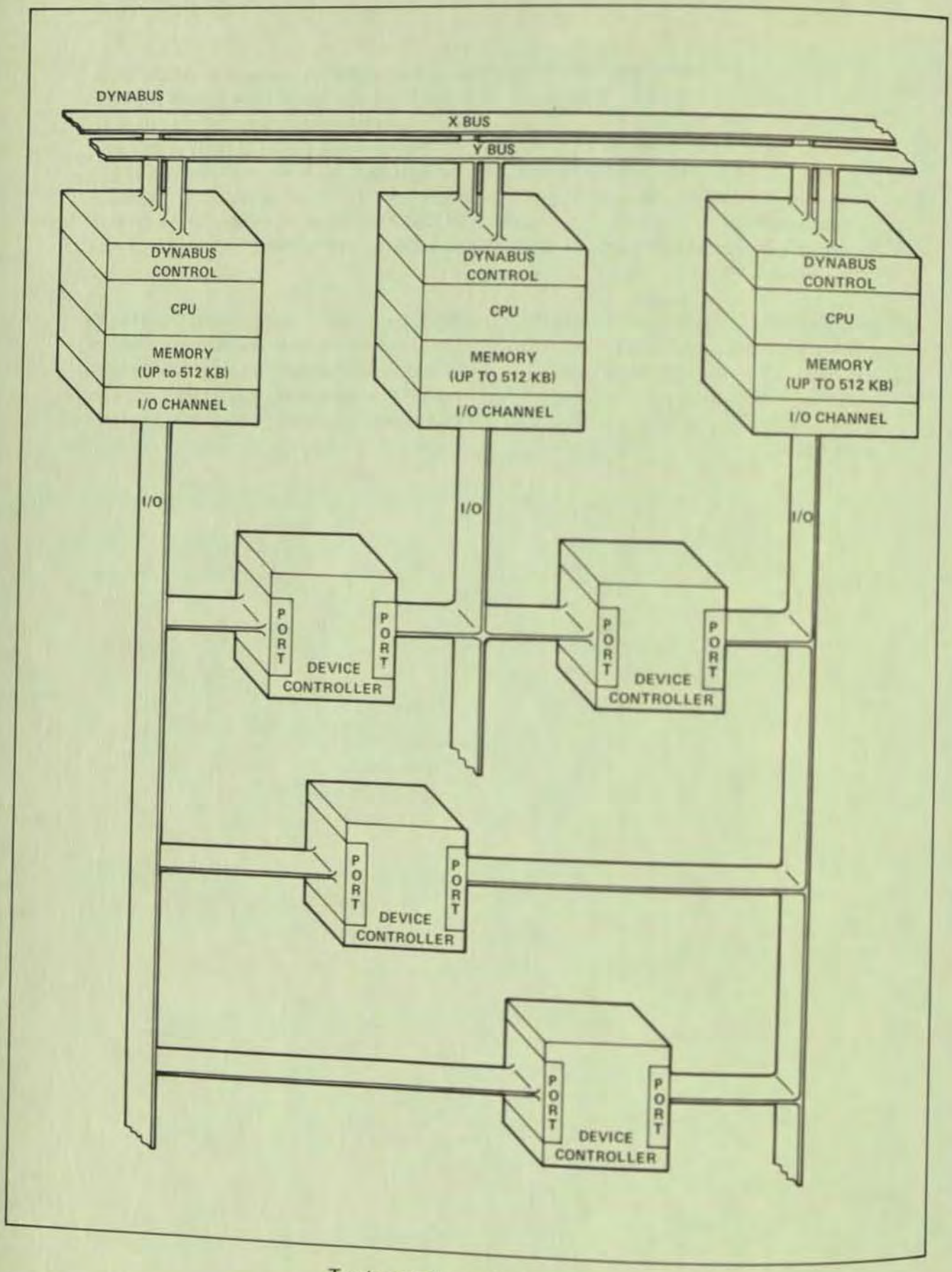
TANDEM

NonStop™ SYSTEMS

INTRODUCTION

The Tandem 16 System, described in this document, represents a major departure from existing mini/midi computer architecture. For the first time, a complete system has been designed to meet the growing demand for on-line, transaction processing systems with "fail-safe" capability. Using standard Tandem 16 hardware and software modules, the user may build a system to match necessary requirements exactly, both in throughput and in system reliability. No special or custom designed hardware or software is necessary. Equally important, the Tandem 16 can grow to meet increasing throughput demands with no change in operating system or applications software and without loss of the system during expansion.

One Tandem 16 processor module is a powerful computer. Two or more Tandem 16 processor modules connected together by Tandem's high-speed interprocessor bus structure (DYNABUS) provide an extremely powerful multiprocessor system. This capability coupled with the Tandem 16's unique *NonStop*™ or "fail-safe" features make it the ideal choice for systems requiring economy today plus assured power to meet tomorrow's increased demands.



Tandem 16 Computer System

The Tandem 16 Computer System fills three important and interrelated needs: it is a computer system where applications run *NonStop* regardless of a module failure, it provides a computer system that can support high transaction rates to large on-line data bases, and it provides a system that is easily adaptable to any application.

The basic design philosophy of the Tandem 16 Computer System is that no single module failure will stop or contaminate the system. This assurance of *NonStop* operation is sometimes called "fail-safe" when no loss of throughput occurs as a result of a failure or "fail-soft" when some slowdown occurs but full processing capabilities are maintained. The Tandem 16 can provide both "fail-safe" and "fail-soft' modes of operation.

- PROCESSOR MODULES

A single Tandem 16 Computer System may contain from two to sixteen processor modules; each module contains a micro-programmed central processing unit, its own memory (up to 512k bytes), and its own micro-programmed input/output channel. Each processor module is fully capable of operating independently of all other processor modules yet can be configured to back up other processor modules.

- INTERPROCESSOR BUSES (DYNABUS)

Each processor module is connected to all other processor modules via redundant high speed interprocessor buses. Programs running in one processor module communicate with programs running in other processor modules by means of these buses. Each interprocessor bus is fully autonomous, operating independently of (but simultaneously with) the other bus. The use of two buses assures that two paths exist between all processor modules in the system.

- INPUT/OUTPUT CHANNEL

Input/output devices (i.e, magnetic tape units, disc drives, terminals, etc.) are interfaced to the computer system via dual-port i/o controllers. Dual-port means that each i/o controller is connected to the input/output channels of two processor modules. This provides two paths of communication to each input/output device. A dual-port controller is "owned" by (i.e., will accept commands from) only one processor module. But in case of a failure, the other processor module can take control programmatically. The dual-port controllers are designed so that the number of components common to both paths are at a minimum.

- TANDEM 16/TRANSACTION OPERATING SYSTEM (T/TOS)

Overseeing system operation is the Tandem 16 Operating System. The operating system provides the multiprocessing (parallel processing in separate processor modules), multiprogramming (interleaved processing in one processor module), and *NonStop* capabilities of the Tandem 16 Computer System. A copy of T/TOS resides in each processor module.

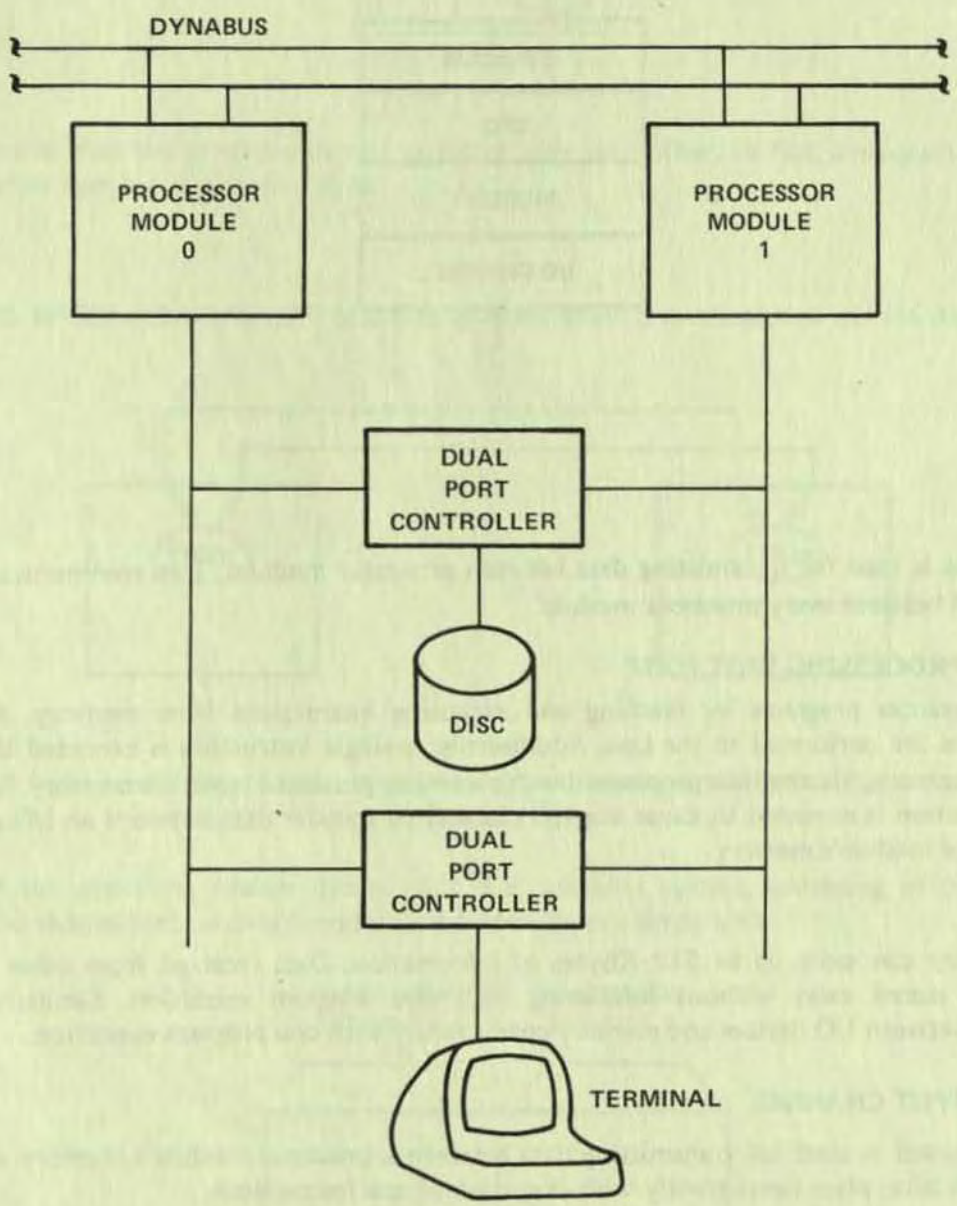
The operating system automatically schedules application programs for execution according to an application-assigned priority, provides memory management functions (automatic overlaying, swapping to disc, etc.), and gives application programs the capability to start programs executing in any processor module from any processor module.

- FILE SYSTEM

As part of the operating system, the Tandem 16 File System provides a single interface between programs and the outside world. The file system, which handles all data transfers, provides the capability for any program running in the system to communicate with any other program running in the system as well as any input/output device connected to the system; programmers need not be aware of the physical location of the device/program. Up to 4,094 bytes can be transferred in one file system operation.

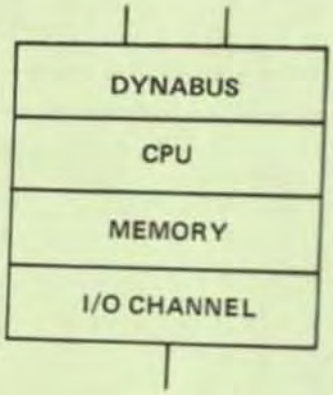
HOW NONSTOP WORKS

A minimum TANDEM 16 NonStop™ System:



- PROCESSOR MODULE

  A Processor Module is comprised of:



DYNABUS

The Dynabus is used for transmitting data between processor modules. Two communication paths are provided between every processor module.

CENTRAL PROCESSING UNIT (CPU)

The cpu executes programs by fetching and executing instructions from memory. Arithmetic computations are performed in the cpu. Additionally, a single instruction is executed to transmit data from memory, via the interprocessor bus, to another processor module's memory. Likewise, a single instruction is executed to cause the I/O channel to transfer data between an I/O device and this processor module's memory.

MEMORY

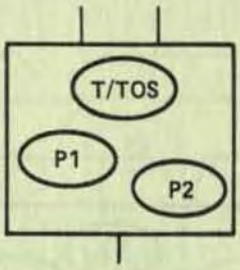
Main memory can store up to 512 Kbytes of information. Data received from other processor modules is stored away without interfering with cpu program execution. Similarly, data is transferred between I/O devices and memory concurrently with cpu program execution.

INPUT/OUTPUT CHANNEL

The I/O channel is used for transmitting data between a processor module's memory and its I/O devices. This takes place concurrently with execution of cpu instructions.

- T/TOS

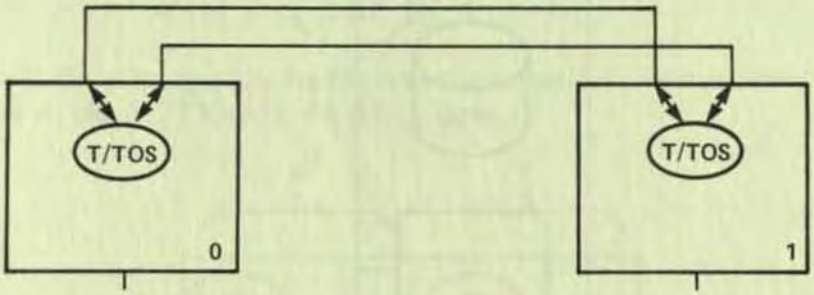
The TANDEM 16 Operating System (T/TOS) provides the capability for multiple programs to run concurrently in the same processor module:



P = PROGRAM

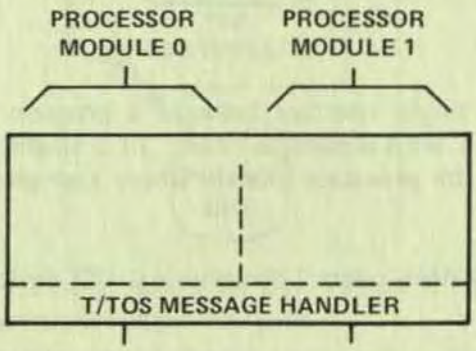
T/TOS ensures that the programs do not interfere with each other. In fact, a program need not be aware of other running programs – at all.

The T/TOS MESSAGE HANDLER provides interprocessor communication via the interprocessor buses:



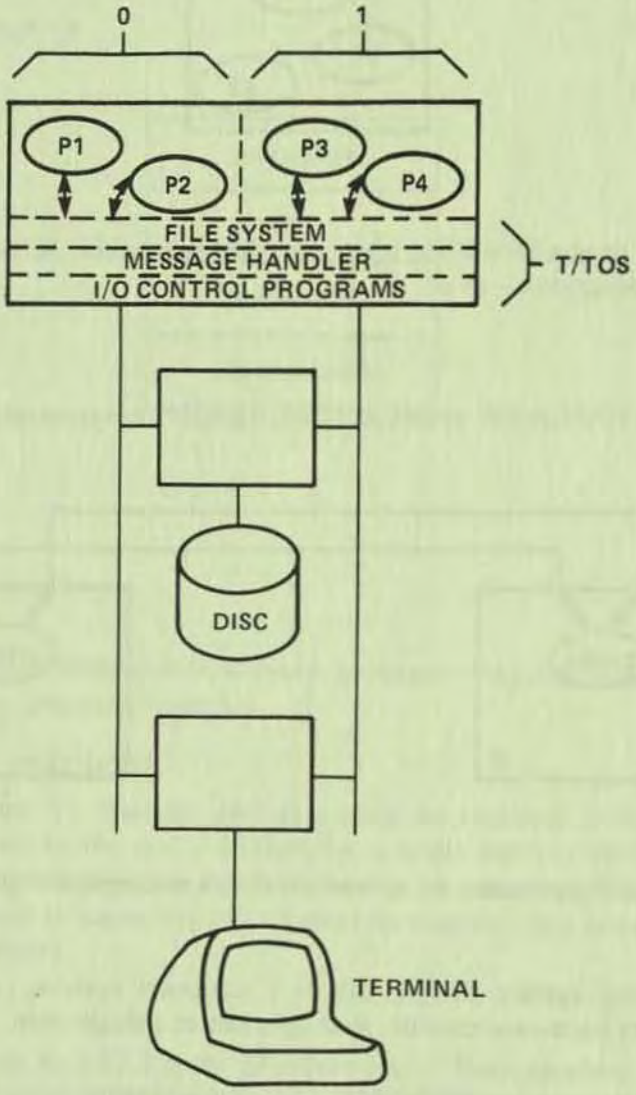
T/TOS uses both buses if necessary to guarantee that a message will get to the desired processor module.

Because of the operating system design, this is a computer system, consisting of two or more separate (and redundant) hardware module, that operates as a single unit.



- FILE SYSTEM

The FILE SYSTEM part of T/TOS is used by programs to communicate with each other and with I/O devices.

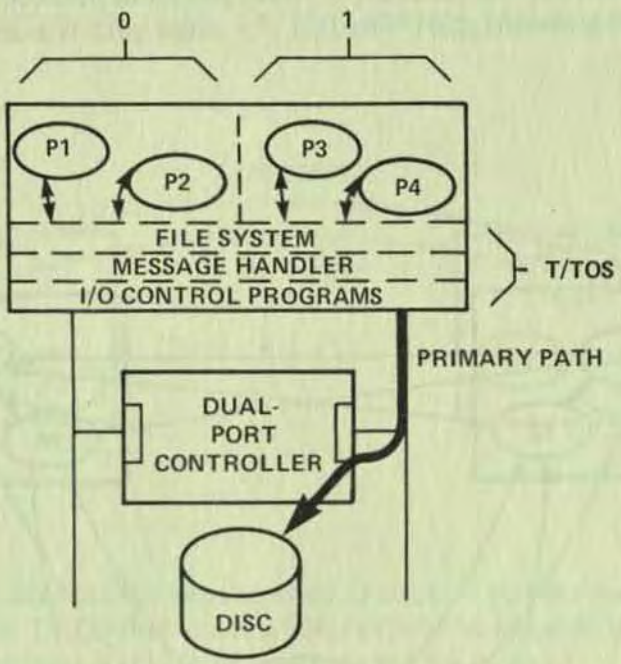


The file system provides a single interface between a program and the outside world. Other programs and all I/O devices are accessed as "files" in a single, uniform manner. The physical location of I/O devices and the processor module where a program is executing is transparent to application programs.

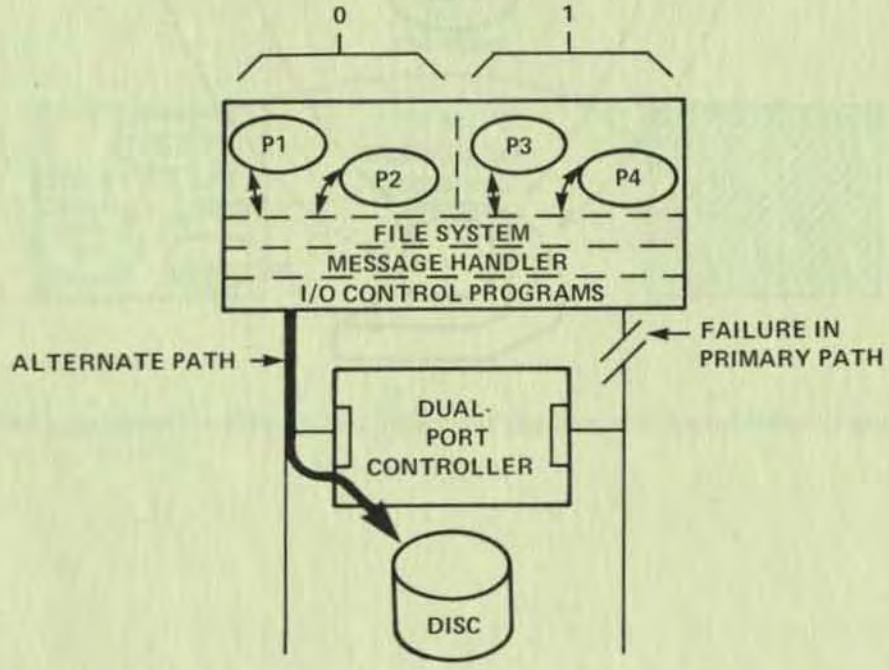
(The I/O CONTROL PROGRAMS control the physical I/O devices on instructions from the file system.)

The use of dual port controllers guarantees a communication path to each I/O device even if a failure occurs.

Each device has a PRIMARY PATH over which communication normally occurs:



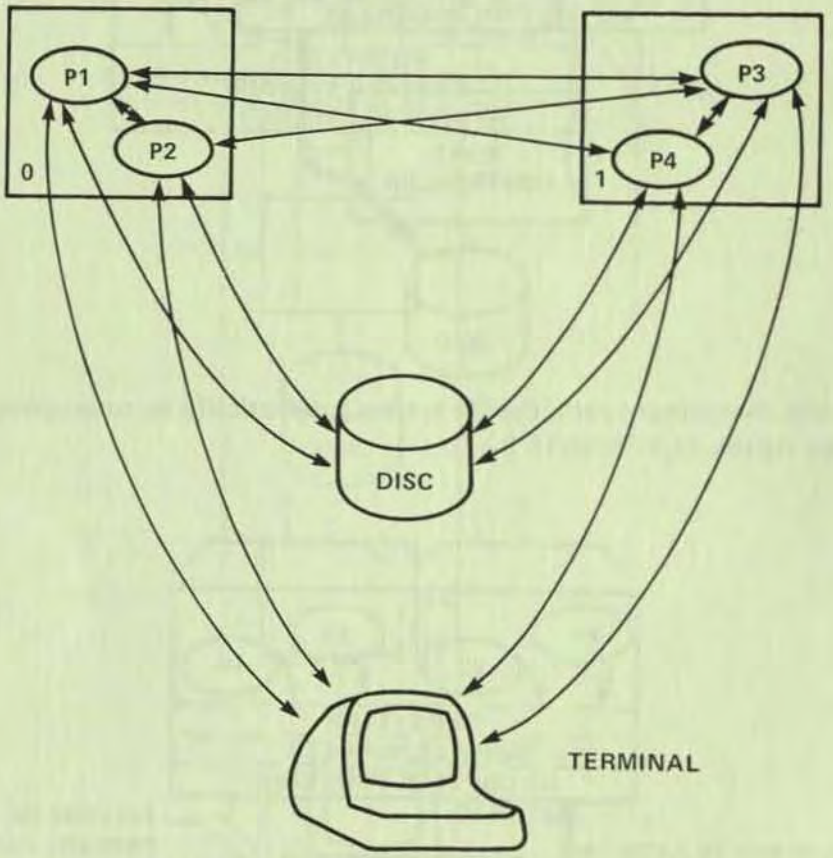
If a failure occurs in the primary path, the file system automatically reroutes communication to the affected I/O device via the ALTERNATE PATH:



All communication to the device now occurs via the original alternate path. When the original primary path is restored, it becomes the "alternate" path.

Because of the file system design:

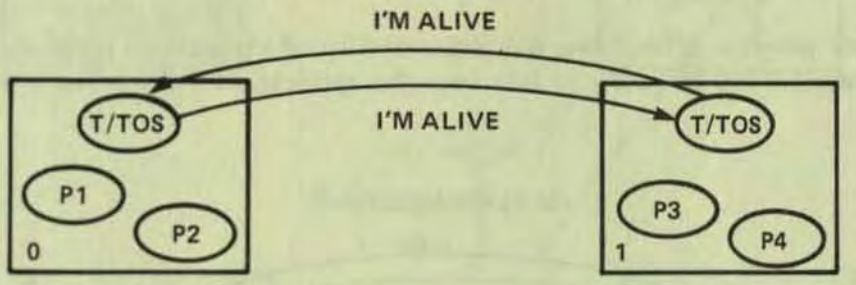
programs running in independent and redundant hardware modules can communicate with each other and with any I/O device. The hardware provides two paths between processor modules and to I/O devices. The operating system guarantees that if a single path is available, communication will occur.



- PROCESSOR MODULE CHECKING

The operating system provides another service:

While application programs are executing, T/TOS in each processor module periodically transmits an "I'm alive" message to all other processor modules in the system. Each processor module, in turn, periodically checks for receipt of an "I'm alive" message from every other processor module.



If T/TOS finds that a message has not been received, it assumes that the non-transmitting processor module has malfunctioned. T/TOS then sends a "CPU DOWN" message to interested programs in its processor module. (This message is sent to programs via the file system.)

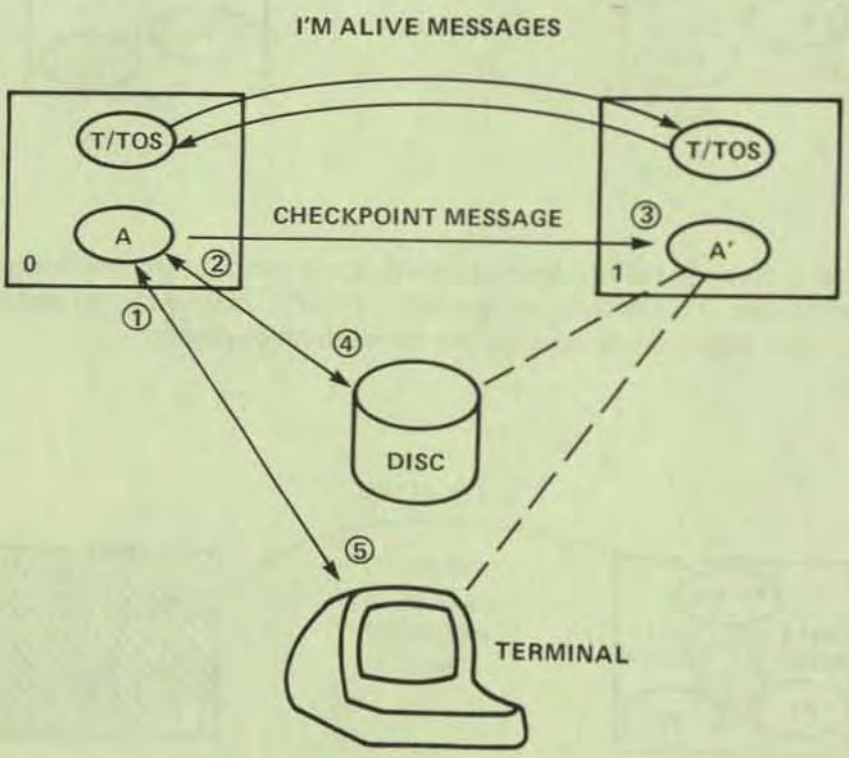
This means that a program running in one processor module will be informed if another processor module fails.

- A NONSTOP APPLICATION

A NonStop application consists of a primary program running in one processor module (called A in this example) and a backup program running in another module (called A').

The primary program, while operable, performs all of the applications work. At critical points in the application (such as prior to altering a disc file), the primary program sends a message containing "checkpointing" information to the backup program.

While the primary program is operable, the responsibility of the backup program is to accept the checkpointing messages and be ready to take over the application if the primary program becomes inoperable.



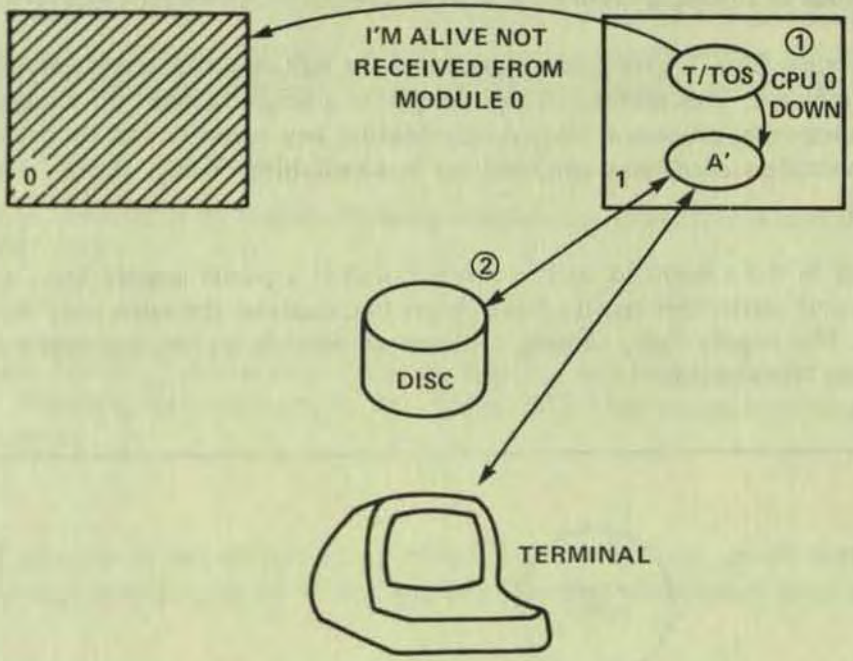
THE PROGRAM: A

① READ (a record from the terminal)
② READ (a record from the disc)
③ WRITE (the old disc record to A') Checkpoint
④ WRITE (the updated record to disc)
⑤ WRITE (the result on the terminal)

THE PROGRAM: A'

READ (the checkpoint message from A)

Then, if A's processor module fails:



PROGRAM A' ACTION

① READ (the cpu 0 down message)

② WRITE (to the disc using the last checkpoint message to restore the record)

Then continue with the same program as A.

READ (a record from the terminal)
READ (a record from the disc)
.
.
.
.

Except that there is no backup for A' at this time, so no checkpoint message is sent.

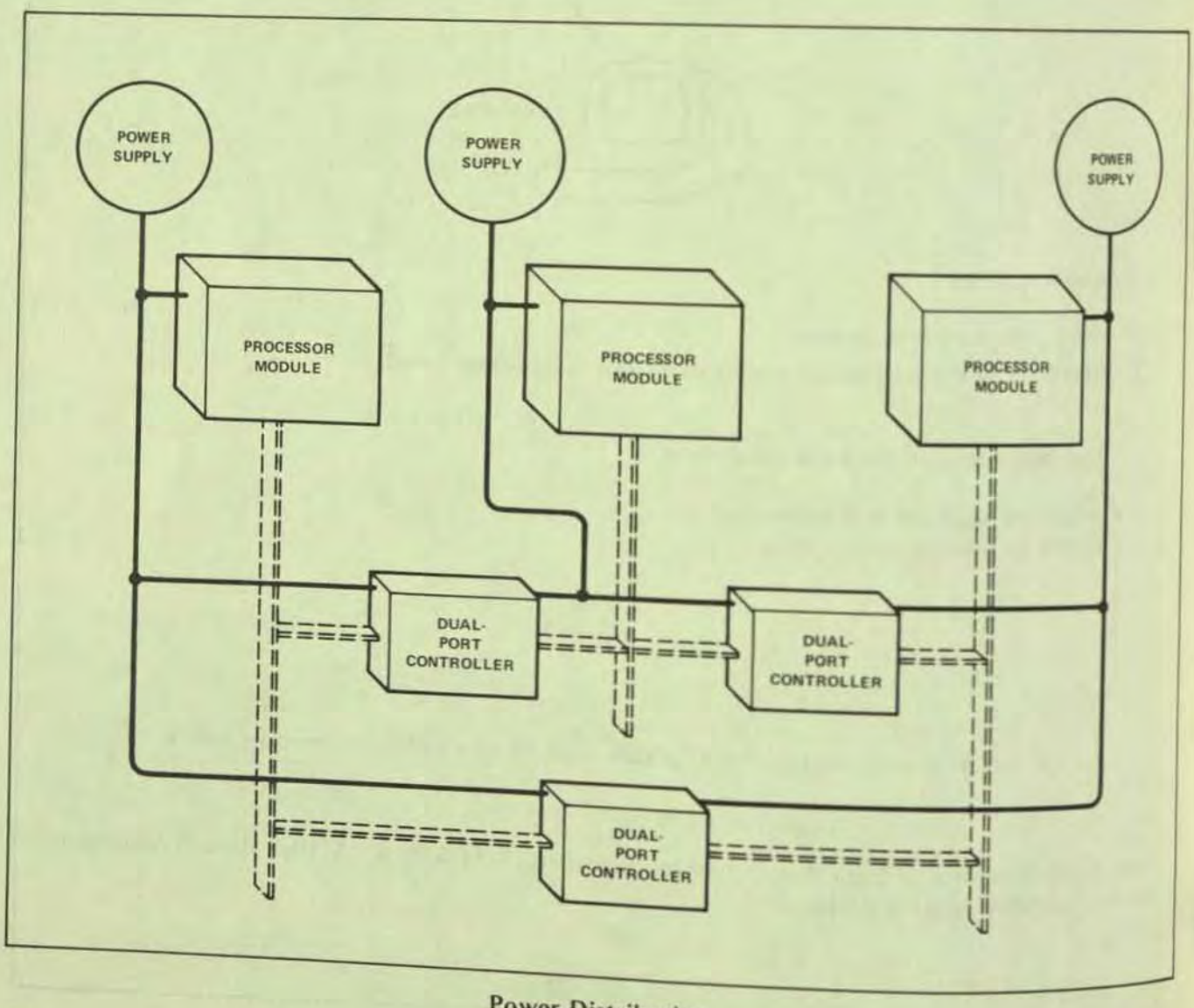
When cpu 0 is repaired, a "cpu 0 up" message is sent by T/TOS to A'. A' then starts A running in cpu 0. (And A becomes the backup.)

WHY THE TANDEM 16 IS FAILURE TOLERANT

The ability of the Tandem 16 Computer System to provide an environment where applications can continue to run regardless of a module failure is due to its unique hardware and software design:

- Interprocessor communication is provided by means of the redundant interprocessor buses, therefore no shared memory is required. This eliminates a point where a single failure could stop a system and also prevents one malfunctioning processor from contaminating any memory but its own. Likewise, the use of dual-port i/o controllers eliminates the need for bus switching devices (again, where a single failure could stop a system).

- Power is distributed in the system in such a manner that if a power supply fails, a backup module is available. The dual-port controllers receive power from two sources: the same supplies as their associated processor modules. If a supply fails, causing a processor module to become inoperative, the alternate processor module can take control.



Power Distribution

- The operating system in each processor module saves the current operating state in the event a power failure occurs and resumes all operations when power is restored.

- If an uncorrectable error is found in memory, the operating system determines if the associated area is critical to system operation. If it is not, the area is flagged as bad and not used again until the memory is repaired. (Typically, the memory would be repaired during system preventive maintenance. However, the the associated processor module could be taken off line to repair the memory, leaving the remainder of the system operable.)

- Critical portions of the operating system are main memory resident; this assures their availability in the event a disc failure occurs.

- The cooling system for the Tandem 16 is designed in such a way that if a failure occurs, ample cooling is still available. In addition, fan modules can be replaced while the system is running (without interfering with system operation).

- Any operational module in the system (e.g., processor, i/o controller, power supply, fan, etc.) can be removed from the system and replaced on-line without stopping operation of other system modules.

- Because the Tandem 16 is modular in organization, it can be configured for any degree of *NONSTOP*ability desired.

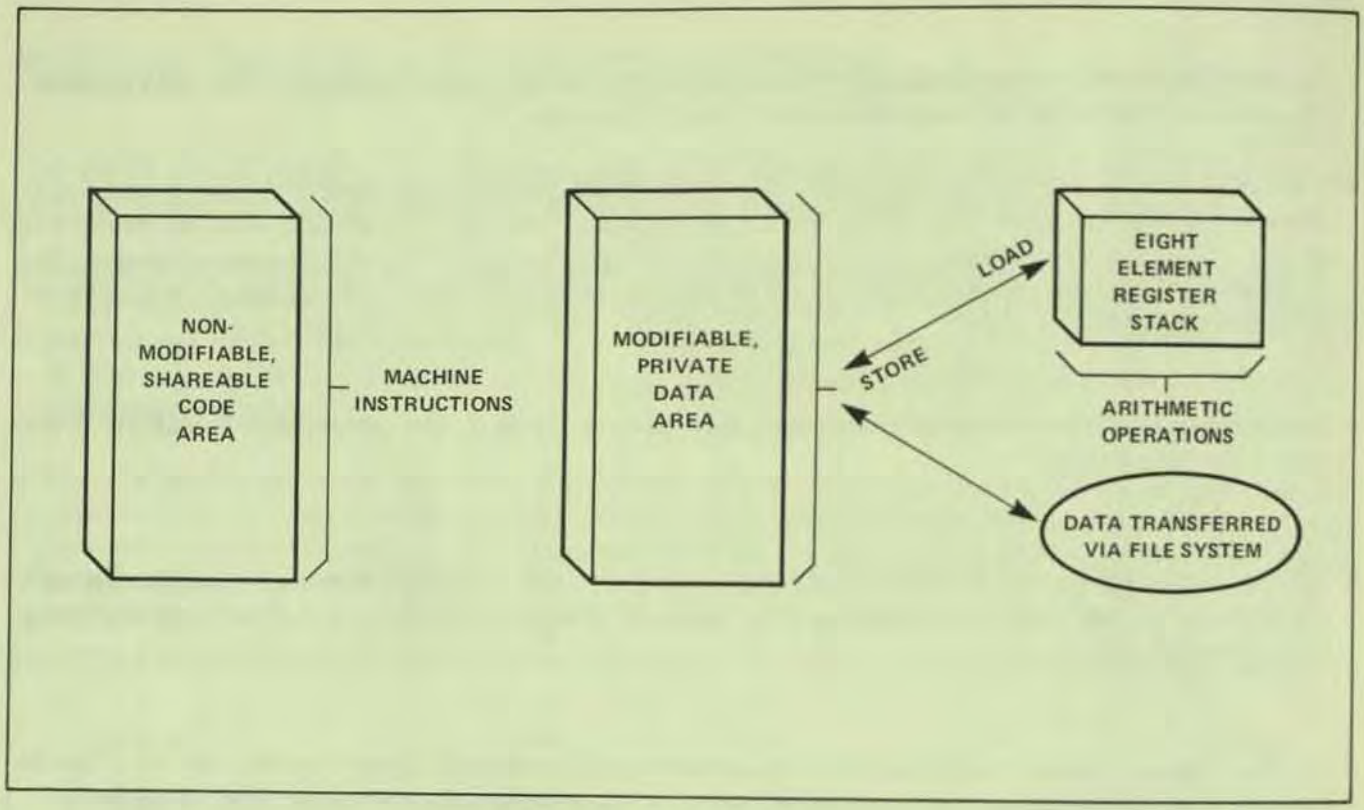
WHY THE TANDEM 16 SUPPORTS HIGH TRANSACTION RATES

A number of system characteristics result in the high throughput rates attainable by the Tandem 16. They are –

- **Program Organization**

Tandem 16 programs while executing in memory are physically separated into two parts: a code part containing machine instructions and program constants and a data part containing program variables. The code part of a program is actually read-only storage (i.e., there are no machine instructions for writing into the code area).

The fact that the code part contains pure code and cannot be modified means that it can be shared by a number of users. In fact, operating system library routines that are executed on behalf of application programs are shared by all application programs running in a given processor module (i.e., only one copy need reside in memory).



Program Organization

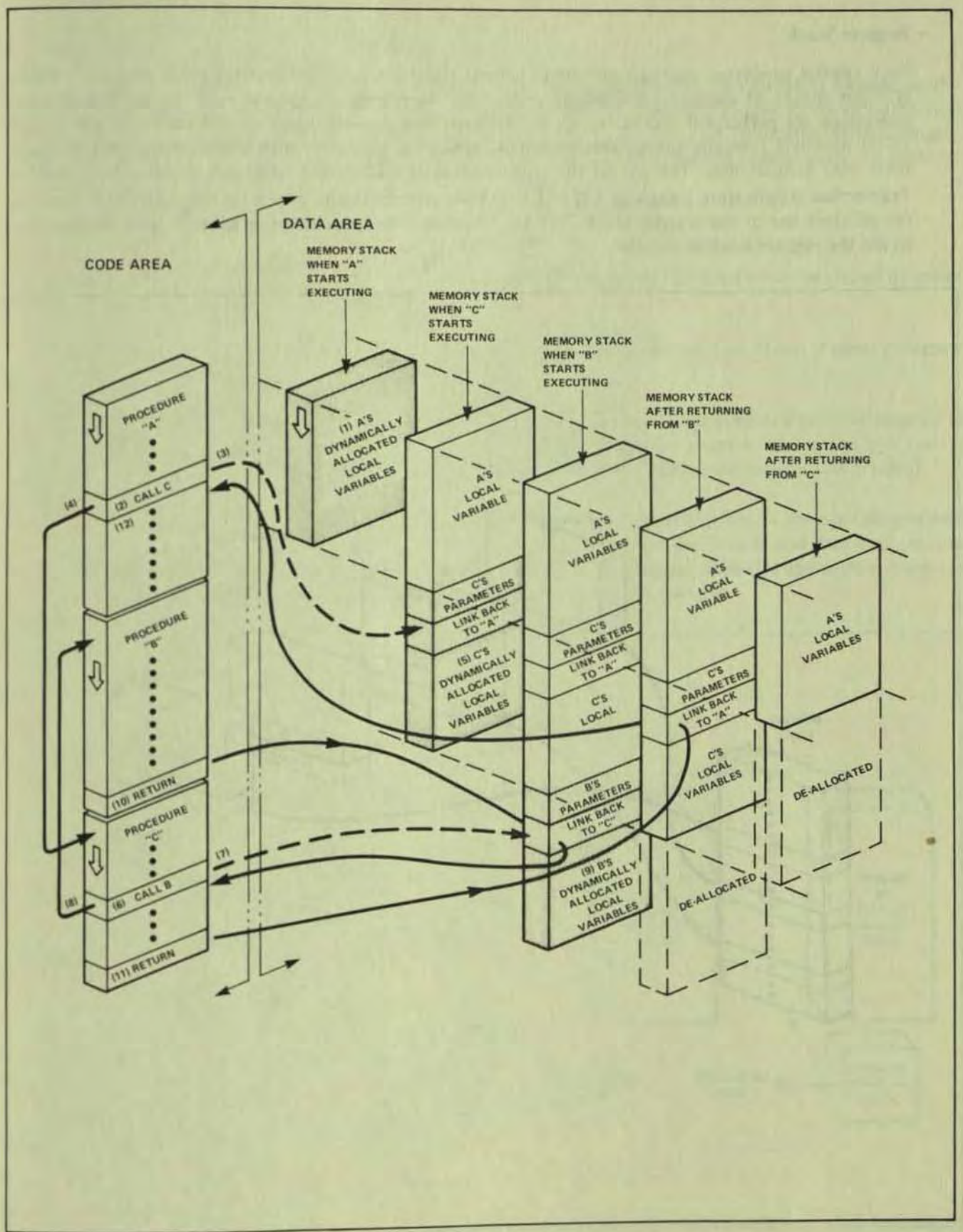
– Procedure Oriented

Programs are functionally separated into blocks of machine instructions called procedures. A procedure, like a program, has its own private data area (actually in the program's data area). The real power of procedures is that they can be called into execution from any point in a program (including other procedures and themselves); the hardware automatically saves the calling environment when a procedure starts executing and restores the calling environment when the procedure finishes. A programmer can write procedures that receive parameter information (arguments), perform computations using the parameters, then return results to the caller (the machine instructions for passing parameters and returning results are generated automatically for programmers using Tandem's Transaction Application Language – T/TAL).

Operating system and file system functions are actually invoked by calling procedures that are part of the operating system (special machine instructions exist that call operating system procedures as efficiently as an application program's own procedures).

– Memory Stack

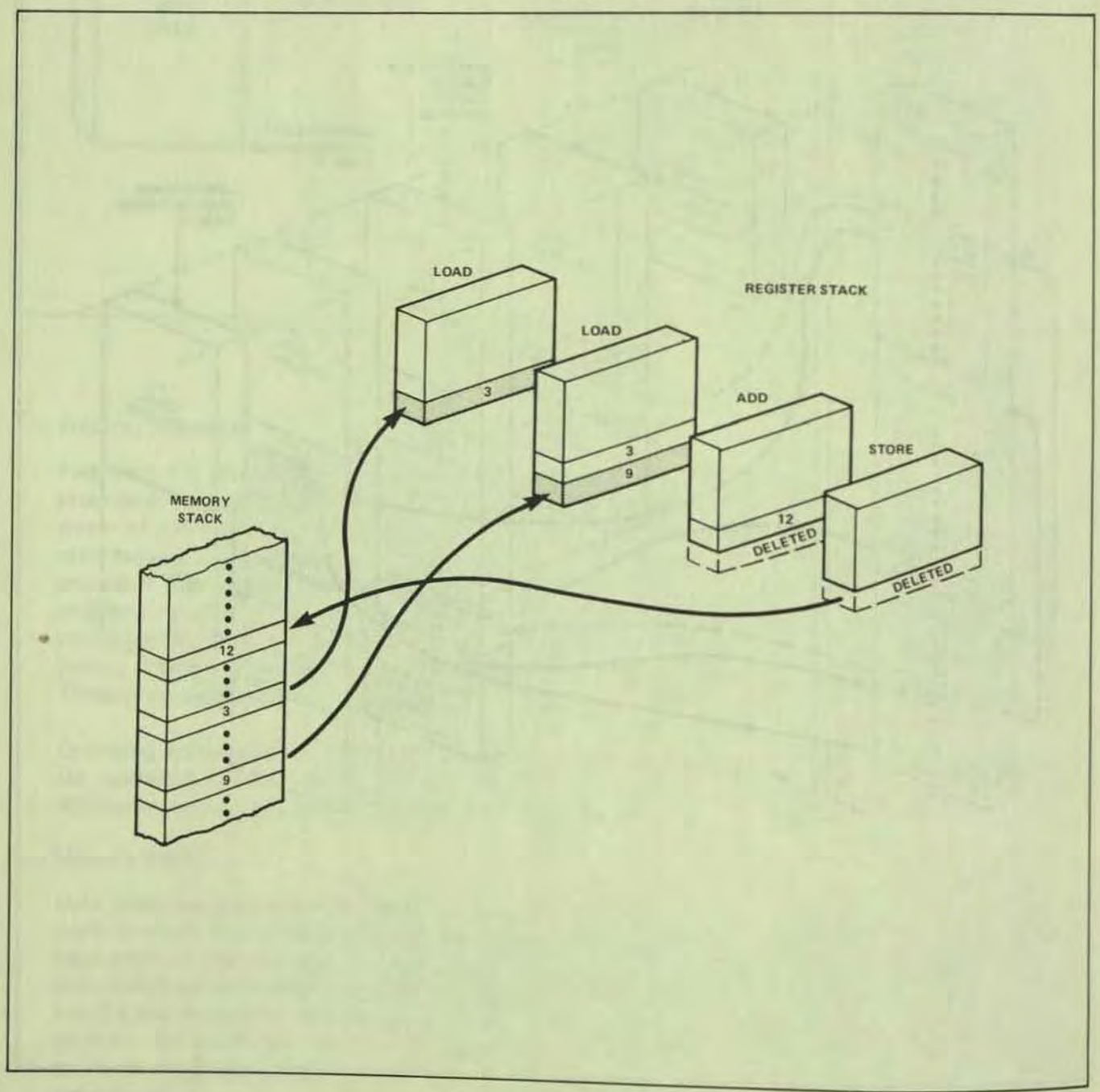
Data areas for programs are organized in main memory as stacks. A stack is a storage allocation method where the last item added is the first item removed. The CPU has registers that automatically keep track of the last area allocated in a stack. The use of the stack means that data areas for a procedure's private variables are allocated dynamically (when the procedure is called into execution), keeping the amount of memory space required by a program to a dynamic minimum. The stack also provides the mechanism for passing parameters to procedures and saving and restoring the calling environment (this applies to calling both an application's own procedures and operating system procedures).



Memory Stack

– Register Stack

Each central processor contains an eight element register stack. The register stack provides a highly efficient means of executing arithmetic operations; operands are loaded into the stack, arithmetic operations are performed, the operands are deleted, and a result is left on the stack. An add of two 16-bit numbers typically takes 500 nanoseconds; storing the result into the memory stack typically takes 900 nanoseconds. The use of the register stack is transparent to programmers using Tandem's Transaction Application Language (T/TAL). T/TAL automatically generates the machine instructions for efficient use of the register stack. T/TAL, however, does provide programmers with the capability to use the register stack explicitly.



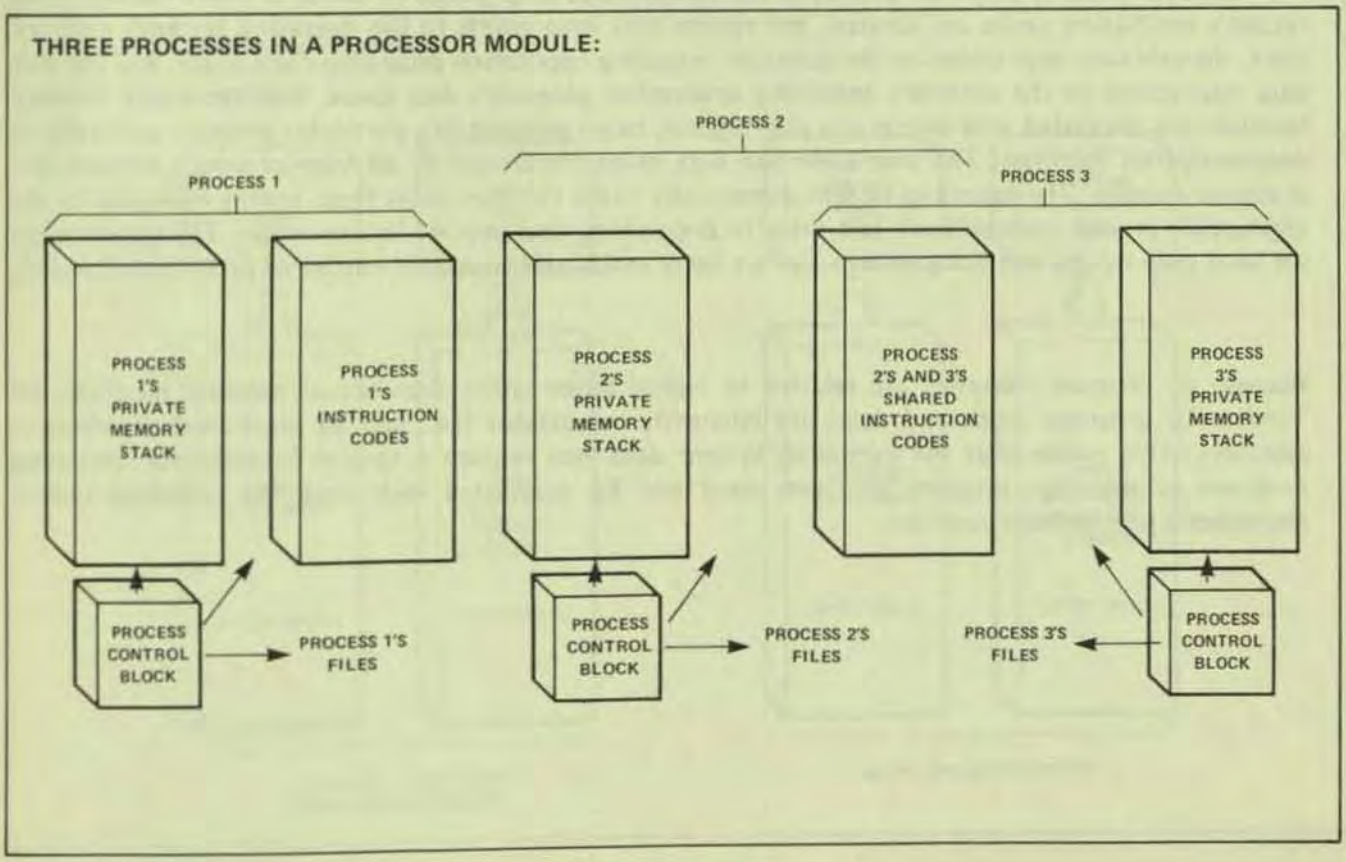
Register Stack

- **Process Structure**

Programs (both application and system programs) are run by the Tandem 16 Operating System on the basis of processes. Two definitions: the term "program" is used to identify a static group of instruction codes and initialized data (like the output of a compiler), the term "process" indicates the dynamically changing states of an executing program. The same program can be executing concurrently a number of times; each execution is a different process.

A process consists of:

- An area in memory containing the instruction codes to be executed (this area may be shared by other processes).
- An area in memory containing a memory stack that is private to the process. (Even if other processes use the same code area, each has a private data area.)
- A process identification number (process id) assigned by the operating system when the program is first called for execution (the process id indicates the processor number where the program is executing, the number of the process in that processor, and the time that the process was created).
- A process control block (PCB), identified by a process id, that is used (only) by the operating system to control process execution. The PCB contains pointers to the process's code and data areas, retains the current state of the process in the event the process is suspended, indicates any system resources currently needed by the process, and defines any files in use by the process.



Process Structure

The process structure provides the mechanism for executing programs in a multiprogramming (interleaved processing) environment. An operating system function called the "dispatcher" assigns processor time to the various processes present in the system. A process, when ready to execute, is placed in a ready list according to its priority number. When one process completes executing or is suspended (while i/o occurs), the highest priority process ready for execution is given control of the processor (a "ready" high priority process automatically causes any lower priority process to be suspended).

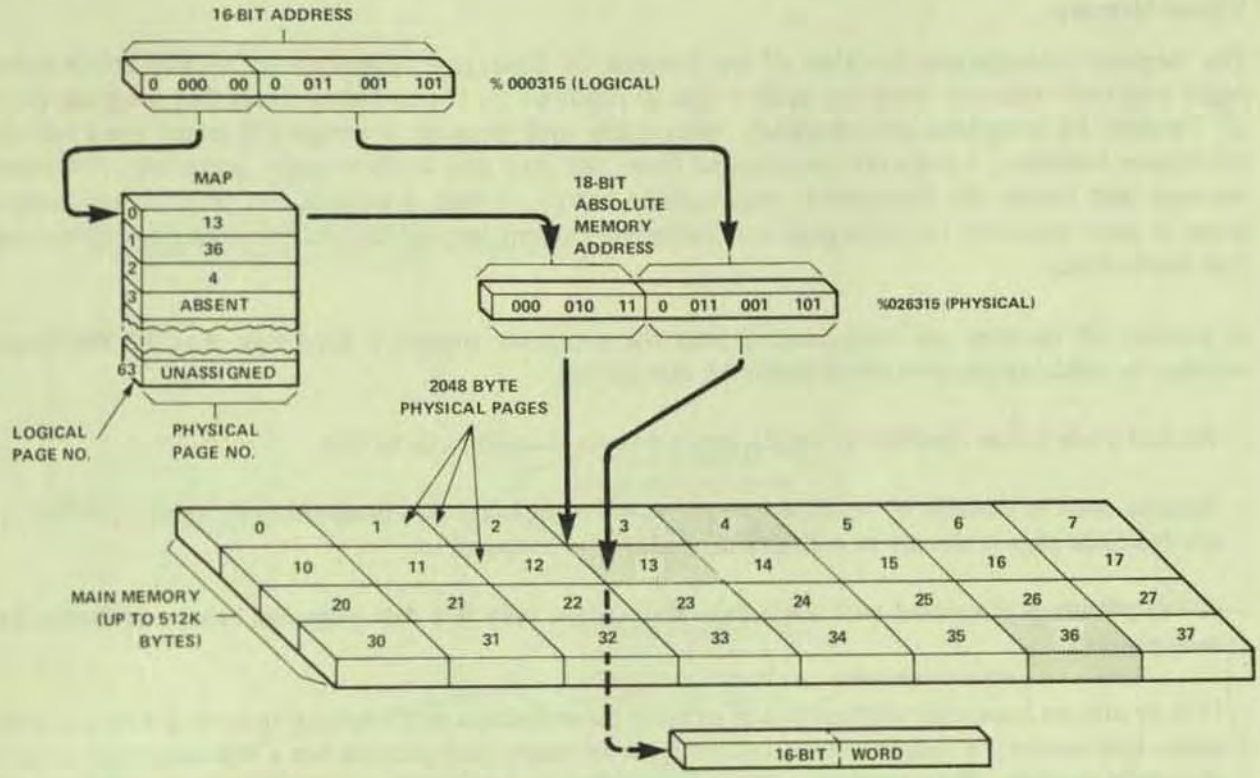
The use of the process structure permits each program to execute as though it has sole use of a processor module and peripherals attached to the system.

- Memory Mapping

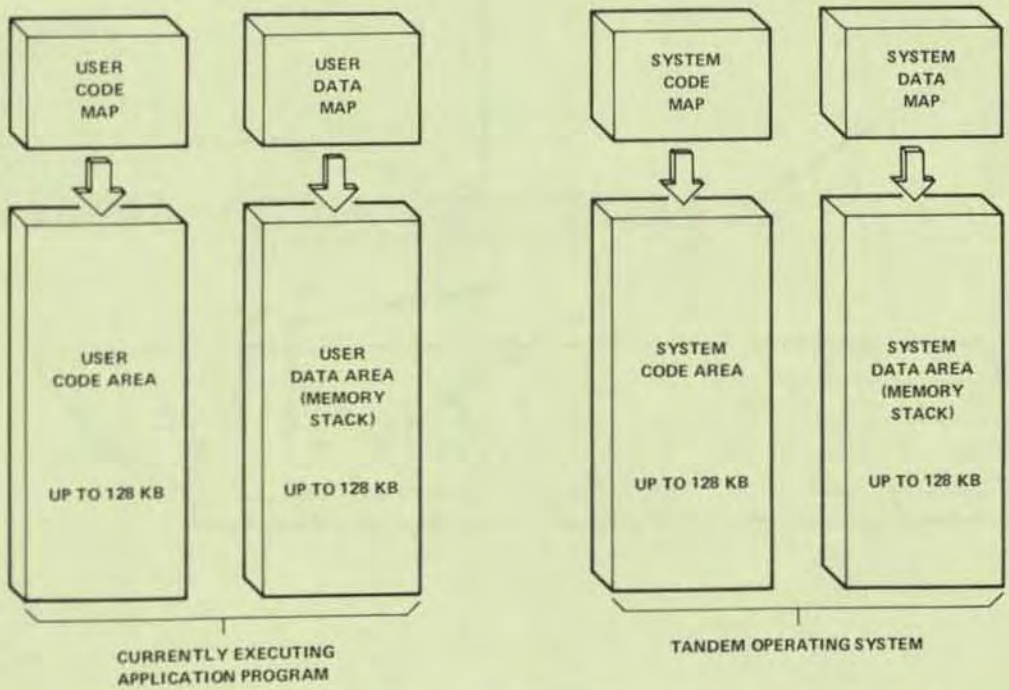
Main memory space is arranged in 2048-byte pages. A memory mapping scheme associates the 16-bit addresses used in a program with the physical pages in memory. The maximum number of pages allotted a process is 128 pages: 64 code pages and 64 data pages. The pages as addressed in a program are referred to as logical pages, the actual pages in memory are called physical pages. The first page in a program's code space and data space are referred to as logical page 0. Because of the mapping scheme, a program's memory pages need not be located contiguously.

Four separate memory maps are provided: the system code map points to the area where the operating system's instruction codes are located, the system data map points to the operating system's memory stack, the user code map points to the currently executing application program's code space, and the user data map points to the currently executing application program's data space. Because actual memory locations are associated with entries in a map register, pages assigned to a particular program can reside in non-contiguous locations. The user code and data maps are shared by all user processes present in a processor module. The operating system dynamically loads the user maps from a table indicated by the appropriate process control block just prior to dispatching that process for execution. The system maps are used only by the operating system and are never modified (except for entries of non-resident pages).

Because all program addresses are relative to logical pages rather than actual memory locations, all Tandem 16 programs (code and data) are inherently relocatable (i.e., can be positioned anywhere in memory). This means that the operating system does not require a special function for relocating programs in memory; program addresses need not be readjusted each time the operating system dispatches a process for execution.



*FOUR MAPS SEPARATE CODE FROM DATA; PROTECT THE OPERATING SYSTEM FROM APPLICATION PROGRAMS



Memory Mapping

- **Virtual Memory**

The memory management function of the Tandem 16 Operating system automatically brings memory pages into main memory from the system disc as required by the currently executing program. Because all Tandem 16 programs are inherently relocatable and because a program's pages need not be in contiguous locations, a page can be swapped from disc into any available page in memory (the memory manager just makes the appropriate map register entry). In fact, a process can execute with only two pages in main memory: the code page containing the current instruction and the data page referenced by that instruction.

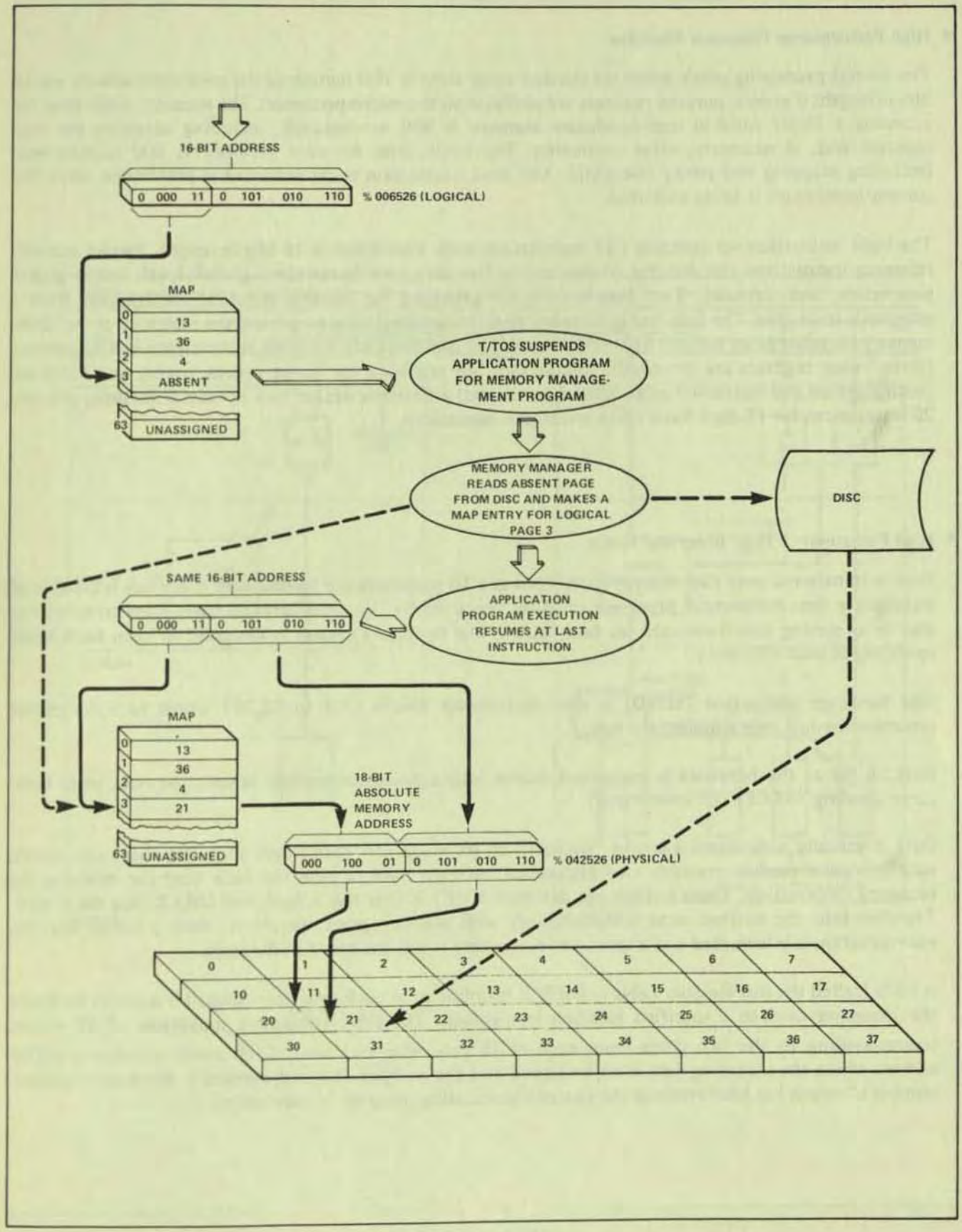
A number of features are incorporated into the processor module's hardware that aid the memory manager in reducing the amount of swapping that occurs.

– Because code is non-modifiable, code pages are never swapped out to disc.

– Because code is sharable by multiple programs, only one copy of a program need exist in memory; if a needed code page is already in memory it need not be swapped in.

– A "dirty" bit is associated with each data map entry; only if a data page has been modified is it ever swapped out.

– History bits are associated with each map entry to record access and overlays occurring with a particular page. The memory manager also maintains a list of maps (each process has a separate map) active in a processor module. When memory space is needed for an overlay, the memory manager selects the map that is at the head of the map list then selects the least accessed page in that map for overlaying. The memory manager ensures that processes are selected on an equal basis for potential overlays by putting the last map selected for overlay at the tail of the map list.



Virtual Memory

- **High Performance Processor Modules**

The central processing unit's micro-instruction cycle time is 100 nanoseconds; microinstructions are 32 bits in length; 6 general purpose registers are available to the micro-processor. The memory cycle time for accessing a 16-bit word in semi-conductor memory is 500 nanoseconds, including accessing the map registers and, if necessary, error correction. The cycle time for core memory is 800 nanoseconds (including mapping and parity checking). The next instruction to be executed is prefetched while the current instruction is being executed.

The basic instruction set contains 123 instructions; each instruction is 16 bits in length. Twelve memory reference instructions can directly address any of five data areas in memory: global, local, system global, parameters, and sublocal. Two instructions are provided for reading constant information from a program's code area. The data and code memory reference instructions can use the contents of the direct memory location as an indirect reference to another location; any of these instructions can be indexed (three index registers are provided). Instructions are provided for string moves, scans, and compares (both eight-bit and sixteen-bit quantities). The decimal arithmetic option to a processor module, provides 20 instructions for 18-digit fixed point arithmetic operations.

- **High Performance Inter-processor Buses**

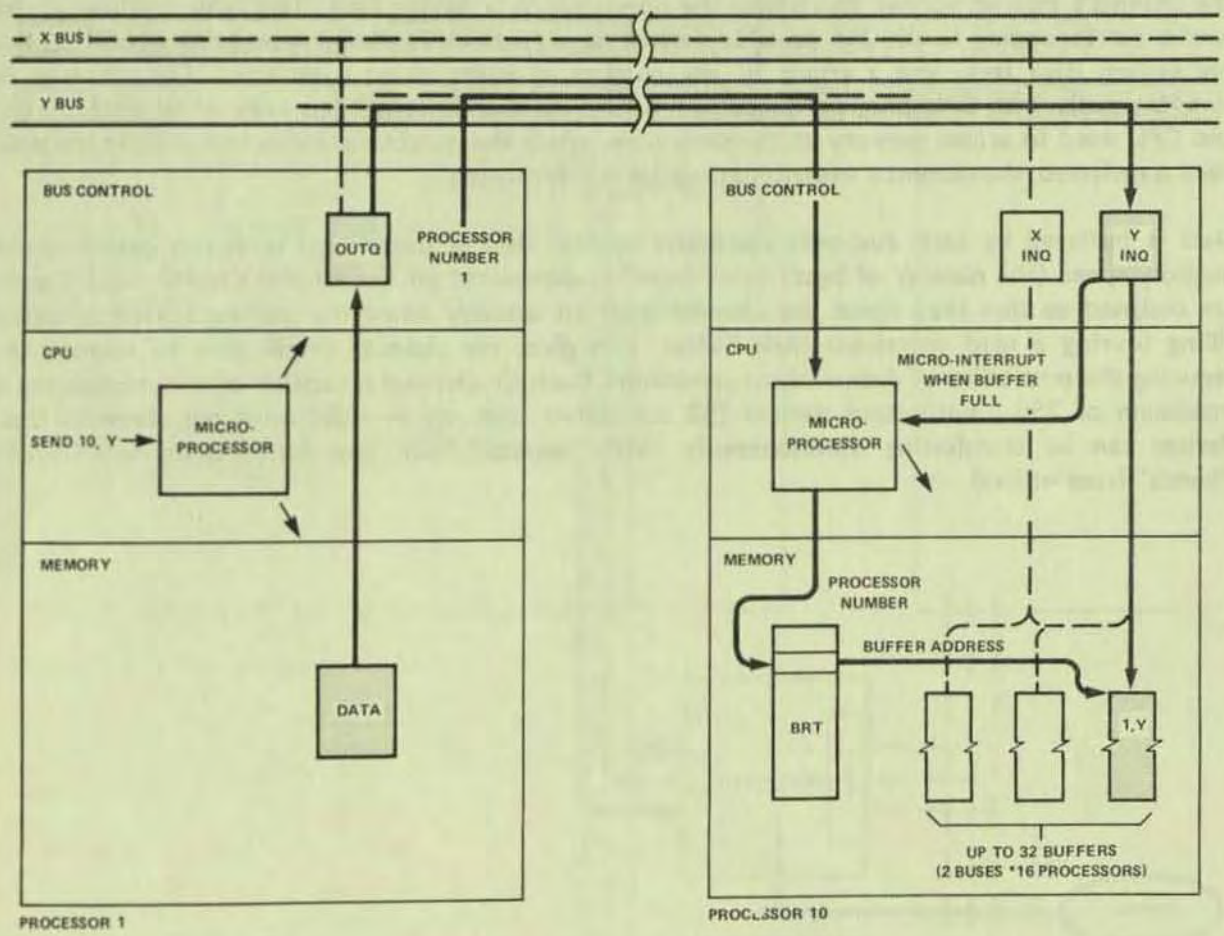
Data is transferred over each interprocessor bus at a 10 megabyte per second rate. Each bus is capable of transferring data between all processor modules concurrently on a multiplexed basis. Data transfers can also be occurring simultaneously on both buses. The operating system is designed to keep both buses operating at peak efficiency.

One hardware instruction (SEND) is used to transmit blocks of 1 to 32,767 words to a designated processor module over a designated bus.

Data, as far as the hardware is concerned, comes into a processor module unsolicited (i.e., there is no corresponding "RECEIVE" instruction).

Data is actually sent across a bus in "packets" of 16 words (15 data words plus one checksum word); each processor module contains two high-speed 16-word buffers (one for each bus) for receiving the incoming information. These buffers are designated INQ X (for the X bus) and INQ Y (for the Y bus). Transfers into the buffers occur simultaneously with microprogram execution; when a buffer fills, the microprogram is interrupted and a special microroutine stores the block in memory.

A table (called the Bus Receive Table – BRT) is maintained in each processor module's memory to direct the incoming data to a specified location in memory. The table contains a maximum of 32 entries (corresponding to the two buses from each of 16 processor modules). Each entry specifies a buffer address where the incoming data is to be stored and the number of words expected. When the expected number of words has been received the currently executing program is interrupted.



Interprocessor Buses

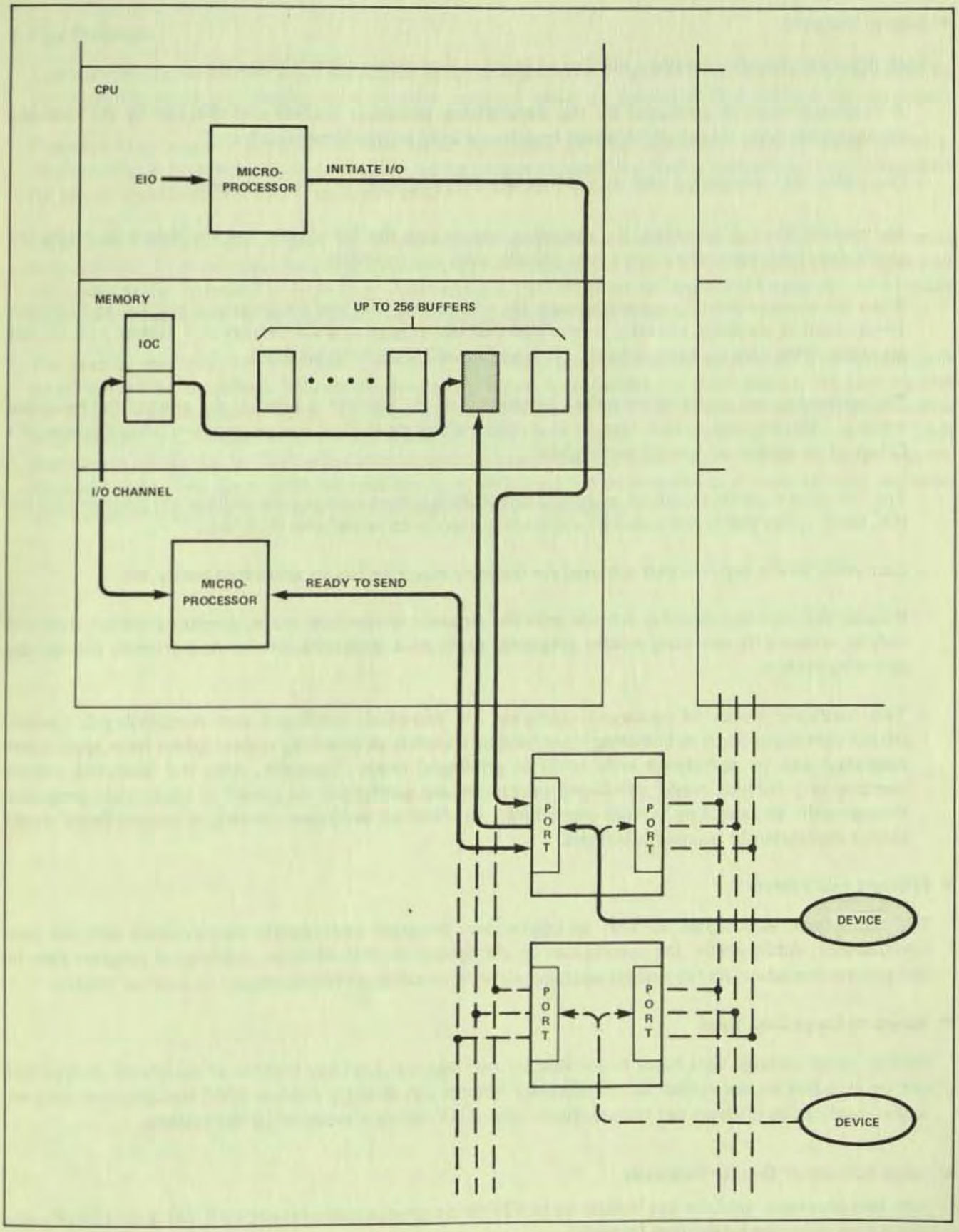
- **High Performance Input/Output Channel**

  Input/output transfers occur directly between memory and i/o devices concurrently with program execution. A single i/o operation is capable of transferring data in blocks of from one to 4094 bytes.

  One hardware instruction (EIO) is used to initiate input/output operations. Once i/o is initiated, a special microprocessor contained in the i/o channel controls the transfer of data between an i/o device and memory.

  A table (called the i/o control table — IOC) is maintained in each processor module's memory to permit the channel's microprocessor to control the operation on a device basis. The table contains up to 256 entries corresponding to the 256 possible devices on a channel; each entry contains a buffer address (in the system data area) and a count of the number of bytes to be transferred. Data transfer occurs simultaneously with CPU program execution; CPU execution is suspended only when both the channel and CPU need to access memory at the same time. When the number of bytes indicated in the IOC have been transferred, the currently executing program is interrupted.

  Data is buffered by each dual-port controller so that data is transferred in bursts over a channel at memory speed (the number of bytes in a "burst" is dependent on a controller's buffer size). Controllers are designed so that they signal the channel prior to actually emptying (during a write operation) or filling (during a read operation) their buffer. This gives the channel ample time to respond thereby reducing the possibility of data overrun conditions. Each i/o channel is capable of communicating with a maximum of 256 input/output devices (32 controllers with up to eight units per controller); all 256 devices can be transferring simultaneously (with "bursts" from one device being interleaved with "bursts" from others).



Input/Output Channel

- **System Integrity**

  Each processor module contains a number of features that assure system integrity:

  - A checksum word is generated by the transmitting processor module and checked by the receiving processor for every packet of 15 words transferred over an interprocessor bus.

  - One parity bit is associated with each 16-bit word transmitted over an i/o channel.

  - An interval timer is provided; the operating system and the file system use the timers to notify the application program in the event a data transfer does not complete.

  - When the semiconductor memory is used, six error correction bits are generated and stored with each 16-bit word in memory; circuitry is provided that corrects all single bit errors and detects ALL double bit errors. With core memory, a parity bit is generated for each 16-bit word.

  - The addressing and count information controlling an i/o transfer is kept in the controlling processor module. This prevents a controller from contaminating more than one processor module because of a failure of an address or word count register.

  - The file system protects against an input/output device from erroneously writing into memory (in the IOC table, either the device's count field is set to zero or its write only bit is set).

  - Each entry in the registers that are used for memory mapping has an associated parity bit.

  - Because the memory mapping scheme provides separate system/user maps, operating system areas can only be accessed by operating system programs; application programs cannot inadvertently destroy the operating system.

  - Two hardware modes of processor operation are provided: privileged and nonprivileged. Certain critical operations (such as initiating input/output transfers or accessing system tables from application programs) can be performed only while in privileged mode. Typically, only the operating system operates in privileged mode; privileged operations are performed on behalf of application programs through calls to operating system procedures. Application programs running in nonprivileged mode cannot inadvertently become privileged.

- **Efficient File System**

  The file system is designed so that an application program can execute concurrently with its own input/output. Additionally, the system can be configured so that while an application program runs in one processor module, its file system operations can be occurring simultaneously in another module.

- **Access to Large Data Bases**

  On-line access to large data bases is possible for two reasons: the large number of peripheral devices that can be attached to the system (a 16 processor system can directly address 2048 input/output devices) and an application program can communicate with ANY device connected to the system.

- **Large Number of On-line Terminals**

  Any two processor modules can handle up to 128 data communication lines; each line is capable of communicating with one single-drop terminal.

- **Fast Response**

Fast response to on-line inquiries is possible because application programs are scheduled for execution by the operating system according to a priority assigned when a program is first readied for execution.

Programs that require precedence over other programs can be scheduled with a higher priority. Additionally, a program can dynamically change its own execution priority, permitting it to compensate for any unusual load conditions that may arise.

Frequently used and critical portions of the operating system are main memory resident (i.e., are never swapped out). This enables operating system functions to be called into execution without having to wait for a page to be swapped in from disc. Additionally, part or all of an application program can be made main memory resident for applications requiring a guaranteed response time.

The area in memory where instruction codes are stored (i.e., referenced by the user and system code maps) cannot be modified. This means that, if a program is suspended for some reason, the memory area can be overwritten immediately; it's not necessary to write this area out to disc. A program's data area is only written out to disc if it has been changed since last loaded into memory. Nonmodifiable code provides an additional benefit: when one program is suspended for some reason, another program can use the same code. This saves time because the code area need not be swapped into main memory and saves space because only one copy of a program need reside in memory.

WHY THE TANDEM 16 IS EASILY ADAPTABLE TO MANY APPLICATIONS

The Tandem 16 Computer System's software has been specifically engineered so that applications can be defined and implemented with a minimum of time and expense.

- Many of the responsibilities normally handled by applications programs with other systems are taken care of automatically by the Tandem 16's operating system (T/TOS).

  – The virtual memory scheme incorporated into the Tandem 16 enables programmers to concentrate fully on the intended application. Programs or portions of programs are swapped between memory and disc automatically by the operating system; the swapping is invisible to the application program.

  – The multiprogramming, multiprocessing features of the Tandem 16 operating system permit programs to be written without regard for other programs running in the system and without regard for the processor module in which a program is eventually run.

  – Programmers can write programs that communicate with input/output devices and other programs without actually knowing where the devices are connected to the system or where other programs are physically executing the system.

  – Using the file system, the hardware aspect of input/output transfers is transparent to application programmers; completion interrupts and system dependent error conditions are handled automatically.

  – Transfers over the interprocessor buses are handled entirely by the operating system; the bus operation is completely invisible to application programmers using the file system.

  – For application dependent error recovery routines (such as not ready on a magnetic tape), the file system provides an error number that specifically describes any errors encountered during an input/output operation.

- User/System Interface

  Application programs and programmers typically make use of the Tandem 16 Computer System through these means:

  – Command Interpreter Program (COMINT)

  Programmer/operator control (versus program control) over the system is by means of a Tandem-supplied program called the Command Interpreter (COMINT). The Command Interpreter performs its functions by conversing with a user through an on-line terminal device; the command interpreter prompts the user for a command, the user enters a command, the command interpreter executes the command, then prompts the user for another command. (Typically, the operating system is configured so that the Command Interpreter program initially executes on one of the console devices. From this point, COMINT can be run on other terminals connected to the system.)

  Functions that the Command Interpreter performs are: obtaining and altering the current operational status of the system, obtaining information about disc files, creating and purging disc files, and running programs (application programs and Tandem-supplied programs such as EDIT, TAL, TOSYSGEN, or COMINT).

For example to obtain the operational status of the system, the operator enters the following command:

```
:STATUS
└— Command Interpreter prompt
```

The command interpreter displays system status information on the terminal.

Entering a command name not known by the Command Interpreter results in an attempt to run a system program by that name. Parameter information can be passed to the program at run time. For example, to run a program called COPY and specify two file names (INFILE and OUTFILE) to the program, the following is entered:

```
:COPY/IN INFILE, OUT OUTFILE/
```

Non-system programs are run by entering a RUN command and the program name. Parameter information can also be passed at this time. For example:

```
:RUN myprog/IN afile, OUT $LP/
```

runs a program called "myprog" and passes the file names "afile" and "$LP" as parameters.

– **Editor Program (EDIT)**

The EDIT or program is used by applications programmers to enter and/or modify source language programs through an interactive terminal.

The editor program is typically run through use of the command interpreter program:

```
:EDIT MYSRCE
```

(MYSRCE is a text file to be accessed by the editor)

Editor functions are invoked interactively by issuing commands to the editor program via an online terminal:

```
*LIST ALL OUT $LP
└— editor prompt
```

(lists the entire contents of the current text file on the system line printer)

– **Tandem/Transaction Application Language Compiler Program (TAL)**

The T/TAL Compiler program is used to prepare ready-to-run programs from source programs written in Tandem's Transaction Application Language (T/TAL). T/TAL is a high-level language, designed especially for transaction processing. A typical statement written using T/TAL that compares two character strings might be:

```
IF INARRAY = "$RECEIVE" THEN . . . .;
```

Or to call a procedure for execution

```
CALL COMPUTETAX(AMOUNT,RATE,TAX);
```

(The name assigned to the procedure is COMPUTETAX; AMOUNT and RATE are parameters; TAX is the result)

The T/TAL compiler program is typically run through use of the Command Interpreter program:

```
:TAL/IN MYSRCE, OUT $LP/MYOBJECT
```

(MYSRCE is the file containing T/TAL source statements, MYOBJECT is the disc file where the ready-to-run object program is stored by the compiler, $_P indicates where the compiler listing is to be sent)

– File System Calls

The file system provides access to all input/output devices in a uniform manner. File system operations are invoked through calls to library procedures that are part of the operating system. Page mode and conversational mode terminals, other devices, discs and portions of discs, and other programs are accessed as files. This permits programs to be written without regard for the actual physical location of the device or program to be accessed. File names are assigned to devices when the system is configured, to disc files when created, and to other programs when they are run. As far as the application program is concerned, interprogram communication appears identical to input/output with physical devices.

A typical T/TAL statement to write to a page mode terminal might be:

```
CALL WRITE (TERMFILE, BUFFER, 600 NUMWRITTEN);
```

(WRITE is a file system procedure; TERMFILE is a file number assigned to a terminal by the file system; BUFFER is an array in the user's memory stack containing the data to be written on the terminal; 600 is the number of bytes to be written; NUMWRITTEN is the actual number written on the terminal).

– Tandem 16 System Generator Program (TOSYSGEN)

A Tandem 16 Computer System is fully configured and ready-to-run when installed on the computer site. A Tandem-supplied system configuration program (TOSYSGEN) is available for tailoring the operating system to suit a particular application. Input/output devices can be added or reassigned, i/o device characteristics are configured, space is assigned for system buffers, programs can be designated that automatically start executing (i.e., without using the Command Interpreter) when the operating system is loaded into the system.

To configure a system, the EDIT program is used to "fill in the blanks" in a disc file that is supplied with the system. The TOSYSGEN program is then run. It reads the configuration file and generates a fully configured operating system (usually on a tape) that is ready to be loaded into the system.

SUMMARY OF TANDEM 16 FEATURES

Tandem 16 Computer System (fail-safe, transaction oriented, easily adaptable)

- Two to sixteen processor modules
- Dual, high-speed interprocessor buses
- High-speed, burst-multiplexed input/output channel
- Multiprocessing, multiprogramming, transaction oriented, fail-safe operating system (T/TOS)
- Virtual file system

Processor Modules

- Microprogrammed (100 nanosecond cycle time)
- Sixteen bit data paths and memory addressing
- Up to 512k bytes memory per processor module
- Memory mapping (four separate maps: system data, system code, user data, user code)
- Up to 128k bytes addressable through each map (for a total of 512k bytes addressable)
- 500 nanosecond semiconductor memory access time (including mapping and error correction)
- 800 nanosecond core memory access time (including mapping and parity checking)
- 123 instructions (including string manipulation and doubleword arithmetic)
- Four-Word (QUAD) Decimal Option: 20 additional instructions (including multiply, divide, rounding, convert quad value to ASCII, convert ASCII to quad value)
- Stack architecture (memory stack and register stack)
- Procedure oriented hardware
- Memory references: direct or indirect with or without indexing to global, local, procedure parameters, top of stack, or system global data areas and to code area. Word, doubleword, byte, and four-word, addressing
- No machine instruction can write into code area (non-modifiable code)
- All programs are inherently re-entrant and relocatable
- I/O transfer, bus receive, and instruction execution occur concurrently
- Next instruction prefetched while current instruction executes
- Hardware power fail/auto restart
- Hardware multiply/divide
- Maximum of five microseconds to call operating system procedures

Interprocessor Buses

- Two paths between each processor module
- 10 megabyte transfer for each bus

- Packet-multiplexed transfers between any number of processor modules
- Block transfers of 1 to 32,767 bytes
- Both buses used simultaneously

Input/Output Channels

- Dual-port Controllers
- Data transfers occur at memory speed (4 megabytes per second with semiconductor memory)
- Up to 256 devices per processor module
- Burst-multiplexed transfers from any number of devices
- Block transfers of 1 to 4,094 bytes

T/TOS Operating System

- Multiprocessing
- Multiprogramming
- Geographical independence
- Process structure
- Memory management function makes virtual memory invisible to users
- Processes scheduled for execution according to priority (0-255)
- Sharable code areas handled automatically
- Up to 256 processes per processor module

File System

- Two paths to each i/o device; file system automatically switches paths
- All devices and programs are accessed as files
- Geographic independence
- Timeout associated with each i/o operation

For more information regarding the Tandem 16 Computer System, refer to the following manuals:

- Tandem 16 System Description: provides a detailed description of the system from a hardware and software standpoint, provides information on making an application NonStop, and tells how to configure the operating system for a particular application and how to run the Tandem-supplied TOSYSGEN program.

- Tandem 16 Programming Manual: describes the T/TAL programming language, how to use the file system, how to use the process control, utility, general purpose, and fail-safe procedures, how an application interfaces with the command interpreter program, and how to use the DEBUG facility.

- Tandem 16 Operating Manual: describes how to use the Tandem-supplied utility programs such as COMINT (the command interpreter), EDIT (the text editor), and BACKUP and RESTORE (for backing up disc files), PUP (the peripheral utility program, for initializing and mounting disc packs, listing device characteristics, etc.), provides a list of system error messages, and describes the system load procedures.

TANDEM

GUARDIAN OPERATING SYSTEM

FEATURES

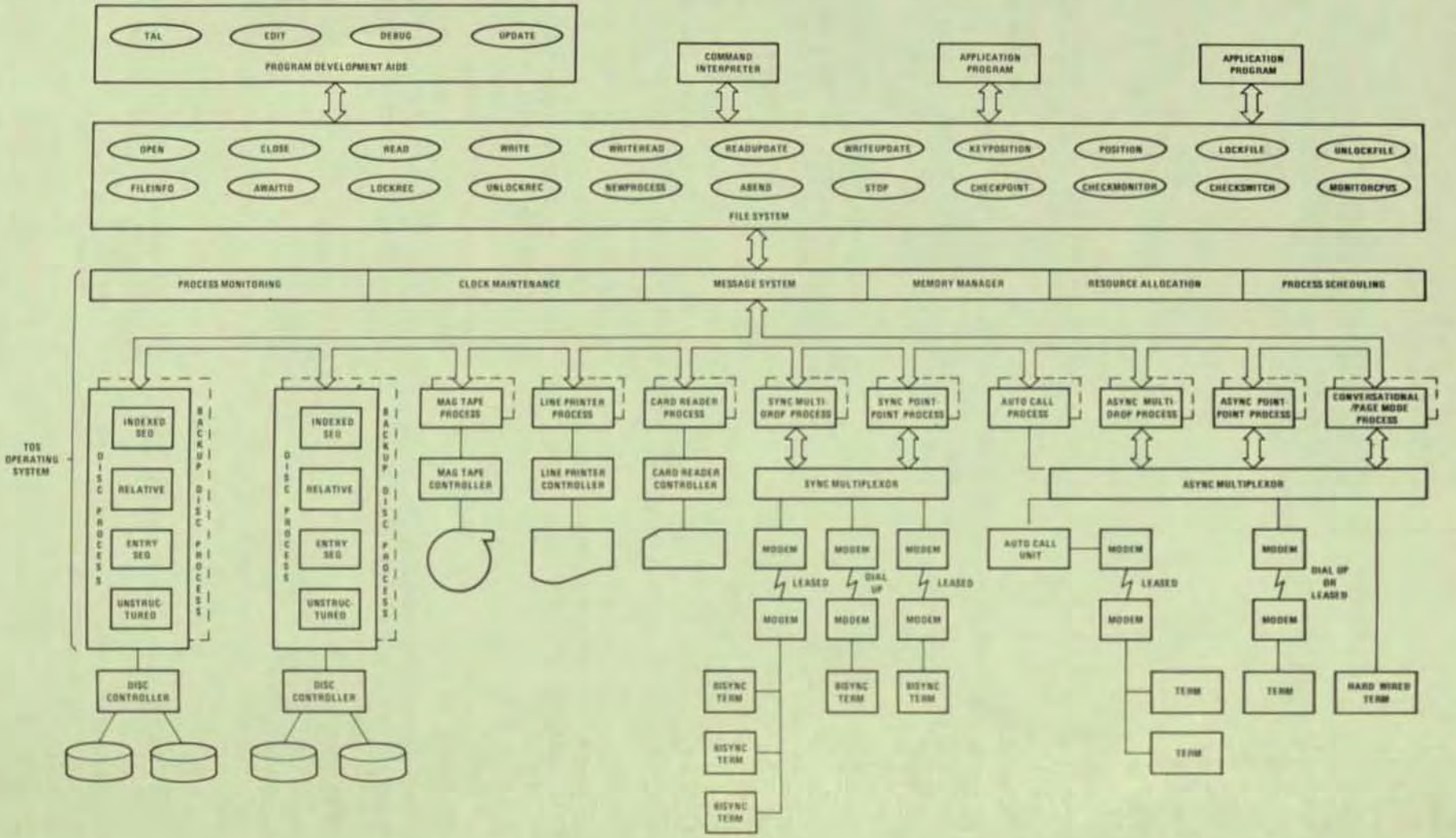
- **Fail-Safe GUARDIAN Operating System for Continuous Non-Stop™ Running of Transaction-Oriented Applications**
- **Multiprogramming/Multiprocessing Virtual Memory Management System to Support High Transaction Rates from Numerous High-Speed Terminals**
- **Redundant File Management System with Symbolic File Access, File Security, File and Record Locking, Concurrent Input/Output and Disc Volume Interchangeability without Reprogramming**
- **Fail-Safe Message System with Checkpointing for Fault-Tolerant Programs, Process Control to Establish, Change, Suspend or Delete Processes, and Automatic Resource Allocation and Memory Mapping**
- **Fail-Safe Utility Procedures for Automatic Data Conversion, Time and Date Logging, and Calling the Debug Facility**
- **Efficient High-Level Transaction Application Language (TAL) Compiler for Easy Implementation of Application Programs**
- **Interactive Command Interpreter (COMINT), Text File Editor (EDIT), Object File Editor (UPDATE), and Debugger (DEBUG) for Fast and Economical Program Development**
- **On-Line Program-Concurrent Diagnostics for System Security and Program Integrity.**

INTRODUCTION

The GUARDIAN Operating System is a true multi-processor, multi-programming *fault tolerant* operating system. GUARDIAN has been specifically architectured so that applications can be defined and implemented to run continuously (even during hardware failures). Many of the responsibilities normally handled by applications programmers with other systems are taken care of automatically by GUARDIAN:

- The virtual memory scheme incorporated into GUARDIAN enables programmers to concentrate fully on the intended application. Programs or portions of programs are swapped between memory and disc automatically. This swapping is invisible to the applications program.
- The multiprogramming, multiprocessing features of GUARDIAN permit programs to be written without regard for other programs running in the system.
- Programmers can write programs that communicate with input/output devices and other programs without actually knowing to which processors these devices are connected or in which processors other programs are located.
- The hardware aspect of input/output transfers is transparent to application programmers. Interrupts and system dependent error conditions are handled automatically.
- Transfers over the interprocessor bus (DYNABUS)™ are handled automatically. The bus operation is completely invisible to application programmers.
- For application dependent error recovery routines, the system provides an error number that specifically describes any errors encountered during an input/output operation.

The *Fail-Safe* features of GUARDIAN make it uniquely suited for the fast development, easy expandability, and reliable operation of on-line transaction-oriented applications. With other systems, these functions were the responsibility of the application programmer.



Guardian Operating System

VIRTUAL MEMORY MANAGEMENT

Within each processor the operating system is responsible for allocation of execution time to multiple programs on a priority basis; allocation of buffer space and control blocks; process synchronization; fault and trap handling; and interval clock maintenance. In addition, GUARDIAN isolates the application from physical memory constraints by providing an efficient virtual memory management system. Paging hardware is provided in the form of four 128K byte memory maps (user code user data, system code and system data). All code is both shar able by multiple programs and non-modifiable, two features which reduce overlay and swapping overhead. In addition, hardware is used to record frequency of access to all memory pages and modification of data pages, thus providing a low overhead method of determining the correct page to be replaced.

FILE SYSTEM

Provided as part of GUARDIAN, the *File Manager* is a flexible, easy-to-use device-independent interface. It allows an application program to communicate with disc files, serial I/O devices, conversational page mode and multidrop terminals, and other application programs all through one standard set of routines. The File Manager allows program execution to continue concurrent with execution of the file request. In fact, the input/output may be taking place in a different processor module.

The File Manager also includes a number of features unique to the *NonStop* environment. When a processor or I/O port fails, the File Manager automatically reroutes subsequent requests to a back-up processor module. In addition, through a very efficient program-to-program communication mechanism, the File Manager provides the application with a simple method of keeping back-up programs informed of current operations so that a smooth transition may be made.

File Access – For I/O devices normally dedicated to a single process, such as Terminals or Line Printers, the device is assigned a *symbolic* name. The programmer need not know the physical address of the device. This provides a simple means to add and/or reconfigure I/O devices without reprogramming the application. In the case of disc devices, a file name represents a user-specified portion of the disc storage space. The File Manager makes no distinction between sequential and random access to a disc file. A file pointer (relative byte address) determines where a data transfer is to begin. The file pointer is normally automatically incremented for sequential access but may also be set explicitly for random access. An access mode specifies the operations to be performed on a file:

- **Read/Write (Default Mode)**
- **Read Only**
- **Write Only**

File Procedures – To implement file operations, calls are made to file management procedures. All files are accessed through the same set of procedures which provides a single uniform access method. These procedures include:

• **CREATE**	a new file
• **OPEN**	a file for access
• **READ**	data from a file
• **WRITE**	data to a file
• **WRITEREAD**	write/read data to/from a file
• **READUPDATE**	read data from a file for subsequent update
• **WRITEUPDATE**	write updated data to a file
• **CLOSE**	a file to access
• **PURGE**	a file from a disc

Numerous other procedures are provided for device-dependent operations.

File Security – The versatile File Manager provides the ability to limit file access to an individual, a group of individuals, or an individual within a group. This limitation is *password* protected. Four types of file operations (read, write, execute and purge) may be separately limited to either the individual (owner) who created the file the owner's group, or any individual within the group. A program uses an *exclusion mode* to limit file access. The exclusion modes are:

- **Shared Access (Default Mode)**
- **Exclusive Access**
- **Protected Access**

File *locking* is provided so that cooperative application programs may share file operations as a disc file. Also, disc volumes may be removed and replaced without reprogramming the application and without loss of file security.

MESSAGE MANAGEMENT SYSTEM

The GUARDIAN message system handles all communications between Tandem 16 processor modules, system processes and application programs. It frees the user of the responsibility of routing messages to the correct processor, verifying that it got there correctly, and deciding which program is to receive it in the destination processor. A program need not be aware of which of the 16 Tandem processors will ultimately run it. In fact, the same program may be executing simultaneously on all processors. In addition, a program may access any device on the system, even if the device is not physically connected to the processor in which the program is running. This allows the system to be expanded without reprogramming the application.

Process Control – A *process* is the execution of a program under control of the GUARDIAN Operating System. Process control procedures are used to interactively call processes. These processes are:

- **NEWPROCESS** – create a new process (run a program)
- **DELAY** – suspend the calling process for a specified interval of time
- **PRIORITY** – change the process execution priority
- **STOP** – delete a process with a normal indication
- **ABEND** – delete a process with an abnormal indication

For example, to have a process delete itself and close its files, the procedure CALL STOP is used.

System Messages – GUARDIAN sends messages directly to application processes to inform the application of certain system conditions. Some of these messages are:

- **Processor Module Failed**
- **Process Stopped Execution**
- **Processor Module Reloaded**

Certain critical error conditions prevent normal execution of a process. These errors cause traps to GUARDIAN Trap Handlers.

Checkpointing – Fail-safe, non-stop operating environments require one or more primary/backup *process pairs*. The primary process executes in one processor while the backup process monitors in another processor. With this type structure, the backup process is kept informed of the primary's execution state of the primary process via periodic *checkpoint* messages sent to the backup process. Each process in a process pair has the same set of files open. This ensures that the backup process has immediate access to the files in the event of a primary processor's failure. GUARDIAN provides several procedures to aid in the development of fault-tolerant programs:

- **CHECKOPEN** –
  is called by a primary process to open a file in its backup process
- **CHECKPOINT** –
  is called by a primary process to checkpoint its current state to its backup process
- **CHECKMONITOR** –
  is called by a backup process to monitor its primary and take appropriate action in the event of the primary's failure
- **CHECKSWITCH** –
  is called by a primary process to switch control to its backup process
- **CHECKCLOSE** –
  is called by a primary process to close a file in its backup process

COMMAND INTERPRETER (COMINT)

Tandem's COMINT is a high-level man-machine interface which allows the user to converse with the system. The user may obtain or alter the current operational status of the system; create, verify and purge disc files; and run programs (both Tandem-supplied and application programs). Normally, the user initially executes COMINT on a system console. From this point on, COMINT may be specified to be run on any other terminals connected to the system.

PROGRAM DEVELOPMENT AIDS

Included as part of the standard software provided with every Tandem 16 system is a comprehensive set of program development tools. These programs, which run under control of the Command Interpreter, allow the user to develop application programs with a minimum of effort. Included are a high-level compiler, a source file editor, object file editor and interactive debugging facility.

Transaction Application Language (T/TAL) Tandem's *Transaction Application Language* is a high-level, block structured language designed for the easy implementation of transaction-oriented applications. T/TAL provides many high level constructs, including IF THEN, FOR, DO UNTIL, WHILE and CASE. It allows the programmer to write self-documenting programs and eliminates the time consuming errors inherent in assembly language programming. Special string manipulation operations are included to facilitate fast processing of transaction data; among them are MOVE, COMPARE and SCAN strings. While providing all these flexible features, T/TAL does not sacrifice execution efficiency. A highly optimized compiler, it produces object programs as efficient as those written in an assembly language.

Edit – The *Text Editor* is a very flexible, interactive text file editor that can be used to prepare both program source files and documentation. EDIT can be run from either conversational or page-mode terminals, and provides an additional interface to allow it to be driven by other programs.

Update – The *Object File Editor* allows the user to make changes to previously compiled programs. In addition, the output of multiple compilations can be combined through the facilities of UPDATE.

Debug – An interactive program debugging facility supported by GUARDIAN enables the user to test application programs. It provides program breakpoints, tracing of variables, and access to all of the code and data of the program, all from an interactive terminal.

GUARDIAN OPERATING SYSTEM SOFTWARE

Overseeing Tandem 16 System operation is the Tandem 16 Guardian Operating System. Guardian provides the multi-processing (parallel processing in separate processor modules), multiprogramming (interleaved processing in one processor module), and NonStop capabilities of the Tandem 16 Computer System.

In a typical system, master copies of the Guardian Operating System, configured for the specific application, are kept in a "system" area on a disc volume. Critical and frequently used parts of Guardian are resident in each processor module's memory. As such, the system's capabilities are maintained even if a module fails . Non-critical or less frequently used parts of Guardian are virtual and are brought into a processor module's memory from disc only when needed.

Several functions of Guardian are transparent to application programs. These include:

- The preparation of a program for execution in virtual memory when a request is made to run a program.
- The capability for processes to communicate with each other regardless of the processor module where they are executing.
- Providing the virtual memory function by automatically bringing absent memory pages in from disc when needed.
- Scheduling processor module time among multiple processes according to their application-assigned priorities (a "process" is an executing program).

Guardian provides an additional and extremely important function. Concurrent with application program execution, Guardian continually checks the integrity of the system. This is accomplished as follows: Guardian in each processor module, at a predefined interval, transmits "I'm alive" messages to Guardian in every processor module (this interval is typically one second). Following this transmission, Guardian in each processor module checks for receipt of an "I'm alive" message from every other processor module. If Guardian in one processor module finds that a message has not been received from another processor module, it first verifies that it can transmit a message to its own processor module; if it can, it assumes that the non-transmitting processor module is inoperative; if it can't, it takes action to insure that its own module does not impair the operation of other processor modules. In either case, Guardian then informs system processes and interested application processes of the failure.

An important service provided by Guardian is file management. File management is the means by which application programs perform input/output operations in the Tandem 16 computer system. A "file" can be all or a portion of a disc pack, a non-disc device such as a terminal or line printer, a process (i.e., running program), or the operator console. Files are identified by symbolic "file names." This frees the programmer from needing to know the physical addresses of I/O devices and permits addition and reconfiguration of input/output devices without the need to rewrite or recompile programs.

File management operations are performed by calling procedures that are part of the operating system. All files are accessed through this same set of procedures, thereby providing a single, uniform access method. Additionally, the file management procedures are designed so as to eliminate the peculiarities of various devices.

An application program "sees" operating system services as a set of "library" procedures. The library procedures have names such as "READ," "WRITE," "OPEN," etc. To request an operating system service (e.g., input), a call to the appropriate operating system procedure is written in the application program (e.g., "READ"). (The operating system library procedures exist in the System Code area and therefore are shared by all processes.)

Another Guardian feature is "mirrored" disc volumes. A "mirrored" disc volume consists of a pair of physically independent disc devices. The purpose of a mirror volume is to insure that the information on a disc volume (i.e., operating system, programs, virtual-swapping-area, and application data) will be available even if one of the disc drives fail. When a write is made to a mirror volume, Guardian records the data to be written on the packs of both disc devices. As a result, both disc packs contain identical information. If one of the disc devices becomes inoperable, Guardian performs all reads and writes to the other operable disc volume. The operation of a mirror volume is entirely transparent to both application programs and system users.

Process control is another important service provided by Guardian. Through the process control functions, an application process can run and stop processes in any processor module in the system and can monitor the operation of any processor module or any process running in the system. If a module fails or a process stops, or if a failed module becomes operable, Guardian will notify the application process. Process control functions are invoked by making call to operating system procedures.

All Tandem System programs and operating system procedures are designed to operate with standard character-mode terminals, with or without communications modems. No special programming is required. Tandem also provides the ENVOY Data Communications Manager as part of its standard operating system. ENVOY supplies data communications services for asynchronous and binary synchronous communications networks in both point-to-point and multipoint networks. For remote job entry with batch data transmission, the Tandem EXCHANGE system can be used by an operator or programmatically. The EXCHANGE system is an extremely flexible 2780/3780 emulator software package that has capability beyond the classical IBM product.

SECURITY

The Guardian Security System is designed to fulfill three objectives:

1. Prevent inadvertent destruction of files through purging or overwriting.

2. Prevent unauthorized access to sensitive data files by programmers or operations personnel.

3. Prevent unauthorized interference with executing programs (processes).

Additionally, the Guardian Security System is designed so as not to interfere with application design in systems where security is not desired.

Additional security may be provided by the application program. Some examples of application program security checks are:

. Limitation of Capability at a Terminal

  It is not necessary to have a Guardian Command Interpreter executing at an application terminal. Therefore, the application program has control over what the terminal operator sees. The application program can limit the functions that the terminal operator can perform.

. Physical Security

  Programs which alter or produce reports of sensitive data may include routines which check the terminal from which they are run. This allows the application to restrict the running of the program to a specific terminal which is physically secure (e.g., in a locked office for which there is only one key).

. Special Devices

  These include authorization terminals such as badge readers, fingerprint readers, etc.

Individuals that have access to the system are called "users." In general, there are four classes of system users:

- Standard User

  A "standard" user is allowed to perform standard operations such as creating and purging disc files, running programs, displaying system status, etc. Additionally, a standard user is limited as to the processes it can stop or debug.

- Group Manager

  A "group manager" user is permitted to perform the standard operations as well as designate new system users.

- System Operator

  A "system operator" user is permitted to perform the standard operations as well as reload processor modules, set the system time-of-day clock, and alter the operating state of the interprocessor buses.

- Superid

  The "superid" user has total freedom to perform any operation in the system. This includes debugging privileged programs, accessing any file, logging on as any user without knowing the user's password, adding new groups to the security system, running privileged programs which have not been "licensed."

Additionally, for systems where security is not desired, all standard users can be defined as "null" users. In a system such as this, all users have equal access to all files in the system. However, such a system must still have a "superid" user and perhaps a "system operator" user so that their related functions can be performed. Another alternative is to have all users access the system as the "superid."

Before a user can access the system, the user must "log on." Log on is accomplished by supplying an application-predefined user name to the system by means of the Command Interpreter LOGON Command. The user name supplied to LOGON is of the form

⟨group name⟩ . ⟨user name⟩

⟨group name⟩ identifies an individual as a member of a group (e.g., a department).

⟨user name⟩ identifies the individual within the group.

GUARDIAN/EXPAND NETWORK SUBSYSTEM

The EXPAND NonStop Network is unique because it is an extension of an existing network operating system. Every Tandem/16 Computer System comprises from 2 to 16 separate central processors. Running in the NonStop mode requires constant communication among the processors. Consequently, the Guardian Operating System includes a sophisticated message handling system to control communications between processors and between processes (programs) running in one processor or running in separate processors. In effect, every Tandem/16 System is a local network controlled by the Guardian Operating System. The EXPAND NonStop Network expands the scope of the Guardian Operating System to allow communications among as many as 255 Tandem/16 systems.

The Guardian/EXPAND Network system, combined with the unique architecture of the Tandem/16 Computer System, provides network users with a number of features unequalled by other computer vendors:

- NonStop Nodes

  The fault-tolerant hardware and software of the Tandem/16 System eliminates the computer as a source of network failures.

  > Note: Individual Tandem/16 Systems within the network are called "nodes" to distinguish the computer system from the network system.

- A Distributed System

  The Guardian/EXPAND Network makes it possible to configure a network of Tandem/16 fault-tolerant computer systems so that a user of any node in the network can access the resources of any other node (processors, files, or physical devices) without regard for the physical location of the resource. To the user, the Guardian/EXPAND Network appears to be one large set of computer resources rather than a collection of separate systems.

- Dynamic Message Routing

  The Guardian/EXPAND Network constantly monitors the communications paths. When a transmission fails, the system retries until the transmission succeeds or until the system determines that the communication path has been broken. When the communication path has been broken, the Guardian/EXPAND system automatically reroutes the message via a different communications path.

- Best Path Message Routing

  The Guardian/EXPAND system monitors the communication lines and automatically selects the best path. The best path is the one that takes the least time. The system selects the fastest rather than the shortest path because this optimum end-to-end protocol can reduce communications costs. When a communications line fails, Guardian/EXPAND reroutes messages using the next fastest available path. When a new line is added, the system takes advantage of any new best paths created by the addition.

- Precisely Tailored Hardware

  Different nodes within a network typically have different computing requirements. The Tandem/16 System allows users to place exactly the right amount of computing power at each site. Even though the nodes within the network may range from a basic two processor system to a sixteen processor system with billions of bytes of online disc storage, the systems retain total compatibility of data, software, and application programs.

- Logical Growth

  The Tandem/16 architecture allows for incremental hardware expansion. A user can add memory, central processors, or peripheral devices as computing requirements grow. Similarly, the Guardian/EXPAND Network allows incremental expansion. Nodes can be added or removed and communication paths can be changed, all without reconfiguring existing systems.

  Notice that the Guardian/EXPAND Network can forestall the need for hardware expansion since the resources of every system in the network are accessible.

- Data Integrity

  The Guardian/EXPAND Network incorporates multiple safeguards to ensure that message packets are received correctly and that data cannot be lost in transmission.

- Human Engineering

  Because the EXPAND Network is an extension of the Guardian Operating System, the user interface to the network is through the Guardian Command Interpreter. For example, to run the text editor program on the local system, the user enters the command EDIT at his terminal. To run the program on a remote system,

the user simply enters the symbolic name of the remote system before the command: \OHIO EDIT. In effect, this command connects the user terminal with the remote OHIO system and starts the next editor program on that system. The only difference the user may notice in running on the remote system rather than the local system is that response time may be slightly longer since the communication lines cannot match the performance of the local processor.

- Simple Programming Interface

  The EXPAND Network relieves programmers of the need to deal with a cumbersome telecommunication access method. Since programs communicate with each other via Guardian's message system, the programmer uses the same commands to communicate with a program in the same processor, another processor, or another system.

Full appreciation of the EXPAND Network System requires an understanding of how tightly the EXPAND Network System is integrated with the Guardian Operating System and the Tandem/16's architecture.

INTRODUCTION

TANDEM HARDWARE

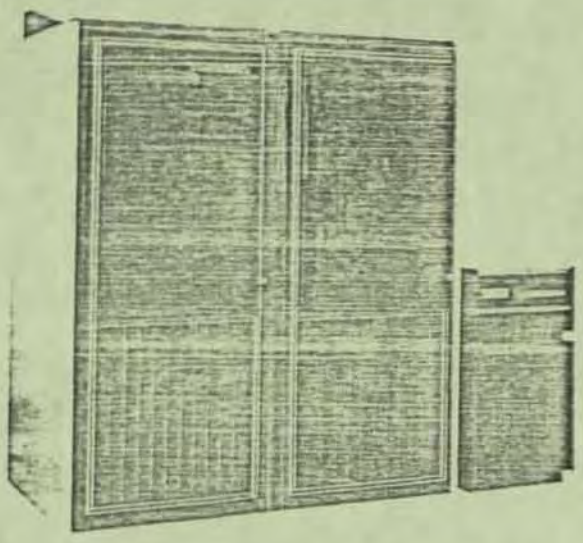
The Tandem 16 System represents a major departure from existing computer architecture. For the first time, a complete system has been designed to meet the growing demand for on-line transaction processing systems with failure tolerance capabilities. The basic design philosophy of the Tandem 16 Computer System is that no single module failure will stop or contaminate the system. Using standard Tandem hardware and software modules, the user may build a system to match necessary requirements exactly, both in throughput and reliability. No special or custom designed hardware or software is necessary. Equally important, the Tandem 16 System can grow to meet increasing demands with no change in operating system, applications software, or existing hardware. This growth is possible even without loss of the system during expansion.

One Tandem 16 processor module is a powerful computer. The Tandem 16 Computer System consists of two to sixteen processor modules. Processor modules are interfaced to one another by means of the two interprocessor buses. A processor module may interface to I/O devices by means of its input/output channel.

Each processor module contains the functions that normally comprise a complete computer system: central processing unit, memory and input/output channel. Therefore, each processor module is capable of operating independently of, and simultaneous with, all other processor modules in the system.

The interprocessor buses (which are always under control of the Tandem Operating System - GUARDIAN) are used to transfer data between the memories of processor modules.

Data is transferred between an input/output device (i.e., discs, terminals, line printers, etc.) and a processor module by means of an input/output channel. Each processor module has one I/O channel that is capable of communicating with up to 256 I/O devices. I/O devices are interfaced to I/O channels by dual-port controllers. Each dual-port controller is connected to the I/O channels of any two processor modules. Therefore, each I/O device can be controlled by either of two processor modules. (In actual practice, an I/O device is controlled exclusively by one processor module until a failure occurs such that the processor module can no longer communicate with the I/O device. If such a failure occurs, the other processor module takes control of the I/O device.)



TANDEM

NonStop™ STANDARD PACKAGED SYSTEMS

- NonStop™ Multi-Processor Systems for On-Line Transaction-Oriented Applications
- Autonomous/Dual DYNABUS™ for Inter-Processor Communications (13 Mega Bytes/second each)
- Fast (800 or 500 nSec) Core or Semiconductor Memory with Virtual Memory Control and Automatic/Dynamic Program Allocation
- High-Speed Redundant Dual-Port I/O Channels with Dedicated Microprocessed Interrupt and Block-Multiplexed DMA (2.5 or 4.0 Mega Bytes/Second)
- Versatile System Applications Support with Tandem/Guardian and Tandem/Transaction Applications Language (T/TAL)
- 100 ns cycle time Processors

SYSTEM CONFIGURATIONS

The Tandem NonStop systems (illustrated on the reverse side of this page) are packaged especially for critical on-line transaction-oriented applications where high reliability and low program overhead are essential. All systems feature dual processors with Tandem's unique DYNABUS for high-speed inter-processor communications, and redundant dual-port I/O channels for input/output resiliency. Each individual processor module contains two microprogrammed processors: one dedicated to programs and one dedicated to input/output control and data transfers. In addition, each central processor provides Power Fail/Auto Restart, an Interval Timer, Hardware Multiply/Divide, DMA, Dynamic Memory Mapping, Virtual Memory control, and a Bootstrap Loader.

TANDEM T16/212-1 SYSTEM

The Tandem T16/212-1 System consists of a dual processor package utilizing core memory modules. Each of the two processors contains 192K bytes of 800 nanosecond core memory with parity. The memory is arranged in 64K byte modules (32K words of 17 bits each). The T16/212-1 may be easily expanded in the field to 448K bytes of memory on each processor. The T16/212-1 I/O System provides high speed dual-port I/O Channels with block-multiplexed DMA. The data rate of each I/O Channel is 2.5 MBytes/second. The system cabinet provides 14 vacant slots for expansion of I/O Controllers.

TANDEM T16/244-1 SYSTEM

The Tandem T16/244-1 System consists of a dual processor package utilizing semiconductor (MOS) memory modules. Each of the two processors contain 192K bytes of 500 nanosecond semiconductor memory with error detection and correction. The memory is arranged in 96K byte modules (48K words of 22 bits each, 16 for data, 6 for ERCC). The T16/244-1 may easily be expanded in the field to contain two additional processors for a total of four processors and each processor may be expanded to 512K bytes of memory. Battery backup is supplied as a standard feature on semiconductor memory. The T16/244-1 I/O System provides high speed dual-port I/O Channels with block multiplexed DMA. The data rate of each I/O Channel is 4.0 MBytes/second. The System Cabinet provides 14 vacant slots for expansion of I/O Controllers.

SUBSYSTEM CONFIGURATIONS

All Tandem packaged systems are supported with a wide selection of peripheral subsystems to meet the heavy demands of diversified applications and high-volume data bases.

Disc Subsystems – Because disc requirements are highly application dependent the user will add disc controllers and drives as needed. The user can choose from a wide variety of disc controllers and drives including 10MB, 50MB and

TANDEM NonStop™ STANDARD PACKAGED SYSTEMS

160MB capacities. Through the use of appropriate controllers and software the user can configure redundant "mirrored" data bases for increased system integrity.

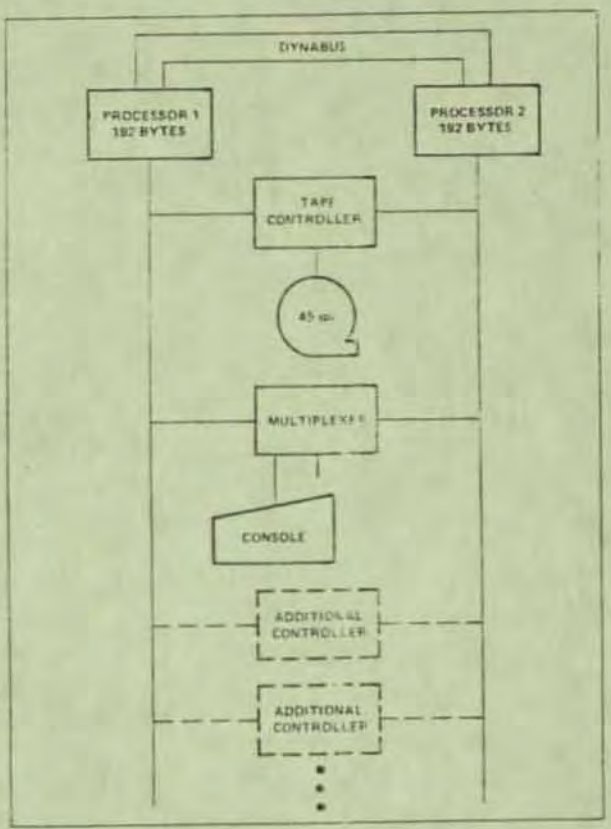
Tape Subsystems – A Magnetic Tape Drive and Controller make up the Tape Subsystem. The tape unit is a 45 ips, 9-track, 800/1600 Dual Density NRZI/PE BPI drive with read-after-write and power-fail/auto-restart electronics. The dual-port controller can handle up to two independently connected tape drives with no daisy chaining required. The Controller may be powered from either one of the dual processors.

Terminal Subsystem – The basic Terminal Subsystem consists of a Multiplexer and the system Console. The Multiplexer is comprised of an Asynchronous Controller. The dual-port controller provides two independent 50 to 19.2K baud communication lines for modem or direct-wire connection compatible with single- or multi-drop terminals. One line is dedicated to the system's 30 CPS, 132-column hard-copy console.

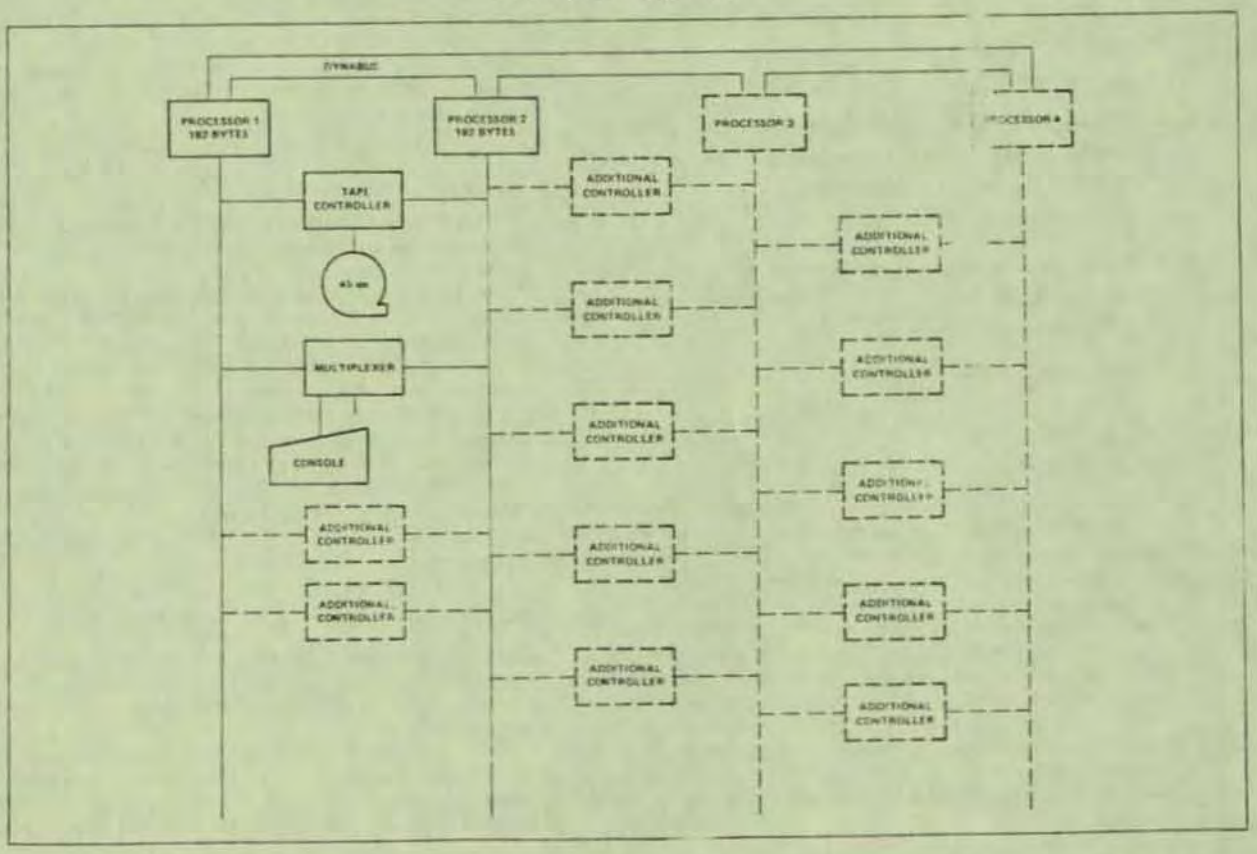
APPLICATIONS SUPPORT

Tandem packaged Non-Stop systems come with full applications program development support: the Tandem/GUARDIAN Operating System, Source File Editor (EDIT), Object File Editor (UPDATE), and an interactive Debugger (DEBUG). The Tandem/Transaction Applications Language (T/TAL) compiler allows programs to be written for NonStop applications in a high level language. This provides self documenting programs and eliminates the time consuming errors inherent in assembly language. In addition, utilities are provided for backup/restore, file creation, system generation, and system operation.

SYSTEMS T16/212-1



SYSTEMS T16/244-1



The following system modules may be added to TANDEM packaged systems. The modules may also be combined to configure systems other than the standard packaged systems if desired.

PRODUCT NUMBER	DESCRIPTION
	SEMICONDUCTOR MEMORY PROCESSORS
T16/1403	General Purpose Processor consisting of: Two (2) pipelined microprocessors, one for programs and one for I/O. Complete DMA only I/O system (4.0M bytes/sec). Virtual Memory control. Memory mapping and protection for up to 512K bytes of main memory. Hardware MPY/DIV. Power-fail/auto-restart. Bootstrap Loader. Interval Timer. Control Panel. Dual interprocessor message hardware. Provision for up to 32 I/O controllers. 122 instructions including string manipulation and double word arithmetic. 96K bytes (48K words) of semiconductor memory arranged as 22 bit words (16 data bits and 6 error detection/correction bits) *all* single bit errors are corrected and *all* double bit errors are detected. Memory cycle time of 500 nsecs. Battery Backup for semiconductor memory.
	CORE MEMORY PROCESSORS
T16/1102	Same as T16/1402 with the following exceptions: 1) Core memory (cycle time 800 nsecs) is used in place of semiconductor memory. Core memory utilizes one parity bit per two bytes. 2) The I/O channel speed is reduced to 2.5 Megabytes/sec, memory capacity is 64K bytes (32K words).
	MEMORY MODULES
T16/2102	Core Memory module, consists of a 64K byte memory plane with a read access time of 500 ns and a cycle time of 800 ns. The module is arranged as 32K words of 17 bits (16 data bits one parity bit) up to 7 of the modules may be controlled by a single processor.
T16/2402	Semiconductor memory module, consists of 64K byte memory module with a cycle time of 500 nsecs. The module is arranged as 32K words of 22 bits, 16 data bits and 6 error detection/correction bits. The module enables detection and correction of all single bit errors and detection of *all* double bit errors. Up to 8 of these modules may be controlled by a single processor.
T16/2403	Semiconductor memory module, consists of 96K byte memory module with a cycle time of 500 nsecs. The module is arranged as 48K words of 22 bits, 16 data bits and 6 error detection/correction bits. The module enables detection and correction of *all* single bit errors and detection of *all* double bit errors. Up to 5 of these modules may be controlled by a single processor.
	TERMINAL SUBSYSTEMS
T16/6202	Synchronous Controller, dual channel connected and may be powered from either processor to which it is connected. Will control from 1 to 4 synchronous communication lines, either point-to-point or multidrop. A single line can run up to 56K bps. The aggregate data rate for all four lines cannot exceed 160K bps. The controller performs all character translation (ASCII or EBCDIC), Block Check Character generation, and autopolling. Includes loop-back test module, down-line loading capability.
T16/6301	Asynchronous Controller, dual channel connected and may be powered from either processor to which it is connected. Will control up to two terminal lines either hard-wired or modem connected. Line speed is programmable from 50 to 19.2K bps. Accommodates up to two extensions (see T16/6302). Includes loop-back test cable.
T16/6302	Asynchronous Extension Board, provides additional control for 15 asynchronous lines. Speed of each line is programmable from 50 to 19.2K bps. Each line can be hard-wired or modem connected. Pre-requisite: T16/6301.
T16/6001	Console Subsystem, consists of: 1. Dual channel connected controller, T16/6301 with (1) port for a console device; 2. One (1) T16/6604 30 cps, 132 column hard-copy console, C/L connected.
T16/6603	Terminal, Hard Copy, 30 cps, 132 columns, includes 25' cable, RS232 interface.
T16/6604	Terminal, Hard Copy, same as T16/6603 except 20mA current loop interface.
T16/6401	Terminal, CRT, character mode, 24 lines x 80 characters, speed switch selectable 110-19.2K bps, local or modem attachment, includes 25' cable, RS232 interface.
T16/6402	Terminal, CRT, character mode, 24 lines x 80 characters, speed switch selectable 110-19.2K bps, includes 25' cable, 20mA current loop interface.
T16/6511	Terminal, CRT, character or page mode, local editing and function keys, 24 lines x 80 characters, protected, unprotected, full/half bright reverse, blinking, speed switch selectable 110-19.2K bps, 25' cable, RS232 interface, local or modem attachment.
T16/6512	Terminal, CRT, character or page mode, local editing and function keys, 24 lines x 80 characters, protected, unprotected, full/half bright, reverse, blinking, speed switch selectable 110-19.2K bps, 25' cable, 20mA current loop interface.
T16/6552	Terminal, CRT, same as T16/6511 except uses Polling Protocol for Multidrop Communications Lines.
	DISC SUBSYSTEMS
T16/3102	Disc Controller (Small Discs), dual channel connected, can be powered from either processor, can control 1 to 4 drives (any mix of T16/4101 or T16/4102), each with separate connection (signal & data cable for each drive), uses 2314 recording technique.

TANDEM T16 SYSTEM MODULES

PRODUCT NUMBER	DESCRIPTION
	DISC SUBSYSTEMS (Continued)
T16/3105	Disc Controller (Large Discs), dual channel connected, can be powered from either processor, can control 1 to 8 drives (T16/4103, 4104, 4105), microprocessor controlled with 4K RAM buffer, dual-write feature, uses [redacted]ied SMD recording technique.
T16/4101	Disc, Moving Head, 10 MB, pedestal mounted, one fixed platter (5.0 MB formatted), one removable platter (5.0 MB formatted, top loading), 30 ms average seek, 12.5 ms latency, 312KB transfer rate.
T16/4102	Disc Moving Head, pedestal mounted, uses one removable 11-high pack, 50 MB formatted, 30 ms average seek, 12.5 ms latency, 312KB transfer rate.
T16/4103	Disc, Moving Head, pedestal mounted, uses one removable 11-high pack, 160 MB formatted, 28 ms average seek, 8.35 ms latency, 806KB transfer rate.
T16/4104	Disc, Moving Head, pedestal mounted, uses one removable 11-high pack, 240 MB formatted, 28 ms average seek, 8.35 ms latency, 1.2MB transfer rate.
T16/4105	Disc, Moving Head, pedestal mounted, uses one removable 5-high pack, 64MB formatted, 30 ms average seek, 8.35 ms latency, 1.2MB transfer rate.
	MAGNETIC TAPE & PUNCHED CARD SUBSYSTEMS
T16/3201	Magnetic Tape Controller, dual channel connected, can be powered from either processor, can control up to two (2) magnetic tape drives (T16/5101 800 bpi), each separately connected (signal and data cable for each drive), tape speed can be selected at 45, 75 or 125 ips.
T16/3202	Magnetic Tape Controller, dual channel connected, can be powered from either processor, can control up to two (2) magnetic tape drives (any combination of T16/5101 at 800 bpi or T16/5103 at 1600 bpi) each separately connected (signal and data cable for each drive), tape speed can be selected at 45, 75 or 125 ips.
T16/5101	Magnetic Tape Drive, 45 ips, 9-track, 800 bpi, NRZI, includes cabinet and 25' cable.
T16/5103	Magnetic Tape Drive, 45 ips, 9-track, dual density (800 bpi NRZI and 1600 bpi P/E) which is operator switch selectable, includes cabinet and 25' cable.
T16/3305	Card Reader/Line Printer Controller, dual channel connected, can be powered from either processor, controls one (1) reader and one (1) line printer.
T16/5301	Card Reader, 600 cpm, std. 80 column cards.
	LINEPRINTER SUBSYSTEMS
T16/3302	Line Printer Controller, dual channel connected, can be powered from either processor, controls one (1) of the following: T16/5502/5503/5504/5505.
T16/5502	Line Printer, 132 columns, 300 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5503	Line Printer, 132 columns, 600 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5504	Line Printer, 132 columns, 900 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5505	Line Printer, 132 columns, 1500 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5508	Serial Printer, 132 columns, 91-160 lpm depending on number of characters printed, 96 character ASCII set, C/L connected to asynchronous multiplexer.
	GENERAL CONTROLLERS
T16/3401	Universal Interface, incorporates two (2) independent I/O channel connections. Each is capable of providing a complete path between processor and I/O devices. The controller may be powered from either of the processors to which it is connected; in the event of a power failure, the controller will automatically draw its power from the second processor. This controller will control either a line printer, card reader, or any other device having a 16 line parallel interface. The line drivers and receivers are TTL level for one device, differential for the other.

TANDEM

PATHWAY

NonStop™ TRANSACTION PROCESSING SYSTEM

PATHWAY **software gives users the only transaction processing system capable of NonStop™ operation, assuring data integrity and continuous availability. Specifically designed to take full advantage of unique multiprocessing capabilities of the TANDEM 16 computer, *PATHWAY* software eases the task of developing applications for the on-line transaction processing environment.**

With the *PATHWAY* system, programmers no longer need to be concerned about terminal characteristics when writing applications programs. TANDEM supplies all the programs, procedures and application structures necessary to get user-written applications up and running in less time.

PATHWAY **software features:**

- **The only NonStop Transaction Processing System assuring continuous availability and data integrity**
- **Impressive software development tools designed to significantly reduce the cost of applications development**
- **Application structures that ease the task of designing and maintaining programs**
- **Increased transaction throughput by utilizing the full advantages of the unique fault-tolerant multiprocessing capabilities of TANDEM**
- **True distributed processing through the TANDEM EXPAND Network, allowing applications to run in any CPU in any system, regardless of the physical location of terminals and data**
- **Access to multiple applications from the same terminal for increased flexibility**
- **All the necessary terminal handling software allowing the user to concentrate on application design**
- **On-line addition, modification or deletion of transaction types, screen characteristics, applications and terminals.**

NonStop Transaction Processing System

Designed to simplify the way TANDEM 16 users develop on-line transaction processing applications, the ***PATHWAY*** system is an important addition to the total TANDEM product line. ***PATHWAY*** software is a key development tool which assists in bringing up new applications faster and easier.

The ***PATHWAY*** software package supplies all the procedures, programs and application structures necessary to allow users to write single-threaded application program modules. Furthermore, the user-written application modules can be designed without concern for terminal characteristics and communications protocols.

TANDEM's ***PATHWAY*** software furnishes all the major components necessary to implement a NonStop Transaction Processing System: a terminal control process, a COBOL-like screen language, an application monitor, and an interactive screen definition facility.

An Impressive Terminal Control Process

All terminal oriented functions are isolated into TANDEM supplied Terminal Control Processes (TCPs). Each TCP interacts with one or more terminals, providing each terminal with a "center of control" where the overall processing flow of that terminal is supported.

The TCP performs four major application functions: terminal interface (multi-terminal I/O handler), field validation (data consistency checks), data mapping (data conversion and formatting), and transaction control (application scheduling and transaction flow).

Although a single TCP can control multiple terminals, each terminal is logically independent of the others. The TCP maintains distinct data areas and control information for each terminal, and several terminals may be performing the same or different application functions at any one time. The TCP automatically allocates the use of shared resources.

Users of ***PATHWAY*** software may run multiple TCPs for better distribution of available resources. In addition, users may start and stop TCPs on-line in response to changes in the transaction environment.

A Powerful, COBOL-Like Screen Formatting Language

The handling of each terminal is defined by the user in a high-level language known as Screen COBOL. With Screen COBOL (a subset of ANSI '74 COBOL with extensions optimized for screen handling), the user defines the screen formats, input and output data mapping, data validation and consistency checks, transaction routing and overall application control of the ***PATHWAY*** system. The Screen COBOL compiler produces an intermediate code file, which is executed by the Terminal Control Process.

A Screen COBOL program may be simple, defining one or two screen formats and handling a single application. Or, it may increase in complexity, defining many screen formats and handling many transaction types. However, a Screen COBOL program is written once for a single terminal type and may be executed multiple times to support multiple terminals of the same type.

A User-Controlled Application Monitor

Overall control of the ***PATHWAY*** system is carried out by the Application Monitor, a TANDEM-supplied program which is used to supervise and control all the working processes in the transaction processing system.

The Application Monitor is a multi-terminal control process which manages load sharing to eliminate bottlenecks. The Application Monitor performs on-line addition, modification or deletion of transaction types, screen formats and terminals — all under user control.

The Application Monitor is the first program that is run to bring up the transaction processing system. From that point, the Application Monitor controls the start-up of all other working processes of the system. The user first defines the Application Monitor parameters, characteristics of the terminals, terminal control processes, and user-written application processes of the system. The user can then start, stop, and alter the operation of the processes and devices. The Application Monitor reports error conditions in the system, and can display on command the status of the various system components.

Interactive Screen Builder

The ***PATHWAY*** Transaction Processing System includes an interactive screen builder facility that allows for the design of screen formats directly on the terminal screen. The screen builder program then generates the appropriate Screen COBOL source statements that describe the screen format. The new screen description may then be added to an existing Screen COBOL program — to add an extension to an existing application or to become the basis for a new Screen COBOL program.

The screen builder will also take an existing Screen COBOL screen description, display the screen, and allow the designer to make modifications directly at the terminal. This procedure allows the designer and the ultimate terminal user to prepare screens together, and further reduces the time required to bring an on-line application into production.

Writing Applications with PATHWAY Software

The ***PATHWAY*** system makes the task of writing on-line applications as simple as writing batch programs. Under control of the Screen COBOL program, the TCP passes a transaction record to a user-written application program which deals with the record very straightforwardly. It receives the record, processes it in a single-threaded manner (making accesses to the ENSCRIBE data base as required) and replies to the TCP. The application program then waits for the next request.

The transaction record sent to the application has already passed the data validation checks, and is independent of the type of terminal that originated the transaction. The applications can therefore handle several terminal types.

For ease of design and programming flexibility, these user applications may be written in COBOL, FORTRAN, MUMPS or T/TAL (TANDEM/Transaction Application Language.)

The ***PATHWAY*** Transaction Processing System allows multiple copies of an application to be run for higher throughput. In addition, ***PATHWAY*** software contains facilities to dynamically start and stop application modules in response to changes in the transaction environment.

PATHWAY software also allows many different application programs to be present in the same system — even accessible from the same terminal — allowing a highly modular approach to application design and providing phased, on-line application development.

TANDEM's ***PATHWAY*** Transaction Processing System can support terminals of several different types including the TANDEM 6520 multi-page display terminal and 3270-compatible terminals.

The goal of the ***PATHWAY*** software is to simplify the design and programming of applications used in on-line transaction processing environments, where predefined transactions originate at terminals to access and update the data base. The ***PATHWAY*** system eases the burden of developing these applications on a TANDEM computer by providing complete terminal handling and application monitoring software.

TANDEM

TANDEM COMPUTERS INCORPORATED, 19333 Vallco Parkway, Cupertino, CA 95014. Toll Free (800) 538-9360 or (408) 996-6000 in California. Offices throughout the United States, Canada, Europe and the United Kingdom. Distributors in Australia, Finland, Mexico and Venezuela.

DS107903

PATHWAY TRANSACTION PROCESSING SYSTEM

The PATHWAY Transaction Processing System combines a set of special Terminal Control Processes, a new screen formatting language, a user-controlled application monitor, and an interactive screen definition facility to provide for Tandem users' significant reductions in programming requirements and simplicity in development requirements for on-line transaction processing applications.

PATHWAY takes full advantage of Tandem's fault-tolerant capabilities to assure data integrity and continuous availability.

Other features include:

- the ability to access multiple applications from the same terminal
- the capability to perform on-line addition, modification or deletion of transaction types, screen definitions, applications and terminals

Until now, a programmer typically has had to spend a large portion of development time addressing terminal characteristics. PATHWAY software provides the necessary procedures, programs, and structures to relieve him of these tasks, thereby increasing productivity and also making on-line application development more accessible to a broader base of programmers. On-line applications become as easy to write as batch applications, opening up on-line transaction processing to a whole new world of users and applications.

Simplifying application design and programming even further, PATHWAY divides terminal control and file manipulation into separate programs, which means the user needs only be concerned with "single-threaded processing." The Tandem supplied software performs all necessary data checking and format validation.

With the PATHWAY system, all terminal oriented functions are isolated within the terminal control processes (TCPs) supplied by Tandem. Although a single TCP can control multiple terminals, each terminal is logically independent of the others, even to the extent of maintaining distinct data areas and control information.

Terminal handling programs in a PATHWAY system need only be written once for any single terminal type. They are defined by the user in Screen COBOL, a new, high-level language similar in form to COBOL.

With Screen COBOL, the user defines the screen formats, input and output data mapping, data validation, message routing and other functions, which are interpreted by the TCP through an intermediate code file. A Screen COBOL program may be executed multiple times to support multiple terminals of the same type.

The PATHWAY system also includes an application monitor program which supervises and controls all working processes. The application monitor, which is a multi-terminal control program which allows for load sharing, is the very first program to be executed, and is responsible for initiating the rest of the system. It permits the user to selectively start, stop, and alter the operation of processes and terminals, and request a report on the status of processes or error conditions.

An interactive screen definition facility in the PATHWAY system supports on-line design and modification of screen formats directly at the terminal. This feature maximizes flexibility and further reduces the time required to bring on-line applications into production.

For ease of design and programming flexibility, user application programs can be written in COBOL, FORTRAN, MUMPS or T/TAL (Tandem/Transaction Application Language). PATHWAY supports terminals of several different types, including the new Tandem 6520 and 6524 multi-block display terminals.

TANDEM

NonStop™ STANDARD PACKAGED SYSTEMS

- **NonStop™ Multi-Processor Systems for On-Line Transaction-Oriented Applications**
- **Autonomous/Dual DYNABUS™ for Inter-Processor Communications (13 Mega Bytes/second each)**
- **Fast (800 or 500 nSec) Core or Semiconductor Memory with Virtual Memory Control and Automatic/Dynamic Program Allocation**
- **High-Speed Redundant Dual-Port I/O Channels with Dedicated Microprocessed Interrupt and Block-Multiplexed DMA (2.5 or 4.0 Mega Bytes/Second)**
- **Versatile System Applications Support with Tandem/Guardian and Tandem/Transaction Applications Language (T/TAL)**
- **100 ns cycle time Processors**

SYSTEM CONFIGURATIONS

The Tandem NonStop systems (illustrated on the reverse side of this page) are packaged especially for critical on-line transaction-oriented applications where high reliability and low program overhead are essential. All systems feature dual processors with Tandem's unique DYNABUS for high-speed inter-processor communications, and redundant dual-port I/O channels for input/output resiliency. Each individual processor module contains two microprogrammed processors: one dedicated to programs and one dedicated to input/output control and data transfers. In addition, each central processor provides Power Fail/Auto Restart, an Interval Timer, Hardware Multiply/Divide, DMA, Dynamic Memory Mapping, Virtual Memory control, and a Bootstrap Loader.

TANDEM T16/212-1 SYSTEM

The Tandem T16/212-1 System consists of a dual processor package utilizing core memory modules. Each of the two processors contains 192K bytes of 800 nanosecond core memory with parity. The memory is arranged in 64K byte modules (32K words of 17 bits each). The T16/212-1 may be easily expanded in the field to 448K bytes of memory on each processor. The T16/212-1 I/O System provides high speed dual-port I/O Channels with block-multiplexed DMA. The data rate of each I/O Channel is 2.5 MBytes/second. The system cabinet provides 14 vacant slots for expansion of I/O Controllers.

TANDEM T16/244-1 SYSTEM

The Tandem T16/244-1 System consists of a dual processor package utilizing semiconductor (MOS) memory modules. Each of the two processors contain 192K bytes of 500 nanosecond semiconductor memory with error detection and correction. The memory is arranged in 96K byte modules (48K words of 22 bits each, 16 for data, 6 for ERCC). The T16/244-1 may easily be expanded in the field to contain two additional processors for a total of four processors and each processor may be expanded to 512K bytes of memory. Battery backup is supplied as a standard feature on semiconductor memory. The T16/244-1 I/O System provides high speed dual-port I/O Channels with block multiplexed DMA. The data rate of each I/O Channel is 4.0 MBytes/second. The System Cabinet provides 14 vacant slots for expansion of I/O Controllers.

SUBSYSTEM CONFIGURATIONS

All Tandem packaged systems are supported with a wide selection of peripheral subsystems to meet the heavy demands of diversified applications and high-volume data bases.

Disc Subsystems – Because disc requirements are highly application dependent the user will add disc controllers and drives as needed. The user can choose from a wide variety of disc controllers and drives including 10MB, 50MB and

TANDEM NonStop™ STANDARD PACKAGED SYSTEMS

TANDEM NonStop™ STANDARD PACKAGED SYSTEMS

160MB capacities. Through the use of appropriate controllers and software the user can configure redundant "mirrored" data bases for increased system integrity.

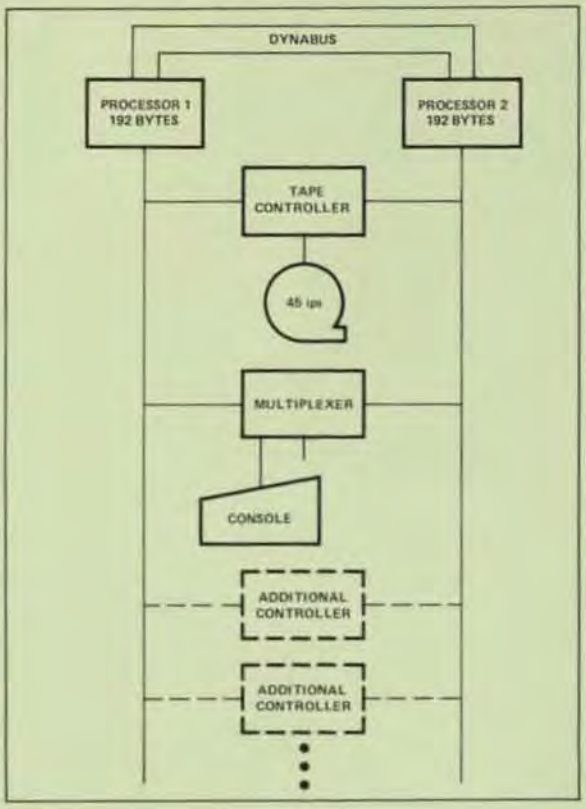
Tape Subsystems – A Magnetic Tape Drive and Controller make up the Tape Subsystem. The tape unit is a 45 ips, 9-track, 800/1600 Dual Density NRZI/PE BPI drive with read-after-write and power-fail/auto-restart electronics. The dual-port controller can handle up to two independently connected tape drives with no daisy chaining required. The Controller may be powered from either one of the dual processors.

Terminal Subsystem – The basic Terminal Subsystem consists of a Multiplexer and the system Console. The Multiplexer is comprised of an Asynchronous Controller. The dual-port controller provides two independent 50 to 19.2K baud communication lines for modem or direct-wire connection compatible with single- or multi-drop terminals. One line is dedicated to the system's 30 CPS, 132-column hard-copy console.

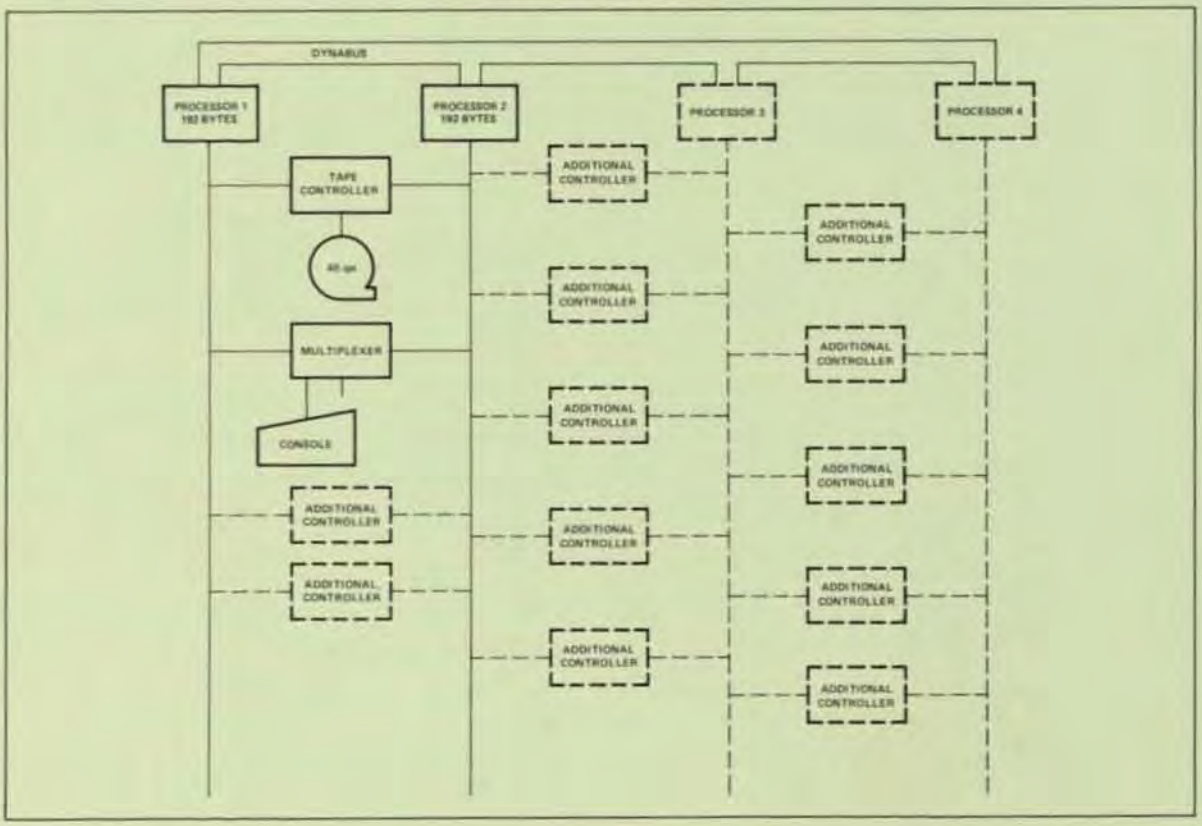
APPLICATIONS SUPPORT

Tandem packaged Non-Stop systems come with full applications program development support: the Tandem/GUARDIAN Operating System, Source File Editor (EDIT), Object File Editor (UPDATE), and an interactive Debugger (DEBUG). The Tandem/Transaction Applications Language (T/TAL) compiler allows programs to be written for NonStop applications in a high level language. This provides self documenting programs and eliminates the time consuming errors inherent in assembly language. In addition, utilities are provided for backup/restore, file creation, system generation, and system operation.

SYSTEMS T16/212-1



SYSTEMS T16/244-1



TANDEM COMPUTERS, INC. ■ 20605 Valley Green Dr. ■ Cupertino, Calif. 95014 ■ (408) 255-4800

TANDEM SYSTEM MODULES

The following system modules may be added to TANDEM packaged systems. The modules may also be combined to configure systems other than the standard packaged systems if desired.

PRODUCT NUMBER	DESCRIPTION
	SEMICONDUCTOR MEMORY PROCESSORS
T16/1403	General Purpose Processor consisting of: Two (2) pipelined microprocessors, one for programs and one for I/O. Complete DMA only I/O system (4.0M bytes/sec). Virtual Memory control. Memory mapping and protection for up to 512K bytes of main memory. Hardware MPY/DIV. Power-fail/auto-restart. Bootstrap Loader. Interval Timer. Control Panel. Dual interprocessor message hardware. Provision for up to 32 I/O controllers. 122 instructions including string manipulation and double word arithmetic. 96K bytes (48K words) of semiconductor memory arranged as 22 bit words (16 data bits and 6 error detection/correction bits) *all* single bit errors are corrected and *all* double bit errors are detected. Memory cycle time of 500 nsecs. Battery Backup for semiconductor memory.
	CORE MEMORY PROCESSORS
T16/1102	Same as T16/1402 with the following exceptions: 1) Core memory (cycle time 800 nsecs) is used in place of semiconductor memory. Core memory utilizes one parity bit per two bytes. 2) The I/O channel speed is reduced to 2.5 Megabytes/sec, memory capacity is 64K bytes (32K words).
	MEMORY MODULES
T16/2102	Core Memory module, consists of a 64K byte memory plane with a read access time of 500 ns and a cycle time of 800 ns. The module is arranged as 32K words of 17 bits (16 data bits one parity bit) up to 7 of the modules may be controlled by a single processor.
T16/2402	Semiconductor memory module, consists of 64K byte memory module with a cycle time of 500 nsecs. The module is arranged as 32K words of 22 bits, 16 data bits and 6 error detection/correction bits. The module enables detection and correction of all single bit errors and detection of *all* double bit errors. Up to 8 of these modules may be controlled by a single processor.
T16/2403	Semiconductor memory module, consists of 96K byte memory module with a cycle time of 500 nsecs. The module is arranged as 48K words of 22 bits, 16 data bits and 6 error detection/correction bits. The module enables detection and correction of *all* single bit errors and detection of *all* double bit errors. Up to 5 of these modules may be controlled by a single processor.
	TERMINAL SUBSYSTEMS
T16/6202	Synchronous Controller, dual channel connected and may be powered from either processor to which it is connected. Will control from 1 to 4 synchronous communication lines, either point-to-point or multidrop. A single line can run up to 56K bps. The aggregate data rate for all four lines cannot exceed 160K bps. The controller performs all character translation (ASCII or EBCDIC), Block Check Character generation, and autopolling. Includes loop-back test module, down-line loading capability.
T16/6301	Asynchronous Controller, dual channel connected and may be powered from either processor to which it is connected. Will control up to two terminal lines either hard-wired or modem connected. Line speed is programmable from 50 to 19.2K bps. Accommodates up to two extensions (see T16/6302). Includes loop-back test cable.
T16/6302	Asynchronous Extension Board, provides additional control for 15 asynchronous lines. Speed of each line is programmable from 50 to 19.2K bps. Each line can be hard-wired or modem connected. Pre-requisite: T16/6301.
T16/6001	Console Subystem, consists of: 1. Dual channel connected controller, T16/6301 with (1) port for a console device; 2. One (1) T16/6604 30 cps, 132 column hard-copy console, C/L connected.
T16/6603	Terminal, Hard Copy, 30 cps, 132 columns, includes 25' cable, RS232 interface.
T16/6604	Terminal, Hard Copy, same as T16/6603 except 20mA current loop interface.
T16/6401	Terminal, CRT, character mode, 24 lines x 80 characters, speed switch selectable 110-19.2K bps, local or modem attachment, includes 25' cable, RS232 interface.
T16/6402	Terminal, CRT, character mode, 24 lines x 80 characters, speed switch selectable 110-19.2K bps, includes 25' cable, 20mA current loop interface.
T16/6511	Terminal, CRT, character or page mode, local editing and function keys, 24 lines x 80 characters, protected, unprotected, full/half bright reverse, blinking, speed switch selectable 110-19.2K bps, 25' cable, RS232 interface, local or modem attachment.
T16/6512	Terminal, CRT, character or page mode, local editing and function keys, 24 lines x 80 characters, protected, unprotected, full/half bright, reverse, blinking, speed switch selectable 110-19.2K bps, 25' cable, 20mA current loop interface.
T16/6552	Terminal, CRT, same as T16/6511 except uses Polling Protocol for Multidrop Communications Lines.
	DISC SUBSYSTEMS
T16/3102	Disc Controller (Small Discs), dual channel connected, can be powered from either processor, can control 1 to 4 drives (any mix of T16/4101 or T16/4102), each with separate connection (signal & data cable for each drive), uses 2314 recording technique.

TANDEM SYSTEM MODULES

PRODUCT NUMBER	DESCRIPTION
	DISC SUBSYSTEMS (Continued)
T16/3105	Disc Controller (Large Discs), dual channel connected, can be powered from either processor, can control 1 to 8 drives (T16/4103, 4104, 4105), microprocessor controlled with 4K RAM buffer, dual-write feature, uses modified SMD recording technique.
T16/4101	Disc, Moving Head, 10 MB, pedestal mounted, one fixed platter (5.0 MB formatted), one removable platter (5.0 MB formatted, top loading), 30 ms average seek, 12.5 ms latency, 312KB transfer rate.
T16/4102	Disc Moving Head, pedestal mounted, uses one removable 11-high pack, 50 MB formatted, 30 ms average seek, 12.5 ms latency, 312KB transfer rate.
T16/4103	Disc, Moving Head, pedestal mounted, uses one removable 11-high pack, 160 MB formatted, 28 ms average seek, 8.35 ms latency, 806KB transfer rate.
T16/4104	Disc, Moving Head, pedestal mounted, uses one removable 11-high pack, 240 MB formatted, 28 ms average seek, 8.35 ms latency, 1.2MB transfer rate.
T16/4105	Disc, Moving Head, pedestal mounted, uses one removable 5-high pack, 64MB formatted, 30 ms average seek, 8.35 ms latency, 1.2MB transfer rate.
	MAGNETIC TAPE & PUNCHED CARD SUBSYSTEMS
T16/3201	Magnetic Tape Controller, dual channel connected, can be powered from either processor, can control up to two (2) magnetic tape drives (T16/5101 800 bpi), each separately connected (signal and data cable for each drive), tape speed can be selected at 45, 75 or 125 ips.
T16/3202	Magnetic Tape Controller, dual channel connected, can be powered from either processor, can control up to two (2) magnetic tape drives (any combination of T16/5101 at 800 bpi or T16/5103 at 1600 bpi) each separately connected (signal and data cable for each drive), tape speed can be selected at 45, 75 or 125 ips.
T16/5101	Magnetic Tape Drive, 45 ips, 9-track, 800 bpi, NRZI, includes cabinet and 25' cable.
T16/5103	Magnetic Tape Drive, 45 ips, 9-track, dual density (800 bpi NRZI and 1600 bpi P/E) which is operator switch selectable, includes cabinet and 25' cable.
T16/3305	Card Reader/Line Printer Controller, dual channel connected, can be powered from either processor, controls one (1) reader and one (1) line printer.
T16/5301	Card Reader, 600 cpm, std. 80 column cards.
	LINEPRINTER SUBSYSTEMS
T16/3302	Line Printer Controller, dual channel connected, can be powered from either processor, controls one (1) of the following: T16/5502/5503/5504/5505.
T16/5502	Line Printer, 132 columns, 300 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5503	Line Printer, 132 columns, 600 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5504	Line Printer, 132 columns, 900 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5505	Line Printer, 132 columns, 1500 lpm drum printer, 64 character ASCII set, 12-channel VFU, 25' cable.
T16/5508	Serial Printer, 132 columns, 91-160 lpm depending on number of characters printed, 96 character ASCII set, C/L connected to asynchronous multiplexer.
	GENERAL CONTROLLERS
T16/3401	Universal Interface, incorporates two (2) independent I/O channel connections. Each is capable of providing a complete path between processor and I/O devices. The controller may be powered from either of the processors to which it is connected; in the event of a power failure, the controller will automatically draw its power from the second processor. This controller will control either a line printer, card reader, or any other device having a 16 line parallel interface. The line drivers and receivers are TTL level for one device, differential for the other.

TANDEM

GUARDIAN OPERATING SYSTEM

FEATURES

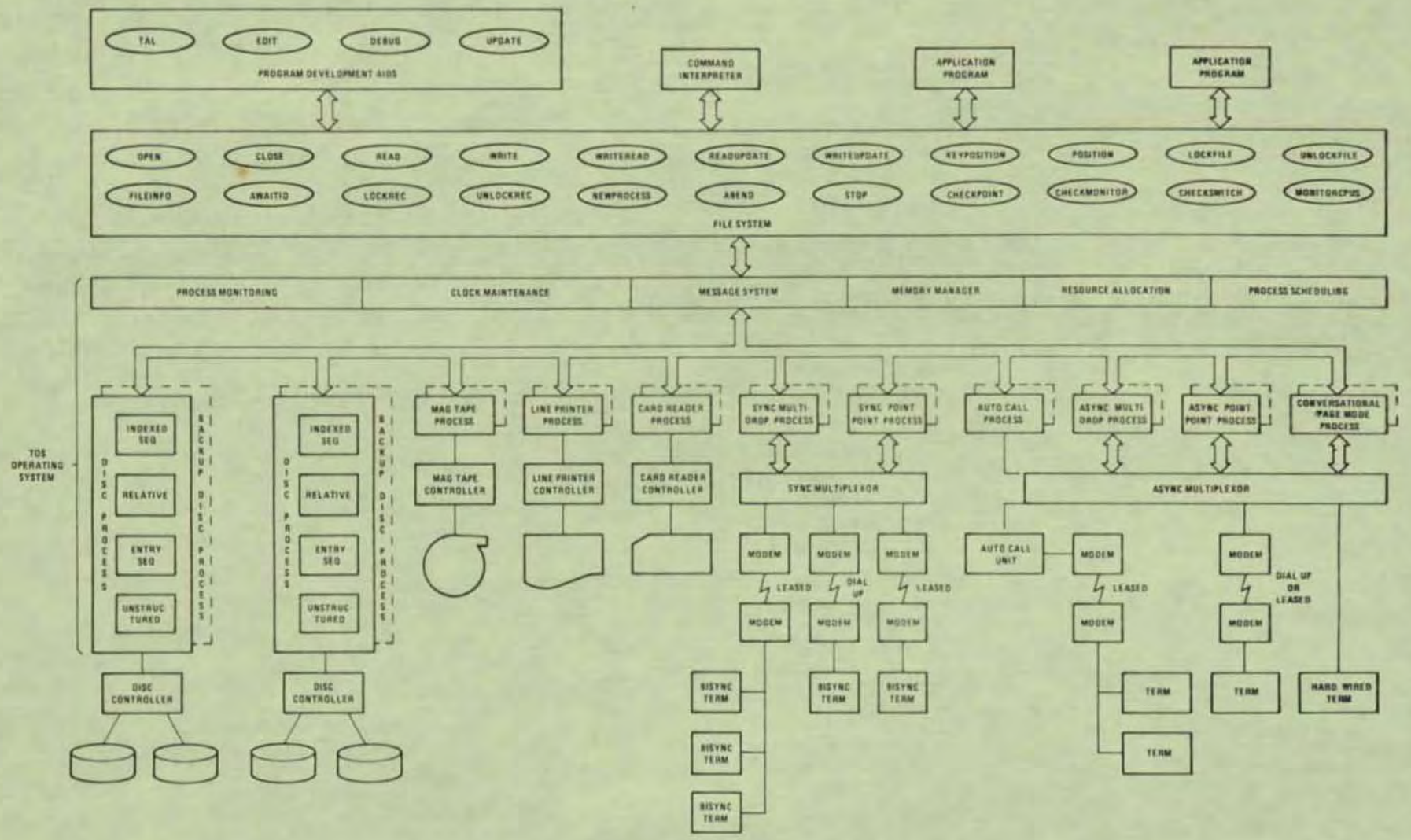
- **Fail-Safe GUARDIAN Operating System for Continuous Non-Stop™ Running of Transaction-Oriented Applications**
- **Multiprogramming/Multiprocessing Virtual Memory Management System to Support High Transaction Rates from Numerous High-Speed Terminals**
- **Redundant File Management System with Symbolic File Access, File Security, File and Record Locking, Concurrent Input/Output and Disc Volume Interchangeability without Reprogramming**
- **Fail-Safe Message System with Checkpointing for Fault-Tolerant Programs, Process Control to Establish, Change, Suspend or Delete Processes, and Automatic Resource Allocation and Memory Mapping**
- **Fail-Safe Utility Procedures for Automatic Data Conversion, Time and Date Logging, and Calling the Debug Facility**
- **Efficient High-Level Transaction Application Language (TAL) Compiler for Easy Implementation of Application Programs**
- **Interactive Command Interpreter (COMINT), Text File Editor (EDIT), Object File Editor (UPDATE), and Debugger (DEBUG) for Fast and Economical Program Development**
- **On-Line Program-Concurrent Diagnostics for System Security and Program Integrity.**

INTRODUCTION

The GUARDIAN Operating System is a true multi-processor, multi-programming *fault tolerant* operating system. GUARDIAN has been specifically architectured so that applications can be defined and implemented to run continuously (even during hardware failures). Many of the responsibilities normally handled by applications programmers with other systems are taken care of automatically by GUARDIAN:

- The virtual memory scheme incorporated into GUARDIAN enables programmers to concentrate fully on the intended application. Programs or portions of programs are swapped between memory and disc automatically. This swapping is invisible to the applications program.
- The multiprogramming, multiprocessing features of GUARDIAN permit programs to be written without regard for other programs running in the system.
- Programmers can write programs that communicate with input/output devices and other programs without actually knowing to which processors these devices are connected or in which processors other programs are located.
- The hardware aspect of input/output transfers is transparent to application programmers. Interrupts and system dependent error conditions are handled automatically.
- Transfers over the interprocessor bus (DYNABUS)™ are handled automatically. The bus operation is completely invisible to application programmers.
- For application dependent error recovery routines, the system provides an error number that specifically describes any errors encountered during an input/output operation.

The *Fail-Safe* features of GUARDIAN make it uniquely suited for the fast development, easy expandability, and reliable operation of on-line transaction-oriented applications. With other systems, these functions were the responsibility of the application programmer.



Guardian Operating System

VIRTUAL MEMORY MANAGEMENT

Within each processor the operating system is responsible for allocation of execution time to multiple programs on a priority basis; allocation of buffer space and control blocks; process synchronization; fault and trap handling; and interval clock maintenance. In addition, GUARDIAN isolates the application from physical memory constraints by providing an efficient virtual memory management system. Paging hardware is provided in the form of four 128K byte memory maps (user code user data, system code and system data). All code is both shar able by multiple programs and non-modifiable, two features which reduce overlay and swapping overhead. In addition, hardware is used to record frequency of access to all memory pages and modification of data pages, thus providing a low overhead method of determining the correct page to be replaced.

FILE SYSTEM

Provided as part of GUARDIAN, the *File Manager* is a flexible, easy-to-use device-independent interface. It allows an application program to communicate with disc files, serial I/O devices, conversational page mode and multidrop terminals, and other application programs all through one standard set of routines. The File Manager allows program execution to continue concurrent with execution of the file request. In fact, the input/output may be taking place in a different processor module.

The File Manager also includes a number of features unique to the *NonStop* environment. When a processor or I/O port fails, the File Manager automatically reroutes subsequent requests to a back-up processor module. In addition, through a very efficient program-to-program communication mechanism, the File Manager provides the application with a simple method of keeping back-up programs informed of current operations so that a smooth transition may be made.

File Access – For I/O devices normally dedicated to a single process, such as Terminals or Line Printers, the device is assigned a *symbolic* name. The programmer need not know the physical address of the device. This provides a simple means to add and/or reconfigure I/O devices without reprogramming the application. In the case of disc devices, a file name represents a user-specified portion of the disc storage space. The File Manager makes no distinction between sequential and random access to a disc file. A file pointer (relative byte address) determines where a data transfer is to begin. The file pointer is normally automatically incremented for sequential access but may also be set explicitly for random access. An access mode specifies the operations to be performed on a file:

- **Read/Write (Default Mode)**
- **Read Only**
- **Write Only**

File Procedures – To implement file operations, calls are made to file management procedures. All files are accessed through the same set of procedures which provides a single uniform access method. These procedures include:

• **CREATE**	a new file
• **OPEN**	a file for access
• **READ**	data from a file
• **WRITE**	data to a file
• **WRITEREAD**	write/read data to/from a file
• **READUPDATE**	read data from a file for subsequent update
• **WRITEUPDATE**	write updated data to a file
• **CLOSE**	a file to access
• **PURGE**	a file from a disc

Numerous other procedures are provided for device-dependent operations.

File Security – The versatile File Manager provides the ability to limit file access to an individual, a group of individuals, or an individual within a group. This limitation is *password* protected. Four types of file operations (read, write, execute and purge) may be separately limited to either the individual (owner) who created the file the owner's group, or any individual within the group. A program uses an *exclusion mode* to limit file access. The exclusion modes are:

- **Shared Access (Default Mode)**
- **Exclusive Access**
- **Protected Access**

File *locking* is provided so that cooperative application programs may share file operations as a disc file. Also, disc volumes may be removed and replaced without reprogramming the application and without loss of file security.

MESSAGE MANAGEMENT SYSTEM

The GUARDIAN message system handles all communications between Tandem 16 processor modules, system processes and application programs. It frees the user of the responsibility of routing messages to the correct processor, verifying that it got there correctly, and deciding which program is to receive it in the destination processor. A program need not be aware of which of the 16 Tandem processors will ultimately run it. In fact, the same program may be executing simultaneously on all processors. In addition, a program may access any device on the system, even if the device is not physically connected to the processor in which the program is running. This allows the system to be expanded without reprogramming the application.

TANDEM GUARDIAN OPERATING SYSTEM

Process Control – A *process* is the execution of a program under control of the GUARDIAN Operating System. Process control procedures are used to interactively call processes. These processes are:

- **NEWPROCESS** – create a new process (run a program)
- **DELAY** – suspend the calling process for a specified interval of time
- **PRIORITY** – change the process execution priority
- **STOP** – delete a process with a normal indication
- **ABEND** – delete a process with an abnormal indication

For example, to have a process delete itself and close its files, the procedure CALL STOP is used.

System Messages – GUARDIAN sends messages directly to application processes to inform the application of certain system conditions. Some of these messages are:

- **Processor Module Failed**
- **Process Stopped Execution**
- **Processor Module Reloaded**

Certain critical error conditions prevent normal execution of a process. These errors cause traps to GUARDIAN Trap Handlers.

Checkpointing – Fail-safe, non-stop operating environments require one or more primary/backup *process pairs*. The primary process executes in one processor while the backup process monitors in another processor. With this type structure, the backup process is kept informed of the primary's execution state of the primary process via periodic *checkpoint* messages sent to the backup process. Each process in a process pair has the same set of files open. This ensures that the backup process has immediate access to the files in the event of a primary processor's failure. GUARDIAN provides several procedures to aid in the development of fault-tolerant programs:

- **CHECKOPEN** –
  is called by a primary process to open a file in its backup process
- **CHECKPOINT** –
  is called by a primary process to checkpoint its current state to its backup process
- **CHECKMONITOR** –
  is called by a backup process to monitor its primary and take appropriate action in the event of the primary's failure
- **CHECKSWITCH** –
  is called by a primary process to switch control to its backup process
- **CHECKCLOSE** –
  is called by a primary process to close a file in its backup process

COMMAND INTERPRETER (COMINT)

Tandem's COMINT is a high-level man-machine interface which allows the user to converse with the system. The user may obtain or alter the current operational status of the system; create, verify and purge disc files; and run programs (both Tandem-supplied and application programs). Normally, the user initially executes COMINT on a system console. From this point on, COMINT may be specified to be run on any other terminals connected to the system.

PROGRAM DEVELOPMENT AIDS

Included as part of the standard software provided with every Tandem 16 system is a comprehensive set of program development tools. These programs, which run under control of the Command Interpreter, allow the user to develop application programs with a minimum of effort. Included are a high-level compiler, a source file editor, object file editor and interactive debugging facility.

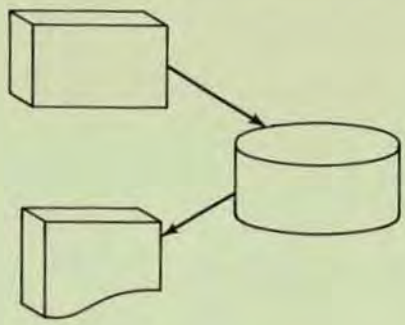
Transaction Application Language (T/TAL) Tandem's *Transaction Application Language* is a high-level, block structured language designed for the easy implementation of transaction-oriented applications. T/TAL provides many high level constructs, including IF THEN, FOR, DO UNTIL, WHILE and CASE. It allows the programmer to write self-documenting programs and eliminates the time consuming errors inherent in assembly language programming. Special string manipulation operations are included to facilitate fast processing of transaction data; among them are MOVE, COMPARE and SCAN strings. While providing all these flexible features, T/TAL does not sacrifice execution efficiency. A highly optimized compiler, it produces object programs as efficient as those written in an assembly language.

Edit – The *Text Editor* is a very flexible, interactive text file editor that can be used to prepare both program source files and documentation. EDIT can be run from either conversational or page-mode terminals, and provides an additional interface to allow it to be driven by other programs.

Update – The *Object File Editor* allows the user to make changes to previously compiled programs. In addition, the output of multiple compilations can be combined through the facilities of UPDATE.

Debug – An interactive program debugging facility supported by GUARDIAN enables the user to test application programs. It provides program breakpoints, tracing of variables, and access to all of the code and data of the program, all from an interactive terminal.

TANDEM COMPUTERS, INC. ■ 20605 Valley Green Dr. ■ Cupertino, Calif. 95014 ■ (408) 255-4800



TANDEM SPOOLER

- **SPOOLER runs NonStop(tm)**
- **User can supply own Print Processes**
- **SPOOLCOM allows operator inspection and/or alteration of job parameters**
- **Routing structure permits individual or broadcast locations**
- **SPOOLER library procedures allow blocking and compression**
- **Multiple files can be spooled from one application**
- **SPOOLER parameters can be specified programmatically**
- **Forms Alignment**

INTRODUCTION

The Tandem SPOOLER provides a means of storing application output in holding areas for later retrieval. Output may be passed to other processes or printed on one or more devices.

The SPOOLER is actually many processes working in unison to provide spooling facilities. These processes include SPOOLER Supervisor, SPOOLER Collectors, Print Processes, and SPOOLCOM.

SPOOLER Supervisor functions as the SPOOLER monitor and communicates with the other SPOOLER processes to determine which tasks to perform or schedule. SPOOLER Control is actually a server process interfacing with 1) SPOOLCOM, 2) applications calling SPOOLERCOMMAND or SPOOLERSTATUS procedures, 3) SPOOLER Collection Processes, and 4) SPOOL Print Processes.

SPOOLER Collectors accept output from application processes and store it on disc. There can be one or more SPOOLER Collectors.

Print Processes retrieve spooled data and print it. The Tandem supplied print processes are capable of handling multiple jobs and devices. Users may supply their own Print Processes.

SPOOLCOM is an operator/user interface with the SPOOLER subsystem. It can be run interactively on a terminal or can be passed commands from an application process. SPOOLCOM performs such functions as downing a device or ordering extra copies of a report.

APPLICATION PROCESS INTERFACE

At the simplest level an application process can open the SPOOLER one or more times to perform spooled output. The standard file management procedures WRITE, CONTROL, and SETMODE are used.

The application may also use the SPOOLER library procedures to implement more advanced features of the SPOOLER. The library procedures are:

SPOOLSTART
— start a job and specify spooling attributes

SPOOLWRITE
— write a print line

SPOOLEND
— end the spool job

SPOOLCONTROL
— control functions

SPOOLSETMODE
— setmode functions

Through SPOOLSTART the application can specify location, form name, priority of printing, number of copies, report name, and hold before/after printing.

TANDEM SPOOLER

TANDEM SPOOLER

SPOOLCOM

SPOOLCOM is used to initiate and control the operation of the spooling system by accepting commands interactively from an operator at a terminal or programmatically from a user application process. SPOOLCOM commands include:

DEV
- controls devices

JOB
- control jobs

PRINT
- control Print Processes

COLLECT
- control collection processes

LOC
- set up and modify destination structure

SPOOLER
- control the Spooling System

HELP
- list SPOOLCOM Commands

EXIT
- terminate SPOOLCOM

SPOOLERCOMMAND
- issue a control command

SPOOLERSTATUS
- retrieve status information

SPOOLERREQUEST
- Request information for a specific job necessary to start printing.

FC
- fix command

COMMENT

USER-WRITTEN PRINT PROCESSES

The Print Process library procedures allow user-written processes for devices not supported by SPOOLER or the retrieval of spooled data by an application process. The procedures are:

PRINTINIT
- Initialize communication with the SPOOLER control process

PRINTCOMPLETE
- Accepts messages from the SPOOLER control process

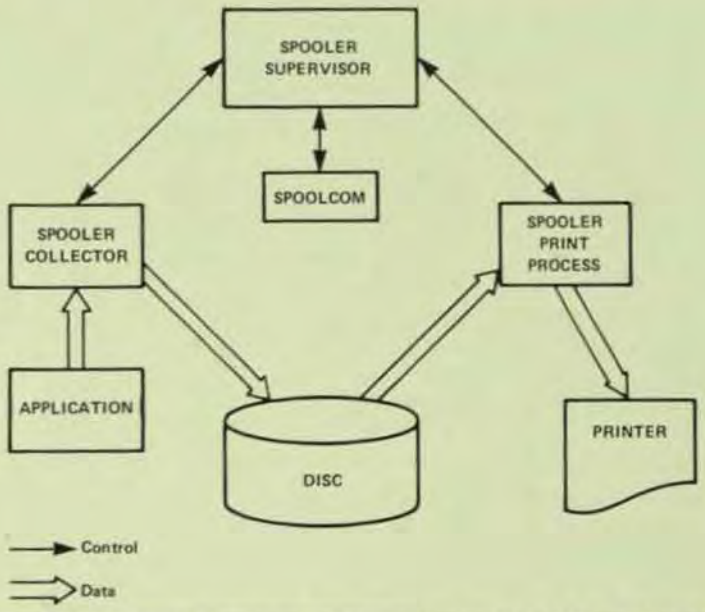
PRINTREADCOMMAND
- Interprets control messages from the control process

PRINTSTART
- Initialize data required to print a job

PRINTREAD
- Read a line of spooled data

PRINTINFO
- Obtain information on an active job

PRINTSTATUS
- Inform SPOOLER control process of an event such as end of file or error on the device.



TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000 T16/8743-00

TANDEM

DATA DEFINITION LANGUAGE

FEATURES

- **Centralized Data Base Provides Common Resource for Numerous Application Programs, Ensures Data Consistency, Reduces Redundant Data, and Allows Easy Maintainability without Sacrificing System Security**
- **Automatic Space Management Allocates and Optimizes Storage Resources Enhancing File and Record Activities**
- **Data Definition Language (DDL) to Facilitate the Definition of a Centralized Data Base Schema**
- **Easy-to-Use EDIT and SCHEMA Programs to Facilitate Compilation of Application Programs with High-Level Language TAL Compiler and ENSCRIBE Data Base Record Manager Procedures**

INTRODUCTION

Tandem's *Data Definition Language* allows centralized administration of a data base to accommodate any number of diversified application programs. And the applications programmer need not know where files associated with a specific application are located. Since the data base is described as a *schema,* and all data field references are defined in the schema, changes to record layout, and additions and/or changes to record types and/or fields may be accomplished *without* code modifications to existing programs. The schema also provides the necessary information for query and reporting programs. The *Data Definition Language* (DDL) is used to describe the schema.

DATA BASE DEFINITION

As illustrated on the next page, the programmer defines a data base in a *Schema Definition File* using the Tandem-supplied EDIT program. This example is used to define records for a customer data base. The *name* assigned to the record type is CUSTOMER. The fields are named ACCTNO, NAME, FIRST, STREET, CITY, STATE, ZIP, BAL. CHARACTER means that the field contains a string of ASCII characters and the number defines the length of each field in bytes. BINARY indicates arithmetic data. A 0 following KEYTAG means that the field ACCTNO is the record's primary key and "NM" means that the field NAME is an alternate key (other fields are not defined as key fields).

APPLICATION PROGRAM COMPILATION

The *Schema Definition File* is used as the input to the Tandem-supplied SCHEMA program. This program generates a T/TAL library file that, when compiled along with the application program, produces an object application program that is tailored to the particular data base. (See example on the next page).

DATA BASE ACCESS

Record types are accessed through ENSCRIBE provided with the GUARDIAN Operating System. Fields in a record type are accessed by *field identifiers.* As shown in the examples, a field identifier is a concatenation of the name of the record and the name of the field. For example, the field identifiers for the CUSTOMER record defined in the example are:

<field name>	<field identifier>
ACCTNO	CUSTOMER ^ ACCTNO
NAME	CUSTOMER ^ NAME
STATE	CUSTOMER ^ STATE

Each field defined as type CHARACTER also has a corresponding *field length identifier* that can be used to determine the length in bytes of a field. A *field length identifier* is the concatenation of the *record name,* the *field name,* and the string LEN. For example, the *field length identifier* for the NAME field is:

CUSTOMER^NAME^LEN

CUSTOMER^NAME^LEN has the value "25" when referenced in the program.

Each field which was defined to have a key tag will have a *key tag identifier* that can be used to pass the key tag to the file system. The form of this identifier is a concatenation of the *field identifier* and KEY. For example, the *key tag identifier* for the field NAME is:

CUSTOMER^NAME^KEY

CUSTOMER^NAME^KEY has the value of "null" when referenced in a program.

Defining the Data Base

```
* SCHEMA DEFINITION
*
* CONSISTS OF TWO TYPES OF RECORDS
*       1. CUSTOMER MASTER RECORD
*       2. TRANSACTION LOG RECORD
*
RECORD CUSTOMER.
       03 ACCTNO; TYPE CHARACTER  5; KEYTAG 0.
       03 NAME:   TYPE CHARACTER 25; KEYTAG "NM".
       03 FIRST:  TYPE CHARACTER 25.
       03 STREET; TYPE CHARACTER 56.
       03 CITY:   TYPE CHARACTER 25.
       03 STATE;  TYPE CHARACTER 25.
       03 ZIP;    TYPE CHARACTER  5.
       03 BAL:    TYPE BINARY 64,2.
*
RECORD TRANSACT.
       03 ACCTNO; TYPE CHARACTER  5; KEYTAG 0.
       03 NAME;   TYPE CHARACTER 25; KEYTAG "NM".
       03 DATE;                      KEYTAG "DT".
          05 YEAR;    TYPE CHARACTER 2.
          05 MONTH:   TYPE CHARACTER 2.
          05 DAY;     TYPE CHARACTER 2.
       03 CODE:   TYPE CHARACTER  1.
*                  "+" = TRANSACTION IS DEPOSIT
*                  "-" = TRANSACTION IS WITHDRAWL
       03 AMT;    TYPE BINARY 64,2.
       03 TIME;                      KEYTAG "TM".
          05 HOUR:    TYPE CHARACTER 2.
          05 MINUTE:  TYPE CHARACTER 2.
```

Compiling the Schema

```
SCHEMA /IN DDLFILE, OUT $LP/ APPLIB
```

Example: Accessing the Data Base

```
Application Global Declarations
        .
        .
! Including the schema source in the application program

?SOURCE SCHEMA(CUSTOMER,TRANSACT)
ALLOCATE^CUSTOMER;
ALLOCATE^TRANSACT;

PROC USER^APPLICATION;
        .
        .
! Using the Schema Definitions

 CUSTOMER^ACCTNO ':=' FORMOPEN^ACCT FOR CUSTOMER^ACCTNO^LEN;
 CUSTOMER^NAME   ':=' FORMOPEN^NAME FOR CUSTOMER^NAME^LEN;
 CUSTOMER^FIRST  ':=' FORMOPEN^FIRST FOR CUSTOMER^FIRST^LEN;
 CUSTOMER^STREET ':=' FORMOPEN^STREET FOR CUSTOMER^STREET^LEN;
 CUSTOMER^CITY   ':=' FORMOPEN^CITY FOR CUSTOMER^CITY^LEN;
 CUSTOMER^STATE  ':=' FORMOPEN^STATE FOR CUSTOMER^STATE^LEN;
 CUSTOMER^ZIP    ':=' FORMOPEN^ZIP FOR CUSTOMER^ZIP^LEN;
        .
        .
! Insert a new Customer Record into the File
 CALL WRITE(CUST,CUSTOMER,CUSTOMER^LEN,CNT);
        .
        .
        .
! Position to Customer Record for Subsequent Update
 CALL KEYPOSITION(CUST,FORMDEF^ACCT,,,EXACT);
 IF < THEN BEGIN ...
        .
        .
        .
! Get Customer Record and Lock It
 CALL READUPDATELOCK(CUST,CUSTOMER,CUSTOMER^LEN,CNT);
 IF < THEN BEGIN
          CALL FILEINFO(CUST,FILE^ERR);
          IF FILE^ERR=1]
          THEN BEGIN ! RECORD NOT FOUND
                  .
                  .
```

TANDEM

ENSCRIBE™

FEATURES

- Fail Safe File and Record Management for Non-Stop™ Transaction-Oriented Applications
- Multiple Disc File Organizations (Key-Sequenced, Relative and Entry-Sequenced) with Exact, Generic or Approximate Multi-Key Access to All Records
- Automatic Management and Maintenance of Multi-Volume Files
- Mirror Volumes to Provide Fail-Safe Data Base Protection
- Data and Index Compression for Key-Sequenced Files to Optimize Disc Storage Space
- Main Memory Cache Buffering to Increase and Enhance Throughput
- Separate or Simultaneous Record and/or File Lock to Facilitate and Expedite Concurrent Record Access and Provide System Security
- Full Complement of Interactive High-Level Utilities to Reduce Applications Programming Overhead and Costs

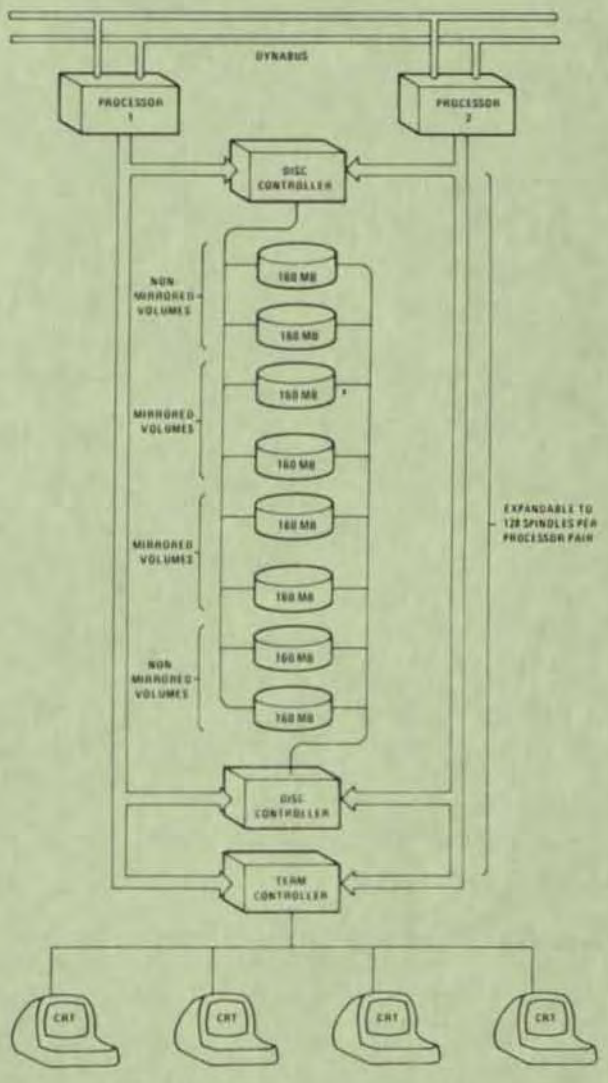


Figure 1. Multi-Volume File Configuration

TANDEM ENSCRIBE™

INTRODUCTION TO ENSCRIBE

The unique and evolutionary ENSCRIBE software package extends Tandem's Non-Stop concept of hardware and operating system failure immunity to fail-safe file and record management. Since ENSCRIBE operates as an integral part of the GUARDIAN Operating System across multiple processors, data base integrity is ensured in the event of a Processor, I/O Controller or Disc Drive failure. ENSCRIBE brings many advanced features not found even in traditional midi- and large-scale computer systems into the minicomputer price range.

DATA BASE INTEGRITY

When a conventional processor or disc drive fails, it is most often necessary to reconstruct the data base from archival tapes or manual records to ensure the integrity of the data base. Conversely, in a Tandem T16 system, when a processor fails, all file management functions are automatically transferred to an alternate processor. Optionally, ENSCRIBE manages a data base using the mirror principle illustrated in Figure 1. In this mode, data is written onto two disc volumes simultaneously. When a disc fails, all file management functions are automatically transferred to the mirror disc volume. Upon restoration of the failed disc, ENSCRIBE copies the data back onto the failed disc. This copying operation is done concurrent with requests to read or update data and is entirely transparent to both the application programmer and user.

MULTI-VOLUME FILES

A file may reside entirely on a single volume or may be partitioned to reside on several volumes. With system expansion to sixteen processors, and assuming that each volume is mirrored, the maximum size for each multi-volume file is nearly four billion bytes. And each partition of a multi-volume file is under the control of a separate processor.

FILE ORGANIZATIONS

ENSCRIBE provides a variety of file organizations for the applications programmer:

- Key-Sequenced (Indexed)
- Relative
- Entry-Sequenced

The programmer need not be burdened with structuring the file. He simply chooses the file organization best suited to a particular application and ENSCRIBE automatically structures the file. This allows a programmer to create, access and maintain data files with efficiency and economy. Moreover, since records are logically positioned, random records may be accessed sequentially. ENSCRIBE also provides Multi-Key Access to data records. Location of records in a Key-Sequenced file may be by approximate, generic or exact key value. ENSCRIBE maintains an index of key values which provides rapid access to data records. When a new record is added to a file, or when a key value has been changed, ENSCRIBE automatically updates the index including all secondary keys. This operation is entirely transparent to the application program.

DATA AND INDEX COMPRESSION

For key-sequenced files, an optional data compression technique may be used to pack more data into a particular disc area. Similarly, an optional index compression is provided for key indices.

MAIN MEMORY CACHE BUFFER

ENSCRIBE provides a Cache Buffer which resides in main memory. The purpose of the cache is, whenever possible, to keep the most recently accessed blocks in main memory speeding up access. System transaction rates may, therefore, be increased by simply allocating more memory to the cache. The cache resides in a separate area from the operating system and application programs.

RECORD AND FILE LOCKING

ENSCRIBE provides both record and file locking for system security. Record locking provides a greater degree of concurrent access to a file. A file lock request must wait for the record to be unlocked before it is granted.

SUMMARY

ENSCRIBE is complemented by Tandem's *Data Definition Language* (DDL) for data base management. ENSCRIBE may be used alone for those applications which do not require centralized data base administration, or in conjunction with DDL to provide full data base access. Thus, the user is provided with an economical growth path from file management to NonStop data base access.

ENSCRIBE FACILITIES

GENERAL SERVICES

ENSCRIBE provides file and record management services for the Tandem GUARDIAN Operating System. These services are interfaced to the application program via T/TAL or COBOL.

ENSCRIBE facilities may be used separately for handling file and record management tasks, or alternatively, may be used as an extension to Tandem's *Data Definition Language* (DDL); a *schema* for centralized data base management. Once a file organization has been established, programs may store, retrieve and/or modify logical data records without concern for file structure. ENSCRIBE facilities provide all necessary access control, data buffering, blocking/deblocking and file structure maintenance.

FILE AND RECORD STRUCTURE SERVICES

ENSCRIBE offers the application programmer a choice of three file organizations: *Key-Sequenced (Indexed), Relative,* or *Entry-Sequenced.* In addition, both fixed-length and variable-length record formats are supported under ENSCRIBE. A major advantage of fixed-length records is high performance during data access. Variable-length records optimize storage space on the disc. Both the file organization and record format must be specified at file definition time.

Key-Sequenced (Indexed) Files – In key-sequenced file organizations, records are stored in ascending order according to the value assigned a primary key field within the record. The primary key field may be defined as any contiguous set of bytes within the record. For example, a customer's name in an invoice file may be defined as the primary key. All records in an indexed file are variable-length. Primary keys must be unique. Up to 255 secondary keys can be specified.

Relative Files – Records are stored in a relative file according to their relative position within the file. The primary key for a relative file is the record number. These record numbers are ordinal values. Each record in a relative file is a fixed-length record. Relative files can have secondary keys.

Entry-Sequenced Files – In an entry-sequenced file, records are stored according to the sequential order in which they are presented to the system. These records may be either fixed or variable length – but once entered, the record's size may not be changed. The primary key for an entry-sequenced file is the record's logical address. Entry-sequenced files can have alternate keys.

Multi-Keyed Records – Under ENSCRIBE, a single file may have up to 255 alternate key fields, and the values contained in these alternate keys need not be unique. Thus, each file may have a primary key and alternate keys. These key fields are used to construct a secondary key index. This scheme is commonly called *Multi-Indexing.* The user need only specify the key values to access records. Moreover, ENSCRIBE automatically keeps all indices up-to-date during record updates and record insertions.

Multi-Indexing – Indexing provides the greatest flexibility in accessing records. Records may be accessed randomly by specifying a key or keys, or sequentially by consecutively accessing the records in the collated order of an index. Since multiple key fields can be defined, multiple indices allows an indexed file to appear as sequential. Moreover, ENSCRIBE provides three indexing options:

- Exact Key Match
- Approximate Key Match
- Generic Key Match

Exact key match means that the record's key field must exactly match the specified key. Approximate match means that the record key may be equal or greater than the search key. This allows a user program to access records without knowing the exact key. Generic key match means that only the initial portion (partial key) of a key need be specified (such as the prefix to a part number in a vendor's record).

Cache Buffering – Transparent to the application program, ENSCRIBE transfers one or more physical data blocks to main memory. Records within these blocks are then made available to the program. The purpose of this cache buffering is to keep the most recently accessed blocks in main

memory. Thus, index blocks for a Key-Sequenced File are likely to remain in the cache. ENSCRIBE automatically ensures that enough buffer space is available via overlaying the least recently accessed blocks, and also ensures that extraneous physical accesses are avoided.

FILE AND RECORD OPERATIONS

At program execution time, the user program may issue requests for various ENSCRIBE services. These services fall into two categories: *file* operations and *record* operations. Figure 2 illustrates a few of the available operations.

File Operations – File operations may be categorized as follows:

- Definition of an ENSCRIBE file organization.
- Opening of an existing ENSCRIBE file for processing.
- Closing of an ENSCRIBE file and termination of processing on that file.
- Examination of the attribute information stored within an ENSCRIBE file.
- Extension of the allocated space in an ENSCRIBE file.

Record Operations – Record operations may be summarized as follows:

- Read a record.
- Find a record.
- Insert a record.
- Update an existing record.
- Delete a record.
- Lock/Unlock a record.

```
INT .filename[0:11] := "$SYSTEM USER    FILE    ",
    .prim^key[0:2] := [5,0,1024],
    .buffer[0:49];
INT file^number, read^cnt, wrt^cnt, cnt^read, cnt^wrt;

STRING key[0:5];

LITERAL prim^ext = 5,
        file^code = 888,
        sec^ext = 2,
        file^type = 3,
        rec^len = 100,
        blk^len = 1000;

! Create a new keyed file
CALL CREATE(file^name,
            prim^ext, ,
            file^code,
            sec^ext,
            file^type,
            rec^len,
            blk^len,
            prim^key);

! Insert a new record into a keyed file
CALL WRITE(file^number,buffer,wrt^cnt,cnt^wrt);

! Find a record by key value
CALL KEYPOSITION(file^number,key);

! Read a record from a file
CALL READ(file^number,buffer,read^cnt,cnt^read);

! Update a record
CALL READUPDATE(file^number,buffer,read^cnt,cnt^read);
       .
       .
   Update the record in buffer
       .
       .
CALL WRITEUPDATE(file^number,buffer,wrt^cnt,cnt^wrt);

! Delete a record from a file
CALL WRITEUPDATE(file^number,buffer,0,cnt^wrt);

! Lock a record
CALL LOCKREC(file^number);
       .
       .
 Next Record Read will be Locked
       .
```

Figure 2. File and Record Management Examples

TANDEM

ENTRY SCREEN FORMATTER

FEATURES

- **Interactive Page-Mode Form Creation on Video Display Terminal for Non-Stop Transaction-Oriented Applications**
- **Efficient and Economical Forms Creation with Automatic or Manual Cursor Positioning, Field Delimiters, Automatic Validity Checking, and Easy Transfer to Disc Files**
- **Fast and Error-Free Forms Display with Automatic Data Entry Validity Checking**
- **Individual Field Access with Length and Validity Check Attributes for User-Defined Error Checking**
- **Tandem-Supplied SCREEN Program to Create New or Modify Old Forms and to Insert Multiple Forms in a Single Disc File**

INTRODUCTION

The ENTRY Screen Formatter provides a simple easy-to-use method of creating and displaying user-defined forms on an interactive page mode video terminal. Extensive validity and error checking is accomplished via the Tandem-supplied ENTRY procedures. When an error has been made in the creation or data entry processes, a flashing cursor is positioned over the field in error and an error message is printed to specifically flag the error.

Entry essentially performs four tasks:

- Creates Forms
- Programmatically Displays Forms
- Programmatically Accesses Individual Fields
- Tests Forms Independent of Application Program

In addition, ENTRY provides the facilities to: modify or update old forms, insert multiple forms in a single disc file, and provide length and field validity checking attributes for user-defined error checking.

FORMS CREATION

Forms creation is performed on a page mode terminal. The user simply designs the form on the screen as it is to appear when used. Fields on the form where terminal operator entries are to be made are indicated by delimiters. Once the form image is created, the user specifies a field name and validity checking attribute for each field. When completed, the form image is written to a designated file on disc.

FORMS DISPLAY

Forms display, forms read and field checking are all performed by application programs through calls to ENTRY procedures. When a form is displayed on a terminal, the operator fills in the fields on the form and presses a function key to transmit the fields into the computer where ENTRY performs the validity check on each field. If a field contains invalid data, the form can be programmatically redisplayed with the invalid entry flashing on the screen.

FIELD ACCESS

Individual fields in a form may be referenced by an application program. A field is referenced by the name assigned when the form was defined. Additionally each field has a length attribute and a validity checking attribute for user-defined error checking.

SCREEN PROGRAM PROCEDURES

ENTRY provides the following procedures to aid terminal I/O with forms created by the SCREEN program:

- EXPAND^SCREEN –
  initializes the applications I/O buffer with the control and data characters required to output a screen to the terminal
- READ^SCREEN –
  fills the applications I/O buffer with the control sequence required to input the operator entry fields from a screen
- CHECK^SCREEN –
  moves the operator entry fields from the applications I/O buffer into the correct fields and does the required validity checking
- BLINK^SCREEN –
  fills the applications I/O buffer with the control sequence required to turn the "blinking" on or off for a specific operator entry field
- POSITION^SCREEN –
  fills the applications I/O buffer with the control sequence required to position the cursor over a specific operator entry field
- FL^SCREEN –
  is used to compute the actual length of the field input by the operator

TANDEM ENTRY SCREEN FORMATTER

ERROR MESSAGES

When a user makes an error during form creation and testing, the error is flagged as follows: the flashing cursor is automatically positioned over the field in error and an error message is displayed on the bottom line of the screen. The user may then correct the field in error and strike any function key to send the corrected form back to the computer. Some typical error messages are:

- FIELD NOT TERMINATED: the operator entry field does not have a terminating delimiter or is greater than 255 characters in length
- ILLEGAL FIELD NAME: the field name was not an alphanumeric string of 8 or fewer characters, starting with an alphabetic character
- ILLEGAL ATTRIBUTE: the checking attribute is either missing or not a legal integer value
- NOT ENOUGH FIELDS DEFINED: the user did not provide enough field names
- TOO MANY FIELDS DEFINED the user provided too many field names or attempted to define more than 255 fields

Example: Creating a Screen Format

```
                N E W     A C C O U N T     F O R M A T

PLEASE FILL IN THE FOLLOWING INFORMATION. IF YOUR DATA DOES NOT
COMPLETELY FILL IN THE SPACE PROVIDED YOU CAN USE THE TAB KEY TO GET
TO THE NEXT ITEM.

LAST NAME        [                          ]
FIRST NAME, M.I.[                           ]
STREET ADDRESS   [                                                  ]
CITY             [                          ]
STATE            [                          ]
ZIP CODE         [     ]

ACCOUNT NUMBER   [     ]

INITIAL DEPOSIT AMOUNT ($NNNNN.NN)   $[         ]

TO ENTER THE DATA PLEASE TYPE FUNCTION KEY F1. IF DATA IS IN ERROR THE
ITEM WILL BE SET BLINKING AND AN ERROR MESSAGE WILL APPEAR AT THE
BOTTOM OF THE SCREEN. PLEASE CORRECT THE DATA AND RE-ENTER. A
CORRECT, COMPLETED FORM WILL BE INDICATED BY A MESSAGE. TYPE FUNCTION
KEY 16 TO RETURN TO THE MENU.

[                                                                    ]

Note: All operator entry fields are defined with [...]. All other
      data is set protected on the screen.
```

Example: Assignment of Field Names & Checking Attributes

```
NAME      1   <--- Any Characters Valid
FIRST     1   <--- Any Characters Valid
STREET    1   <--- Any Characters Valid
CITY      1   <--- Any Characters Valid
STATE     2   <--- Alphabetic Only
ZIP       3   <--- Numeric Only
ACCT      3   <--- Numeric Only
INITIAL   8   <--- Financial Numeric Only
MSG       0   <--- No Checking Performed
```

Example: Coding the Entry Procedures

```
PROC PAGE^OPEN;
 BEGIN

! Initialize the I/O Buffer with Screen Format
 WRTCNT := EXPAND^SCREEN(@FORMOPEN,,SBUFFER,1);
 CALL WRITE(CRT,BUFFER,WRTCNT,CNT);
 IF < THEN BEGIN ...
    .
    .
    .

! Turn Blinking Field Off
 WRTCNT := BLINK^SCREEN(@FORMOPEN,SCREEN,SBUFFER,FORMOPEN^MSG,0);
 CALL WRITE(CRT,BUFFER,WRTCNT,CNT);
 IF < THEN BEGIN ...

! Prepare a buffer for Reading a Screen
 WRTCNT := READ^SCREEN(@FORMOPEN,SBUFFER);
 CALL WRITEREAD(CRT,BUFFER,WRTCNT,FORMOPEN^IOBUF,CNT);
 IF < THEN BEGIN ...
    .
    .
    .

! Perform Validity Checking on Operator Entry Fields
 DO ...  UNTIL CHECK^SCREEN(@FORMOPEN,SCREEN,SBUFFER,OPEN^CHK);

! Turn on Blink, Position Cursor
 WRTCNT := BLINK^SCREEN(@FORMOPEN,SCREEN,SBUFFER,FORMOPEN^MSG,1);
 WRTCNT := POSITION^SCREEN(@FORMOPEN,
                           SCREEN,
                           SBUFFER[WRTCNT],
                           FORMOPEN^ACCT) + WRTCNT;
 CALL WRITE(CRT,BUFFER,WRTCNT,CNT);
 IF ERR THEN BEGIN ...
```

TANDEM
ENVOY™

DATA COMMUNICATIONS MANAGER

FEATURES

- Fail-Safe Data Communications Manager for Non-Stop™ Transaction-Oriented Applications
- Multiple Communications Protocols
  - IBM Binary Synchronous (BSC)
  - ADM-2 Asynchronous
  - TINET Asynchronous
  - Burroughs Synchronous
- Multiple Line Types (Data Links)
  - Point-to-Point (BSC only)
  - Centralized Multipoint Supervisor
  - Centralized Multipoint Tributary
- Trace Facility, Line Usage & Error Statistics, On-line Testing
- Support for Auto Call Facility
- Support for Multiple Modem Types
  - Bell type 103, 202 Asynchronous
  - Bell type 201, 208 Synchronous
- Simplified Applications Programming with Calls to GUARDIAN Operating System Procedures

INTRODUCTION

The ENVOY *Data Communications Manager* provides an efficient, easy-to-use interface between transaction-oriented application programs and the telecommunications network. And since ENVOY is under control of the GUARDIAN *Operating System,* it has the same inherent *fail-safe* features of all other Tandem systems software. As such, ENVOY ensures that data communications are maintained even in the event of a processor or I/O channel failure. ENVOY provides all the control necessary for switched or leased point-to-point and leased multi-point telecommunication networks. An automatic calling option allows unattended stations on switched networks to communicate and exchange data without operator intervention. The ENVOY Data Communications Manager supports Synchronous transmission at speeds up to 56K Bps and Asynchronous transmission at speeds up to 19.2K Bps.

POINT-TO-POINT DATA LINK

ENVOY supports a Binary Synchronous point-to-point data link over either a switched or leased (non-switched) lines. In the switched network mode, each of two stations has a unique telephone number. The originating station dials the answering (remote) station and establishes a connection. After the connection has been established, system security messages may be interchanged to ensure the integrity of the two stations. Information interchange may then proceed. Once the interchange has been completed, the calling station automatically disconnects. The answering station detects the disconnection and then it also automatically disconnects. In the leased (non-switched) mode, the two stations are permanently connected (no number is required), but the data exchange sequence is the same as that described above for switched lines.

MULTIPOINT DATA LINK

A multipoint data link consists of two or more stations communicating over a leased (non-switched) line. With this type of data link, one station is designated the *supervisor* and controls all communications over the link. All other stations are designated *tributaries.* In a *centralized* multipoint link, communications can occur only between the supervisor and a tributary (tributary-to-tributary communication is not allowed). Each tributary station in a multipoint link is identified by a *polling* address and a *selection* address.

To solicit data transfers from the tributary stations, the supervisor station periodically *polls* each tributary station in the multipoint link by transmitting each tributary's polling address. When a tributary that has data to send is polled, further polling of the data link stops. At this point, the tributary responds by transmitting its data to the supervising station.

To transfer data to a tributary station, the supervisor station first *selects* the desired tributary station by transmitting the tributary's selection address. For selection to occur, the tributary station must be monitoring the line and be willing to accept the forthcoming data. Following selection, the supervisor station transmits the data to the tributary station.

TANDEM ENVOY™

TANDEM

ENVOY PROCEDURES

ENVOY functions are implemented via making calls to the GUARDIAN *Operating System's File Management Procedures.* These procedures are as follows:

- The **OPEN** Procedure is used to gain access to a data communications line.
- The **DEFINELIST** Procedure is used by application processes operating as a multi-point supervisory or tributary station. For a supervisory station, **DEFINELIST** specifies the polling address and selection address of each station. For a tributary station, **DEFINELIST** specifies the polling address(s) and selection address(s) that identify the tributary when the multi-point line is polled.
- The **WRITE** Procedure is used to transmit data to a remote station.
- The **READ** Procedure is used to accept data from a remote station.
- The **WRITEREAD** Procedure is used to transmit data to a remote station and then await a reply.
- The **HALTPOLL** Procedure is used by a station operating as a multipoint supervisor to stop continuous polling of all stations on a line.
- The **CHANGELIST** Procedure is used by application processes operating as a multipoint supervisory or tributary station. For a supervisory station, **CHANGELIST** is used to enable/disable polling of a particular station. Additionally, **CHANGELIST** is used to restore polling of non-responding units.
- The **CLOSE** Procedure is used to terminate access to a line.

TRACE FACILITY

ENVOY provides an aid in checkout of communications applications with the Trace Facility. The Trace Facility works with all line types except auto call units.

Trace Facility records line "events" in a Trace Table. Each Trace Table entry provides;

- Sequence Number
- Controller and Line Number
- Line State
- Event
- Miscellaneous Information
- Timestamp

LINE STATISTICS

Line statistics are maintained by ENVOY for each line from the time a line is opened until the line is closed. The statistics are printed on the operator console when a predetermined error threshold is exceeded, when a path switch occurs, or when the line is closed.

ON-LINE TESTING

ENVOYTST (Envoy Test) is used to verify the operation of the data communications process and the synchronous controller operating as the following line types:

- Point-to-point non-switched primary
- Point-to-point non-switched secondary
- Point-to-point switched primary
- Point-to-point switched secondary
- Multipoint supervisor
- Multipoint tributary

TM

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000

T16/8733-00

TAL

TANDEM APPLICATION LANGUAGE

- **HIGH LEVEL LANGUAGE FOR APPLICATIONS AND SYSTEMS PROGRAMMING**
- **PROCEDURE-ORIENTED LANGUAGE FOR TRANSACTION PROCESSING**
- **EXTENSIVE DATA TYPES FOR ARITHMETIC AND STRING DATA TYPES**
- **SPECIAL STRING AND BIT MANIPULATION FUNCTIONS**
- **OBJECT CODE OPTIMIZATION COMPARABLE TO ASSEMBLER CODING**
- **FREE-FORM STRUCTURE**
- **CONSTRUCTS AMENABLE TO STRUCTURED PROGRAMMING**

TRANSACTION APPLICATION LANGUAGE (T/TAL)

The Tandem Transaction Application Language is a high level, block structured language designed for the easy implementation of transaction-oriented applications. T/TAL is similar to ALGOL, PL/I or COBOL in providing procedure blocks and high level constructs such as IF . . . THEN . . . ELSE, DO . . . UNTIL, WHILE . . . DO, FOR, and CASE. Programmers can write self documenting programs with object code generation as efficient as assembler code.

PROCEDURES

The procedure-oriented, block-structured approach allows a programmer to design procedure blocks for functional simplicity. Data can be assigned globally for all procedures to access or can be assigned locally – known only to the procedure in which it is declared. Local data is assigned dynamically and initialized each time the procedure is invoked. The local data feature also allows the use of recursive procedures.

One procedure can call another and pass parameters by value or by reference. The called procedure can optionally return a value through its name.

All procedure code is compiled as re-entrant and, therefore, non-modifiable. This permits code to be shared by many different users thereby minimizing the amount of memory required for multiple applications.

DATA DECLARATIONS

T/TAL provides data types for arithmetic, string and logical operations. Signed integer data can be specified with 15 bits or 31 bits of precision. Fixed data is a scaled decimal with up to 18 digits of precision. String data consists of one or more 8-bit bytes. Logical data consists of 16-bit words.

Data can be declared as simple, single element items or as arrays (multiple elements). In addition a variable can be declared as a pointer variable to facilitate indirect addressing.

Type functions are an integral part of TAL and provide the ability to convert from one data type to another. Type functions are also available to determine if string data represents ASCII NUMERIC, ALPHA or SPECIAL values.

SPECIAL FEATURES

Special string manipulation constructs are available to facilitate efficient processing of transaction data. Included are SCAN, MOVE and COMPARE string operations.

Data can be addressed at the bit level, if desirable. Operations exist in TAL to perform bit deposit, bit extraction, arithmetic shifting and logical shifting.

I/O

All I/O is performed via calls to operating system procedures. Procedures exist to open and close files, read and write data, perform control functions, create and purge files, lock and unlock files, and perform "wait" or "nowait" I/O. Extensive error checking is provided by the File System and error codes are returned to the calling procedure for inspection.

TAL CONSTRUCTS

General Form

```
[label:]  statement;
```

Specific Form

Assignment

```
variable := [variable:= ... ] arithexp;
```

Compound

```
BEGIN
statement1;
statement2;
  .
  .
statementn;
END;
```

Unconditional Branching

```
GOTO label;
```

Invoking Procedures

```
CALL procedurename [(parameterlist)];
RETURN [arithexp];
```

Conditional

```
IF condexp THEN statement [ELSE statement]
CASE <index OF BEGIN
                statement0;
                statement1;
                  .
                  .
                statementn;
                OTHERWISE statement;
                END;
```

Iteration

```
FOR variable := initvalue TO limit [BY step] DO statement;
WHILE condexp DO statement;
DO statement UNTIL condexp;
```

String Manipulation

```
destarray { .*n | .+n } { sourcearray FOR count | constantlist } [->nextaddr]

{SCAN | RSCAN} array { WHILE | UNTIL } testchar [->addr];
```

The listing on the next page shows a simple TAL program including GLOBAL and LOCAL data declarations and a procedure block. Also included is the variable map and procedure map which are used when debuggfng the program.

TANDEM APPLICATION LANGUAGE

TANDEM APPLICATION LANGUAGE

```
TAL - TANDEM COMPUTERS VERSION A01  ( 6/25/76 - 10 AM)                SOURCE LANGUAGE: TAL - TARGET MACHINE TANDEM/16
DATE - TIME :  6/28/76 - 11:56:01                          OPTIONS: ON (LIST,CODE,MAP,WARN,(MAP) - OFF (ICODE,INNERLIST)

    1.    000000 0  0  ?NOCODE
    2.    000000 0  0  ! GLOBAL DATA DECLARATIONS
    3.    000000 0  0  INT .HOMETERM[0:11], .LINE[0:35];
    4.    000060 0  0  STRING .LINES := @LINE '<<' 1;
    5.    000060 0  0  DEFINE NULL = %0#, CRLF = [%15,%12]#;
    6.    000060 0  0  ?NOLIST
    9.    000000 0  0
   10.    000000 0  0  PROC CHANGE^DATE MAIN;
   11.    000000 1  0   BEGIN
   12.    000000 1  1  ! LOCAL DATA DECLARATIONS
   13.    000000 1  1  INT SAVE^ADDR1, SAVE^ADDR2, SAVE^ADDR3, MONTH^IN^DECIMAL, TERM;
   14.    000000 1  1  INT COUNT,STATUS, INPUT^DATA^POINTER;
   15.    000000 1  1  INT LENGTH^YEAR, LENGTH^MONTH, LENGTH^DAY;
   16.    000000 1  1  STRING .YEAR[0:3], .MONTH[0:2], .DAY[0:1], .TEMPS[0:3];
   17.    000000 1  1  STRING .MONTH^TABLE[0:35] := "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC";
   18.    000022 1  1
   19.    000022 1  1          CALL MYTERM(HOMETERM);                  ! GET TERM ID AND OPEN TERMINAL FILE
   20.    000050 1  1          CALL OPEN(HOMETERM,TERM,0);
   21.    000055 1  1   NEXT: LINES ':=' "ENTER DATE YY/MM/DD" & CRLF;
   22.    000072 1  1          CALL WRITEREAD(TERM,LINE,21,0,COUNT);   ! GET DATE AS YY/MM/DD
   23.    000103 1  1          IF COUNT = 0 THEN CALL STOP;            ! IF NO DATA THEN QUIT
   24.    000111 1  1          LINES[COUNT] ':=' NULL;                 ! INITIALIZE FOR SCAN
   25.    000121 1  1  ! GET THE YEAR AND EXPAND IT
   26.    000121 1  1          SCAN LINES UNTIL "/" -> SAVE^ADDR1;     ! SCAN FOR FIRST SLASH
   27.    000125 1  1          LENGTH^YEAR := SAVE^ADDR1 - @LINES;     ! CALCULATE LENGTH OFYEAR FIELD
   28.    000131 1  1          YEAR ':=' "19" & LINES[0] FOR LENGTH^YEAR; ! CONCATENATE "19" TO YEAR FOR A 4 DIGIT VALUE
   29.    000143 1  1          INPUT^DATA^POINTER := SAVE^ADDR1 - @LINES + 1;  ! CALCULATE POINTER TO NEXT FIELD IN DATE
   30.    000150 1  1  ! GET THE MONTH, CHECK FOR VALIDITY, AND CONVERT TO ALPHA
   31.    000150 1  1          SCAN LINES[INPUT^DATA^POINTER] UNTIL "/" -> SAVE^ADDR2;  ! SCAN FOR SECOND SLASH
   32.    000155 1  1          LENGTH^MONTH := SAVE^ADDR2 - SAVE^ADDR1 - 1;  ! CALCULATE LENGTH OF MONTH
   33.    000162 1  1          TEMPS ':=' LINES[INPUT^DATA^POINTER] FOR LENGTH^MONTH & NULL;
   34.    000174 1  1     ! SAVE MONTH NUMERIC VALUE FOR VALIDITY CHECK AND ALPHA LOOKUP
   35.    000174 1  1          CALL NUMIN(TEMPS,MONTH^IN^DECIMAL,10,STATUS); !CONVERT ASCII MONTH TO INTEGER
   36.    000203 1  1          IF STATUS <> 0 THEN BEGIN   ! IF CONVERSION IS NOT GOOD
   37.    000206 1  2                               ERROR: LINES ':=' "INPUT ERROR";
   38.    000215 1  2                                      CALL WRITE(TERM,LINE,11,COUNT);
   39.    000225 1  2                                      GOTO NEXT;
   40.    000234 1  2                               END;
   41.    000234 1  1          IF MONTH^IN^DECIMAL > 12 THEN GOTO ERROR;
   42.    000240 1  1     ! FIND ALPHABETIC VALUE FOR MONTH^IN^DECIMAL
   43.    000240 1  1          MONTH ':=' MONTH^TABLE[3 * (MONTH^IN^DECIMAL - 1)] FOR 3;
   44.    000251 1  1          INPUT^DATA^POINTER := SAVE^ADDR2 - @LINES + 1; ! CALCULATE POINTER TO NEXT FIELD IN DATE
   45.    000256 1  1  ! GET THE DAY AND SAVE
   46.    000256 1  1          SCAN LINES[INPUT^DATA^POINTER] UNTIL NULL -> SAVE^ADDR3;
   47.    000263 1  1          LENGTH^DAY := SAVE^ADDR3 - SAVE^ADDR2 - 1;
   48.    000270 1  1          DAY ':=' LINES[INPUT^DATA^POINTER] FOR LENGTH^DAY;
   49.    000274 1  1  ! FORMAT AND OUTPUT THE NEW DATE
   50.    000274 1  1          LINES ':=' CRLF & MONTH FOR 3 & " " & DAY FOR LENGTH^DAY & ", " & YEAR FOR LENGTH^YEAR+2;
   51.    000331 1  1          CALL WRITE(TERM,LINE,10+LENGTH^DAY+LENGTH^YEAR,COUNT);
   52.    000348 1  1          GOTO NEXT;
   53.    000351 1  1   END; ! END OF PROC

COUNT                          VARIABLE      INT          L+00A    DIRECT
DAY                            VARIABLE      STRING       L+016    INDIRECT
ERROR                          LABEL
INPUT^DATA^POINTER             VARIABLE      INT          L+010    DIRECT
LENGTH^DAY                     VARIABLE      INT          L+013    DIRECT
LENGTH^MONTH                   VARIABLE      INT          L+012    DIRECT
LENGTH^YEAR                    VARIABLE      INT          L+011    DIRECT
MONTH                          VARIABLE      STRING       L+015    INDIRECT
MONTH^IN^DECIMAL               VARIABLE      INT          L+004    DIRECT
MONTH^TABLE                    VARIABLE      STRING       L+020    INDIRECT
NEXT                           LABEL
SAVE^ADDR1                     VARIABLE      INT          L+001    DIRECT
SAVE^ADDR2                     VARIABLE      INT          L+002    DIRECT
SAVE^ADDR3                     VARIABLE      INT          L+003    DIRECT
STATUS                         VARIABLE      INT          L+007    DIRECT
TEMPS                          VARIABLE      STRING       L+017    INDIRECT
TERM                           VARIABLE      INT          L+005    DIRECT
YEAR                           VARIABLE      STRING       L+014    INDIRECT

CHANGE^DATE                    PROC
CRLF                           DEFINE                              [%15,%12]
HOMETERM                       VARIABLE      INT          G+000    INDIRECT
LINE                           VARIABLE      INT          G+001    INDIRECT
LINES                          VARIABLE      STRING       G+002    INDIRECT
MYTERM                         PROC
NULL                           DEFINE                              %0
NUMIN                          PROC          INT
OPEN                           PROC
STOP                           PROC
WRITE                          PROC
WRITEREAD                      PROC

PEP    BASE    LIMIT   ENTRY  ATTRIBUTES  NAME

002   000003  000402  000025    M         CHANGE^DATE

OBJECT FILE NAME IS $SCR.SYSTEM.SAMPLE
NO. ERRORS=0   ; NO. WARNINGS=0
PRIMARY GLOBAL STORAGE=3
SECONDARY GLOBAL STORAGE=0A
CODE SIZE=256
DATA AREA SIZE=2  PAGES
CODE AREA SIZE=1  PAGES
MAXIMUM SYMBOL TABLE SIZE=3A3
ELAPSED TIME =   0: 0:19
```

TANDEM
COBOL

FEATURES

- **Conforms to ANSI COBOL X3.23-1974**
- **Full compatibility with ENSCRIBE file structures**
- **Runs in mixed language environment with T/TAL**
- **Includes Tandem Extension for NonStop(TM) Programming**

INTRODUCTION

Tandem/COBOL runs on the Tandem T16 Computer System – the only multi-processor system designed for NonStop, transaction-oriented, data base applications. Tandem/COBOL utilizes all the capabilities of Tandem's GUARDIAN Operating System and ENSCRIBE Data Base Record Manager – software designed to keep the system up and running while maintaining the integrity of the on-line data base. Thus Tandem/COBOL is compatible with programs written in T/TAL, Tandem's high level language for transaction processing, and is capable of running in a multi-language environment.

GUARDIAN features handled by Tandem/COBOL include;

- NonStop Operation
- Shared, Re-entrant Code
- Virtual Memory System
- Geographic Independence of I/O Devices
- Checkpoint/Checkmonitor Facilities

ENSCRIBE features handled by Tandem/COBOL include;

- Key-sequenced, Entry-sequenced and Relative file structures
- Logical file size to 4 BILLION bytes
- One primary and up to 31 alternate keys
- Optional mirror data base recording

LEVEL OF SUPPORT

NUCLEUS – Level 2

The nucleus gives a basic language for the internal processing of data within the four divisions of a program. Qualification, punctuation characters, data-name formation, connectives, and figurative constants are supported as well as ACCEPT, ADD, ALTER, COMPUTE, DISPLAY, DIVIDE, EXIT, GO, IF, INSPECT, MOVE, MULTIPLY, PERFORM, STOP, STRING, SUBTRACT, and UNSTRING statements.

Extensions to the language include the ability to call T/TAL procedures via the ENTER verb.

TABLE HANDLING – Level 2

Table Handling lets you define three dimensional fixed length tables of contiguous data items. Each item is accessed relative to its position in the table. Each item may be identified through use of a subscript or an index. This level also provides series options and the ability to vary the contents of indices by an increment or decrement. Table handling includes the use of the SET and SEARCH statements.

SEQUENTIAL I-O – Level 2

Sequential I-O reads records of a file one after the other, in the order they were written. Memory areas among files are shared, and the full facilities for the FILE-CONTROL, I-O-CONTROL, and FD entries are included. Within the Procedure Division, CLOSE, OPEN, READ, REWRITE, USE, and WRITE are recognized.

RELATIVE I-O – Level 2

Relative I-O accesses records in either random or sequential manner. Each record in a relative file is uniquely identified by an integer value greater than zero which specifies the record's logical position in a file. It provides full facilities for the FILE-CONTROL, I-O-CONTROL and FD entries.

TANDEM COBOL

TANDEM COBOL

Within the Procedure Division it provides full capabilities for the CLOSE, DELETE, OPEN, REWRITE, START, USE, and WRITE statements.

INDEXED I-O – Level 2

Indexed I-O accesses records in either random or sequential manner. Each record in an indexed file is identified by the value of one or more keys within that record. It provides full facilities for the FILE-CONTROL, I-O-CONTROL and FD entries. Within the Procedure Division it provides full capabilities for the CLOSE, DELETE, OPEN, READ, REWRITE, START, USE, and WRITE statements.

SORT-MERGE – Level 2

Sort-Merge orders one or more files of records, or combines two or more identically ordered files of records, according to a set of keys within each record. Optionally, you may apply some special processing to each of the individual records by input or output procedures using RELEASE and RETURN. This special processing may be applied before and/or after the records are ordered by SORT or after the records have been combined by MERGE. Also, you may sort one or more files, or combine two or more files, one or more times within a given execution of a COBOL program.

REPORT-WRITER – Null Level

SEGMENTATION – Null Level

Segmentation is automatic via Tandem's GUARDIAN Operating System.

LIBRARY – Level 1

Text from a library is copied into the source program using COPY. The copied text is not changed.

DEBUG – Level 1

Debug provides a basic debugging capability, including the ability to specify (1) selective or full procedure monitoring, and (2) optionally compiled debugging statements.

INTER-PROGRAM COMMUNICATION – Level 1

Inter-program Communication means a program can communicate with one or more programs using CALL and CANCEL. This communication is done by (1) transferring control from one program to another within a run unit, and (2) letting both programs have access to the same data items.

COMMUNICATION – Null Level

TANDEM COBOL EXTENSIONS

Several extensions have been added to Tandem/COBOL to permit use of Tandem's GUARDIAN Operating System.

1. Nonstop Extensions

   For Nonstop programming in COBOL, you include a "?NONSTOP" line in your source program. STARTBACKUP and CHECKPOINT are the verbs that make the program nonstop. Normally STARTBACKUP is called once at the beginning of the program to set the nonstop mode. Thereafter the CHECKPOINT verb is used to pass information to the backup process at critical points in the processing. In a nonstop program, checkpoints will also occur automatically upon any OPEN or CLOSE executed after the backup is established. Both of these verbs will set the special register, PROGRAM-STATUS, to indicate the outcome of the checkpointing operation.

2. Extensions to the standard COBOL I/O facility

   Three new verbs, LOCKFILE, UNLOCKFILE, and UNLOCKRECORD, have been introduced to allow the use of the corresponding system file and record locking routines. This addition allows separate processes to share a common data base. READ and REWRITE verbs have been extended to allow the specification of a LOCK or UNLOCK operation. The OPEN syntax has been extended to specify the file access, EXCLUSIVE, SHARED or PROTECTED, and to permit the SYNCDEPTH for files opened in the OUTPUT, I-O or EXTEND mode.

3. Calling TAL procedures

   The ENTER verb has been modified to call TAL procedures. This lets you access any TAL system external or your object library routines. These object library routines are assigned to the COBOL run unit at compile time.

```
      DIMENSION M (10,20)
      READ 100, I, J, K
      DO 200, IS = I, 10
      DO 200, JS = J, 20
  200 M (IS, JS) = K
      IF (I .EQ. J) GO TO 300
```

TANDEM FORTRAN

- **Conforms to full language specifications of ANSI FORTRAN, X3.9-1977**
- **Full use of all ENSCRIBE facilities including multi-key access and locking**
- **RECORD Structures for DDL compatibility**
- **Extensions for NonStop(tm) Programming**
- **Facilities for Interprocess Communications**
- **Support of Floating Point Arithmetic**

INTRODUCTION

Tandem FORTRAN runs on the Tandem T16 Computer System — the only multi-processor architecture designed for NonStop, transaction-oriented, data base applications. Tandem FORTRAN utilizes all the facilities of the GUARDIAN Operating System including Non-Stop operation, re-entrant code, interprocess communications, virtual memory, and ENSCRIBE data base facilities for keyed, relative, and sequential access, multi-key data paths, and concurrent record access. Tandem FORTRAN is capable of running in a multi-language environment.

LEVEL OF SUPPORT

Tandem FORTRAN conforms to the specifications described in the American National Standards Institute for Programming Languages – FORTRAN, X3.9-1977.

Data Types

Six data types are supported; INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL and CHARACTER. In addition data may be arranged as arrays or substrings. Dimensioned arrays can have up to seven subscripts.

Expressions

Expression types can be arithmetic, character, relational or logical. Operators allowed in expressions include:

Arithmetic	**	(Exponentiation)
	/	(Division)
	*	(Multiplication)
	-	(Subtraction)
	+	(Addition)
Character	//	(Concatenation)
Relational	.LT.	(Less Than)
	.LE.	(Less Than or Equal)
	.EQ.	(Equal)
	.NE.	(Not Equal)
	.GT.	(Greater Than)
	.GE.	(Greater Than or Equal)
Logical	.NOT.	
	.AND.	
	.OR.	
	.EQV.	
	.NEQV.	

Specification Statements

Support is included for DIMENSION, COMMON, EQUIVALENCE, IMPLICIT, PARAMETER, EXTERNAL, INTRINSIC, and SAVE statements. In addition, type statements are allowed for INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, and CHARACTER.

DATA Statement

DATA Statements can be used for initialization of variables, arrays, array elements, and substrings.

Assignment Statements

There are four kinds of Assignment statements: Arithmetic, Logical, Statement label (ASSIGN), and Character.

TANDEM FORTRAN

TANDEM FORTRAN

Control Statements

Sixteen Control statements are used to control the execution of a program.

Unconditional GO TO	DO
Computed GO TO	CONTINUE
Assigned GO TO	STOP
Arithmetic IF	PAUSE
Logical IF	END
Block IF	CALL
ELSE IF	RETURN
ELSE	
END IF	

Input/Output Statements

Nine Input/Output statements are supported.

READ	OPEN	BACKSPACE
WRITE	CLOSE	ENDFILE
PRINT	INQUIRE	REWIND

Tandem FORTRAN utilizes the UNIT, FMT, REC, IOSTAT, ERR, and END control parameters for READ and WRITE.

For OPEN the UNIT, IOSTAT, ERR, FILE, STATUS, ACCESS, FORM, RECL, and BLANK open parameters can be used. For CLOSE the UNIT, IOSTAT, ERR, and STATUS parameters can be used.

For INQUIRE specifiers may be used for IOSTAT, ERR, EXIST, OPENED, NUMBER, NAMED, NAME, ACCESS, SEQUENTIAL, DIRECT, FORM, FORMATTED, UNFORMATTED, RECL, NEXTREC, and BLANK.

For BACKSPACE, REWIND, and ENDFILE the UNIT, IOSTAT, and ERR specifiers can be used.

FORMAT Specification

A complete range of FORMAT specifiers is available. Both explicit and list-directed editing are allowed. Editing descriptors allowed are:

Apostrophe	P
H	BN and BZ
Positional	Numeric (I, F, E, D, G)
T, TL, and TR	L
X	A
Slash	List-Directed
Colon	S, SP, and SS

Functions and Subroutines

Four categories of procedures are supported:

Intrinsic Functions
Statement Functions
External Functions
Subroutines

External functions and subroutines may be specified in either FORTRAN or other programming language subprograms.

An ENTRY statement may be used to permit specification of the entry point into a subprogram.

BLOCK DATA Subprogram

Block Data subprograms are allowed for initializing variables and array elements in named common blocks.

TANDEM FORTRAN EXTENSIONS

Several extensions have been made to FORTRAN 77 to enhance its operation on Tandem systems.

1. NonStop Extensions

STARTBACKUP and CHECKPOINT functions allow a FORTRAN program to utilize the NonStop capabilities of GUARDIAN. STARTBACKUP is called once at the beginning of a program to establish the nonstop mode. Thereafter CHECKPOINT is used to pass critical information to the backup process. Checkpoints will automatically occur upon any OPEN or CLOSE after the backup has been created.

2. Structures

Structures provide the ability to define records in FORTRAN. The constructs RECORD and END RECORD are used to define record structures. The Data Definition Language may also be used to transcribe a schema into FORTRAN RECORD structures.

3. ENSCRIBE Extensions

Extensions have been made to the FORTRAN READ and WRITE statements to permit the full use of ENSCRIBE facilities. Thus it is possible with FORTRAN statements to access key-sequenced, relative, and entry-sequenced files by primary or up to 255 alternate keys. Provision has been made to allow exact, approximate or generic positioning into an ENSCRIBE file structure using FORTRAN. Concurrent record access is supported with LOCK mechanisms at either the record or file level.

4. Interprocess Communications

FORTRAN processes can communicate with one another or with processes written in other languages through the standard FORTRAN READ and WRITE statements. Communication to other processes is implemented using the interprocess communication facilities of the GUARDIAN Operating System.

TANDEM
SORT

FEATURES

- Three modes of operation
  - Conversational
  - Via command files
  - Programmatic
- Record Input and Output
  - From/To an External File
  - From/To a User Application Program
- Support for multiple file types
  - Unstructured files
  - ENSCRIBE structured files
  - EDIT files
- SORT keys may be;
  - Ascending or descending
  - STRING or INTEGER
  - Multiple key fields per sort
- SORT Output can be;
  - Complete record
  - Record Sequence Numbers
  - Keys only

INTRODUCTION

The Tandem SORT program reorders a set of records according to the values of sort key fields defined within the records. SORT may be driven by a set of commands conversationally, or by a text file containing the commands, or it may be called from your program.

Records may be passed to SORT from a file, or sent one by one through procedure calls from your program. Similarly, the sorted set of records may be written to a file, or your program may call a procedure to retrieve the records, one per call.

Actual sorting runs as a separate process from the host program. Standard interface procedures are present in the Sort Command Interpreter program or called from your program, which handle process creation, control, and communication.

CONVERSATIONAL MODE

For this mode simply type,

```
:SORT
```

and wait for the prompt "<". Six commands are allowed:

FROM – names the file of unsorted records and describes the records;

TO – names the file for the sorted records and selects output options;

ASCENDING – DESCENDING describe the location and attributes of the sorting keys within the records;

RUN – starts the actual sort.

EXIT – exits from the Sort Command Interpreter.

Example:

```
:SORT/OUT $lp/
<FROM insort,RECORD 56
<TO outsort
<ASC 6:12 INTEGER
<RUN
<EXIT
:
```

TANDEM SORT

TANDEM SORT

COMMAND FILE MODE

For this mode, use EDIT to put your sort commands, one to a line, in a command file

:SORT [/IN <command file>]
[, OUT <list file>] /]

SORT reads the commands from <command file>. When a RUN command is found, and the previous commands describe a valid sort, SORT begins. Once the SORT completes, <command file> is read for more commands. End-of-file on <command file> terminates SORT.

Example:

:SORT/IN myfile,OUT $lp/

Contents of "myfile";

```
FROM datain
TO dataout
ASC 1:3 INTEGER
DESC 10:16 STRING
RUN
ASC 88:89
FROM names
DESC 22:35
TO sortout
RUN
```

PROGRAM SORTS

There are four ways to use SORT in your program.

1. SORT Input from External File
   SORT Output to External File

2. SORT Input from External File
   SORT Output to Program, one record at a time

3. SORT Input from Program
   SORT Output to External File

4. SORT Input from Program
   SORT Output to Program, one record at a time.

Five SORT Interface Procedures are provided.

SORTSTART — Initiate a sort.

SORTSEND — Send input record to SORT

SORTRECEIVE — Retrieve a sorted record

SORTFINISH — Complete the sort

SORTERROR — Format a SORT error message

GENERAL METHOD OF OPERATION

Two things determine how the sort functions internally:

- the amount of data to be sorted,
- the amount of memory you give.

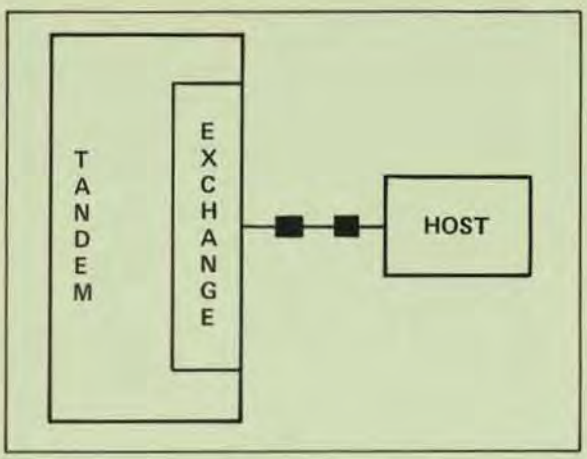
If the amount of data is small enough, the data is sorted within memory.

In most large sorts, the memory is insufficient to sort all the data at once, so SORT splits the input data into sorted pieces it can handle, and puts them in a scratch file. Each piece is referred to as a *run* since the records in each one "run" in sequence.

When there is more than one run formed from the initial data, the sorted output is produced by merging the runs together. This method of sorting is known as "Replacement/Selection".

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000

T16/8736-00



TANDEM

EXCHANGE REMOTE JOB ENTRY SUBSYSTEM

FEATURES

- **Supports 2780 and 3780 Emulation**
- **Batch input from any media**
  - **terminal**
  - **magnetic tape**
  - **disc**
  - **card reader**
  - **another process**
- **Batch output to any media**
  - **terminal**
  - **magnetic tape**
  - **disc**
  - **line printer**
  - **another process**
- **Three Modes of Operation**
  - **Conversational**
  - **Command File**
  - **Programmatic**
- **File transfer ability between Tandem Systems**
- **NonStop Operation Optional**

INTRODUCTION

EXCHANGE is a Tandem Subsystem designed to allow a Tandem System to emulate the functions of a 2780 or 3780 remote batch workstation. Input and output can be from/to any media supported by the Tandem System including disc, magnetic tape, terminals, card readers, line printers, and other processes. Commands to define the input, list and punch files, and characteristics about the connection can be accepted conversationally from a terminal, from a Command File, or programmatically from another process.

General capabilities of EXCHANGE include transmitting and receiving in ASCII or EBCDIC, accepting horizontal tab codes, accepting vertical forms control codes, transmitting or receiving EBCDIC transparent data, short record truncation, blank field compression, transmitting and receiving blocked data link messages, and generation of WACK and TTD control codes when temporarily unable to transmit or receive.

Another useful feature is the ability to transfer files between two workstations. An EXCHANGE subsystem on one Tandem can perform remote file transfers to another Tandem which is also running the EXCHANGE subsystem.

EXCHANGE, at the user option, may be run in a NonStop mode.

EXCHANGE STRUCTURE

EXCHANGE is composed of two program modules; a Server Process and a Command Interpreter.

The Server has all data link handling responsibility and responds to requests to send or accept data to/from a remote system. The Server is typically run in conjunction with the Command Interpreter but can be called directly from an application process. Send and receive data and initial connection parameters are passed to the Server. The Server handles all message assembly/disassembly, blank compression/decompression, horizontal tab expansion, record truncation, and character set translation. The Server accepts and delivers data on a record-by-record basis.

The Command Interpreter provides a conversational interface to the operator. Commands are entered to control the connection type, specify the receive file names and parameters, and identify the files to be sent.

TANDEM EXCHANGE SUBSYSTEM

TANDEM EXCHANGE SUBSYSTEM

EXCHANGE COMMAND INTERPRETER

The EXCHANGE Command Interpreter allows the user to send or receive files by entering commands at a terminal or through a command file. Commands which can be entered are:

CONNECT – defines remote terminal connection

SEND – defines file to be sent

RECEIVE – defines the receiving file

ABORT – stops any transfer in progress

DISCONNECT – ends connection to host

EXIT – exits from EXCHANGE Command Interpreter

STATUS – displays line status and statistics

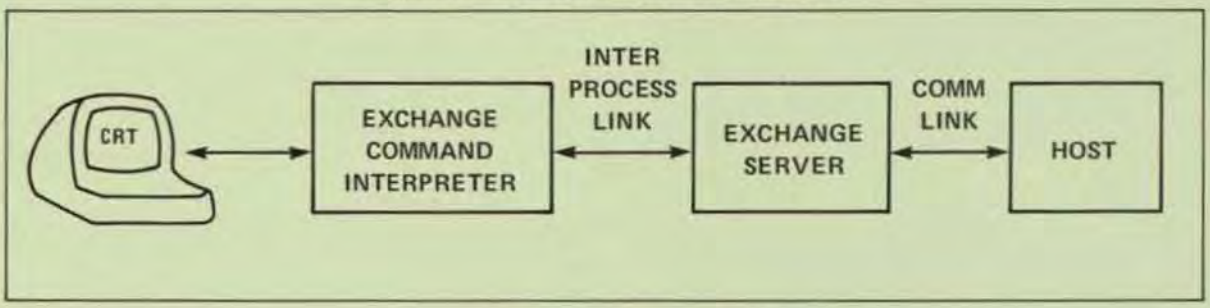
PROGRAMMATIC INTERFACE

A user process may directly interface to the EXCHANGE Server. The required steps in the application process are:

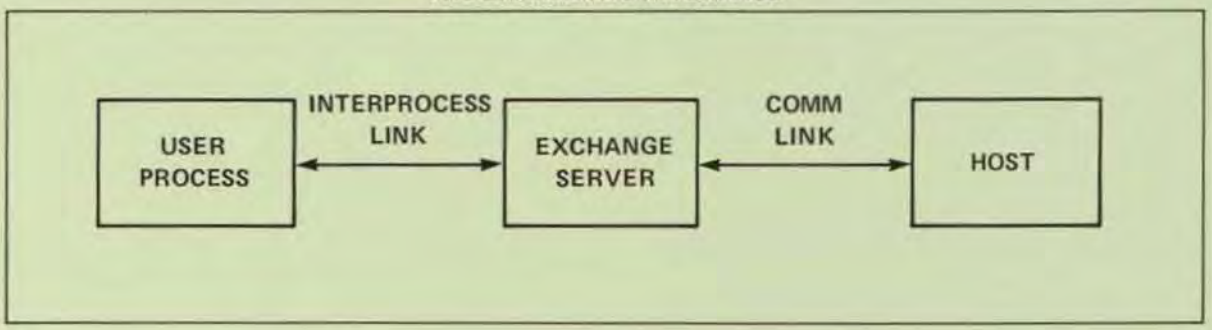
1. Open an interprocess file to the Server.
2. Establish the receive file character set via SETMODE.
3. Establish a data link to the remote system via SETMODE.

At this point the application may send/receive records via simple READ/WRITE type statements. Data link message formatting is handled by the Server, and compressed blanks and embedded horizontal tabs are expanded.

COMMAND INTERPRETER INTERFACE



PROGRAMMATIC INTERFACE



NONSTOP FEATURES

The EXCHANGE Server can be made to run NonStop. In the NonStop mode of operation EXCHANGE will:

1. Maintain a connection between the User, the EXCHANGE Command Interpreter, the EXCHANGE Server, and the Communication Data Link.
2. Maintain a major state; connected, disconnected, connecting.

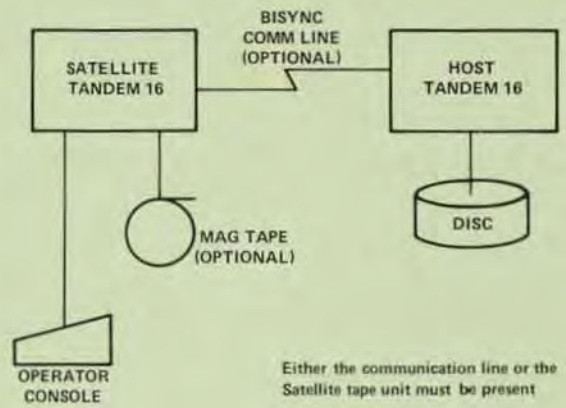
In the NonStop mode of operation each process checkpoints all major state changes. This implies that:

1. Connected or disconnected status is maintained after a failure of either process.
2. Connection attempts are retried if a failure occurs during a connect.
3. If a failure occurs in either process SEND and RECEIVE commands are aborted, but the communication link is maintained.

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, CA 95014 ■ (408) 996-6000

T16/8738-00

SATELLITE SYSTEM HARDWARE



Either the communication line or the Satellite tape unit must be present

TANDEM EXTEND

SATELLITE PROCESSOR OPERATING SYSTEM

INTRODUCTION

- EXTEND is a subset of the GUARDIAN Operating System that allows operation of the Tandem 16 system without the use of disc storage. The Tandem 16 computer on which EXTEND runs, known as the Satellite, is used in conjunction with a "normal" Tandem 16, known as the Host. The Satellite and the Host may be connected by a data communication line.
- The Satellite Tandem 16 may optionally be connected to terminals, tape drives, punched card readers, and so forth. However, the Satellite must have an operator console for the purpose of logging system messages and reloading processor modules. Also, if the Satellite Tandem 16 is not equipped with a tape unit, it must be connected via a communication line to a normal Tandem 16 for the purpose of loading the Satellite's processor modules.

SATELLITE SYSTEM HARDWARE

A Satellite system consists of at least the following hardware:

- Two or more Tandem processor modules, as in a normal system
- An operator console for logging system messages

The Satellite system must also have at least one of the following:

- A magnetic tape unit and a magnetic tape controller, or
- A point-to-point data communication line using bisync protocol, connected to a synchronous controller. The communication line must be connected to a normal Tandem 16, known as the Host.

A Satellite system may include any other hardware devices with the exception of discs.

- The data communication line between the Satellite and the Host is a point-to-point line using Binary Synchronous (bisync) protocol. It may be switched or non-switched (leased). It is connected at both the Host and Satellite ends to a synchronous controller.
- During the time that the Satellite operating system image is being downloaded from the Host, the communication line must be dedicated to the loading operation. At other times, the line may be used by application processes.

SATELLITE SYSTEM SOFTWARE

SCOMINT

Commands may be input interactively to the Satellite via SCOMINT, the Satellite Command Interpreter. SCOMINT is a subset of COMINT, the Command Interpreter that runs on a normal Tandem 16.

SHOST

A program called SHOST is run on the Host Tandem 16 for the purpose of loading Satellite processor modules via a communication line.

TANDEM EXTEND

TANDEM EXTEND

SATELLITE COMMAND INTERPRETER

- The Satellite Command Interpreter, a subset of the Command Interpreter that runs on a normal Tandem 16, serves as the programmer's interface to the Satellite Tandem 16. Its purpose is to accept a command (from a predefined set of commands) entered from a file (usually an online terminal) and initiate the system function indicated by the definition of the command.

  Some of the system functions are:

- to display system status
- to run, debug, and stop programs
- to designate a backup cpu for the Satellite Command Interpreter
- to switch control of the Satellite Command Interpreter to its backup
- to load processor modules
- to set current date and time-of-day
- to alter bus availability status
- to debug privileged programs
- to control the state of the system's peripheral devices
- An important difference between the normal Command Interpreter and the Satellite Command Interpreter is that the Satellite Command Interpreter has no LOGON command. Instead, the user is always "logged on" as SUPER.SUPER. Any user of the Satellite system can debug privileged programs, reload processor modules, and in general perform any operation on the system.
- In a typical system a startup Satellite Command Interpreter begins executing when the system is cold loaded.

HOST PROCESSING

The SHOST program is used to control the Host end of a Satellite reload operation. The normal procedure is for the Satellite operator to call the Host operator and verbally request either a cold load (all Satellite CPU's are down) or a reload (at least one Satellite CPU is up).

The Host operator then uses SHOST to do the following:

- open a communication line to the Satellite
- open a file containing a Satellite operating system image
- if this is a cold load, then send a bootstrap over the communication line
- wait until Satellite operator is done reloading all CPU's
- exit from SHOST

LOADING THE SATELLITE

- Depending on whether or not the Satellite is equipped with a tape unit, different procedures must be followed when performing a cold load or a reload of the system. If the system has a tape unit, then the memory image tape generated by the SYSGEN program is used to load the system at the Satellite site. If the Satellite does not have a tape unit, then the operating system image must be sent over the bisync communication line. Sending a processor image across a communication line is known as "downloading" the processor module. Downloading Satellite processor modules requires the cooperation of an operator at the Host site, who must establish the communication line and (if a cold load is required) send the bootstrap.

SYSTEM GENERATION

- The SYSGEN program is used to generate an operating system for a Satellite Tandem 16. Input to SYSGEN is a set of files: a configuration file and several object code files containing operating system processes, library routines, and (optional) user library routines. The output of SYSGEN is a fully configured operating system.
- The configuration file specifies information such as the number of processors in the system, the memory size of each processor module, the I/O devices configured into the system, and so forth. In addition, the configuration file used to generate a Satellite operating system contains a Program Definition Table, which contains the name and process number of each program that is to be run on the system.

TANDEM
UNIVERSAL INTERFACE

FEATURES

- **Provides 8/16 line parallel interface**
- **Operates half-duplex, bi-directional at up to 4 MBytes/sec**
- **Uses both TTL and Differential logic**
- **Contains dual-channel connected logic for fault-tolerant operations**

INTRODUCTION

The T16/3401 Universal Interface (UI) provides the ability to interface custom equipment to the T16 Computer System. The UI is capable of attaching two devices that have 8 or 16 line parallel data interfaces to the Tandem 16 Computer. The Universal Interface (UI) provides a device data path that is buffered (16 words deep), bi-directional, and capable of operating in half-duplex mode at a sustained data transfer rate of up to 4 Mbyte per second (depending on the channel configuration). It interfaces to one device over positive or ground true TTL lines for short distances (up to 25 ft) and to the second device over differential lines for longer distances (up to 500 ft). The data path between either or both of the two devices and the UI can be either one byte (8 bits) or one word (16 bits) wide.

Configuration of the UI is accomplished by software and by configuration jumpers in the connector hood.

DATA PATH

The UI can be used to interface both input and output devices requiring a data path width of either one byte (8 bits) or one word (16 bits) in half-duplex mode. Data is transferred between the UI and the channel in bursts consisting of several words. For input or output devices that operate in word mode, the UI passes data words directly to and from the device on the DATA 0:15 interface lines (Figure 1). For output devices that operate in byte mode, the UI disassembles data words from the channel and transfers data bytes to the device. For input devices that operate in byte mode, the UI assembles data bytes from the device and transfers data words to the channel. In byte mode, data is transferred to and from the device on the DATA 8:15 interface lines (Figure 1). The width of the device data path is program controlled, such that a device with a one byte wide data path can have additional control signals sent to it on the remaining eight data lines. For example, some terminals have a one byte data path but require 12 bits for cursor addressing.

DATA TRANSFER

Data transferred between the UI and a device may be strobed in a handshake sequence or pulsed without a handshake. In Handshake Mode, the device response can either lead or follow the UI data strobe. In Pulse Mode, the UI data strobe is used to pulse data to the device on write operations with no response from the device, while for read operations, the device response is used to pulse data to the UI with no response from the UI.

DATA PARITY

Odd parity is generated and checked for each data word that is transferred between the channel and the UI. The parity that exists between the UI and each device is defined by configuration jumpers in the connector hood. The jumpers select odd, even or no parity.

DATA TRANSFER TERMINATION

A data transfer sequence between a device and the I/O channel is initiated with an EIO instruction executed by the CPU program. When a specified number of bytes have been transferred, the sequence is terminated in one of three ways: 1) by the I/O channel, 2) by the device, or 3) by the UI.

UI STROBES

The UI is capable of transmitting up to six strobe signals to a device, one of which is the Data Strobe. The other five strobes can be transmitted individually or in combinations under program control.

Two strobe signals are provided that can be pulsed before and after data transfer. The "before" strobe (Strobe 4) is 1 μs in duration and the "after" strobe (Strobe 5) is 10 μs in duration.

TANDEM UNIVERSAL INTERFACE

Two strobe signals (Strobe 2 and Strobe 3) can be held active during the entire data transfer.

One strobe signal (Strobe 1) can be gated during data transfer by an external trigger from the device.

In a ground-true TTL interface, Strobes 1 through 5 can be wired together in the connector hood in any combination to "or" their respective functions.

UI STATUS

Device status is presented to the UI on eight (8) interface lines, STATUS 8:15. These signal lines can be jumpered to the STATRES 0:8 pins in the connector hood to form the desired status results.

Controller (UI) and device status words are transferred to the CPU program in the CPU's register stack in response to EIO, IIO and HIIO instructions.

UI DEVICE ADDRESSING

All devices attached to a processor I/O channel are assigned unique Physical Unit Addresses. The Physical Unit Address is 8 bits in length and consists of a 5 bit Controller Number and a 3 bit Unit Number. The Controller Number is defined by switches on the UI board; Unit Number 0 is the device attached to the TTL interface and Unit Number 1 is the device attached to the differential interface.

UI COMMANDS

Commands are issued to the UI by executing EIO (Execute Input/Output) instructions in the CPU program. The UI command set is comprised of the following commands:

- SENSE
- TAKE OWNERSHIP
- DISABLE PORT
- SET COUNT
- READ
- WRITE

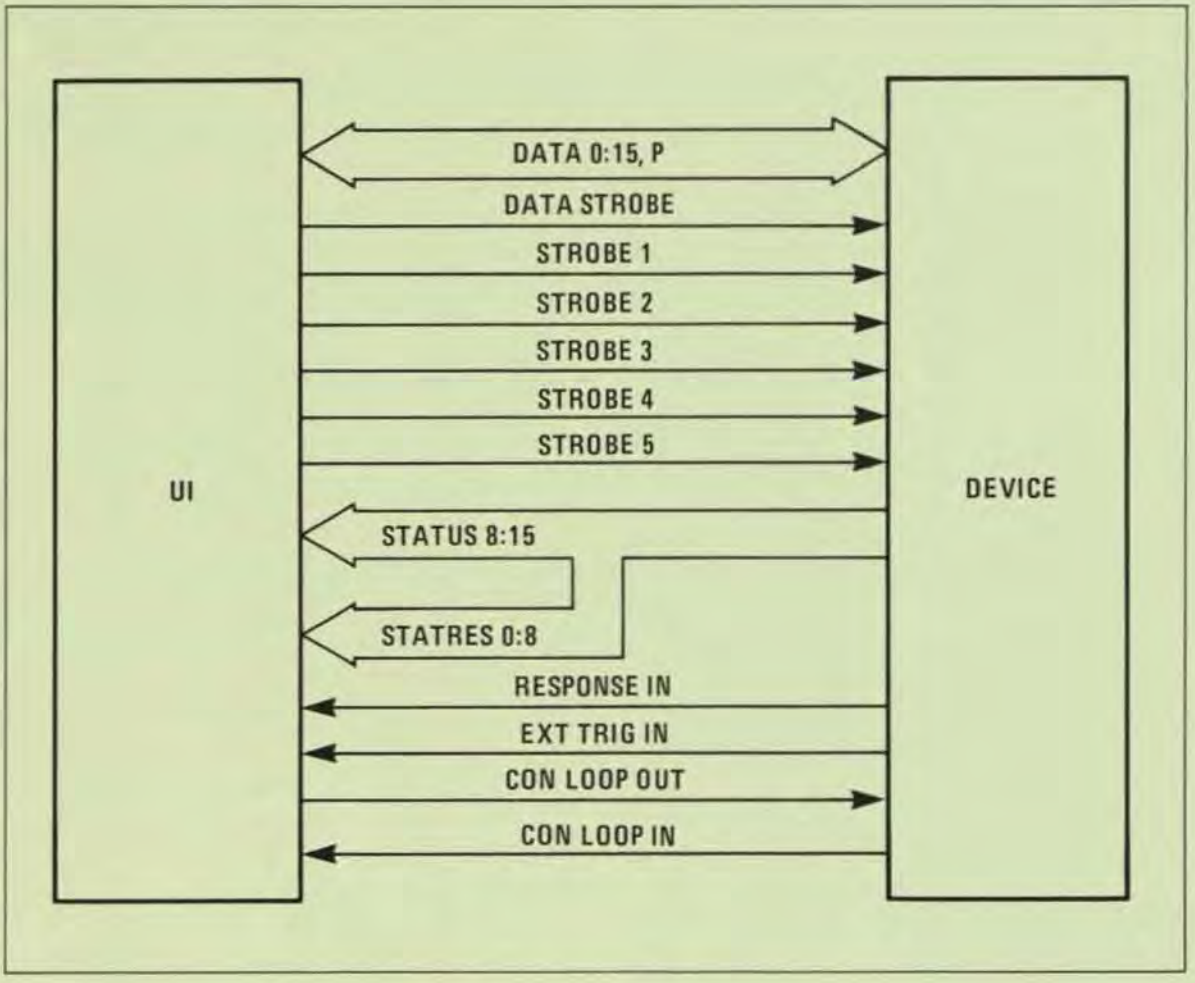
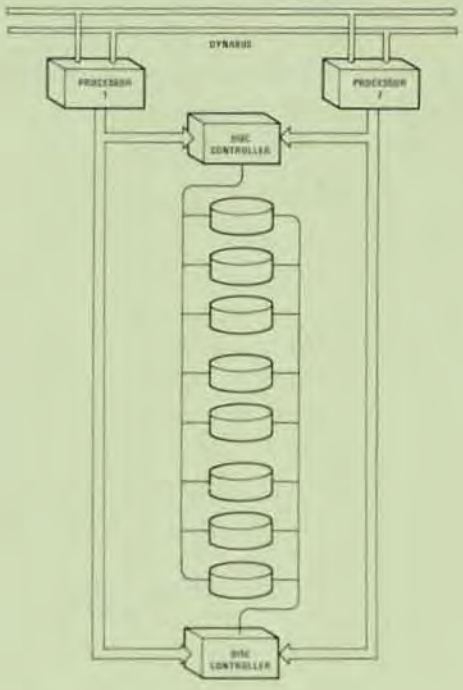


Figure 1. UI/Device Interface



TANDEM

DUAL-PORT DISC SUBSYSTEMS

- **Field Proven Disc Drives for Transaction-oriented Systems**
- **High Capacity**
- **High Performance**
- **Dual-Ported Disc Drives for Data Integrity**
- **Reliable Servo Track Positioning for Accuracy**
- **Intelligent, microprogrammed Disc Controller**
- **Compact, Stand-alone Packaging**

SUBSYSTEM CONFIGURATION

The Tandem Dual-port Disc Subsystem provides high capacity, high performance, high availability and high data integrity for on-line, transaction-oriented systems. When used on the Tandem NonStop (tm) Computer System these disc subsystems provide the most cost effective solution for transaction-oriented applications where data integrity is of paramount importance.

The Model T16/3105 Disc Controller provides a high degree of flexibility and expandability. The disc controller is actually an intelligent, "backend" disc processor capable of supporting up to eight disc drives per controller. Because the controller is micro-programmed a wide variety of disc technologies and data base functions can be supported.

The T16/3105 Disc Controller connects to two (2) I/O channels simultaneously and may be powered from either processor in the event of a single processor failure. Thus, high availability is insured to the controller.

The T16/4103, T16/4104 and T16/4105 Disc Drives provide high capacity and performance through the use of new disc technologies. In addition high availability and data integrity are provided through dual-port connections. Each disc drive can connect to two (2) controllers simultaneously.

T16/3105 DISC CONTROLLER

The T16/3105 Disc Controller is used to interface up to eight disc drives consisting of any mix of T16/4103, T16/4104 or T16/4105 Disc Drives. The controller provides high availability to the drives through its dual-channel connection pioneered by Tandem in its other controllers.

Significant features of the controller include:

(1) Dual-channel connection

(2) 4K RAM Buffer for buffering logical records

(3) Dual recording or the ability to transfer the buffer to one or more disc units without having to re-transmit on the I/O bus

(4) Support for dual-access disc drives

(5) Read without I/O transfer which allows one disc to be copied to another without involving the I/O bus

The disc controller can be configured so that two controllers are connected to the same disc unit. Up to eight drives can be daisy-chained on one controller. With the dual-ported disc option daisy-chain failures will not cause a loss of the data base, a loss of data base integrity, or a loss of an access path to the data base.

TANDEM DUAL-PORT DISC SUBSYSTEMS

TANDEM DUAL-PORT DISC SUBSYSTEMS

T16/4103, 4104 AND 4105 DISC DRIVES

The T16/4103, 4104 and 4105 are stand-alone disc drives using the latest in disc technologies to provide high capacity and performance.

The T16/4103 uses 3330 technology to provide a storage capacity of 160M bytes per pack (formatted). Performance data includes an average access time of 28 ms, average latency of 8.35 ms, and a data transfer rate of 806K bytes/second. Data is recorded on an eleven-high pack. Servo tracking insures accuracy of track following.

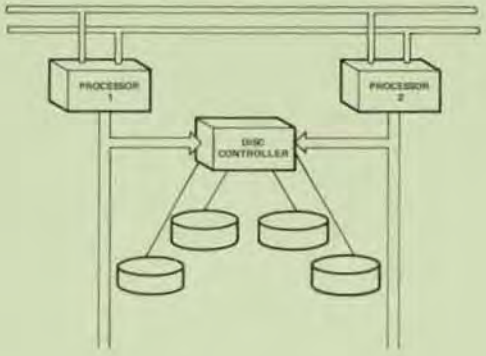
The T16/4104 uses SMD technology to provide a storage capacity of 240M bytes per pack (formatted). Performance data includes an average access time of 28 ms, average latency of 8.35 ms, and a data transfer rate of 1.2M bytes/second. Data is recorded on an eleven-high pack. Servo tracking insures accuracy of track following.

The T16/4105 uses SMD technology to provide a storage capacity of 64M bytes per pack (formatted). Performance data includes an average access time of 30 ms, average latency of 8.35 ms, and a data transfer rate of 1.2M bytes/second. Data is recorded on a five-high pack. Servo tracking insures accuracy of track following.

Each of the drives has an option of being dual-ported, the ability to connect to two (2) T16/3105 Disc Controllers simultaneously. If a disc controller or the daisy-chain should fail, the other controller connection will insure access to the data base.

Each drive uses a linear DC motor (voice coil) for accurate and high speed seeking. Reliable TTL and ECL circuits are used throughout.

SPECIFICATIONS	T16/4103	T16/4104	T16/4105
RECORDING CAPACITY			
Capacity (Unformatted)	200M Bytes	300M Bytes	80M Bytes
Capacity (Formatted)	160M Bytes	240M Bytes	64M Bytes
Recording Mode	MFM	MFM	MFM
Recording Density	4040 BPI	6060 BPI	6038 BPI
Tracks per Surface	808 + 7 alt	808 + 7 alt	808 + 15 alt
Tracks per Inch	384	384	384
Data Surfaces	19	19	5
Servo Surfaces	1	1	1
PROCESSING SPEED			
Data Transfer Rate	806K Bytes/Sec	1.2M Bytes/Sec	1.2M Bytes/Sec
Spindle Speed	3600 RPM	3600 RPM	3600 RPM
Latency (average)	8.35 ms	8.35 ms	8.35 ms
ACCESS SPEEDS			
Minimum (one track)	10 ms	10 ms	6 ms
Average	28 ms	28 ms	30 ms
Maximum	55 ms	55 ms	55 ms
DISC PACKS (all packs must be flag free)	3M 936/11 CDC 9882 Memorex Mark XI Dysan	Dysan (300MB)	CDC 9877
PHYSICAL			
Dimensions (D x W x H)	34 x 19 x 38	34 x 19 x 38	34 x 19 x 34
Weight	465 lbs	465 lbs	243 lbs
ELECTRICAL			
Voltage	208V, 60 Hz 220/240V, 50 Hz	208V, 60 Hz 220/240V, 50 Hz	120V, 60 Hz 220/240V, 50 Hz
Phases	Three	Three	Single
Operating Amps	6.4	6.4	8.2
ENVIRONMENTAL			
Temperature (operating)	60–90 F	60–90 F	59–90 F
Humidity (operating)	20–80%	20–80%	20–80%
Heat Dissipation	5800 BTU/hr	5800 BTU/hr	1800 BTU/hr



TANDEM
SINGLE - PORT DISC SUBSYSTEMS

- **Tough, Service-Engineered Disc Drives for Critical NonStop™ Real Time Transaction-Oriented Applications**
- **Low-Cost High Performance Medium- to High-Capacity (10 to 50 Megabytes) Disc Drives with Proven Dependability, System Versatility and Modular Maintainability**
- **Economical High-Speed Reliability with Advanced Hybrid and CMOS Technology**
- **Fast Data Transfer Rate (312K Bytes/Second) for Efficient and Economical Data Processing**
- **Wide Choice of Optional Features to Facilitate NonStop™ System Configuration or Expansion in the Field.**

SUBSYSTEM CONFIGURATION

The Tandem Disc Subsystems were particularly selected to provide low-cost, high reliability, high data capacity, fast data transfer and easy as well as economical maintainability. All these parameters are crucial to the exacting requirements of a non-stop transaction-oriented environment.

A standard Disc Subsystem consists of a Disc Controller and one or more Moving Head Disc Drives. The Controller features two independent I/O Channel ports and may be powered from either system processor in the event of a single processor failure. In addition, a disc failure cannot corrupt the system processor's memory since the address and count words are held in the I/O processor – not the disc controller.

The Model T16/4101 is a 10MB capacity drive using moving heads while the Model T16/4102 is a 50MB capacity drive using moving heads. The Model T16/3101 Disc Controller can accommodate 1 to 4 disc drives in any mix of T16/4101 and T16/4102 Disc Drive configurations. Each drive connected to the controller has its own signal and data cable. No daisy chain is used.

MODEL T16/4101 DISC DRIVE

The T16/4101 Disc Drive is a low-cost, medium capacity (10M Byte) disc drive featuring a 312KB transfer rate, high-speed random-access and pedestal-mounted cartridge disc drive. The unit is comprised of one fixed platter of 5MB and one removable top-loading platter of 5MB with average seek and latency times of 35 and 12.5 milliseconds, respectively. This rugged unit also features: a service-oriented modular design which eliminates all field adjustments, and an automatic cleaning system which purges and cleans the disc area to reduce costly preventive maintenance.

MODEL T16/4102 DISC DRIVE

The Model T16/4102 Disc Drive is a moderately-priced, high-capacity (50M Bytes) disc drive ideally suited for large data bases. This high-performance unit is a random-access, pedestal-mounted disc pack drive and is directly I/O interchangeable with the Model T16/4101 cartridge disc drive (described above) when the Model T16/3102 Controller is installed. The unit uses CDC® 9873 Disk Packs, or equivalent, for the storage and retrieval of data. Each removable pack contains 11 discs. Average seek time is 35 ms. Average latency is 12.5 ms. The transfer rate is 312KB.

TANDEM SINGLE-PORT DISC SUBSYSTEMS

SPECIFICATIONS

PARAMETER	MODEL	
	4101	4102
RECORDING CAPACITY		
Capacity (Maximum Megabytes, Formatted)	10MB	50MB
Recording Mode	Double Freq.	Double Freq.
Recording Density (BPI - Nominal)	2200	2200
Tracks Per Surface (Maximum)	408	406
PROCESSING SPEED		
Data Transfer Rate (K Bytes/Second)	312	312
Spindle Speed (RPM)	2400	2400
ACCESS TIME (Direct Seek)		
Full Stroke (Milliseconds)	60	70
Average (Milliseconds)	35	35
One-Track (Milliseconds)	7	10
DISC CARTRIDGE/PACK		
Number of Discs	1	11
Usable Surfaces	2	20
Pack Type	5440 equiv.	2314-2 equiv.

PARAMETER	MODEL	
	4101	4102
RECORDING HEADS	4	20
PHYSICAL:		
Dimensions (L x W x H Inches)	29-3/4 x 18-1/2 x 34	32-1/2 x 27-1/2 x 38
Weight (Pounds)	252	700
ELECTRICAL:		
Voltage	1	2
Frequency (Hz) Disc	50 ±1.0 60 ±1.0	50 ±1.0 60 ±1.0
Power Rating (Operating AMPS)	4.6	5.0
Power Rating (Surge AMPS)	9.2	22.0
ENVIRONMENTAL:		
Operating Temperature (°F)	60 – 90	60 – 90
Operating Humidity (Non-Condensing)	10 – 90%	10 – 80%
Non-Operating Temperature (°F)	-30 to 150	-30 to 150
Non-Operating Humidity (Non-Condensing)	5 – 95%	5 – 95%

Notes: (1) Integral 100 – 250 VAC ±10% at 50 or 60 Hz (Standard), single phase or two phase.

(2) 208 VAC ±10% at 60 Hz or 220 VAC ±10% at 50 Hz (Standard). 200, 230, 240, or 250 VAC ±10% at 50 or 60 Hz (Optional), three phase.

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000

T16/8714-00

TANDEM

MAGNETIC TAPE SUBSYSTEMS

- **Ultra-Reliable, Multiple-Speed (45, 75 or 125 IPS) Tape Drives for Critical NonStop™ Transaction-Oriented Applications**
- **NRZI (800 BPI) and/or PE (1600 BPI) Recording Format and ANSI/ECMA Format Compatibility**
- **IBM 9-track Compatible with Vertical Parity Generation and Check, Longitudinal Parity Generation and Check, CRCC Generation, Write Echo and Over-Write**
- **Time-Saving Performance with Speedy Local or Remote Fast-Forward (150 IPS), Rewind (300 IPS) and Unload Modes**
- **Economical Maintainability with Modular Design, Extensive Diagnostics and Self-Contained Diagnostic Module for Off-Line Testing**

SUBSYSTEM CONFIGURATION

Tandem's Magnetic Tape Subsystem provides the economy, reliability, and performance expected in a non-stop computer system. With fewer parts, front-end access and superior specifications, no other tape drive is easier to maintain. The standard unit features electronic write deskew electronics, with forward and reverse read deskew, as well as a unique tension arm sensor which eliminates all conventional lamps, photocells and rheostats. In summary, this rugged unit has been human-engineered to reduce operational system overhead and cut service costs.

A basic Magnetic Tape Subsystem consists of one or two Model T16/5101 or T16/5103 Magnetic Tape Drives and a Model T16/3201 or T16/3202 Magnetic Tape Controller. The standard tape unit is a 45 ips, 9-track, PE 1600 BPI or NRZI 800 BPI drive with read-after-write and power-fail/auto-restart electronics. The dual-port Controller handles up to two independently connected tape drives without the conventional requirement for daisy-chaining. In the event of a single processor failure, the dual-port Controller is powered from the second processor. Similarly, a faulty tape drive cannot corrupt a processor since the address and count words are held in the I/O processor – not the tape controller.

MODEL T16/5101 TAPE DRIVE

The Model T16/5101 Magnetic Tape Drive is a moderately-priced, medium-speed (45 ips) direct-drive unit which accommodates 10-1/2-inch reels, 8-1/2-inch reels or mini reels. The drive is fully compatible with IBM 9-track formats. Recording is NRZI at 800 BPI.

TRANSPORT – The tape drive is mounted on a precision casting which provides a completely rigid drive structure to eliminate shimming. A unique reel drive sensor precisely positions the buffer area and eliminates many parts so that there is virtually nothing to wear out. The head gate and buffer arms retract automatically to simplify tape threading. Unloading is completely automatic. A single REWIND/UNLOAD command causes the tape to rewind to loadpoint and unload after loadpoint.

MODEL T16/5103 TAPE DRIVE

The Model T16/5103 Magnetic Tape Drive is a moderately-priced, medium speed (45 ips) direct drive unit which can accommodate 10-1/2, 8-1/2 or 6-1/2 inch reels of 1/2" tape. The drive is fully compatible with IBM 9-track formats. Recording mode is dual-density, 800 BPI NRZI or 1600 BPI PE,

switch selectable. The transport mechanism is identical to the T16/5101.

MODEL T16/3201 MAGNETIC TAPE CONTROLLER

The Model T16/3201 Magnetic Tape Controller provides for a maximum two tape transports. Each transport may be run at 800 BPI (NRZI). Tape speed selection is available at 45, 75 or 125 IPS. NRZI recording mode conforms to American National Standards (ANSI X3.22-1973) for 800 BPI.

MODEL T16/3202 MAGNETIC TAPE CONTROLLER

The Model T16/3102 Magnetic Tape Controller provides for a maximum two tape transports. Each transport may be run at 800 BPI (NRZI) or 1600 BPI (PE). Tape speed selection is available at 45, 75 or 125 IPS. NRZI recording mode conforms to American National Standards (ANSI X3.22-1973) for 800 BPI. Phase Encoded (PE) recording conforms to American National Standards (ANSI X3.39-1973) for 1600 BPI.

SPECIFICATIONS (T16/5101 and T16/5103)

Controls and indicators	Power On, Forward/Load, Reverse, Rewind/Unload, Fast Forward, Density, On Line, Stop/Reset, Load Point, High Density, File Protect, Not Ready, Local, On-Line
Power input	Voltage: 100-250 VAC ±10% with transformer taps, 48 to 63 Hz, 430 watts maximum single phase
Operating environment	Ambient temperature: 2° to 55°C (transport without tape)
	Relative humidity: 20% to 95% (non-condensing)
Features	Vertical parity check
	Vertical parity generate
	Write echo
	Longitudinal parity check
	Longitudinal character generation
	Cyclic redundancy check character generation
	Phase Encode electronics or NRZI electronics — Vertical parity generation and check; longitudinal parity generation and check; CRCC generation, check and correction; and correction of single-track errors during reread operations.
	Quick release hold-down knob for lower reel assembly.

TANDEM

LINEPRINTER SUBSYSTEMS

- **Economical Medium to High-Speed (300 to 1500 lpm) Field-Proven Performance**
- **Wide Selection of Vertical and Horizontal Form Formats**
- **Versatile ASCII or OCR Compatible Character Sets (64 or 96 Characters in 132 columns)**
- **Rugged Solid-State Reliability and Easy Modularized Maintainability with Self-Contained Service Aids**

SUBSYSTEM CONFIGURATION

The Tandem Lineprinter Subsystems were especially selected to offer the user a wide choice in price/performance trade-offs to meet individual transaction-oriented application needs. The standard subsystems consist of one or more Series T16/5500 Lineprinters and a Model T16/3302 Lineprinter Controller. The controller incorporates Tandem's NonStop™ feature of two independent I/O ports. In the event of a single processor failure, the controller automatically switches over to the second processor. In addition, a Lineprinter failure cannot corrupt a processor since the critical control parameters are held in the I/O processor – not the Lineprinter Controller.

The Tandem Lineprinter Subsystems were human-engineered to provide easy on-line non-stop maintenance and expandability in the field. The rugged modularity concept employed ensures that these subsystems perform their function with a minimum of preventive and/or corrective maintenance downtime.

MODEL T16/5502 LINEPRINTER

The Model T16/5502 Lineprinter is a medium-priced, low-speed (300 lpm) drum unit featuring a 12-channel Vertical Format Unit (VFU). The VFU utiljzes IBM-compatible carriage tapes. The unit provides 132 columns of either 64 or 96 (optional) ASCII characters with a spacing of 10 characters/inch horizontally and 6 or 8 lines/inch vertically. Optionally, the unit features OCR character set with appropriate fonts. For service and maintenance, an optional self-test code generator is also available. This unit accepts up to 6-part fanfold edge-punched forms 4 to 16.75 inches wide.

MODEL T16/5503 LINEPRINTER

The Model T16/5503 Lineprinter is a low-priced, medium speed (600 lpm) drum unit incorporating the same features as the Model T16/5502 described above. The minimum printing rate is maintained at 436 lpm even when the optional 96 character set is incorporated.

MODEL T16/5504 LINEPRINTER

The Model T16/5504 Lineprinter is a moderately-priced, medium-speed (900 lpm) drum unit identical to the Model T16/5502 described above. This rugged, compact unit also accepts up to 6-part forms.

MODEL T16/5505 LINEPRINTER

The Model T16/5505 Lineprinter is a high-performance, high-speed (1500 lpm) train unit ideally suited for high-volume on-line transaction-oriented applications where speed and reliability are critical. The unit incorporates all the features of the Model T16/5502 and, in addition, provides a powered paper stacker. This unit also handles up to 6-part fanfold edge-punched forms 4 to 16.75 inches wide.

OPTIONS

DESCRIPTION	MODEL			
	5502	5503	5504	5505
ASCII 96-character Set	YES	YES	YES	YES
OCR Character Set	YES	YES	YES	YES
Vertical Format Unit (VFU)	YES	YES	YES	YES
Elapsed Time Meter	YES	YES	YES	YES
Self-Test Feature	YES	YES	YES	YES
Static Eliminator	YES	YES	YES	YES
Pedestal Mount	STD	STD	STD	STD

Notes: ① Standard

SPECIFICATIONS

PARAMETER	MODEL			
	5502	5503	5504	5505
PHYSICAL:				
Dimensions (L x W x H)	26 x 33 x 45	26 x 33 x 45	36 x 33 x 45	24 x 48 x 46
Weight (Pounds)	340	370	420	800
ELECTRICAL:				
Voltage (Single Phase VAC)	1	1	1	230 ± 10%
Frequency (Hz)	60 ± 2.0 50 ± 2.0	60 ± 2.0 50 ± 2.0	60 ± 2.0 50 ± 2.0	60 ± 3.0 50 ± 3.0
Power Rating (Watts)	525	680	825	<3 KW
ENVIRONMENTAL:				
Temperature (°C)	10 – 37	10 – 37	10 – 37	10 – 43
Humidity (Non-Condensing)	30 – 80%	30 – 80%	30 – 80%	10 – 90%
PRINTING:				
Type	Drum	Drum	Drum	Drum
Speed (64 Character Set lpm)	340	600	900	1500
Speed (96 Character Set lpm)	240	436	660	1220
Columns (Standard)	132	132	132	132
Horizontal Spacing (Characters/Inch)	10	10	10	10
Vertical Spacing (Lines/Inch)	6/8	6/8	6/8	6/8
Paper Widths (Inches)	4 – 16.75	4 – 16.75	4 – 16.75	4 – 16.75
TRANSMISSION:				
Code (Standard)	ASCII	ASCII	ASCII	ASCII

Note: ① 100, 115, 125, 200, 220 or 240 ± 10%

TANDEM
COMMUNICATION CONTROLLERS

SYNCHRONOUS CONTROLLER

FEATURES

- **Supports four lines per controller with speeds up to 56K Bps per line**
- **Full or Half Duplex synchronous operation**
- **Automatic Generation and Detection of Block Check Characters with support for VRC, LRC and CRC16 Modes of Operation**
- **Automatic Code Translation to ASCII & EBCDIC**
- **Automatic Polling Capability provided by the Controller for Multipoint Environments**
- **DMA Access to Main Memory**
- **Supports Transparency and Auto-insertion of "DLE" and "SYN" characters**
- **Supports Bell-type 201, 203, 208 and 209 modems**

INTRODUCTION

The T16/6201 Synchronous Communications Adapter, utilizing microprogrammed technology, provides a high speed, intelligent interface for synchronous communications environments. When used with Tandem's ENVOY Data Communication Manager a powerful, yet flexible data communications system can be designed with a minimum of effort.

All input/output is directly between main memory and the controller. CPU processing is interrupted only when a transfer of a message completes or a line error condition is encountered.

The controller generates and appends check character information upon transmission and performs error checking upon reception.

The controller automatically recognizes the transmission of transparent text. If the control sequence [DLE-STX] is encountered at the beginning of a message transfer, the controller enters the transparent text mode. In this mode, the controller will insert a control [DLE] character in front of a data [DLE] character when transmitting. Conversely, when receiving transparent text, the controller will delete a control [DLE] character in front a data [DLE] character. The controller remains in the transparent text mode until an [ETB], [ETX], or [EOT] control sequence is encountered.

The controller's internal character code, as far as detecting control character sequences is concerned, is ASCII. The controller has the capability to translate ASCII code to EBCDIC code upon transmission and to translate EBCDIC code to ASCII code upon reception. Because of this capability, application processes need deal only with the ASCII character set.

The polling of multipoint stations is, for the most part, handled by the controller. ENVOY formats a polling list (on behalf of the application process) for the controller to use to poll the multipoint tributary stations. ENVOY then commands the controller to begin polling. CPU processing is interrupted only when a polled station responds.

The controller has the capability to recognize if a line is being polled or selected. For each line, the controller stores the first byte of the station's polling address and the first byte of the station's selection address. Only when the line is polled or selected and the corresponding poll or select byte matches is CPU processing interrupted.

OPERATION

Each line can be configured dynamically for

- Translate Enable
- Transparent Text Capability
- Full or Half Duplex Operation
- Polling Address
- Selection Address

In addition, instructions are available to:

- Answer/Hang Up the Phone
  Sets or Clears the modem control signal Data Terminal Ready (DTR).
- Initiate Write
  Initiates a Write operation
- Initialize Read Control
  Sets up a Read Operation which will follow an automatic line turnaround
- Initiate Read
  Initiates a Read operation
- Terminate Read
  Terminates a Read Continuous operation
- Stop Polling
  Terminates auto polling for a line and interrupts

ASYNCHRONOUS CONTROLLER

FEATURES

- **2 to 32 asynchronous lines per controller with line speeds from 50 to 19.2K Bps**
- **Point-to-Point or Multipoint Modes of Operation**
- **DMA Access to Main Memory on all I/O Transfers**
- **EIA (RS232) or Current Loop Interfaces**
- **Modem support for Bell-type 103 and 202 (including reverse channel)**
- **Each line individually programmable with respect to;**
  - **Baud Rate.**
  - **Character Size.**
  - **Computer Parity Generation.**
  - **Computer Parity Checking.**
  - **Connection Type.**
  - **Enable/disable Checking for Signal Characters.**
  - **Half-Duplex Modem Turn-around Character(s).**
  - **Read Completion on ETX Character.**
  - **Default Transfer Mode.**
  - **Conversational Mode Line Termination Character.**
  - **Conversational Mode Automatic Linefeed on Input.**
  - **Conversational Mode Backspace Type.**
  - **Conversational Mode Carriage Return/Line Feed Delay.**
  - **Conversational Mode Forms Control Delay.**
  - **Page Mode Page Termination Character.**
  - **Page Mode Psuedo-Polling Trigger Character.**

INTRODUCTION

The T16/6301/6302 Asynchronous Controller provides a flexible interface for all types of asynchronous communications. For point-to-point applications the standard GUARDIAN I/O Subsystem will support any RS232 or Current Loop Terminal by merely configuring the line in SYSGEN. Standard I/O calls (READ, WRITE, WRITEREAD) provide access to the terminal. For multipoint applications ENVOY provides the ability to interface polling terminals such as Tandem's T16/6552 CRT Polling Terminal or TI's TINET Data Entry Pads. The Asynchronous Controller is sufficiently fast to support 32 lines all running at 19.2K Bps.

Both conversational and page mode terminals are supported. Conversational terminals can specify, as a SYSGEN parameter, the characters which will be interpreted as line termination, line cancel, backspace and end-of-file. Page mode terminals can specify, as a SYSGEN parameter, the characters which will be interpreted as Page Termination and Pseudo-poll Triggers.

OPERATION

Instructions are available to;

- Initiate a Read
- Initiate a Write
- Initiate a Write Read
  Writes data to a terminal, then waits for data to be read back from terminal
- Setmode
  Sets or clears
  - single spacing
  - auto linefeed
  - conversation/page mode
  - signal characters
  - parity checking
  - break ownership
  - access mode
  - read termination on ETX
  - read termination on signal character
- Control
  Used for form control and modem connect/disconnect

TANDEM

TERMINAL SUBSYSTEMS

- **High Performance Terminals for NonStop(tm) Transaction-Oriented Applications**
- **Select from Keyboard/Printer, CRTs or Receive-Only Printer**
- **Asynchronous RS232 or 20mA Current Loop Interfaces**
- **CRT Speeds to 19.2K bps**
- **Serial Printers Speeds to 200 cps**
- **Field Proven Reliability**

SUBSYSTEM CONFIGURATIONS

The Tandem Terminal Subsystems offer the user a wide selection of operational benefits to maximize the Non-Stop features of the system and minimize program development overhead. All terminals meet or exceed ANSI, EIA and ECMA standards in addition to being UL certified. A wide range of options are also offered to meet the individual needs of a customized or diversified application.

A basic Terminal Subsystem consists of a Multiplexer, a System Console and up to 16 additional terminal lines. The Multiplexer is comprised of a Model T16/6301 Asynchronous Controller and one or two Model T16/6302 Asynchronous Extension Interfaces. The dual-port Controller handles up to 32 independent communication lines for modem or direct-wire connection compatible with single- or multi-drop terminals. One line is dedicated to the hard- or soft-copy system console. Each Asynchronous Extension provides an interface for up to 15 additional terminal lines.

MODELS T16/6603-6604 KEYBOARD/PRINTER TERMINALS

The Models T16/6603 and T16/6604 terminals provide hard-copy output combined with keyboard input at a low cost. Both models offer a 59 character keyboard, 30 characters per second printing using a 5 X 7 dot matrix, 132 characters per line and the ability to print all 96 ASCII characters while transmitting 128 ASCII characters. Each model is suitable for operator logging, error message logging, and operator input to the Tandem system.

Both models are identical in every respect except interfacing. The Model T16/6603 uses an EIA RS232 interface while the T16/6604 uses a 20 mA current loop interface permitting hard-wired operation up to 1500 feet away.

MODELS T16/6401-6402 CRT TERMINALS

The Models T16/6401 and T16/6402 CRT Terminals provide the performance and reliability of cathode ray tubes at less cost than keyboard/printer terminals. Both models incorporate a compact 12 inch CRT which displays 24 lines by 80 characters, or 1920 characters per screen. The standard keyboard provides 59 keys for ease of data entry. Upper Case/Lower Case may be chosen as an option. Transmission rates are switch selectable from 75 to 19,200 bps. Thus display rates up to 1920 characters per second are attainable.

Both models are identical in every respect except interfacing. The Model T16/6401 uses an EIA RS232 interface while the T16/6402 uses a 20mA current loop interface permitting hard-wired operation up to 1500 feet away.

MODELS T16/6511-6512 PAGE-MODE CRT TERMINALS

The Models T16/6511 and T16/6512 CRT Terminals provide increased capability over the T16/6401-6402 CRT Terminals by the use of page-mode display screens and added keyboard functions. Both models incorporate a compact 12 inch CRT which displays 24

lines by 80 characters, or 1920 characters per screen. The standard screen provides the ability to define protected and data entry fields for page-mode operation. The standard keyboard provides 68 keys for ease of data entry. In addition to the standard keys a ten-key pad, 16 function keys, edit function keys, and cursor control keys are provided. Upper Case/Lower Case is switch selectable. Transmission rate for both models is 9600 bps. Thus display rates up to 960 characters per second are attainable.

Both models are identical in every respect except interfacing. The Model T16/6511 uses an EIA RS232 interface while the T16/6512 uses a 20mA current loop interface permitting hard-wired operation up to 1500 feet away.

MODEL 5508 SERIAL PRINTER

The Model T16/5508 Serial Printer provides high performance serial printing at a modest cost. Using a 9-wire ballistic head, the T16/5508 generates 7 X 9 dot matrix characters at speeds of 200 characters per second. Printing is bi-directional to provide maximum throughput for NonStop applications. Both upper and lower case characters can be printed. A versatile electronic VFU provides flexible forms control.

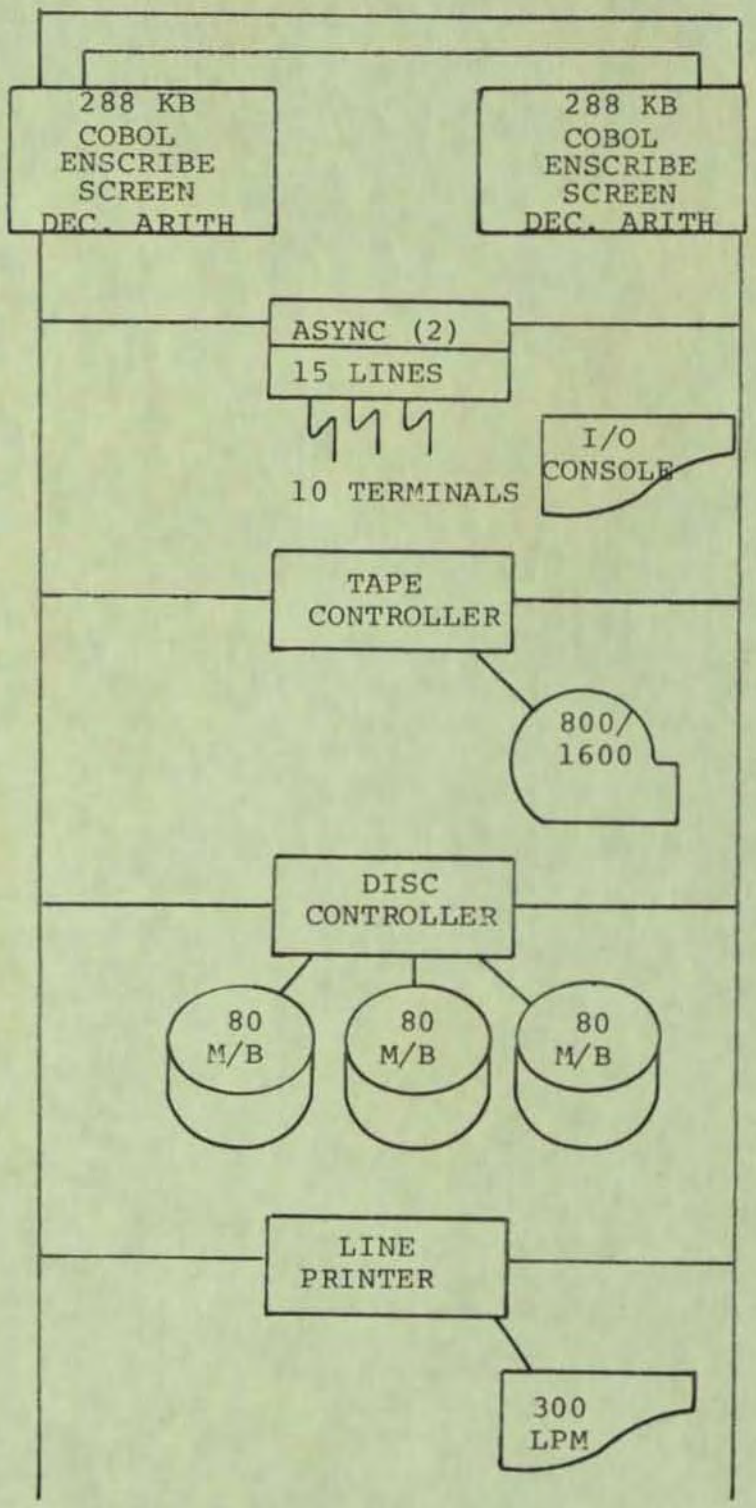
The T16/5508 uses a 20mA current loop interface. No special line printer interface is required since a current loop port on the Asynchronous Multiplexer can be used. The T16/5508 can be hard-wired up to 1500 feet from the Tandem system.

SPECIFICATIONS

	T16/6603-6604	T16/6401-6402	T16/6511-6512	T16/5508
Device Type	KYBD/PTR	CRT	CRT	R/O PTR
PRINT/DISPLAY/KYBD				
Type	Impact	CRT	CRT	Impact
Character Gen (Dot Matrix)	5 X 7	5 X 7	5 X 9	7 X 9
Columns	132	80	80	132
Lines	n/a	24	24	n/a
Character Display	n/a	1920	1920	n/a
Horizontal Spacing	10	n/a	n/a	10
Diagonal Display	n/a	12"	12"	n/a
VFU	no	n/a	n/a	yes
Keyboard	59 keys	59 keys	116 keys	n/a
Printing	Unidirectional	n/a	n/a	Bidirectional
Print Speed	30 cps	1920 cps	960 cps	200 cps
Transmission				
Code	ASCII	ASCII	ASCII	ASCII
Connection				
RS232	6603	6401	6511	n/a
20mA CL	6604	6402	6512	yes
Line Speed (bps)	300	75 to 19.2K	9600	9600
Mounting	Pedestal	Table-top	Table-top	Table-Top
Electrical				
Voltage	110V, 60Hz 220V, 50Hz	110V, 60Hz 230V, 50Hz	110V, 60Hz 230V, 50Hz	110V, 60Hz 220V, 50Hz
Operating Amps	3.1	0.8	2.0	4.0
Physical				
Dimensions (LxWxH)	24x28x34	20x16x13	24x21x13	16x23x13
Weight (lbs)	120	32	50	45
Environmental				
Temperature	50-104F	41-122F	41-122F	39-95F
Humidity	10-90%	5-95%	5-95%	20-90%

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, CA 95014 ■ (408) 996-6000

TANDEM 16 SYSTEM



TANDEM 16 SYSTEM

System Package	$ 87,100
Memory - 288 KB	19,200
Dec. Arithmetic	3,000
COBOL	1,000
ENSCRIBE	2,000
15 Lines	4,300
10 Page Mode Terminal	24,000
Terminal Patch	775
Disc Controller	10,500
(3) 80 M/Bytes	~~46,500~~
Disc Patch Panel	775
Line Printer Controller	2,800
Line Printer, 300 LPM	11,500
	$213,450

SOFTWARE

COBOL	$ 7,000
ENSCRIBE	4,000
SCREEN	2,000
SPOOLER	2,000
	$15,000

TANDEM

DATA DEFINITION LANGUAGE

FEATURES

- **Centralized Data Base Provides Common Resource for Numerous Application Programs, Ensures Data Consistency, Reduces Redundant Data, and Allows Easy Maintainability without Sacrificing System Security**
- **Automatic Space Management Allocates and Optimizes Storage Resources Enhancing File and Record Activities**
- **Data Definition Language (DDL) to Facilitate the Definition of a Centralized Data Base Schema**
- **Easy-to-Use EDIT and SCHEMA Programs to Facilitate Compilation of Application Programs with High-Level Language TAL Compiler and ENSCRIBE Data Base Record Manager Procedures**

INTRODUCTION

Tandem's *Data Definition Language* allows centralized administration of a data base to accommodate any number of diversified application programs. And the applications programmer need not know where files associated with a specific application are located. Since the data base is described as a *schema,* and all data field references are defined in the schema, changes to record layout, and additions and/or changes to record types and/or fields may be accomplished *without* code modifications to existing programs. The schema also provides the necessary information for query and reporting programs. The *Data Definition Language* (DDL) is used to describe the schema.

DATA BASE DEFINITION

As illustrated on the next page, the programmer defines a data base in a *Schema Definition File* using the Tandem-supplied EDIT program. This example is used to define records for a customer data base. The *name* assigned to the record type is CUSTOMER. The fields are named ACCTNO, NAME, FIRST, STREET, CITY, STATE, ZIP, BAL. CHARACTER means that the field contains a string of ASCII characters and the number defines the length of each field in bytes. BINARY indicates arithmetic data. A O following KEYTAG means that the field ACCTNO is the record's primary key and "NM" means that the field NAME is an alternate key (other fields are not defined as key fields).

APPLICATION PROGRAM COMPILATION

The *Schema Definition File* is used as the input to the Tandem-supplied SCHEMA program. This program generates a T/TAL library file that, when compiled along with the application program, produces an object application program that is tailored to the particular data base. (See example on the next page).

DATA BASE ACCESS

Record types are accessed through ENSCRIBE provided with the GUARDIAN Operating System. Fields in a record type are accessed by *field identifiers.* As shown in the examples, a field identifier is a concatenation of the name of the record and the name of the field. For example, the field identifiers for the CUSTOMER record defined in the example are:

<field name>	<field identifier>
ACCTNO	CUSTOMER^ACCTNO
NAME	CUSTOMER^NAME
STATE	CUSTOMER^STATE

Each field defined as type CHARACTER also has a corresponding *field length identifier* that can be used to determine the length in bytes of a field. A *field length identifier* is the concatenation of the *record name,* the *field name,* and the string LEN. For example, the *field length identifier* for the NAME field is:

CUSTOMER^NAME^LEN

CUSTOMER^NAME^LEN has the value "25" when referenced in the program.

Each field which was defined to have a key tag will have a *key tag identifier* that can be used to pass the key tag to the file system. The form of this identifier is a concatenation of the *field identifier* and KEY. For example, the *key tag identifier* for the field NAME is:

CUSTOMER^NAME^KEY

CUSTOMER^NAME^KEY has the value of "null" when referenced in a program.

Defining the Data Base

```
* SCHEMA DEFINITION
*
* CONSISTS OF TWO TYPES OF RECORDS
*       1. CUSTOMER MASTER RECORD
*       2. TRANSACTION LOG RECORD
*
RECORD CUSTOMER.
       03 ACCTNO; TYPE CHARACTER  5; KEYTAG 0.
       03 NAME;   TYPE CHARACTER 25; KEYTAG "NM".
       03 FIRST;  TYPE CHARACTER 25.
       03 STREET; TYPE CHARACTER 56.
       03 CITY;   TYPE CHARACTER 25.
       03 STATE;  TYPE CHARACTER 25.
       03 ZIP;    TYPE CHARACTER  5.
       03 BAL;    TYPE BINARY 64,2.
*
RECORD TRANSACT.
       03 ACCTNO; TYPE CHARACTER  5; KEYTAG 0.
       03 NAME;   TYPE CHARACTER 25; KEYTAG "NM".
       03 DATE;                      KEYTAG "DT".
          05 YEAR;    TYPE CHARACTER 2.
          05 MONTH;   TYPE CHARACTER 2.
          05 DAY;     TYPE CHARACTER 2.
       03 CODE;   TYPE CHARACTER  1.
*                "+" = TRANSACTION IS DEPOSIT
*                "-" = TRANSACTION IS WITHDRAWL
       03 AMT;    TYPE BINARY 64,2.
       03 TIME;                      KEYTAG "TM".
          05 HOUR;    TYPE CHARACTER 2.
          05 MINUTE;  TYPE CHARACTER 2.
```

Compiling the Schema

```
SCHEMA /IN DDLFILE, OUT $LP/ APPLIB
```

Example: Accessing the Data Base

```
Application Global Declarations
        .
        .
! Including the schema source in the application program

?SOURCE SCHEMA(CUSTOMER,TRANSACT)
ALLOCATE^CUSTOMER;
ALLOCATE^TRANSACT;

PROC USER^APPLICATION;
        .
        .
! Using the Schema Definitions

 CUSTOMER^ACCTNO ':=' FORMOPEN^ACCT FOR CUSTOMER^ACCTNO^LEN;
 CUSTOMER^NAME   ':=' FORMOPEN^NAME FOR CUSTOMER^NAME^LEN;
 CUSTOMER^FIRST  ':=' FORMOPEN^FIRST FOR CUSTOMER^FIRST^LEN;
 CUSTOMER^STREET ':=' FORMOPEN^STREET FOR CUSTOMER^STREET^LEN;
 CUSTOMER^CITY   ':=' FORMOPEN^CITY FOR CUSTOMER^CITY^LEN;
 CUSTOMER^STATE  ':=' FORMOPEN^STATE FOR CUSTOMER^STATE^LEN;
 CUSTOMER^ZIP    ':=' FORMOPEN^ZIP FOR CUSTOMER^ZIP^LEN;
        .
        .
! Insert a new Customer Record into the File
 CALL WRITE(CUST,CUSTOMER,CUSTOMER^LEN,CNT);
        .
        .
        .
! Position to Customer Record for Subsequent Update
 CALL KEYPOSITION(CUST,FORMDEF^ACCT,,,EXACT);
 IF < THEN BEGIN ...
        .
        .
        .
! Get Customer Record and Lock It
 CALL READUPDATELOCK(CUST,CUSTOMER,CUSTOMER^LEN,CNT);
 IF < THEN BEGIN
          CALL FILEINFO(CUST,FILE^ERR);
          IF FILE^ERR=11
          THEN BEGIN ! RECORD NOT FOUND
                  .
                  .
```

TANDEM
COMMUNICATION CONTROLLERS

SYNCHRONOUS CONTROLLER

FEATURES

- **Supports four lines per controller with speeds up to 56K Bps per line**
- **Full or Half Duplex synchronous operation**
- **Automatic Generation and Detection of Block Check Characters with support for VRC, LRC and CRC16 Modes of Operation**
- **Automatic Code Translation to ASCII & EBCDIC**
- **Automatic Polling Capability provided by the Controller for Multipoint Environments**
- **DMA Access to Main Memory**
- **Supports Transparency and Auto-insertion of "DLE" and "SYN" characters**
- **Supports Bell-type 201, 203, 208 and 209 modems**

INTRODUCTION

The T16/6201 Synchronous Communications Adapter, utilizing microprogrammed technology, provides a high speed, intelligent interface for synchronous communications environments. When used with Tandem's ENVOY Data Communication Manager a powerful, yet flexible data communications system can be designed with a minimum of effort.

All input/output is directly between main memory and the controller. CPU processing is interrupted only when a transfer of a message completes or a line error condition is encountered.

The controller generates and appends check character information upon transmission and performs error checking upon reception.

The controller automatically recognizes the transmission of transparent text. If the control sequence [DLE-STX] is encountered at the beginning of a message transfer, the controller enters the transparent text mode. In this mode, the controller will insert a control [DLE] character in front of a data [DLE] character when transmitting. Conversely, when receiving transparent text, the controller will delete a control [DLE] character in front a data [DLE] character. The controller remains in the transparent text mode until an [ETB], [ETX], or [EOT] control sequence is encountered.

The controller's internal character code, as far as detecting control character sequences is concerned, is ASCII. The controller has the capability to translate ASCII code to EBCDIC code upon transmission and to translate EBCDIC code to ASCII code upon reception. Because of this capability, application processes need deal only with the ASCII character set.

The polling of multipoint stations is, for the most part, handled by the controller. ENVOY formats a polling list (on behalf of the application process) for the controller to use to poll the multipoint tributary stations. ENVOY then commands the controller to begin polling. CPU processing is interrupted only when a polled station responds.

The controller has the capability to recognize if a line is being polled or selected. For each line, the controller stores the first byte of the station's polling address and the first byte of the station's selection address. Only when the line is polled or selected and the corresponding poll or select byte matches is CPU processing interrupted.

OPERATION

Each line can be configured dynamically for

- Translate Enable
- Transparent Text Capability
- Full or Half Duplex Operation
- Polling Address
- Selection Address

In addition, instructions are available to;

- Answer/Hang Up the Phone
  Sets or Clears the modem control signal Data Terminal Ready (DTR).
- Initiate Write
  Initiates a Write operation
- Initialize Read Control
  Sets up a Read Operation which will follow an automatic line turnaround
- Initiate Read
  Initiates a Read operation
- Terminate Read
  Terminates a Read Continuous operation
- Stop Polling
  Terminates auto polling for a line and interrupts

ASYNCHRONOUS CONTROLLER

FEATURES

- **2 to 32 asynchronous lines per controller with line speeds from 50 to 19.2K Bps**
- **Point-to-Point or Multipoint Modes of Operation**
- **DMA Access to Main Memory on all I/O Transfers**
- **EIA (RS232) or Current Loop Interfaces**
- **Modem support for Bell-type 103 and 202 (including reverse channel)**
- **Each line individually programmable with respect to;**
  - **Baud Rate.**
  - **Character Size.**
  - **Computer Parity Generation.**
  - **Computer Parity Checking.**
  - **Connection Type.**
  - **Enable/disable Checking for Signal Characters.**
  - **Half-Duplex Modem Turn-around Character(s).**
  - **Read Completion on ETX Character.**
  - **Default Transfer Mode.**
  - **Conversational Mode Line Termination Character.**
  - **Conversational Mode Automatic Linefeed on Input.**
  - **Conversational Mode Backspace Type.**
  - **Conversational Mode Carriage Return/Line Feed Delay.**
  - **Conversational Mode Forms Control Delay.**
  - **Page Mode Page Termination Character.**
  - **Page Mode Psuedo-Polling Trigger Character.**

INTRODUCTION

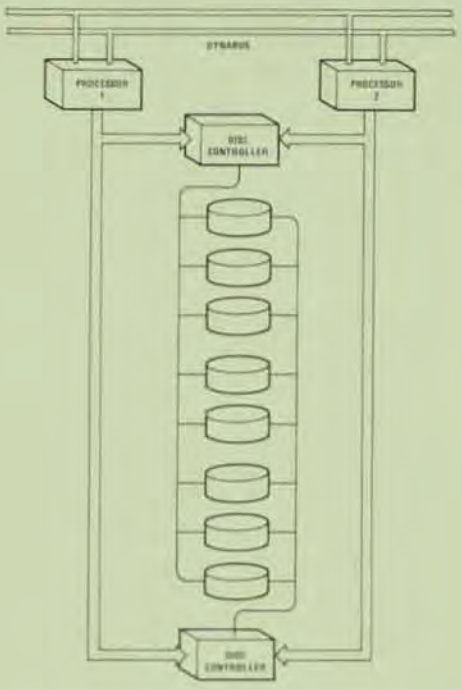
The T16/6301/6302 Asynchronous Controller provides a flexible interface for all types of asynchronous communications. For point-to-point applications the standard GUARDIAN I/O Subsystem will support any RS232 or Current Loop Terminal by merely configuring the line in SYSGEN. Standard I/O calls (READ, WRITE, WRITEREAD) provide access to the terminal. For multipoint applications ENVOY provides the ability to interface polling terminals such as Tandem's T16/6552 CRT Polling Terminal or TI's TINET Data Entry Pads. The Asynchronous Controller is sufficiently fast to support 32 lines all running at 19.2K Bps.

Both conversational and page mode terminals are supported. Conversational terminals can specify, as a SYSGEN parameter, the characters which will be interpreted as line termination, line cancel, backspace and end-of-file. Page mode terminals can specify, as a SYSGEN parameter, the characters which will be interpreted as Page Termination and Pseudo-poll Triggers.

OPERATION

Instructions are available to;

- Initiate a Read
- Initiate a Write
- Initiate a Write Read
  Writes data to a terminal, then waits for data to be read back from terminal
- Setmode
  Sets or clears
  - single spacing
  - auto linefeed
  - conversation/page mode
  - signal characters
  - parity checking
  - break ownership
  - access mode
  - read termination on ETX
  - read termination on signal character
- Control
  Used for form control and modem connect/disconnect



TANDEM

DUAL-PORT DISC SUBSYSTEMS

- **Field Proven Disc Drives for Transaction-oriented Systems**
- **High Capacity**
- **High Performance**
- **Dual-Ported Disc Drives for Data Integrity**
- **Reliable Servo Track Positioning for Accuracy**
- **Intelligent, microprogrammed Disc Controller**
- **Compact, Stand-alone Packaging**

SUBSYSTEM CONFIGURATION

The Tandem Dual-port Disc Subsystem provides high capacity, high performance, high availability and high data integrity for on-line, transaction-oriented systems. When used on the Tandem NonStop (tm) Computer System these disc subsystems provide the most cost effective solution for transaction-oriented applications where data integrity is of paramount importance.

The Model T16/3105 Disc Controller provides a high degree of flexibility and expandability. The disc controller is actually an intelligent, "backend" disc processor capable of supporting up to eight disc drives per controller. Because the controller is micro-programmed a wide variety of disc technologies and data base functions can be supported.

The T16/3105 Disc Controller connects to two (2) I/O channels simultaneously and may be powered from either processor in the event of a single processor failure. Thus, high availability is insured to the controller.

The T16/4103, T16/4104 and T16/4105 Disc Drives provide high capacity and performance through the use of new disc technologies. In addition high availability and data integrity are provided through dual-port connections. Each disc drive can connect to two (2) controllers simultaneously.

T16/3105 DISC CONTROLLER

The T16/3105 Disc Controller is used to interface up to eight disc drives consisting of any mix of T16/4103, T16/4104 or T16/4105 Disc Drives. The controller provides high availability to the drives through its dual-channel connection pioneered by Tandem in its other controllers.

Significant features of the controller include:

(1) Dual-channel connection

(2) 4K RAM Buffer for buffering logical records

(3) Dual recording or the ability to transfer the buffer to one or more disc units without having to re-transmit on the I/O bus

(4) Support for dual-access disc drives

(5) Read without I/O transfer which allows one disc to be copied to another without involving the I/O bus

The disc controller can be configured so that two controllers are connected to the same disc unit. Up to eight drives can be daisy-chained on one controller. With the dual-ported disc option daisy-chain failures will not cause a loss of the data base, a loss of data base integrity, or a loss of an access path to the data base.

TANDEM DUAL-PORT DISC SUBSYSTEMS

TANDEM DUAL-PORT DISC SUBSYSTEMS

T16/4103, 4104 AND 4105 DISC DRIVES

The T16/4103, 4104 and 4105 are stand-alone disc drives using the latest in disc technologies to provide high capacity and performance.

The T16/4103 uses 3330 technology to provide a storage capacity of 160M bytes per pack (formatted). Performance data includes an average access time of 28 ms, average latency of 8.35 ms, and a data transfer rate of 806K bytes/second. Data is recorded on an eleven-high pack. Servo tracking insures accuracy of track following.

The T16/4104 uses SMD technology to provide a storage capacity of 240M bytes per pack (formatted). Performance data includes an average access time of 28 ms, average latency of 8.35 ms, and a data transfer rate of 1.2M bytes/second. Data is recorded on an eleven-high pack. Servo tracking insures accuracy of track following.

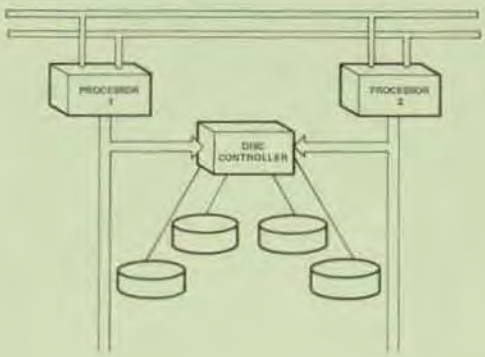
The T16/4105 uses SMD technology to provide a storage capacity of 64M bytes per pack (formatted). Performance data includes an average access time of 30 ms, average latency of 8.35 ms, and a data transfer rate of 1.2M bytes/second. Data is recorded on a five-high pack. Servo tracking insures accuracy of track following.

Each of the drives has an option of being dual-ported, the ability to connect to two (2) T16/3105 Disc Controllers simultaneously. If a disc controller or the daisy-chain should fail, the other controller connection will insure access to the data base.

Each drive uses a linear DC motor (voice coil) for accurate and high speed seeking. Reliable TTL and ECL circuits are used throughout.

SPECIFICATIONS	T16/4103	T16/4104	T16/4105
RECORDING CAPACITY			
Capacity (Unformatted)	200M Bytes	300M Bytes	80M Bytes
Capacity (Formatted)	160M Bytes	240M Bytes	64M Bytes
Recording Mode	MFM	MFM	MFM
Recording Density	4040 BPI	6060 BPI	6038 BPI
Tracks per Surface	808 + 7 alt	808 + 7 alt	808 + 15 alt
Tracks per Inch	384	384	384
Data Surfaces	19	19	5
Servo Surfaces	1	1	1
PROCESSING SPEED			
Data Transfer Rate	806K Bytes/Sec	1.2M Bytes/Sec	1.2M Bytes/Sec
Spindle Speed	3600 RPM	3600 RPM	3600 RPM
Latency (average)	8.35 ms	8.35 ms	8.35 ms
ACCESS SPEEDS			
Minimum (one track)	10 ms	10 ms	6 ms
Average	28 ms	28 ms	30 ms
Maximum	55 ms	55 ms	55 ms
DISC PACKS	3M 936/11	Dysan (300MB)	CDC 9877
(all packs must be flag free)	CDC 9882 Memorex Mark XI Dysan		
PHYSICAL			
Dimensions (D x W x H)	34 x 19 x 38	34 x 19 x 38	34 x 19 x 34
Weight	465 lbs	465 lbs	243 lbs
ELECTRICAL			
Voltage	208V, 60 Hz 220/240V, 50 Hz	208V, 60 Hz 220/240V, 50 Hz	120V, 60 Hz 220/240V, 50 Hz
Phases	Three	Three	Single
Operating Amps	6.4	6.4	8.2
ENVIRONMENTAL			
Temperature (operating)	60–90 F	60–90 F	59–90 F
Humidity (operating)	20–80%	20–80%	20–80%
Heat Dissipation	5800 BTU/hr	5800 BTU/hr	1800 BTU/hr

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, CA 95014 ■ (408) 996-6000



TANDEM
SINGLE - PORT DISC SUBSYSTEMS

- **Tough, Service-Engineered Disc Drives for Critical NonStop™ Real Time Transaction-Oriented Applications**
- **Low-Cost High Performance Medium- to High-Capacity (10 to 50 Megabytes) Disc Drives with Proven Dependability, System Versatility and Modular Maintainability**
- **Economical High-Speed Reliability with Advanced Hybrid and CMOS Technology**
- **Fast Data Transfer Rate (312K Bytes/Second) for Efficient and Economical Data Processing**
- **Wide Choice of Optional Features to Facilitate NonStop™ System Configuration or Expansion in the Field.**

SUBSYSTEM CONFIGURATION

The Tandem Disc Subsystems were particularly selected to provide low-cost, high reliability, high data capacity, fast data transfer and easy as well as economical maintainability. All these parameters are crucial to the exacting requirements of a non-stop transaction-oriented environment.

A standard Disc Subsystem consists of a Disc Controller and one or more Moving Head Disc Drives. The Controller features two independent I/O Channel ports and may be powered from either system processor in the event of a single processor failure. In addition, a disc failure cannot corrupt the system processor's memory since the address and count words are held in the I/O processor – not the disc controller.

The Model T16/4101 is a 10MB capacity drive using moving heads while the Model T16/4102 is a 50MB capacity drive using moving heads. The Model T16/3101 Disc Controller can accommodate 1 to 4 disc drives in any mix of T16/4101 and T16/4102 Disc Drive configurations. Each drive connected to the controller has its own signal and data cable. No daisy chain is used.

MODEL T16/4101 DISC DRIVE

The T16/4101 Disc Drive is a low-cost, medium capacity (10M Byte) disc drive featuring a 312KB transfer rate, high-speed random-access and pedestal-mounted cartridge disc drive. The unit is comprised of one fixed platter of 5MB and one removable top-loading platter of 5MB with average seek and latency times of 35 and 12.5 milliseconds, respectively. This rugged unit also features: a service-oriented modular design which eliminates all field adjustments, and an automatic cleaning system which purges and cleans the disc area to reduce costly preventive maintenance.

MODEL T16/4102 DISC DRIVE

The Model T16/4102 Disc Drive is a moderately-priced, high-capacity (50M Bytes) disc drive ideally suited for large data bases. This high-performance unit is a random-access, pedestal-mounted disc pack drive and is directly I/O interchangeable with the Model T16/4101 cartridge disc drive (described above) when the Model T16/3102 Controller is installed. The unit uses CDC® 9873 Disk Packs, or equivalent, for the storage and retrieval of data. Each removable pack contains 11 discs. Average seek time is 35 ms. Average latency is 12.5 ms. The transfer rate is 312KB.

TANDEM SINGLE-PORT DISC SUBSYSTEMS

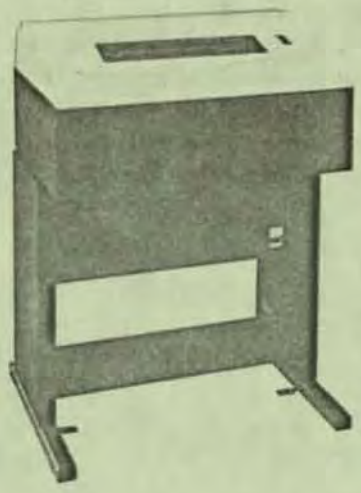
SPECIFICATIONS

PARAMETER	MODEL	
	4101	4102
RECORDING CAPACITY		
Capacity (Maximum Megabytes, Formatted)	10MB	50MB
Recording Mode	Double Freq.	Double Freq.
Recording Density (BPI - Nominal)	2200	2200
Tracks Per Surface (Maximum)	408	406
PROCESSING SPEED		
Data Transfer Rate (K Bytes/Second)	312	312
Spindle Speed (RPM)	2400	2400
ACCESS TIME (Direct Seek)		
Full Stroke (Milliseconds)	60	70
Average (Milliseconds)	35	35
One-Track (Milliseconds)	7	10
DISC CARTRIDGE/PACK		
Number of Discs	1	11
Usable Surfaces	2	20
Pack Type	5440 equiv.	2314-2 equiv.

PARAMETER	MODEL	
	4101	4102
RECORDING HEADS	4	20
PHYSICAL:		
Dimensions (L x W x H Inches)	29-3/4 x 18-1/2 x 34	32-1/2 x 27-1/2 x 38
Weight (Pounds)	252	700
ELECTRICAL:		
Voltage	1	2
Frequency (Hz) Disc	50 ±1.0 60 ±1.0	50 ±1.0 60 ±1.0
Power Rating (Operating AMPS)	4.6	5.0
Power Rating (Surge AMPS)	9.2	22.0
ENVIRONMENTAL:		
Operating Temperature (°F)	60 – 90	60 – 90
Operating Humidity (Non-Condensing)	10 – 90%	10 – 80%
Non-Operating Temperature (°F)	-30 to 150	-30 to 150
Non-Operating Humidity (Non-Condensing)	5 – 95%	5 – 95%

Notes: (1) Integral 100 – 250 VAC ±10% at 50 or 60 Hz (Standard), single phase or two phase.

(2) 208 VAC ±10% at 60 Hz or 220 VAC ±10% at 50 Hz (Standard). 200, 230, 240, or 250 VAC ±10% at 50 or 60 Hz (Optional), three phase.



TANDEM

LINEPRINTER SUBSYSTEMS

- **Economical Medium to High-Speed (300 to 1500 lpm) Field-Proven Performance**
- **Wide Selection of Vertical and Horizontal Form Formats**
- **Versatile ASCII or OCR Compatible Character Sets (64 or 96 Characters in 132 columns)**
- **Rugged Solid-State Reliability and Easy Modularized Maintainability with Self-Contained Service Aids**

SUBSYSTEM CONFIGURATION

The Tandem Lineprinter Subsystems were especially selected to offer the user a wide choice in price/performance trade-offs to meet individual transaction-oriented application needs. The standard subsystems consist of one or more Series T16/5500 Lineprinters and a Model T16/3302 Lineprinter Controller. The controller incorporates Tandem's NonStop™ feature of two independent I/O ports. In the event of a single processor failure, the controller automatically switches over to the second processor. In addition, a Lineprinter failure cannot corrupt a processor since the critical control parameters are held in the I/O processor — not the Lineprinter Controller.

The Tandem Lineprinter Subsystems were human-engineered to provide easy on-line non-stop maintenance and expandability in the field. The rugged modularity concept employed ensures that these subsystems perform their function with a minimum of preventive and/or corrective maintenance downtime.

MODEL T16/5502 LINEPRINTER

The Model T16/5502 Lineprinter is a medium-priced, low-speed (300 lpm) drum unit featuring a 12-channel Vertical Format Unit (VFU). The VFU utilizes IBM-compatible carriage tapes. The unit provides 132 columns of either 64 or 96 (optional) ASCII characters with a spacing of 10 characters/inch horizontally and 6 or 8 lines/inch vertically. Optionally, the unit features OCR character set with appropriate fonts. For service and maintenance, an optional self-test code generator is also available. This unit accepts up to 6-part fanfold edge-punched forms 4 to 16.75 inches wide.

MODEL T16/5503 LINEPRINTER

The Model T16/5503 Lineprinter is a low-priced, medium speed (600 lpm) drum unit incorporating the same features as the Model T16/5502 described above. The minimum printing rate is maintained at 436 lpm even when the optional 96 character set is incorporated.

LINE PRINTER SUBSYSTEMS

MODEL T16/5504 LINEPRINTER

The Model T16/5504 Lineprinter is a moderately-priced, medium-speed (900 lpm) drum unit identical to the Model T16/5502 described above. This rugged, compact unit also accepts up to 6-part forms.

MODEL T16/5505 LINEPRINTER

The Model T16/5505 Lineprinter is a high-performance, high-speed (1500 lpm) train unit ideally suited for high-volume on-line transaction-oriented applications where speed and reliability are critical. The unit incorporates all the features of the Model T16/5502 and, in addition, provides a powered paper stacker. This unit also handles up to 6-part fanfold edge-punched forms 4 to 16.75 inches wide.

OPTIONS

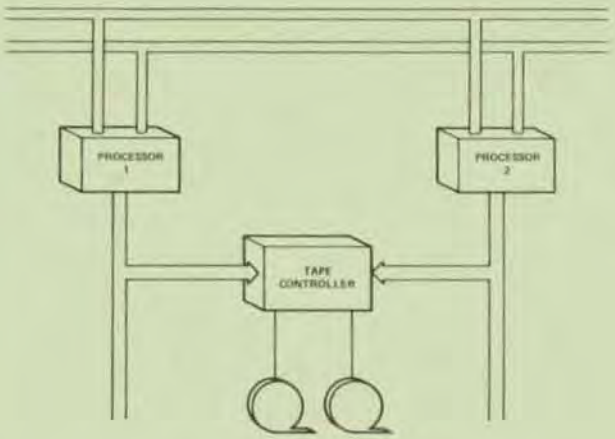
DESCRIPTION	MODEL			
	5502	5503	5504	5505
ASCII 96-character Set	YES	YES	YES	YES
OCR Character Set	YES	YES	YES	YES
Vertical Format Unit (VFU)	YES	YES	YES	YES
Elapsed Time Meter	YES	YES	YES	YES
Self-Test Feature	YES	YES	YES	YES
Static Eliminator	YES	YES	YES	YES
Pedestal Mount	STD	STD	STD	STD

Notes: ① Standard

SPECIFICATIONS

PARAMETER	MODEL			
	5502	5503	5504	5505
PHYSICAL:				
Dimensions (L x W x H)	26 x 33 x 45	26 x 33 x 45	36 x 33 x 45	24 x 48 x 46
Weight (Pounds)	340	370	420	800
ELECTRICAL:				
Voltage (Single Phase VAC)	1	1	1	230 ± 10%
Frequency (Hz)	60 ± 2.0 50 ± 2.0	60 ± 2.0 50 ± 2.0	60 ± 2.0 50 ± 2.0	60 ± 3.0 50 ± 3.0
Power Rating (Watts)	525	680	825	<3 KW
ENVIRONMENTAL:				
Temperature (°C)	10 – 37	10 – 37	10 – 37	10 – 43
Humidity (Non-Condensing)	30 – 80%	30 – 80%	30 – 80%	10 – 90%
PRINTING:				
Type	Drum	Drum	Drum	Drum
Speed (64 Character Set lpm)	340	600	900	1500
Speed (96 Character Set lpm)	240	436	660	1220
Columns (Standard)	132	132	132	132
Horizontal Spacing (Characters/Inch)	10	10	10	10
Vertical Spacing (Lines/Inch)	6/8	6/8	6/8	6/8
Paper Widths (Inches)	4 – 16.75	4 – 16.75	4 – 16.75	4 – 16.75
TRANSMISSION:				
Code (Standard)	ASCII	ASCII	ASCII	ASCII

Note: ① 100, 115, 125, 200, 220 or 240 ± 10%



TANDEM MAGNETIC TAPE SUBSYSTEMS

- **Flexible and reliable NonStop(tm) Magnetic Tape Controllers**
- **800 bpi NRZI and/or 1600 bpi PE Magnetic Tape Drives**
- **45 ips or 125 ips Magnetic Tape Drives**
- **9 track recording, IBM and ANSI compatible**
- **Ease of serviceability**

SUBSYSTEM CONFIGURATION

The Tandem Magnetic Tape Subsystem provides reliability and performance for Non-Stop transaction-oriented systems requiring magnetic tape capability. A basic Magnetic Tape Subsystem consists of a NonStop magnetic tape controller and one or more magnetic tape drives. Two controllers are available – the T16/3201 Magnetic Tape Controller handles 800 bpi NRZI formats and the T16/3202 Magnetic Tape Controller handles both 800 bpi NRZI and 1600 bpi Phase Encoded (PE) formats.

Both controllers are dual-ported. Each connects to two I/O channels simultaneously so that in the event of a single failure (processor, power supply or I/O channel) the other port can maintain a data and control path to the magnetic tape drives.

Each controller can control one or two drives.

Three tape drives are available. They can be configured in 800 or 1600 bpi, NRZI or PE recording formats, and have tape speeds of 45 or 125 ips.

The drives were designed to provide ease of maintenance. All major service functions can be performed from the front door opening.

MODEL T16/5101 MAGNETIC TAPE DRIVE

The Model T16/5101 Magnetic Tape Drive is a moderately priced, medium speed direct drive unit. Tape speed is 45 ips using 800 bpi NRZI recording formats. The effective data rate is 36KB.

The drive uses 9 track recording and is compatible with IBM and ANSI standards. The drive uses standard 1/2" tape and can accommodate reels up to 10.5 inches in diameter using standard IBM hubs.

Data integrity is maintained by the use of VRC, LRC and CRCC checking.

MODEL T16/5103 MAGNETIC TAPE DRIVE

The Model T16/5103 Magnetic Tape Drive is a moderately priced, medium speed direct drive unit. Tape speed is 45 ips. Recording mode is dual-density using either 1600 bpi Phase Encoded (PE) recording formats or 800 bpi NRZI recording formats. The effective data rate is 72KB or 36KB depending on the operator-settable switch.

TANDEM MAGNETIC TAPE SUBSYSTEMS

TANDEM MAGNETIC TAPE SUBSYSTEMS

The drive uses 9 track recording and is compatible with IBM and ANSI standards. The drive uses standard 1/2" tape and can accommodate reels up to 10.5 inches in diameter using standard IBM hubs.

Data integrity is maintained by the use of VRC, LRC and CRCC checking.

MODEL T16/5104 MAGNETIC TAPE DRIVE

The Model T16/5104 Magnetic Tape Drive is a moderately priced, high speed vacuum column unit. Tape speed is 125 ips. Recording mode is dual-density using either 1600 bpi Phase Encoded (PE) recording formats or 800 bpi NRZI recording formats. The effective data rate is 200KB or 100KB depending on the operator-settable switch.

The drive uses 9 track recording and is compatible with IBM and ANSI standards. The drive uses standard 1/2" tape and can accommodate reels up to 10.5 inches in diameter using standard IBM hubs.

Data integrity is maintained by the use of VRC, LRC and CRCC checking

MODEL T16/3101 MAGNETIC TAPE CONTROLLER

The Model T16/3101 Magnetic Tape Controller provides the ability to attach up to two NRZI tape drives. Connections are starred rather than daisy-chained to insure maximum utilization. The controller allows each drive to operate at 800 bpi using NRZI recording formats as specified in ANSI X3.22-1973. The controller allows tape speeds of 45, 75 or 125 ips. The controller conforms to IBM standards for 9 track digital recording.

MODEL T16/3102 MAGNETIC TAPE CONTROLLER

The Model T16/3102 Magnetic Tape Controller provides the ability to attach up to two NRZI and/or PE tape drives. Connections are starred rather than daisy-chained to insure maximum utilization. The controller allows each drive to operate at 800 bpi using NRZI recording formats as specified in ANSI X3.22-1973 and/or 1600 bpi using Phase Encoded recording formats as specified in ANSI X3.39-1973. The controller allows tape speeds of 45, 75 or 125 ips. The controller conforms to IBM standards for 9 track digital recording.

SPECIFICATIONS

Characteristics	5101	5103	5104
Tape Speed (ips)	45	45	125
Recording Density (bpi)	800	800/1600	800/1600
Transfer Rate (cps)	36KB	36/72KB	100/200KB
Recording Format	9-track NRZI	9-track NRZI/PE	9-track NRZI/PE
Tape Reels	1/2" std IBM hub	1/2" std IBM hubs	1/2" std IBM hubs
Rewind Time (ips)	300	300	375
Transport	Tension Arm	Tension Arm	Vacuum Column
Checking	LRC, VRC, CRCC	LRC, VRC, CRCC	LRC, VRC, CRCC

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000 T16-8713-01

TANDEM
UNIVERSAL INTERFACE

FEATURES

- **Provides 8/16 line parallel interface**
- **Operates half-duplex, bi-directional at up to 4 MBytes/sec**
- **Uses both TTL and Differential logic**
- **Contains dual-channel connected logic for fault-tolerant operations**

INTRODUCTION

The T16/3401 Universal Interface (UI) provides the ability to interface custom equipment to the T16 Computer System. The UI is capable of attaching two devices that have 8 or 16 line parallel data interfaces to the Tandem 16 Computer. The Universal Interface (UI) provides a device data path that is buffered (16 words deep), bi-directional, and capable of operating in half-duplex mode at a sustained data transfer rate of up to 4 Mbyte per second (depending on the channel configuration). It interfaces to one device over positive or ground true TTL lines for short distances (up to 25 ft) and to the second device over differential lines for longer distances (up to 500 ft). The data path between either or both of the two devices and the UI can be either one byte (8 bits) or one word (16 bits) wide.

Configuration of the UI is accomplished by software and by configuration jumpers in the connector hood.

DATA PATH

The UI can be used to interface both input and output devices requiring a data path width of either one byte (8 bits) or one word (16 bits) in half-duplex mode. Data is transferred between the UI and the channel in bursts consisting of several words. For input or output devices that operate in word mode, the UI passes data words directly to and from the device on the DATA 0:15 interface lines (Figure 1). For output devices that operate in byte mode, the UI disassembles data words from the channel and transfers data bytes to the device. For input devices that operate in byte mode, the UI assembles data bytes from the device and transfers data words to the channel. In byte mode, data is transferred to and from the device on the DATA 8:15 interface lines (Figure 1). The width of the device data path is program controlled, such that a device with a one byte wide data path can have additional control signals sent to it on the remaining eight data lines. For example, some terminals have a one byte data path but require 12 bits for cursor addressing.

DATA TRANSFER

Data transferred between the UI and a device may be strobed in a handshake sequence or pulsed without a handshake. In Handshake Mode, the device response can either lead or follow the UI data strobe. In Pulse Mode, the UI data strobe is used to pulse data to the device on write operations with no response from the device, while for read operations, the device response is used to pulse data to the UI with no response from the UI.

DATA PARITY

Odd parity is generated and checked for each data word that is transferred between the channel and the UI. The parity that exists between the UI and each device is defined by configuration jumpers in the connector hood. The jumpers select odd, even or no parity.

DATA TRANSFER TERMINATION

A data transfer sequence between a device and the I/O channel is initiated with an EIO instruction executed by the CPU program. When a specified number of bytes have been transferred, the sequence is terminated in one of three ways: 1) by the I/O channel, 2) by the device, or 3) by the UI.

UI STROBES

The UI is capable of transmitting up to six strobe signals to a device, one of which is the Data Strobe. The other five strobes can be transmitted individually or in combinations under program control.

Two strobe signals are provided that can be pulsed before and after data transfer. The "before" strobe (Strobe 4) is 1 μs in duration and the "after" strobe (Strobe 5) is 10 μs in duration.

TANDEM UNIVERSAL INTERFACE

Two strobe signals (Strobe 2 and Strobe 3) can be held active during the entire data transfer.

One strobe signal (Strobe 1) can be gated during data transfer by an external trigger from the device.

In a ground-true TTL interface, Strobes 1 through 5 can be wired together in the connector hood in any combination to "or" their respective functions.

UI STATUS

Device status is presented to the UI on eight (8) interface lines, STATUS 8:15. These signal lines can be jumpered to the STATRES 0:8 pins in the connector hood to form the desired status results.

Controller (UI) and device status words are transferred to the CPU program in the CPU's register stack in response to EIO, IIO and HIIO instructions.

UI DEVICE ADDRESSING

All devices attached to a processor I/O channel are assigned unique Physical Unit Addresses. The Physical Unit Address is 8 bits in length and consists of a 5 bit Controller Number and a 3 bit Unit Number. The Controller Number is defined by switches on the UI board; Unit Number 0 is the device attached to the TTL interface and Unit Number 1 is the device attached to the differential interface.

UI COMMANDS

Commands are issued to the UI by executing EIO (Execute Input/Output) instructions in the CPU program. The UI command set is comprised of the following commands:

- SENSE
- TAKE OWNERSHIP
- DISABLE PORT
- SET COUNT
- READ
- WRITE

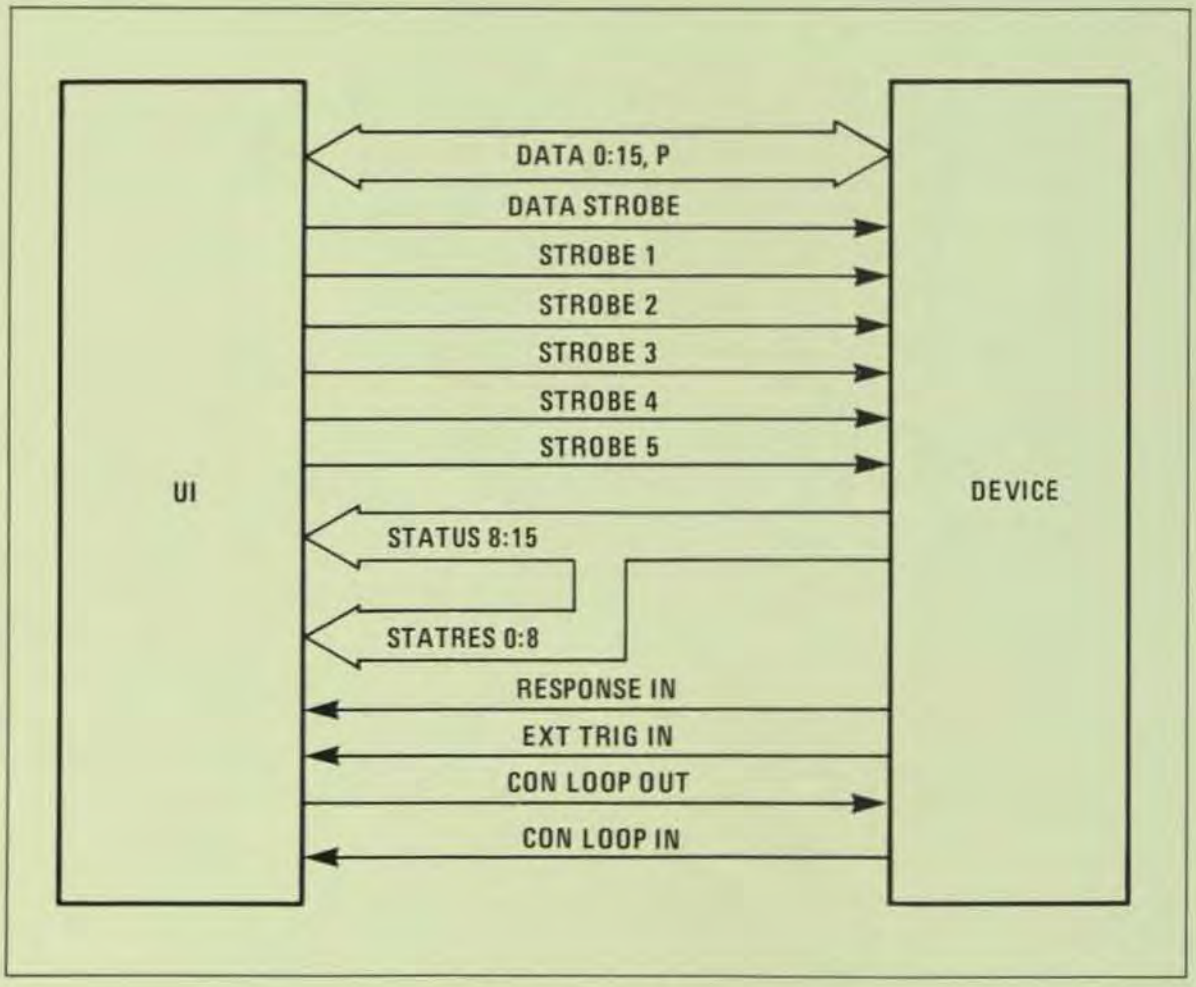


Figure 1. UI/Device Interface

TANDEM

6520

MULTI-PAGE DISPLAY TERMINAL

The Tandem *6520* block mode terminal represents a significant advance in state-of-the art technology. Designed specifically for use in the on-line transaction environment, the *6520* underscores Tandem's commitment to making transaction processing easier and more reliable.

The Tandem *6520* features include:

- **MEMORY PARITY for data integrity**
- **MULTIPLE DISPLAY PAGES for high data throughput and reduced line utilization**
- **FULL COMPLEMENT OF VIDEO AND DATA ATTRIBUTES, EDITING AND PROGRAM FUNCTION KEYS for ease of operation**
- **CONVERSATIONAL AND BLOCK MODES**
- **SYNCHRONOUS AND ASYNCHRONOUS PROTOCOLS for simplified communications**
- **POINT TO POINT AND MULTIPOINT AT SPEEDS UP TO 19.2 K BPS for increased flexibility in data transmissions**
- **RS-232 AND CURRENT LOOP for ease in communication interface.**

Increased Data Integrity

Tandem's commitment to data integrity has now been extended to the ***6520*** terminal through the use of parity on all display memory. No other terminal configuration can as reliably transmit to and receive data from the host processor, monitoring it for accuracy. Nor can other terminals provide as much assurance as the ***6520*** that the data displayed on the screen is the data actually received by the terminal.

The Tandem ***6520*** provides immediate feedback to the operator if a memory parity error occurs, by displaying an error message on the 25th line of the screen. It also terminates any I/O transaction in process to prevent contamination of the data base.

High Data Throughput

The Tandem ***6520*** has an impressive multiple display paging facility which significantly enhances throughput — a critical factor in the on-line transaction environment.

While interacting with an operator on a screen display, the ***6520*** terminal can be transmitting from, or receiving into, any one of the five 1920 character pages at the same time.

The multiple display paging feature gives the host processor the option of writing to, or reading from, any page of the ***6520*** terminal at its discretion.

Ease of Operation

The Tandem ***6520*** provides a full complement of video and data attributes for ease of operation. Video attributes include: reverse video, blinking, non-display, underscore and two programmable brightness levels. A 25th display line is provided for status information.

Data attributes include: protected and unprotected fields, alpha/numeric or alphanumeric only fields, upshifted fields, modified field indication and auto-tab disable fields.

The ***6520*** provides an impressive array of editing and function controls for ease of use. Local and remote editing functions are an integral part of the terminal design, continuing Tandem's commitment to fully distributed processing.

Editing keys include: cursor up, down, left, right, home and return. Screen control keys include erase character/line/field/page, character/line insert and delete, set/clear tabs, forward/back tabulation, roll up and down, next/previous page and numeric key pad.

There are 16 function keys, providing 32 user-programmable functions, for greater flexibility in defining application dependent functions.

Increased Flexibility

The Tandem ***6520*** allows a high degree of flexibility in the ways it may be utilized by Tandem 16 systems, allowing increased ease of configuration and expansion, a very important factor in the on-line transaction oriented environment.

Conversational mode allows the terminal to interact with the host processor on a character-at-a-time basis. Display memory is organized as 120 lines by 40 or 80 characters wide, 24 lines of which are displayable at a time. The screen may be rolled up or down to view all 120 lines.

Block mode operation allows the terminal to transmit and received blocks of characters. Display memory is referenced as five pages of 1920 characters or 10 pages of 960 characters. Fields may be designated as PROTECTED, which will not allow entry into those fields from the keyboard. Through the use of the READ MODIFIED command, only fields which have been modified from the keyboard will be transmitted by the terminal.

Tandem software subsystems — including the VS Block Mode Editor, the EXPAND Network Monitor, and the PATHWAY Transaction Processing System — will take full advantage of the block mode capabilities of the ***6520*** terminal.

Simplified Communications Interface

Synchronous operation allows attachment of the ***6520*** to the Tandem T16/6202 Byte Synchronous Controller in block mode via a half or full duplex RS-232 communication link. The ***6520*** is connected multipoint with one to 63 Tandem ***6520*** terminals on the same communication link.

Asynchronous operation allows attachment of the ***6520*** to the Tandem TI6/6303 or 6304 Asynchronous Communications Controller via a half or full duplex RS-232 communication link either point to point in conversational or block modes, or multipoint in block mode. The ***6520*** may also be connected to the 6303 or 6304 controller via a 20 ma current loop link in point to point conversational or block modes.

Both synchronous and asynchronous communications protocols allow the ***6520*** to operate at nine different speeds between 110 and 19.2 K BPS.

Tandem also offers the ***6524*** terminal which has the same impressive features as the ***6520***, plus the printer port option.

Specifications

DISPLAY FEATURES

Physical
- 12 inch diagonal screen
- 2,000 characters per screen
- 25 lines x 40/80 characters per screen
- 7x9 or 14x9 dot matrix character generation

Memory
- 1 parity bit per byte
- 120 lines x 40/80 characters in conversational mode
- 5/10 pages x 1920/960 characters in block mode

Video Attributes
- Reverse video
- 2 brightness levels
- Blinking
- Underscore
- Non-display
- Blinking underscore/reverse video cursor
- Addressable/readable cursor

Data Attributes
- Protected/unprotected fields
- Modified data tag/partial screen transmit
- Alpha/Numeric/Alphanumeric only fields
- Upshifted fields
- Auto-tab disabled fields

KEYBOARD FEATURES

Style
- Typewriter
- 32 program function keys
- 10 key pad
- Repeating keys
- Audible alarm

Editing Functions
- Roll up/down
- Set/clear tabs
- Forward/reverse tabulation
- Character/line insert and delete
- Line/page erase
- Up/down/left/right/home/return cursor controls
- Previous/next page

COMMUNICATION FEATURES

- Conversational Mode
- Block Mode
- ASCII code
- Synchronous/Asynchronous protocols
- Half/full duplex
- Point to point/multipoint
- RS-232/20ma current loop
- Speeds of 110, 150, 300, 1200, 1800, 2400, 4800, 9600, 19,200 bps

ENVIRONMENTAL FEATURES

Physical
- 13.3" x 17.0" x 22.0" (HxWxL)
- 40 lbs.

Electrical
- 100/120 VAC, 60 HZ, 2 Amps
- 220/240 VAC, 50 HZ, 1 Amp

Temperature/Humidity
- 40F to 95F (5C to 35C)
- 20% to 80% relative humidity (non-condensing)

Mounting
- Table top

TANDEM

TANDEM COMPUTERS INCORPORATED, 19333 Vallco Parkway, Cupertino, CA 95014. Toll Free (800) 538-9360 or (408) 996-6000 in California. Offices throughout the United States, Canada, Europe and the United Kingdom. Distributors in Australia, Finland, Mexico and Venezuela.

DS107902

TANDEM

TERMINAL SUBSYSTEMS

- **High Performance Terminals for NonStop(tm) Transaction-Oriented Applications**
- **Select from Keyboard/Printer, CRTs or Receive-Only Printer**
- **Asynchronous RS232 or 20mA Current Loop Interfaces**
- **CRT Speeds to 19.2K bps**
- **Serial Printers Speeds to 200 cps**
- **Field Proven Reliability**

SUBSYSTEM CONFIGURATIONS

The Tandem Terminal Subsystems offer the user a wide selection of operational benefits to maximize the Non-Stop features of the system and minimize program development overhead. All terminals meet or exceed ANSI, EIA and ECMA standards in addition to being UL certified. A wide range of options are also offered to meet the individual needs of a customized or diversified application.

A basic Terminal Subsystem consists of a Multiplexer, a System Console and up to 16 additional terminal lines. The Multiplexer is comprised of a Model T16/6301 Asynchronous Controller and one or two Model T16/6302 Asynchronous Extension Interfaces. The dual-port Controller handles up to 32 independent communication lines for modem or direct-wire connection compatible with single- or multi-drop terminals. One line is dedicated to the hard- or soft-copy system console. Each Asynchronous Extension provides an interface for up to 15 additional terminal lines.

MODELS T16/6603-6604 KEYBOARD/PRINTER TERMINALS

The Models T16/6603 and T16/6604 terminals provide hard-copy output combined with keyboard input at a low cost. Both models offer a 59 character keyboard, 30 characters per second printing using a 5 X 7 dot matrix, 132 characters per line and the ability to print all 96 ASCII characters while transmitting 128 ASCII characters. Each model is suitable for operator logging, error message logging, and operator input to the Tandem system.

Both models are identical in every respect except interfacing. The Model T16/6603 uses an EIA RS232 interface while the T16/6604 uses a 20 mA current loop interface permitting hard-wired operation up to 1500 feet away.

MODELS T16/6401-6402 CRT TERMINALS

The Models T16/6401 and T16/6402 CRT Terminals provide the performance and reliability of cathode ray tubes at less cost than keyboard/printer terminals. Both models incorporate a compact 12 inch CRT which displays 24 lines by 80 characters, or 1920 characters per screen. The standard keyboard provides 59 keys for ease of data entry. Upper Case/Lower Case may be chosen as an option. Transmission rates are switch selectable from 75 to 19,200 bps. Thus display rates up to 1920 characters per second are attainable.

Both models are identical in every respect except interfacing. The Model T16/6401 uses an EIA RS232 interface while the T16/6402 uses a 20mA current loop interface permitting hard-wired operation up to 1500 feet away.

MODELS T16/6511-6512 PAGE-MODE CRT TERMINALS

The Models T16/6511 and T16/6512 CRT Terminals provide increased capability over the T16/6401-6402 CRT Terminals by the use of page-mode display screens and added keyboard functions. Both models incorporate a compact 12 inch CRT which displays 24

lines by 80 characters, or 1920 characters per screen. The standard screen provides the ability to define protected and data entry fields for page-mode operation. The standard keyboard provides 68 keys for ease of data entry. In addition to the standard keys a ten-key pad, 16 function keys, edit function keys, and cursor control keys are provided. Upper Case/Lower Case is switch selectable. Transmission rate for both models is 9600 bps. Thus display rates up to 960 characters per second are attainable.

Both models are identical in every respect except interfacing. The Model T16/6511 uses an EIA RS232 interface while the T16/6512 uses a 20mA current loop interface permitting hard-wired operation up to 1500 feet away.

MODEL 5508 SERIAL PRINTER

The Model T16/5508 Serial Printer provides high performance serial printing at a modest cost. Using a 9-wire ballistic head, the T16/5508 generates 7 X 9 dot matrix characters at speeds of 200 characters per second. Printing is bi-directional to provide maximum throughput for NonStop applications. Both upper and lower case characters can be printed. A versatile electronic VFU provides flexible forms control.

The T16/5508 uses a 20mA current loop interface. No special line printer interface is required since a current loop port on the Asynchronous Multiplexer can be used. The T16/5508 can be hard-wired up to 1500 feet from the Tandem system.

SPECIFICATIONS

	T16/6603-6604	T16/6401-6402	T16/6511-6512	T16/5508
Device Type	KYBD/PTR	CRT	CRT	R/O PTR
PRINT/DISPLAY/KYBD				
Type	Impact	CRT	CRT	Impact
Character Gen (Dot Matrix)	5 X 7	5 X 7	5 X 9	7 X 9
Columns	132	80	80	132
Lines	n/a	24	24	n/a
Character Display	n/a	1920	1920	n/a
Horizontal Spacing	10	n/a	n/a	10
Diagonal Display	n/a	12"	12"	n/a
VFU	no	n/a	n/a	yes
Keyboard	59 keys	59 keys	116 keys	n/a
Printing	Unidirectional	n/a	n/a	Bidirectional
Print Speed	30 cps	1920 cps	960 cps	200 cps
Transmission				
Code	ASCII	ASCII	ASCII	ASCII
Connection				
RS232	6603	6401	6511	n/a
20mA CL	6604	6402	6512	yes
Line Speed (bps)	300	75 to 19.2K	9600	9600
Mounting	Pedestal	Table-top	Table-top	Table-Top
Electrical				
Voltage	110V, 60Hz 220V, 50Hz	110V, 60Hz 230V, 50Hz	110V, 60Hz 230V, 50Hz	110V, 60Hz 220V, 50Hz
Operating Amps	3.1	0.8	2.0	4.0
Physical				
Dimensions (LxWxH)	24x28x34	20x16x13	24x21x13	16x23x13
Weight (lbs)	120	32	50	45
Environmental				
Temperature	50-104F	41-122F	41-122F	39-95F
Humidity	10-90%	5-95%	5-95%	20-90%

TANDEM

ENTRY SCREEN FORMATTER

FEATURES

- **Interactive Page-Mode Form Creation on Video Display Terminal for Non-Stop Transaction-Oriented Applications**
- **Efficient and Economical Forms Creation with Automatic or Manual Cursor Positioning, Field Delimiters, Automatic Validity Checking, and Easy Transfer to Disc Files**
- **Fast and Error-Free Forms Display with Automatic Data Entry Validity Checking**
- **Individual Field Access with Length and Validity Check Attributes for User-Defined Error Checking**
- **Tandem-Supplied SCREEN Program to Create New or Modify Old Forms and to Insert Multiple Forms in a Single Disc File**

INTRODUCTION

The ENTRY Screen Formatter provides a simple easy-to-use method of creating and displaying user-defined forms on an interactive page mode video terminal. Extensive validity and error checking is accomplished via the Tandem-supplied ENTRY procedures. When an error has been made in the creation or data entry processes, a flashing cursor is positioned over the field in error and an error message is printed to specifically flag the error.

Entry essentially performs four tasks:

- Creates Forms
- Programmatically Displays Forms
- Programmatically Accesses Individual Fields
- Tests Forms Independent of Application Program

In addition, ENTRY provides the facilities to: modify or update old forms, insert multiple forms in a single disc file, and provide length and field validity checking attributes for user-defined error checking.

FORMS CREATION

Forms creation is performed on a page mode terminal. The user simply designs the form on the screen as it is to appear when used. Fields on the form where terminal operator entries are to be made are indicated by delimiters. Once the form image is created, the user specifies a field name and validity checking attribute for each field. When completed the form image is written to a designated file on disc.

FORMS DISPLAY

Forms display, forms read and field checking are all performed by application programs through calls to ENTRY procedures. When a form is displayed on a terminal, the operator fills in the fields on the form and presses a function key to transmit the fields into the computer where ENTRY performs the validity check on each field. If a field contains invalid data, the form can be programmatically redisplayed with the invalid entry flashing on the screen.

FIELD ACCESS

Individual fields in a form may be referenced by an application program. A field is referenced by the name assigned when the form was defined. Additionally each field has a length attribute and a validity checking attribute for user-defined error checking.

SCREEN PROGRAM PROCEDURES

ENTRY provides the following procedures to aid terminal I/O with forms created by the SCREEN program:

- EXPAND^SCREEN –
  initializes the applications I/O buffer with the control and data characters required to output a screen to the terminal
- READ^SCREEN –
  fills the applications I/O buffer with the control sequence required to input the operator entry fields from a screen
- CHECK^SCREEN –
  moves the operator entry fields from the applications I/O buffer into the correct fields and does the required validity checking
- BLINK^SCREEN –
  fills the applications I/O buffer with the control sequence required to turn the "blinking" on or off for a specific operator entry field
- POSITION^SCREEN –
  fills the applications I/O buffer with the control sequence required to position the cursor over a specific operator entry field
- FL^SCREEN –
  is used to compute the actual length of the field input by the operator

TANDEM ENTRY SCREEN FORMATTER

TANDEM ENTRY SCREEN FORMATTER

ERROR MESSAGES

When a user makes an error during form creation and testing, the error is flagged as follows: the flashing cursor is automatically positioned over the field in error and an error message is displayed on the bottom line of the screen. The user may then correct the field in error and strike any function key to send the corrected form back to the computer. Some typical error messages are:

- FIELD NOT TERMINATED: the operator entry field does not have a terminating delimiter or is greater than 255 characters in length
- ILLEGAL FIELD NAME: the field name was not an alphanumeric string of 8 or fewer characters, starting with an alphabetic character
- ILLEGAL ATTRIBUTE: the checking attribute is either missing or not a legal integer value
- NOT ENOUGH FIELDS DEFINED. the user did not provide enough field names
- TOO MANY FIELDS DEFINED the user provided too many field names or attempted to define more than 255 fields

Example: Creating a Screen Format

```
          N E W     A C C O U N T     F O R M A T

PLEASE FILL IN THE FOLLOWING INFORMATION. IF YOUR DATA DOES NOT
COMPLETELY FILL IN THE SPACE PROVIDED YOU CAN USE THE TAB KEY TO GET
TO THE NEXT ITEM.

LAST NAME        [                          ]
FIRST NAME, M.I.[                           ]
STREET ADDRESS   [                                                  ]
CITY             [                          ]
STATE            [                          ]
ZIP CODE         [     ]

ACCOUNT NUMBER   [     ]

INITIAL DEPOSIT AMOUNT ($NNNNN.NN)  $[         ]

TO ENTER THE DATA PLEASE TYPE FUNCTION KEY F1. IF DATA IS IN ERROR THE
ITEM WILL BE SET BLINKING AND AN ERROR MESSAGE WILL APPEAR AT THE
BOTTOM OF THE SCREEN. PLEASE CORRECT THE DATA AND RE-ENTER. A
CORRECT, COMPLETED FORM WILL BE INDICATED BY A MESSAGE. TYPE FUNCTION
KEY 16 TO RETURN TO THE MENU.

[                                                                   ]

Note: All operator entry fields are defined with [...]. All other
      data is set protected on the screen.
```

Example: Assignment of Field Names & Checking Attributes

```
NAME      1   <--- Any Characters Valid
FIRST     1   <--- Any Characters Valid
STREET    1   <--- Any Characters Valid
CITY      1   <--- Any Characters Valid
STATE     2   <--- Alphabetic Only
ZIP       3   <--- Numeric Only
ACCT      3   <--- Numeric Only
INITIAL   8   <--- Financial Numeric Only
MSG       0   <--- No Checking Performed
```

Example: Coding the Entry Procedures

```
PROC PAGE^OPEN;
 BEGIN

! Initialize the I/O Buffer with Screen Format
 WRTCNT := EXPAND^SCREEN(@FORMOPEN,,SBUFFER,1);
 CALL WRITE(CRT,BUFFER,WRTCNT,CNT);
 IF < THEN BEGIN ...
    .
    .
    .

! Turn Blinking Field Off
 WRTCNT := BLINK^SCREEN(@FORMOPEN,SCREEN,SBUFFER,FORMOPEN^MSG,0);
 CALL WRITE(CRT,BUFFER,WRTCNT,CNT);
 IF < THEN BEGIN ...

! Prepare a buffer for Reading a Screen
 WRTCNT := READ^SCREEN(@FORMOPEN,SBUFFER);
 CALL WRITEREAD(CRT,BUFFER,WRTCNT,FORMOPEN^IOBUF,CNT);
 IF < THEN BEGIN ...
     .
     .
     .

! Perform Validity Checking on Operator Entry Fields
 DO ...  UNTIL CHECK^SCREEN(@FORMOPEN,SCREEN,SBUFFER,OPEN^CHK);

! Turn on Blink, Position Cursor
 WRTCNT := BLINK^SCREEN(@FORMOPEN,SCREEN,SBUFFER,FORMOPEN^MSG,1);
 WRTCNT := POSITION^SCREEN(@FORMOPEN,
                           SCREEN,
                           SBUFFER[WRTCNT],
                           FORMOPEN^ACCT) + WRTCNT;
 CALL WRITE(CRT,BUFFER,WRTCNT,CNT);
 IF ERR THEN BEGIN ...
```

TANDEM
SORT

FEATURES

- **Three modes of operation**
  - **Conversational**
  - **Via command files**
  - **Programmatic**
- **Record Input and Output**
  - **From/To an External File**
  - **From/To a User Application Program**
- **Support for multiple file types**
  - **Unstructured files**
  - **ENSCRIBE structured files**
  - **EDIT files**
- **SORT keys may be;**
  - **Ascending or descending**
  - **STRING or INTEGER**
  - **Multiple key fields per sort**
- **SORT Output can be;**
  - **Complete record**
  - **Record Sequence Numbers**
  - **Keys only**

INTRODUCTION

The Tandem SORT program reorders a set of records according to the values of sort key fields defined within the records. SORT may be driven by a set of commands conversationally, or by a text file containing the commands, or it may be called from your program.

Records may be passed to SORT from a file, or sent one by one through procedure calls from your program. Similarly, the sorted set of records may be written to a file, or your program may call a procedure to retrieve the records, one per call.

Actual sorting runs as a separate process from the host program. Standard interface procedures are present in the Sort Command Interpreter program or called from your program, which handle process creation, control, and communication.

CONVERSATIONAL MODE

For this mode simply type,

```
:SORT
```

and wait for the prompt "<". Six commands are allowed:

FROM – names the file of unsorted records and describes the records;

TO – names the file for the sorted records and selects output options;

ASCENDING – DESCENDING describe the location and attributes of the sorting keys within the records;

RUN – starts the actual sort.

EXIT – exits from the Sort Command Interpreter.

Example:

```
:SORT/OUT $lp/
<FROM insort,RECORD 56
<TO outsort
<ASC 6:12 INTEGER
<RUN
<EXIT
:
```

TANDEM SORT

TANDEM SORT

COMMAND FILE MODE

For this mode, use EDIT to put your sort commands, one to a line, in a command file

:SORT [/IN <command file>]
[, OUT <list file>] /]

SORT reads the commands from <command file>. When a RUN command is found, and the previous commands describe a valid sort, SORT begins. Once the SORT completes, <command file> is read for more commands. End-of-file on <command file> terminates SORT.

Example:

```
:SORT/IN myfile,OUT $lp/
```

Contents of "myfile";

```
FROM datain
TO dataout
ASC 1:3 INTEGER
DESC 10:16 STRING
RUN
ASC 88:89
FROM names
DESC 22:35
TO sortout
RUN
```

PROGRAM SORTS

There are four ways to use SORT in your program.

1. SORT Input from External File
   SORT Output to External File

2. SORT Input from External File
   SORT Output to Program, one record at a time

3. SORT Input from Program
   SORT Output to External File

4. SORT Input from Program
   SORT Output to Program, one record at a time.

Five SORT Interface Procedures are provided.

SORTSTART — Initiate a sort.

SORTSEND — Send input record to SORT

SORTRECEIVE — Retrieve a sorted record

SORTFINISH — Complete the sort

SORTERROR — Format a SORT error message

GENERAL METHOD OF OPERATION

Two things determine how the sort functions internally:

- the amount of data to be sorted,
- the amount of memory you give.

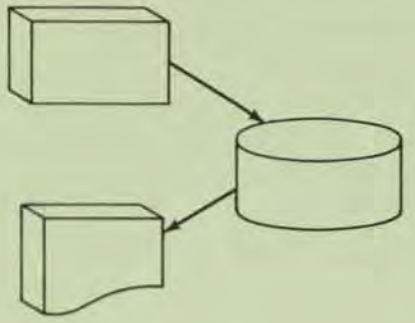
If the amount of data is small enough, the data is sorted within memory.

In most large sorts, the memory is insufficient to sort all the data at once, so SORT splits the input data into sorted pieces it can handle, and puts them in a scratch file. Each piece is referred to as a *run* since the records in each one "run" in sequence.

When there is more than one run formed from the initial data, the sorted output is produced by merging the runs together. This method of sorting is known as "Replacement/Selection".

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000

T16/8736-00



TANDEM SPOOLER

TANDEM SPOOLER

- **SPOOLER runs NonStop(tm)**
- **User can supply own Print Processes**
- **SPOOLCOM allows operator inspection and/or alteration of job parameters**
- **Routing structure permits individual or broadcast locations**
- **SPOOLER library procedures allow blocking and compression**
- **Multiple files can be spooled from one application**
- **SPOOLER parameters can be specified programmatically**
- **Forms Alignment**

INTRODUCTION

The Tandem SPOOLER provides a means of storing application output in holding areas for later retrieval. Output may be passed to other processes or printed on one or more devices.

The SPOOLER is actually many processes working in unison to provide spooling facilities. These processes include SPOOLER Supervisor, SPOOLER Collectors, Print Processes, and SPOOLCOM.

SPOOLER Supervisor functions as the SPOOLER monitor and communicates with the other SPOOLER processes to determine which tasks to perform or schedule. SPOOLER Control is actually a server process interfacing with 1) SPOOLCOM, 2) applications calling SPOOLERCOMMAND or SPOOLERSTATUS procedures, 3) SPOOLER Collection Processes, and 4) SPOOL Print Processes.

SPOOLER Collectors accept output from application processes and store it on disc. There can be one or more SPOOLER Collectors.

Print Processes retrieve spooled data and print it. The Tandem supplied print processes are capable of handling multiple jobs and devices. Users may supply their own Print Processes.

SPOOLCOM is an operator/user interface with the SPOOLER subsystem. It can be run interactively on a terminal or can be passed commands from an application process. SPOOLCOM performs such functions as downing a device or ordering extra copies of a report.

APPLICATION PROCESS INTERFACE

At the simplest level an application process can open the SPOOLER one or more times to perform spooled output. The standard file management procedures WRITE, CONTROL, and SETMODE are used.

The application may also use the SPOOLER library procedures to implement more advanced features of the SPOOLER. The library procedures are:

SPOOLSTART
— start a job and specify spooling attributes

SPOOLWRITE
— write a print line

SPOOLEND
— end the spool job

SPOOLCONTROL
— control functions

SPOOLSETMODE
— setmode functions

Through SPOOLSTART the application can specify location, form name, priority of printing, number of copies, report name, and hold before/after printing.

TANDEM SPOOLER

SPOOLCOM

SPOOLCOM is used to initiate and control the operation of the spooling system by accepting commands interactively from an operator at a terminal or programmatically from a user application process. SPOOLCOM commands include:

DEV
- controls devices

JOB
- control jobs

PRINT
- control Print Processes

COLLECT
- control collection processes

LOC
- set up and modify destination structure

SPOOLER
- control the Spooling System

HELP
- list SPOOLCOM Commands

EXIT
- terminate SPOOLCOM

SPOOLERCOMMAND
- issue a control command

SPOOLERSTATUS
- retrieve status information

SPOOLERREQUEST
- Request information for a specific job necessary to start printing.

FC
- fix command

COMMENT

USER-WRITTEN PRINT PROCESSES

The Print Process library procedures allow user-written processes for devices not supported by SPOOLER or the retrieval of spooled data by an application process. The procedures are:

PRINTINIT
- Initialize communication with the SPOOLER control process

PRINTCOMPLETE
- Accepts messages from the SPOOLER control process

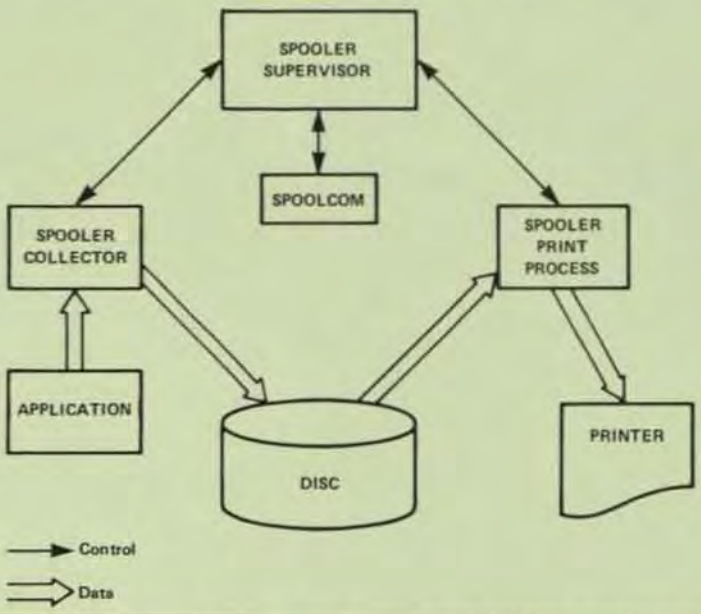
PRINTREADCOMMAND
- Interprets control messages from the control process

PRINTSTART
- Initialize data required to print a job

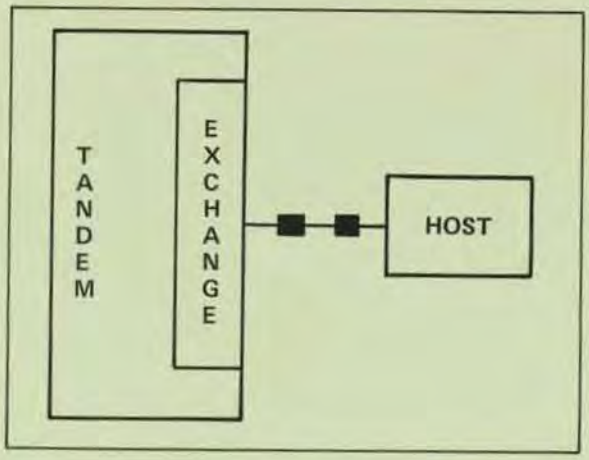
PRINTREAD
- Read a line of spooled data

PRINTINFO
- Obtain information on an active job

PRINTSTATUS
- Inform SPOOLER control process of an event such as end of file or error on the device.



TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000 T16/8743-00



TANDEM

EXCHANGE REMOTE JOB ENTRY SUBSYSTEM

FEATURES

- **Supports 2780 and 3780 Emulation**
- **Batch input from any media**
  - **terminal**
  - **magnetic tape**
  - **disc**
  - **card reader**
  - **another process**
- **Batch output to any media**
  - **terminal**
  - **magnetic tape**
  - **disc**
  - **line printer**
  - **another process**
- **Three Modes of Operation**
  - **Conversational**
  - **Command File**
  - **Programmatic**
- **File transfer ability between Tandem Systems**
- **NonStop Operation Optional**

INTRODUCTION

EXCHANGE is a Tandem Subsystem designed to allow a Tandem System to emulate the functions of a 2780 or 3780 remote batch workstation. Input and output can be from/to any media supported by the Tandem System including disc, magnetic tape, terminals, card readers, line printers, and other processes. Commands to define the input, list and punch files, and characteristics about the connection can be accepted conversationally from a terminal, from a Command File, or programmatically from another process.

General capabilities of EXCHANGE include transmitting and receiving in ASCII or EBCDIC, accepting horizontal tab codes, accepting vertical forms control codes, transmitting or receiving EBCDIC transparent data, short record truncation, blank field compression, transmitting and receiving blocked data link messages, and generation of WACK and TTD control codes when temporarily unable to transmit or receive.

Another useful feature is the ability to transfer files between two workstations. An EXCHANGE subsystem on one Tandem can perform remote file transfers to another Tandem which is also running the EXCHANGE subsystem.

EXCHANGE, at the user option, may be run in a NonStop mode.

EXCHANGE STRUCTURE

EXCHANGE is composed of two program modules; a Server Process and a Command Interpreter.

The Server has all data link handling responsibility and responds to requests to send or accept data to/from a remote system. The Server is typically run in conjunction with the Command Interpreter but can be called directly from an application process. Send and receive data and initial connection parameters are passed to the Server. The Server handles all message assembly/disassembly, blank compression/decompression, horizontal tab expansion, record truncation, and character set translation. The Server accepts and delivers data on a record-by-record basis.

The Command Interpreter provides a conversational interface to the operator. Commands are entered to control the connection type, specify the receive file names and parameters, and identify the files to be sent.

TANDEM EXCHANGE SUBSYSTEM

TANDEM EXCHANGE SUBSYSTEM

EXCHANGE COMMAND INTERPRETER

The EXCHANGE Command Interpreter allows the user to send or receive files by entering commands at a terminal or through a command file. Commands which can be entered are:

CONNECT – defines remote terminal connection

SEND – defines file to be sent

RECEIVE – defines the receiving file

ABORT – stops any transfer in progress

DISCONNECT – ends connection to host

EXIT – exits from EXCHANGE Command Interpreter

STATUS – displays line status and statistics

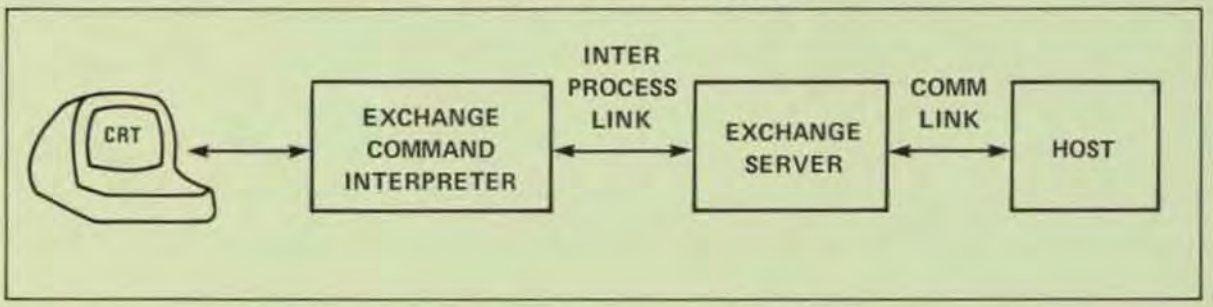
PROGRAMMATIC INTERFACE

A user process may directly interface to the EXCHANGE Server. The required steps in the application process are:

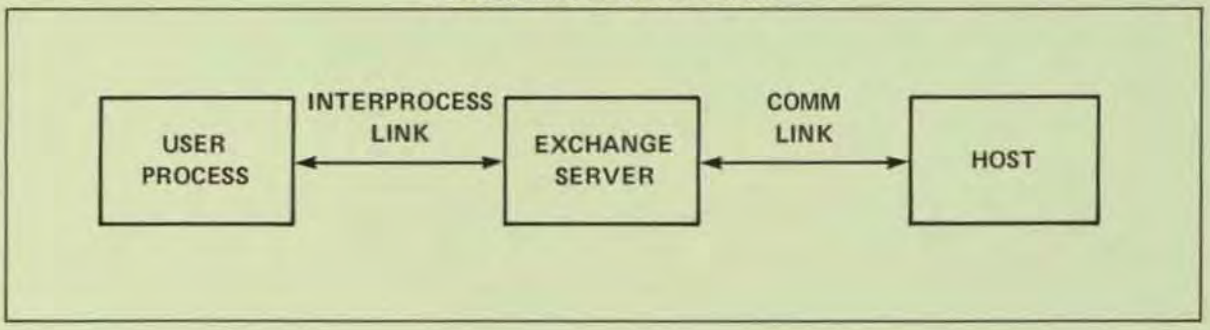
1. Open an interprocess file to the Server.
2. Establish the receive file character set via SETMODE.
3. Establish a data link to the remote system via SETMODE.

At this point the application may send/receive records via simple READ/WRITE type statements. Data link message formatting is handled by the Server, and compressed blanks and embedded horizontal tabs are expanded.

COMMAND INTERPRETER INTERFACE



PROGRAMMATIC INTERFACE



NONSTOP FEATURES

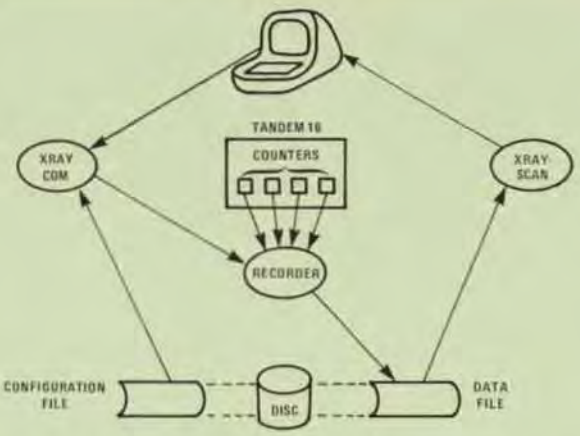
The EXCHANGE Server can be made to run NonStop. In the NonStop mode of operation EXCHANGE will:

1. Maintain a connection between the User, the EXCHANGE Command Interpreter, the EXCHANGE Server, and the Communication Data Link.
2. Maintain a major state; connected, disconnected, connecting.

In the NonStop mode of operation each process checkpoints all major state changes. This implies that:

1. Connected or disconnected status is maintained after a failure of either process.
2. Connection attempts are retried if a failure occurs during a connect.
3. If a failure occurs in either process SEND and RECEIVE commands are aborted, but the communication link is maintained.

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, CA 95014 ■ (408) 996-6000 T16/8738-00



TANDEM XRAY

PERFORMANCE MONITOR FOR TANDEM/16 COMPUTER SYSTEM

- **Diverse applications include Mix Balancing, Growth Management, Online Monitoring and Application Tuning**
- **Provides data usable for resolving a variety of performance issues**
- **Does not require any special hardware or display devices**
- **Includes 2 programs: XRAYCOM for measurement control, and XRAYSCAN for data reduction and analysis**
- **Self-measuring capability shows the exact amount of perturbation XRAY's measurements are causing in the system**

INTRODUCTION

XRAY is a tool for monitoring the performance of a TANDEM/16 computer system. The applications of XRAY span the following areas:

Mix Balancing – the distribution of applications across system hardware so as to eliminate bottlenecks.

Growth Management – the long-term appraisal of system component usage for planning, budgeting, and control purposes.

On Line Monitoring – permitting immediate detection of performance difficulties and providing continuous feedback on system usage.

Application Tuning – highlighting where the application program should be restructured to increase transaction throughput.

The data provided by XRAYSCAN in the form of time plots and reports can be used to resolve a variety of performance issues, for example:

- The usage of the system components (i.e., cps, memories, discs, communication lines) can be monitored on a long-term basis, permitting an orderly management of growth.
- Programs can be distributed among the processors for the most effective use of cpu and memory resources.
- Data base files can be distributed among the system's disc volumes to avoid bottlenecking at a single spindle.
- Programs responsible for excessive cpu, message, or virtual memory activity can be pinpointed.
- Data base files can be restructured for speedy access, and the optimum disc cache size can be determined.
- Response times can be tracked to provide an objective measure of system performance.

ADDITIONAL EQUIPMENT

XRAY will run on a minimum TANDEM/16 with the GUARDIAN Version C operating system and firmware. XRAY does not require ENSCRIBE OR ENVOY, but is fully integrated with both so that all data base and data communications activity can be measured or monitored. XRAY does not require any special hardware or display devices: it can be operated from any asynchronous, point-to-point terminal with a line width of at least 80 characters.

PROGRAMS

The program XRAYCOM is used to control measurements. Data is collected in a disc file. System performance is analyzed by running the program XRAYSCAN against the collected data; thus XRAYSCAN is used for data reduction and analysis.

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

TANDEM XRAY

TANDEM XRAY

An important feature of XRAY is that it can measure itself, indicating exactly how much the system was perturbed by the measurements. This is possible because XRAY allows measurement of individual processes on the system. When measurements are being taken, XRAYCOM installs a data collection process, called the RECORDER, in each processor. XRAY can be directed to measure each RECORDER, letting the user know exactly how much overhead XRAY entails. (Typically, this is less than 1% of the processor's workload during the time of the measurement).

XRAYCOM has five commands:

DATA	designates the disc file in which the RECORDERs store measurement data
CONF	specifies a configuration file, which lists the system components to be measured
GO	starts the measurement and specifies the time interval at which the RECORDERs write measurement data to the data file
EXIT	exit from XRAYCOM. The form "EXIT!" stops the measurement
LIGHTS	displays processor utilization on the cpu panel lights

A typical sequence of commands used to start a measurement would be:

```
: XRAYCOM
+ CONF xrayconf
+ DATA xraydata
+ GO 10
+ EXIT
```

Any of the commands can be issued at any time during a measurement, permitting the measurement configuration, the data file, and the time interval to be changed.

The items to be measured can be extended beyond basic device utilizations by configuring particular sets of files and programs for measurement. The configuration is placed in a EDIT-type file for input to XRAYCOM. A typical configuration file looks like this:

```
PROGRAMS:  SYSPROCS
           $SYSTEM.SYSTEM.*
           $DEVELOP.TESTPROG.*
FILES:     $ORDERS.*.*
           $ADMIN.BUDGET.*
```

EXAMINING THE COLLECTED DATA

The program XRAYSCAN is used for exploring and filtering the data in an XRAY data file. Any particular part of the data file can be selected for examination with the WINDOW command. Then, sets of the measured entities can be chosen for perusal with the entity selection commands.

Report Command	Measured Entities Reported
CPU	processors
LINE	data communication lines
DEVICE	tapes, printers, etc.
TERMINAL	terminals
DISC	discs, and disc file opens
PROCESS	processes
FILE	file opens

For each of the above report commands, a set of items is displayed. The items depend on the nature of the component being reported on. As an example, the items associated with the CPU report command include CPU BUSY, SWAP RATE, DISC RATE, CHIT RATE, SEND RATE, CPU QLEN, DISP RATE, TRAN RATE and RESP TIME.

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

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

restricts the report to those processes in cpu 1 which have used 1.5% of the cpu or more during the portion of the measurement being examined.

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

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

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

Many reports and plots will routinely be printed on a hard copy printer. At a CRT, a hard copy of the last report or plot can be made with the COPY command.

The ability to plot any item over time is an integral function of XRAY. Plots are used to find exception conditions, detect counter overflows, and associate correlated counters. It is simple to generate a plot, and a user may elect to plot any set of items on the same set of axes.

TANDEM

MUMPS

ANSI '77 Standard MUMPS Interpreter

TANDEM MUMPS **is a high-level, interpreted computer language designed for data management. MUMPS is used extensively in commercial and medical applications because of its ease of use, richness of features, ease of learning and superior string manipulation capabilities.**

TANDEM MUMPS **features include:**

- **Program structures that are easy to code, debug and modify;**
- **Modular expansion of the system without reprogramming applications, allowing the most cost-effective use of your investment;**
- **A vendor-supported MUMPS implementation fully integrated with the multifunctional GUARDIAN Operating System;**
- **High availability and reliable operation using *NonStop*™ System facilities;**
- **Concurrent execution of COBOL, FORTRAN or TAL programs for increased programming flexibility;**
- **Easy access to non-MUMPS files for maximum utilization of system resources;**
- **Versatile development tools to cut programming costs.**

TANDEM MUMPS supports all features of the ANSI 1977 MUMPS Language Standard. In addition, ***TANDEM MUMPS*** differs from other implementations in that it runs under the vendor-supported GUARDIAN Operating System. This means that ***TANDEM MUMPS*** has the unique benefits of the ***NonStop*** System which allows applications to continue to run even if a processor, disc, controller or I/O channel should fail during operation.

TANDEM MUMPS Extensions

TANDEM MUMPS provides the most powerful extensions of MUMPS available to date. In addition to all the benefits available with Standard MUMPS, TANDEM supplies an interface to COBOL, FORTRAN, TAL and PATHWAY programs and the reliability of *NonStop* computer systems, plus the following extensions:

1. **String subscripting,** which lets you give variables meaningful names such as NAME("LAST") rather than the cryptic numeric identifiers such as NAME (4). String subscripts can be up to 250 characters long.
2. **Negative and decimal number subscripting,** which lets you subscript variables with negative numbers, TIME(-12), and non-integer numbers, VARIANCE(-.75), MEAN(-25.67).
3. The **$ORDER function** as proposed by the MUMPS Development Committee, which returns all subscript values including non-numeric strings. The $ORDER is supplied in addition to the $NEXT function.
4. The **$ZFINFO function,** which returns information on all system files (e.g., key sequenced, entry sequenced and relative disc files).
5. The **ZTRAP command, $ZTRAP function and $ZTRAP special variable** which together provide a complete error trapping facility that can be tailored to individual applications.
6. **Commands and utilities** that create, edit and delete application and utility routines.
7. Commands that give access to **information on the state of the GUARDIAN Operating System** environment in which MUMPS is running.
8. **READ and WRITE commands** which provide the capability of accessing sequential and key-sequenced files with record sizes up to 4 k bytes.
9. **Extension to the MUMPS READ and WRITE commands,** providing for interprocess communication with COBOL, FORTRAN, and TAL and other MUMPS processes.
10. **Simple interface** to the PATHWAY Transaction Processing System, which allows CRT terminals to interact with applications written in MUMPS, as well as other TANDEM languages.

Versatile Operating Environment

TANDEM MUMPS runs under the GUARDIAN Operating System with full system resources at its disposal. ***TANDEM MUMPS*** users are able to concurrently execute COBOL, FORTRAN and TAL programs, as well as take advantage of the unique features available with a *NonStop* System, including a fault-tolerant mode. With ***TANDEM MUMPS*** you have:

- The capability of initiating any system or user process directly from MUMPS.
- The ability to communicate with COBOL, FORTRAN and TAL programs through the ENSCRIBE Data Base Manager for increased efficiency and optimum utilization of system resources. Through this facility users can read and write disc files created by any of these high-level languages, as well as other MUMPS processes.
- Access to the GUARDIAN Operating System facilities and resources, including the versatile TANDEM-supplied Spooler.
- The ability to utilize the ENFORM Query/Report Writer for fast and easy report generation.
- Access to the EXPAND Fault-Tolerant Communication Network capabilities for full utilization of distributed data bases.
- Access to the impressive text editing facility for program entry and maintenance.

Reliability

The reliability of a ***TANDEM MUMPS*** application is inherent in the basic architecture of the TANDEM 16 itself:

- TANDEM *NonStop* Computers are the only commercially available systems that are able to continue to operate despite the failure of a major component (i.e., CPU, controller, disc).
- Automatic *NonStop* operation of MUMPS routines can be selected with the use of the ZNONSTOP command.

Expandability

TANDEM provides systems that can grow with you. Modular expansion is easy with TANDEM Computers; there is no need for costly reprogramming. TANDEM provides the capability of dynamically expanding your system from 2 to 16 processors. Up to 255 system nodes can be joined in an EXPAND Network — all without additional costs for applications software reprogramming. With proper security, data can be accessed from any node in a network without special programming.

TANDEM MUMPS Applications

MUMPS is a specialized computer language that is intended for applications that emphasize interactive data entry and inspection, manipulation of text and encoded data, and data sharing. MUMPS is most useful and efficient when used for the following applications and conditions:

- When rapid program development is a requirement;
- For dynamic, evolving applications;
- For applications handling data that map into hierarchical, sparse data structures;
- For text manipulation;
- For applications involving interactive data base query and update operations;
- For applications involving extensive error checking of user-entered data.

TANDEM

TANDEM COMPUTERS INCORPORATED, 19333 Vallco Parkway, Cupertino, CA 95014. Toll Free (800) 538-9360 or (408) 996-6000 in California. Offices throughout the United States, Canada, Europe and the United Kingdom. Distributors in Australia, Finland, Mexico and Venezuela.

DS127901

TANDEM MUMPS

MUMPS, an acronym for Massachusetts General Hospital Utility Multi-Programming System, is a high-level, interpreted computer language designed for easy manipulation of string data and of hierarchically-structured data bases.

MUMPS is used extensively in commercial and medical applications because of its richness of features, ease of learning, ease of use, and superior string manipulation capabilities.

Tandem MUMPS differs from most current implementations in that it is not running under its own operating system, rather it runs under the Tandem Guardian operating system. As a result, users are able to execute COBOL, FORTRAN or TAL programs concurrently. And of course MUMPS has the benefits of running on the Tandem NonStop™ system so that applications can stay up and running even if a processor, disc, disc controller or I/O channel should fail during operation.

Standard MUMPS

Tandem Standard MUMPS supports all features of the ANSI 1977 MUMPS Language standard.

Tandem MUMPS Extensions

Tandem supplies an interface to COBOL, FORTRAN, and T/TAL programs, the reliability of NonStop computing, and the following extensions:

- String subscripting, which lets you give variables meaningful names such as NAME("LAST") rather than the cryptic numeric names such as NAME(4). String subscripts can be up to 254 characters long.
- Negative and decimal number subscripting, which lets you subscript variables with negative numbers (PERS(-12)) and non-integer numbers (MATR(23.75), MATR(-25.67)).
- The $ORDER function proposed by the MUMPS Development Committee, which returns all subscript values including non-numeric strings. $ORDER is supplied in addition to the $NEXT function.
- The $ZINSPECT and $ZRINSPECT functions, which perform a "scan until" capability from the front and back, respectively, of a string. These capabilities speed and simplify the stripping of leading zeros and trailing blanks.
- The $ZFINFO function, which returns information on all Tandem files, all of which can be accessed from MUMPS.
- The ZTRAP command, the $ZTRAP function, and $ZTRAP special variable, which provide a complete error trapping facility that can be tailored to individual applications.
- Commands and utilities that create, edit, and delete application and library routines.
- Commands that give access to Tandem environmental information.
- READ and WRITE commands which provide the capability of accessing sequential and key-sequenced files with record sizes up to 4KB.
- Extensions to the MUMPS READ and WRITE commands, which provide interprocess communication between MUMPS, COBOL, FORTRAN, TAL and other MUMPS processes.

The System MUMPS

Tandem MUMPS runs under the Tandem GUARDIAN operating system with full system resources at its disposal. With Tandem MUMPS you have:

- Access to the power of the Tandem Text Editor for code entry and maintenance.
- The capability of initiating any system or user process directly from MUMPS.
- The capability of communicating with COBOL, FORTRAN, and T/TAL as well as with other MUMPS processes.
- The capability of reading and writing disc files created by COBOL, FORTRAN and T/TAL programs through Tandem's ENSCRIBE data base manager.
- Access to the Tandem GUARDIAN/EXPAND communications network capabilities.
- Access to the GUARDIAN operating system facilities and resources, including the versatile Tandem spooler.

Reliability

The reliability of a Tandem MUMPS application is derived from the architecture of the Tandem 16 itself:

- The Tandem 16 is immune to the failure of any single component.
- The Tandem software, including the MUMPS interpreter, complements the NonStop™ environment.
- The hardware and software NonStop™ functions are transparent to the MUMPS application programmer and user.

Expandability

The Tandem system provides the capability of dynamically expanding your system from 2 to 16 processors. It also provides the capability of dynamically expanding into a Tandem network.

MUMPS Applications

MUMPS is a specialized computer language that is most useful and efficient when used for the following applications and conditions:

- When rapid program development is a requirement.
- For dynamic, evolving applications.
- For applications handling data that map into hierarchical, sparse files.
- For text manipulation.
- For applications involving interactive data base query and update operations.
- For applications involving extensive error checking of user-entered data.

TANDEM

TANDEM COMPUTERS, 19333 Vallco Parkway, Cupertino, CA 95014
—800-538-9360—

87901

TANDEM
COBOL

FEATURES

- **Conforms to ANSI COBOL X3.23-1974**
- **Full compatibility with ENSCRIBE file structures**
- **Runs in mixed language environment with T/TAL**
- **Includes Tandem Extension for NonStop(TM) Programming**

INTRODUCTION

Tandem/COBOL runs on the Tandem T16 Computer System – the only multi-processor system designed for NonStop, transaction-oriented, data base applications. Tandem/COBOL utilizes all the capabilities of Tandem's GUARDIAN Operating System and ENSCRIBE Data Base Record Manager – software designed to keep the system up and running while maintaining the integrity of the on-line data base. Thus Tandem/COBOL is compatible with programs written in T/TAL, Tandem's high level language for transaction processing, and is capable of running in a multi-language environment.

GUARDIAN features handled by Tandem/COBOL include;

- NonStop Operation
- Shared, Re-entrant Code
- Virtual Memory System
- Geographic Independence of I/O Devices
- Checkpoint/Checkmonitor Facilities

ENSCRIBE features handled by Tandem/COBOL include;

- Key-sequenced, Entry-sequenced and Relative file structures
- Logical file size to 4 BILLION bytes
- One primary and up to 31 alternate keys
- Optional mirror data base recording

LEVEL OF SUPPORT

NUCLEUS – Level 2

The nucleus gives a basic language for the internal processing of data within the four divisions of a program. Qualification, punctuation characters, data-name formation, connectives, and figurative constants are supported as well as ACCEPT, ADD, ALTER, COMPUTE, DISPLAY, DIVIDE, EXIT, GO, IF, INSPECT, MOVE, MULTIPLY, PERFORM, STOP, STRING, SUBTRACT, and UNSTRING statements.

Extensions to the language include the ability to call T/TAL procedures via the ENTER verb.

TABLE HANDLING – Level 2

Table Handling lets you define three dimensional fixed length tables of contiguous data items. Each item is accessed relative to its position in the table. Each item may be identified through use of a subscript or an index. This level also provides series options and the ability to vary the contents of indices by an increment or decrement. Table handling includes the use of the SET and SEARCH statements.

SEQUENTIAL I-O – Level 2

Sequential I-O reads records of a file one after the other, in the order they were written. Memory areas among files are shared, and the full facilities for the FILE-CONTROL, I-O-CONTROL, and FD entries are included. Within the Procedure Division, CLOSE, OPEN, READ, REWRITE, USE, and WRITE are recognized.

RELATIVE I-O – Level 2

Relative I-O accesses records in either random or sequential manner. Each record in a relative file is uniquely identified by an integer value greater than zero which specifies the record's logical position in a file. It provides full facilities for the FILE-CONTROL, I-O-CONTROL and FD entries.

TANDEM COBOL

TANDEM COBOL

Within the Procedure Division it provides full capabilities for the CLOSE, DELETE, OPEN, REWRITE, START, USE, and WRITE statements.

INDEXED I-O – Level 2

Indexed I-O accesses records in either random or sequential manner. Each record in an indexed file is identified by the value of one or more keys within that record. It provides full facilities for the FILE-CONTROL, I-O-CONTROL and FD entries. Within the Procedure Division it provides full capabilities for the CLOSE, DELETE, OPEN, READ, REWRITE, START, USE, and WRITE statements.

SORT-MERGE – Level 2

Sort-Merge orders one or more files of records, or combines two or more identically ordered files of records, according to a set of keys within each record. Optionally, you may apply some special processing to each of the individual records by input or output procedures using RELEASE and RETURN. This special processing may be applied before and/or after the records are ordered by SORT or after the records have been combined by MERGE. Also, you may sort one or more files, or combine two or more files, one or more times within a given execution of a COBOL program.

REPORT-WRITER – Null Level

SEGMENTATION – Null Level

Segmentation is automatic via Tandem's GUARDIAN Operating System.

LIBRARY – Level 1

Text from a library is copied into the source program using COPY. The copied text is not changed.

DEBUG – Level 1

Debug provides a basic debugging capability, including the ability to specify (1) selective or full procedure monitoring, and (2) optionally compiled debugging statements.

INTER-PROGRAM COMMUNICATION – Level 1

Inter-program Communication means a program can communicate with one or more programs using CALL and CANCEL. This communication is done by (1) transferring control from one program to another within a run unit, and (2) letting both programs have access to the same data items.

COMMUNICATION – Null Level

TANDEM COBOL EXTENSIONS

Several extensions have been added to Tandem/COBOL to permit use of Tandem's GUARDIAN Operating System.

1. Nonstop Extensions

   For Nonstop programming in COBOL, you include a "?NONSTOP" line in your source program. STARTBACKUP and CHECKPOINT are the verbs that make the program nonstop. Normally STARTBACKUP is called once at the beginning of the program to set the nonstop mode. Thereafter the CHECKPOINT verb is used to pass information to the backup process at critical points in the processing. In a nonstop program, checkpoints will also occur automatically upon any OPEN or CLOSE executed after the backup is established. Both of these verbs will set the special register, PROGRAM-STATUS, to indicate the outcome of the checkpointing operation.

2. Extensions to the standard COBOL I/O facility

   Three new verbs, LOCKFILE, UNLOCKFILE, and UNLOCKRECORD, have been introduced to allow the use of the corresponding system file and record locking routines. This addition allows separate processes to share a common data base. READ and REWRITE verbs have been extended to allow the specification of a LOCK or UNLOCK operation. The OPEN syntax has been extended to specify the file access, EXCLUSIVE, SHARED or PROTECTED, and to permit the SYNCDEPTH for files opened in the OUTPUT, I-O or EXTEND mode.

3. Calling TAL procedures

   The ENTER verb has been modified to call TAL procedures. This lets you access any TAL system external or your object library routines. These object library routines are assigned to the COBOL run unit at compile time.

```
      DIMENSION M (10,20)
      READ 100, I, J, K
      DO 200, IS = I, 10
      DO 200, JS = J, 20
  200 M (IS, JS) = K
      IF (I .EQ. J) GO TO 300
```

TANDEM FORTRAN

- **Conforms to full language specifications of ANSI FORTRAN, X3.9-1977**
- **Full use of all ENSCRIBE facilities including multi-key access and locking**
- **RECORD Structures for DDL compatibility**
- **Extensions for NonStop(tm) Programming**
- **Facilities for Interprocess Communications**
- **Support of Floating Point Arithmetic**

INTRODUCTION

Tandem FORTRAN runs on the Tandem T16 Computer System – the only multi-processor architecture designed for NonStop, transaction-oriented, data base applications. Tandem FORTRAN utilizes all the facilities of the GUARDIAN Operating System including Non-Stop operation, re-entrant code, interprocess communications, virtual memory, and ENSCRIBE data base facilities for keyed, relative, and sequential access, multi-key data paths, and concurrent record access. Tandem FORTRAN is capable of running in a multi-language environment.

LEVEL OF SUPPORT

Tandem FORTRAN conforms to the specifications described in the American National Standards Institute for Programming Languages – FORTRAN, X3.9-1977.

Data Types

Six data types are supported; INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL and CHARACTER. In addition data may be arranged as arrays or substrings. Dimensioned arrays can have up to seven subscripts.

Expressions

Expression types can be arithmetic, character, relational or logical. Operators allowed in expressions include:

Arithmetic	**	(Exponentiation)
	/	(Division)
	*	(Multiplication)
	–	(Subtraction)
	+	(Addition)
Character	//	(Concatenation)
Relational	.LT.	(Less Than)
	.LE.	(Less Than or Equal)
	.EQ.	(Equal)
	.NE.	(Not Equal)
	.GT.	(Greater Than)
	.GE.	(Greater Than or Equal)
Logical	.NOT.	
	.AND.	
	.OR.	
	.EQV.	
	.NEQV.	

Specification Statements

Support is included for DIMENSION, COMMON, EQUIVALENCE, IMPLICIT, PARAMETER, EXTERNAL, INTRINSIC, and SAVE statements. In addition, type statements are allowed for INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, and CHARACTER.

DATA Statement

DATA Statements can be used for initialization of variables, arrays, array elements, and substrings.

Assignment Statements

There are four kinds of Assignment statements: Arithmetic, Logical, Statement label (ASSIGN), and Character.

TANDEM FORTRAN

TANDEM FORTRAN

Control Statements

Sixteen Control statements are used to control the execution of a program.

Unconditional GO TO	DO
Computed GO TO	CONTINUE
Assigned GO TO	STOP
Arithmetic IF	PAUSE
Logical IF	END
Block IF	CALL
ELSE IF	RETURN
ELSE	
END IF	

Input/Output Statements

Nine Input/Output statements are supported.

READ	OPEN	BACKSPACE
WRITE	CLOSE	ENDFILE
PRINT	INQUIRE	REWIND

Tandem FORTRAN utilizes the UNIT, FMT, REC, IOSTAT, ERR, and END control parameters for READ and WRITE.

For OPEN the UNIT, IOSTAT, ERR, FILE, STATUS, ACCESS, FORM, RECL, and BLANK open parameters can be used. For CLOSE the UNIT, IOSTAT, ERR, and STATUS parameters can be used.

For INQUIRE specifiers may be used for IOSTAT, ERR, EXIST, OPENED, NUMBER, NAMED, NAME, ACCESS, SEQUENTIAL, DIRECT, FORM, FORMATTED, UNFORMATTED, RECL, NEXTREC, and BLANK.

For BACKSPACE, REWIND, and ENDFILE the UNIT, IOSTAT, and ERR specifiers can be used.

FORMAT Specification

A complete range of FORMAT specifiers is available. Both explicit and list-directed editing are allowed. Editing descriptors allowed are:

Apostrophe	P
H	BN and BZ
Positional	Numeric (I, F, E, D, G)
T, TL, and TR	L
X	A
Slash	List-Directed
Colon	S, SP, and SS

Functions and Subroutines

Four categories of procedures are supported:

Intrinsic Functions
Statement Functions
External Functions
Subroutines

External functions and subroutines may be specified in either FORTRAN or other programming language subprograms.

An ENTRY statement may be used to permit specification of the entry point into a subprogram.

BLOCK DATA Subprogram

Block Data subprograms are allowed for initializing variables and array elements in named common blocks.

TANDEM FORTRAN EXTENSIONS

Several extensions have been made to FORTRAN 77 to enhance its operation on Tandem systems.

1. NonStop Extensions

STARTBACKUP and CHECKPOINT functions allow a FORTRAN program to utilize the NonStop capabilities of GUARDIAN. STARTBACKUP is called once at the beginning of a program to establish the nonstop mode. Thereafter CHECKPOINT is used to pass critical information to the backup process. Checkpoints will automatically occur upon any OPEN or CLOSE after the backup has been created.

2. Structures

Structures provide the ability to define records in FORTRAN. The constructs RECORD and END RECORD are used to define record structures. The Data Definition Language may also be used to transcribe a schema into FORTRAN RECORD structures.

3. ENSCRIBE Extensions

Extensions have been made to the FORTRAN READ and WRITE statements to permit the full use of ENSCRIBE facilities. Thus it is possible with FORTRAN statements to access key-sequenced, relative, and entry-sequenced files by primary or up to 255 alternate keys. Provision has been made to allow exact, approximate or generic positioning into an ENSCRIBE file structure using FORTRAN. Concurrent record access is supported with LOCK mechanisms at either the record or file level.

4. Interprocess Communications

FORTRAN processes can communicate with one another or with processes written in other languages through the standard FORTRAN READ and WRITE statements. Communication to other processes is implemented using the interprocess communication facilities of the GUARDIAN Operating System.

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, CA 95014 ■ (408) 996-6000

T16/8742-00

TAL

TANDEM APPLICATION LANGUAGE

- **HIGH LEVEL LANGUAGE FOR APPLICATIONS AND SYSTEMS PROGRAMMING**
- **PROCEDURE-ORIENTED LANGUAGE FOR TRANSACTION PROCESSING**
- **EXTENSIVE DATA TYPES FOR ARITHMETIC AND STRING DATA TYPES**
- **SPECIAL STRING AND BIT MANIPULATION FUNCTIONS**
- **OBJECT CODE OPTIMIZATION COMPARABLE TO ASSEMBLER CODING**
- **FREE-FORM STRUCTURE**
- **CONSTRUCTS AMENABLE TO STRUCTURED PROGRAMMING**

TRANSACTION APPLICATION LANGUAGE (T/TAL)

The Tandem Transaction Application Language is a high level, block structured language designed for the easy implementation of transaction-oriented applications. T/TAL is similar to ALGOL, PL/I or COBOL in providing procedure blocks and high level constructs such as IF . . . THEN . . . ELSE, DO . . . UNTIL, WHILE . . . DO, FOR, and CASE. Programmers can write self documenting programs with object code generation as efficient as assembler code.

PROCEDURES

The procedure-oriented, block-structured approach allows a programmer to design procedure blocks for functional simplicity. Data can be assigned globally for all procedures to access or can be assigned locally — known only to the procedure in which it is declared. Local data is assigned dynamically and initialized each time the procedure is invoked. The local data feature also allows the use of recursive procedures.

One procedure can call another and pass parameters by value or by reference. The called procedure can optionally return a value through its name.

All procedure code is compiled as re-entrant and, therefore, non-modifiable. This permits code to be shared by many different users thereby minimizing the amount of memory required for multiple applications.

DATA DECLARATIONS

T/TAL provides data types for arithmetic, string and logical operations. Signed integer data can be specified with 15 bits or 31 bits of precision. Fixed data is a scaled decimal with up to 18 digits of precision. String data consists of one or more 8-bit bytes. Logical data consists of 16-bit words.

Data can be declared as simple, single element items or as arrays (multiple elements). In addition a variable can be declared as a pointer variable to facilitate indirect addressing.

Type functions are an integral part of TAL and provide the ability to convert from one data type to another. Type functions are also available to determine if string data represents ASCII NUMERIC, ALPHA or SPECIAL values.

SPECIAL FEATURES

Special string manipulation constructs are available to facilitate efficient processing of transaction data. Included are SCAN, MOVE and COMPARE string operations.

Data can be addressed at the bit level, if desirable. Operations exist in TAL to perform bit deposit, bit extraction, arithmetic shifting and logical shifting.

I/O

All I/O is performed via calls to operating system procedures. Procedures exist to open and close files, read and write data, perform control functions, create and purge files, lock and unlock files, and perform "wait" or "nowait" I/O. Extensive error checking is provided by the File System and error codes are returned to the calling procedure for inspection.

TAL CONSTRUCTS

General Form

```
[label:]    statement;
```

Specific Form

Assignment

```
variable := [variable := ...] arithexp;
```

Compound

```
BEGIN
statement1;
statement2;
.
.
statementn;
END;
```

Unconditional Branching

```
GOTO label;
```

Invoking Procedures

```
CALL procedurename [(parameterlist)] ;
RETURN [arithexp] ;
```

Conditional

```
IF condexp THEN statement [ELSE statement] ;
CASE index OF BEGIN
        statement0;
        statement1;
        .
        .
        statementn;
        OTHERWISE statement;
        END;
```

Iteration

```
FOR variable := initvalue TO limit [BY step] DO statement;
WHILE condexp DO statement;
DO statement UNTIL condexp;
```

String Manipulation

```
destarray { ':=' } { sourcearray FOR count } [→nextaddr] ;
          { ':=' } { constantlist          }

{SCAN } array { WHILE } testchar [→addr] ;
{RSCAN }      { UNTIL }
```

The listing on the next page shows a simple TAL program including GLOBAL and LOCAL data declarations and a procedure block. Also included is the variable map and procedure map which are used when debugging the program.

TANDEM APPLICATION LANGUAGE

```
TAL - TANDEM COMPUTERS VERSION A01  ( 6/25/76 - 10 AM)              SOURCE LANGUAGE: TAL - TARGET MACHINE TANDEM/16
DATE - TIME :  6/28/76 - 11:56:41                          OPTIONS: ON (LIST,CODE,MAP,WARN,LMAP) - OFF (ICODE,INNERLIST)

    1.    000000 0  0  ?NOCODE
    2.    000000 0  0  !  GLOBAL DATA DECLARATIONS
    3.    000000 0  0  INT .HOMETERM[0:11], .LINE[0:35];
    4.    000060 0  0  STRING .LINES := @LINE '<<' 1;
    5.    000060 0  0  DEFINE NULL = %0#, CRLF = [%15,%12]#;
    6.    000060 0  0  ?NOLIST
    9.    000000 0  0
   10.    000000 0  0  PROC CHANGE^DATE MAIN;
   11.    000000 1  0   BEGIN
   12.    000000 1  1  !  LOCAL DATA DECLARATIONS
   13.    000000 1  1  INT SAVE^ADDR1, SAVE^ADDR2, SAVE^ADDR3, MONTH^IN^DECIMAL, TERM;
   14.    000000 1  1  INT COUNT,STATUS, INPUT^DATA^POINTER;
   15.    000000 1  1  INT LENGTH^YEAR, LENGTH^MONTH, LENGTH^DAY;
   16.    000000 1  1  STRING .YEAR[0:3], .MONTH[0:2], .DAY[0:1], .TEMPS[0:3];
   17.    000000 1  1  STRING .MONTH^TABLE[0:35] := "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC";
   18.    000022 1  1
   19.    000022 1  1          CALL MYTERM(HOMETERM);                    ! GET TERM ID AND OPEN TERMINAL FILE
   20.    000050 1  1          CALL OPEN(HOMETERM,TERM,0);
   21.    000055 1  1   NEXT: LINES ':=' "ENTER DATE YY/MM/DD" & CRLF;
   22.    000072 1  1          CALL WRITEREAD(TERM,LINE,21,9,COUNT);     ! GET DATE AS YY/MM/DD
   23.    000103 1  1          IF COUNT = 0 THEN CALL STOP;              ! IF NO DATA THEN QUIT
   24.    000111 1  1          LINES[COUNT] ':=' NULL;                   ! INITIALIZE FOR SCAN
   25.    000121 1  1  ! GET THE YEAR AND EXPAND IT
   26.    000121 1  1          SCAN LINES UNTIL "/" -> SAVE^ADDR1;       ! SCAN FOR FIRST SLASH
   27.    000125 1  1          LENGTH^YEAR := SAVE^ADDR1 - @LINES;       ! CALCULATE LENGTH OFYEAR FIELD
   28.    000131 1  1          YEAR ':=' "19" & LINES[0] FOR LENGTH^YEAR; ! CONCATENATE "19" TO YEAR FOR A 4 DIGIT VALUE
   29.    000143 1  1          INPUT^DATA^POINTER := SAVE^ADDR1 - @LINES + 1;  ! CALCULATE POINTER TO NEXT FIELD IN DATE
   30.    000150 1  1  ! GET THE MONTH, CHECK FOR VALIDITY, AND CONVERT TO ALPHA
   31.    000150 1  1          SCAN LINES[INPUT^DATA^POINTER] UNTIL "/" -> SAVE^ADDR2;  ! SCAN FOR SECOND SLASH
   32.    000155 1  1          LENGTH^MONTH := SAVE^ADDR2 - SAVE^ADDR1 - 1;  ! CALCULATE LENGTH OF MONTH
   33.    000142 1  1          TEMPS ':=' LINES[INPUT^DATA^POINTER] FOR LENGTH^MONTH & NULL;
   34.    000174 1  1      ! SAVE MONTH NUMERIC VALUE FOR VALIDITY CHECK AND ALPHA LOOKUP
   35.    000174 1  1          CALL NUMIN(TEMPS,MONTH^IN^DECIMAL,10,STATUS); !CONVERT ASCII MONTH TO INTEGER
   36.    000203 1  1          IF STATUS <> 0 THEN BEGIN   ! IF CONVERSION IS NOT GOOD
   37.    000206 1  2                              ERROR: LINES ':=' "INPUT ERROR";
   38.    000215 1  2                                     CALL WRITE(TERM,LINE,11,COUNT);
   39.    000225 1  2                                     GOTO NEXT;
   40.    000234 1  2                              END;
   41.    000230 1  1          IF MONTH^IN^DECIMAL > 12 THEN GOTO ERROR;
   42.    000240 1  1      ! FIND ALPHABETIC VALUE FOR MONTH^IN^DECIMAL
   43.    000240 1  1          MONTH ':=' MONTH^TABLE[3 * (MONTH^IN^DECIMAL - 1)] FOR 3;
   44.    000251 1  1          INPUT^DATA^POINTER := SAVE^ADDR2 - @LINES + 1; ! CALCULATE POINTER TO NEXT FIELD IN DATE
   45.    000256 1  1  ! GET THE DAY AND SAVE
   46.    000256 1  1          SCAN LINES[INPUT^DATA^POINTER] UNTIL NULL -> SAVE^ADDR3;
   47.    000263 1  1          LENGTH^DAY := SAVE^ADDR3 - SAVE^ADDR2 - 1;
   48.    000270 1  1          DAY ':=' LINES[INPUT^DATA^POINTER] FOR LENGTH^DAY;
   49.    000274 1  1  ! FORMAT AND OUTPUT THE NEW DATE
   50.    000274 1  1          LINES ':=' CRLF & MONTH FOR 3 & " " & DAY FOR LENGTH^DAY & ", " & YEAR FOR LENGTH^YEAR+2;
   51.    000331 1  1          CALL WRITE(TERM,LINE,10+LENGTH^DAY+LENGTH^YEAR,COUNT);
   52.    000344 1  1          GOTO NEXT;
   53.    000351 1  1    END; ! END OF PROC

COUNT                          VARIABLE     INT          L+006    DIRECT
DAY                            VARIABLE     STRING       L+016    INDIRECT
ERROR                          LABEL
INPUT^DATA^POINTER             VARIABLE     INT          L+010    DIRECT
LENGTH^DAY                     VARIABLE     INT          L+013    DIRECT
LENGTH^MONTH                   VARIABLE     INT          L+012    DIRECT
LENGTH^YEAR                    VARIABLE     INT          L+011    DIRECT
MONTH                          VARIABLE     STRING       L+015    INDIRECT
MONTH^IN^DECIMAL               VARIABLE     INT          L+004    DIRECT
MONTH^TABLE                    VARIABLE     STRING       L+020    INDIRECT
NEXT                           LABEL
SAVE^ADDR1                     VARIABLE     INT          L+001    DIRECT
SAVE^ADDR2                     VARIABLE     INT          L+002    DIRECT
SAVE^ADDR3                     VARIABLE     INT          L+003    DIRECT
STATUS                         VARIABLE     INT          L+007    DIRECT
TEMPS                          VARIABLE     STRING       L+017    INDIRECT
TERM                           VARIABLE     INT          L+005    DIRECT
YEAR                           VARIABLE     STRING       L+014    INDIRECT

CHANGE^DATE                    PROC
CRLF                           DEFINE                               [%15,%12]
HOMETERM                       VARIABLE     INT          G+000    INDIRECT
LINE                           VARIABLE     INT          G+001    INDIRECT
LINES                          VARIABLE     STRING       G+002    INDIRECT
MYTERM                         PROC
NULL                           DEFINE                               %0
NUMIN                          PROC         INT
OPEN                           PROC
STOP                           PROC
WRITE                          PROC
WRITEREAD                      PROC

PEP    BASE    LIMIT   ENTRY  ATTRIBUTES  NAME

002   000003  000402  000025    M         CHANGE^DATE

OBJECT FILE NAME IS $SCR.SYSTEM.SAMPLE
NO. ERRORS=0    ; NO. WARNINGS=0
PRIMARY GLOBAL STORAGE=3
SECONDARY GLOBAL STORAGE=0A
CODE SIZE=256
DATA AREA SIZE=2  PAGES
CODE AREA SIZE=1  PAGES
MAXIMUM SYMBOL TABLE SIZE=383
ELAPSED TIME =   0: 0:19
```

TANDEM

ENSCRIBE™

FEATURES

- Fail Safe File and Record Management for Non-Stop™ Transaction-Oriented Applications
- Multiple Disc File Organizations (Key-Sequenced, Relative and Entry-Sequenced) with Exact, Generic or Approximate Multi-Key Access to All Records
- Automatic Management and Maintenance of Multi-Volume Files
- Mirror Volumes to Provide Fail-Safe Data Base Protection
- Data and Index Compression for Key-Sequenced Files to Optimize Disc Storage Space
- Main Memory Cache Buffering to Increase and Enhance Throughput
- Separate or Simultaneous Record and/or File Lock to Facilitate and Expedite Concurrent Record Access and Provide System Security
- Full Complement of Interactive High-Level Utilities to Reduce Applications Programming Overhead and Costs

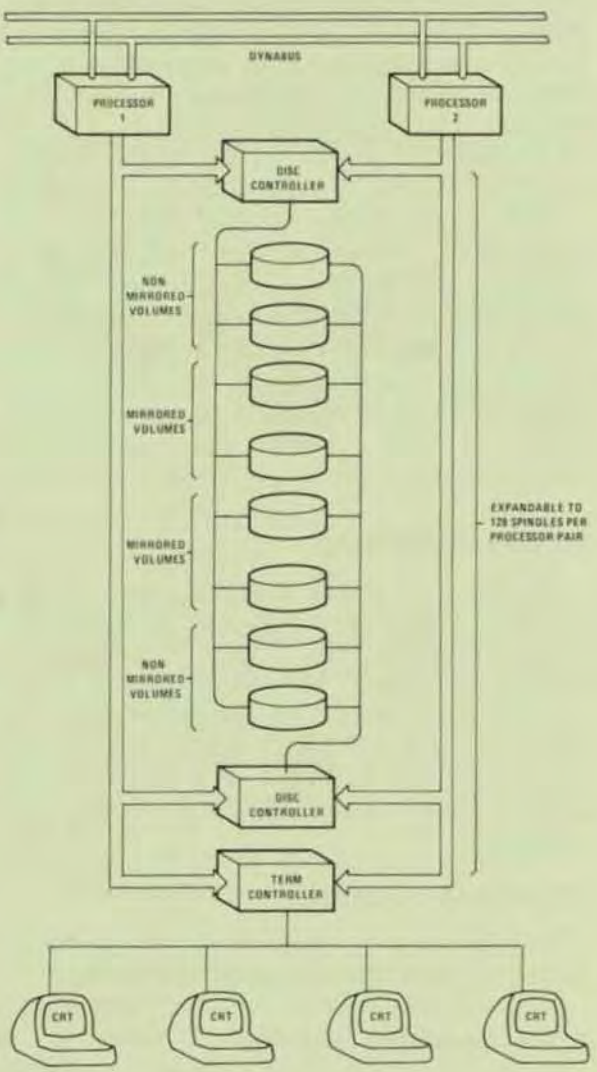


Figure 1. Multi-Volume File Configuration

TANDEM ENSCRIBE™

INTRODUCTION TO ENSCRIBE

The unique and evolutionary ENSCRIBE software package extends Tandem's Non-Stop concept of hardware and operating system failure immunity to fail-safe file and record management. Since ENSCRIBE operates as an integral part of the GUARDIAN Operating System across multiple processors, data base integrity is ensured in the event of a Processor, I/O Controller or Disc Drive failure. ENSCRIBE brings many advanced features not found even in traditional midi- and large-scale computer systems into the minicomputer price range.

DATA BASE INTEGRITY

When a conventional processor or disc drive fails, it is most often necessary to reconstruct the data base from archival tapes or manual records to ensure the integrity of the data base. Conversely, in a Tandem T16 system, when a processor fails, all file management functions are automatically transferred to an alternate processor. Optionally, ENSCRIBE manages a data base using the mirror principle illustrated in Figure 1. In this mode, data is written onto two disc volumes simultaneously. When a disc fails, all file management functions are automatically transferred to the mirror disc volume. Upon restoration of the failed disc, ENSCRIBE copies the data back onto the failed disc. This copying operation is done concurrent with requests to read or update data and is entirely transparent to both the application programmer and user.

MULTI-VOLUME FILES

A file may reside entirely on a single volume or may be partitioned to reside on several volumes. With system expansion to sixteen processors, and assuming that each volume is mirrored, the maximum size for each multi-volume file is nearly four billion bytes. And each partition of a multi-volume file is under the control of a separate processor.

FILE ORGANIZATIONS

ENSCRIBE provides a variety of file organizations for the applications programmer:

- Key-Sequenced (Indexed)
- Relative
- Entry-Sequenced

The programmer need not be burdened with structuring the file. He simply chooses the file organization best suited to a particular application and ENSCRIBE automatically structures the file. This allows a programmer to create, access and maintain data files with efficiency and economy. Moreover, since records are logically positioned, random records may be accessed sequentially. ENSCRIBE also provides Multi-Key Access to data records. Location of records in a Key-Sequenced file may be by approximate, generic or exact key value. ENSCRIBE maintains an index of key values which provides rapid access to data records. When a new record is added to a file, or when a key value has been changed, ENSCRIBE automatically updates the index including all secondary keys. This operation is entirely transparent to the application program.

DATA AND INDEX COMPRESSION

For key-sequenced files, an optional data compression technique may be used to pack more data into a particular disc area. Similarly, an optional index compression is provided for key indices.

MAIN MEMORY CACHE BUFFER

ENSCRIBE provides a Cache Buffer which resides in main memory. The purpose of the cache is, whenever possible, to keep the most recently accessed blocks in main memory speeding up access. System transaction rates may, therefore, be increased by simply allocating more memory to the cache. The cache resides in a separate area from the operating system and application programs.

RECORD AND FILE LOCKING

ENSCRIBE provides both record and file locking for system security. Record locking provides a greater degree of concurrent access to a file. A file lock request must wait for the record to be unlocked before it is granted.

SUMMARY

ENSCRIBE is complemented by Tandem's *Data Definition Language* (DDL) for data base management. ENSCRIBE may be used alone for those applications which do not require centralized data base administration, or in conjunction with DDL to provide full data base access. Thus, the user is provided with an economical growth path from file management to NonStop data base access.

ENSCRIBE FACILITIES

GENERAL SERVICES

ENSCRIBE provides file and record management services for the Tandem GUARDIAN Operating System. These services are interfaced to the application program via T/TAL or COBOL.

ENSCRIBE facilities may be used separately for handling file and record management tasks, or alternatively, may be used as an extension to Tandem's *Data Definition Language* (DDL); a *schema* for centralized data base management. Once a file organization has been established, programs may store, retrieve and/or modify logical data records without concern for file structure. ENSCRIBE facilities provide all necessary access control, data buffering, blocking/deblocking and file structure maintenance.

FILE AND RECORD STRUCTURE SERVICES

ENSCRIBE offers the application programmer a choice of three file organizations: *Key-Sequenced (Indexed), Relative,* or *Entry-Sequenced.* In addition, both fixed-length and variable-length record formats are supported under ENSCRIBE. A major advantage of fixed-length records is high performance during data access. Variable-length records optimize storage space on the disc. Both the file organization and record format must be specified at file definition time.

Key-Sequenced (Indexed) Files – In key-sequenced file organizations, records are stored in ascending order according to the value assigned a primary key field within the record. The primary key field may be defined as any contiguous set of bytes within the record. For example, a customer's name in an invoice file may be defined as the primary key. All records in an indexed file are variable-length. Primary keys must be unique. Up to 255 secondary keys can be specified.

Relative Files – Records are stored in a relative file according to their relative position within the file. The primary key for a relative file is the record number. These record numbers are ordinal values. Each record in a relative file is a fixed-length record. Relative files can have secondary keys.

Entry-Sequenced Files – In an entry-sequenced file, records are stored according to the sequential order in which they are presented to the system. These records may be either fixed or variable length – but once entered, the record's size may not be changed. The primary key for an entry-sequenced file is the record's logical address. Entry-sequenced files can have alternate keys.

Multi-Keyed Records – Under ENSCRIBE, a single file may have up to 255 alternate key fields, and the values contained in these alternate keys need not be unique. Thus, each file may have a primary key and alternate keys. These key fields are used to construct a secondary key index. This scheme is commonly called *Multi-Indexing.* The user need only specify the key values to access records. Moreover, ENSCRIBE automatically keeps all indices up-to-date during record updates and record insertions.

Multi-Indexing – Indexing provides the greatest flexibility in accessing records. Records may be accessed randomly by specifying a key or keys, or sequentially by consecutively accessing the records in the collated order of an index. Since multiple key fields can be defined, multiple indices allows an indexed file to appear as sequential. Moreover, ENSCRIBE provides three indexing options:

- Exact Key Match
- Approximate Key Match
- Generic Key Match

Exact key match means that the record's key field must exactly match the specified key. Approximate match means that the record key may be equal or greater than the search key. This allows a user program to access records without knowing the exact key. Generic key match means that only the initial portion (partial key) of a key need be specified (such as the prefix to a part number in a vendor's record).

Cache Buffering – Transparent to the application program, ENSCRIBE transfers one or more physical data blocks to main memory. Records within these blocks are then made available to the program. The purpose of this cache buffering is to keep the most recently accessed blocks in main

memory. Thus, index blocks for a Key-Sequenced File are likely to remain in the cache. ENSCRIBE automatically ensures that enough buffer space is available via overlaying the least recently accessed blocks, and also ensures that extraneous physical accesses are avoided.

FILE AND RECORD OPERATIONS

At program execution time, the user program may issue requests for various ENSCRIBE services. These services fall into two categories: *file* operations and *record* operations. Figure 2 illustrates a few of the available operations.

File Operations – File operations may be categorized as follows:

- Definition of an ENSCRIBE file organization.
- Opening of an existing ENSCRIBE file for processing.
- Closing of an ENSCRIBE file and termination of processing on that file.
- Examination of the attribute information stored within an ENSCRIBE file.
- Extension of the allocated space in an ENSCRIBE file.

Record Operations – Record operations may be summarized as follows:

- Read a record.
- Find a record.
- Insert a record.
- Update an existing record.
- Delete a record.
- Lock/Unlock a record.

```
INT .filename[0:11] := "$SYSTEM USER    FILE    ",
    .prim^key[0:2] := [5,0,1024],
    .buffer[0:49];
INT file^number, read^cnt, wrt^cnt, cnt^read, cnt^wrt;

STRING key[0:5];

LITERAL prim^ext = 5,
        file^code = 888,
        sec^ext = 2,
        file^type = 3,
        rec^len = 100,
        blk^len = 1000;

! Create a new keyed file
CALL CREATE(file^name,
            prim^ext, ,
            file^code,
            sec^ext,
            file^type,
            rec^len,
            blk^len,
            prim^key);

! Insert a new record into a keyed file
CALL WRITE(file^number,buffer,wrt^cnt,cnt^wrt);

! Find a record by key value
CALL KEYPOSITION(file^number,key);

! Read a record from a file
CALL READ(file^number,buffer,read^cnt,cnt^read);

! Update a record
CALL READUPDATE(file^number,buffer,read^cnt,cnt^read);
       .
       .
   Update the record in buffer
       .
       .
CALL WRITEUPDATE(file^number,buffer,wrt^cnt,cnt^wrt);

! Delete a record from a file
CALL WRITEUPDATE(file^number,buffer,0,cnt^wrt);

! Lock a record
CALL LOCKREC(file^number);
       .
       .
 Next Record Read will be Locked
       .
```

Figure 2. File and Record Management Examples

TANDEM

Enform – Query/Report Writing Language

Features

Query

- Powerful English-like relational query language easily used by non-programming personnel
- Retrieves data from multiple files which may be related in ways not anticipated during database design
- Used with Expand, allows queries on a distributed database
- Automatically develops the most efficient strategy to extract data from your database
- Keywords may be easily redefined to a different language, such as German, Spanish or French
- Uses Tandem's DDL statements to define database

Report

- Produces reports at a fraction of the programming cost and time of conventional languages, such as COBOL
- Reporting options allow sorting and summarizing of retrieved data as well as evaluation of built-in or user-defined functions
- Automatically spaces information on a page and supplies headings
- Accumulates information with your own formula, such as calculating commissions for sales people
- Formats numbers with commas, decimal point, currency sign
- Can be used from TAL, COBOL or FORTRAN programs to replace tedious report-formatting coding

ENFORM will make a difference in the time and energy required to extract a report from your database because you don't have to write a program, or be a programmer, to use it. ENFORM automatically writes the information gathered into rows and columns with headings and page numbers. If you want something different from the automatic settings, you may override them.

Enform Queries

Gathering the data from your database is done with the query portion of ENFORM. If your database was designed with specific relationships in mind, then you can take full advantage of them with ENFORM. However, what if you want the files related in a different manner, a manner not originally planned into the database? ENFORM handles this with no trouble; just tell it how you want the files related through a LINK statement.

Furthermore, ENFORM automatically selects an efficient method of searching for the data you requested. It uses existing keys and alternate keys to speed up the search (and cut down the cost of producing the report).

Defining the Database

The only task you must do before using ENFORM is define the database records with Tandem's Data Definition Language (DDL):

```
RECORD order.
   05    order-number;    PIC "9(4)".
   05    customer;        PIC "X(35)".
   05    part-number;     PIC "9(5)".
   05    quantity;        PIC "999".
RECORD parts.
   05    part-number;     PIC "9(5)".
   05    cost;            PIC "9(5) V99".
   05    part-name;       PIC "X(30)".
   05    stock-amount;    PIC "9(4)".
```

Ease of Use
ENFORM's statements and commands look like English words. For example:

```
OPEN order, parts
LINK order to parts VIA part-number;
LIST BY order-number,
   customer, part-number, quantity,
   (quantity * cost) HEADING "TOTAL COST",
   subtotal;
```

Keywords don't have to look like English – they may resemble any language you prefer, such as German or Spanish, by redefining them.

Entering the Statements
You can use ENFORM interactively by typing in the statements, one at a time, or put the commands into a file for later execution. You can save the "compiled" output from an ENFORM session so the ENFORM statements don't have to be checked and processed each time the report is created.

Choosing the Report Data
Criteria for selecting the data you want is defined with a WHERE statement:

```
OPEN parts
LIST part-number, part-name
   WHERE stock-amount LESS THAN 5
      AND part-number GREATER THAN 250;
```

Enform Reports

ENFORM's report phase automates many of the formatting details most reports require, such as centering titles and headings, spacing columns of data, underlining headings, skipping to a new page, writing headings. Special keywords further enhance these report duties, such as totaling, subtotaling, centering data, overriding headings, formatting numbers with commas, periods or currency sign.

Accumulation Operations
Common accumulation operations – percentage, cumulative sum, sum, average, count, maximum, minimum – are built into ENFORM:

```
LIST BY part-number
   MIN (cost)
   MAX (cost)
WHERE part-number EQUAL TO 5000 THRU 5999;
```

If you have an accumulation scheme not satisfied by these operations, you can build an expression to meet your needs.

Sample Report

10/04/78 – 10:39:35 AM
ORDER DETAIL REPORT OCT 04TH 1978

ORDER-NUMBER	PART-NUMBER	COST	QUANTITY	TOTAL-COST
35	244	87,000.00	1	87,000.00
	2001	1,500.00	2	3,000.00
	2403	9,600.00	4	38,400.00
	3103	10,500.00	2	21,000.00
	3302	2,800.00	1	2,800.00
	4103	24,500.00	2	49,000.00
	6301	2,900.00	1	2,900.00
	6302	4,300.00	2	8,600.00
*				212,700.00

TO: FRESNO STATE BANK FRESNO, CALIFORNIA
ORDER-DATE: 03/03/78 DELIVERY-DATE: 08/10/78

38	244	87,000.00	1	87,000.00
	2402	7,500.00	3	22,500.00
	3102	4,800.00	1	4,800.00
	4102	14,500.00	1	14,500.00
	5502	11,500.00	1	11,500.00
	6201	5,800.00	1	5,800.00
	6302	4,300.00	1	4,300.00
	6402	1,500.00	2	3,000.00
*				153,400.00

TO: BROWN MEDICAL CO., SAN FRANCISCO, CA
ORDER-DATE: 03/19/78 DELIVERY-DATE: 08/20/78

TANDEM

TANDEM COMPUTERS INC., 19333 Vallco Parkway, Cupertino, California 95014 • Toll Free 800-538-9360 or (408)996-6000 in California. Branch offices throughout the United States. Foreign Offices in Frankfurt, Dusseldorf and Munich; West Germany • Uxbridge, Middlesex; England • Zurich; Switzerland • Toronto; Canada.

TANDEM
ENVOY™
DATA COMMUNICATIONS MANAGER

FEATURES

- **Fail-Safe Data Communications Manager for Non-Stop™ Transaction-Oriented Applications**
- **Multiple Communications Protocols**
  - **IBM Binary Synchronous (BSC)**
  - **ADM-2 Asynchronous**
  - **TINET Asynchronous**
  - **Burroughs Synchronous**
- **Multiple Line Types (Data Links)**
  - **Point-to-Point (BSC only)**
  - **Centralized Multipoint Supervisor**
  - **Centralized Multipoint Tributary**
- **Trace Facility, Line Usage & Error Statistics, On-line Testing**
- **Support for Auto Call Facility**
- **Support for Multiple Modem Types**
  - **Bell type 103, 202 Asynchronous**
  - **Bell type 201, 208 Synchronous**
- **Simplified Applications Programming with Calls to GUARDIAN Operating System Procedures**

INTRODUCTION

The ENVOY *Data Communications Manager* provides an efficient, easy-to-use interface between transaction-oriented application programs and the telecommunications network. And since ENVOY is under control of the GUARDIAN *Operating System,* it has the same inherent *fail-safe* features of all other Tandem systems software. As such, ENVOY ensures that data communications are maintained even in the event of a processor or I/O channel failure. ENVOY provides all the control necessary for switched or leased point-to-point and leased multi-point telecommunication networks. An automatic calling option allows unattended stations on switched networks to communicate and exchange data without operator intervention. The ENVOY Data Communications Manager supports Synchronous transmission at speeds up to 56K Bps and Asynchronous transmission at speeds up to 19.2K Bps.

POINT-TO-POINT DATA LINK

ENVOY supports a Binary Synchronous point-to-point data link over either a switched or leased (non-switched) lines. In the switched network mode, each of two stations has a unique telephone number. The originating station dials the answering (remote) station and establishes a connection. After the connection has been established, system security messages may be interchanged to ensure the integrity of the two stations. Information interchange may then proceed. Once the interchange has been completed, the calling station automatically disconnects. The answering station detects the disconnection and then it also automatically disconnects. In the leased (non-switched) mode, the two stations are permanently connected (no number is required), but the data exchange sequence is the same as that described above for switched lines.

MULTIPOINT DATA LINK

A multipoint data link consists of two or more stations communicating over a leased (non-switched) line. With this type of data link, one station is designated the *supervisor* and controls all communications over the link. All other stations are designated *tributaries.* In a *centralized* multipoint link, communications can occur only between the supervisor and a tributary (tributary-to-tributary communication is not allowed). Each tributary station in a multipoint link is identified by a *polling* address and a *selection* address.

To solicit data transfers from the tributary stations, the supervisor station periodically *polls* each tributary station in the multipoint link by transmitting each tributary's polling address. When a tributary that has data to send is polled, further polling of the data link stops. At this point, the tributary responds by transmitting its data to the supervising station.

To transfer data to a tributary station, the supervisor station first *selects* the desired tributary station by transmitting the tributary's selection address. For selection to occur, the tributary station must be monitoring the line and be willing to accept the forthcoming data. Following selection, the supervisor station transmits the data to the tributary station.

TANDEM ENVOY™

TANDEM ENVOY™

ENVOY PROCEDURES

ENVOY functions are implemented via making calls to the GUARDIAN *Operating System's File Management Procedures*. These procedures are as follows:

- The **OPEN** Procedure is used to gain access to a data communications line.
- The **DEFINELIST** Procedure is used by application processes operating as a multi-point supervisory or tributary station. For a supervisory station, **DEFINELIST** specifies the polling address and selection address of each station. For a tributary station, **DEFINELIST** specifies the polling address(s) and selection address(s) that identify the tributary when the multi-point line is polled.
- The **WRITE** Procedure is used to transmit data to a remote station.
- The **READ** Procedure is used to accept data from a remote station.
- The **WRITEREAD** Procedure is used to transmit data to a remote station and then await a reply.
- The **HALTPOLL** Procedure is used by a station operating as a multipoint supervisor to stop continuous polling of all stations on a line.
- The **CHANGELIST** Procedure is used by application processes operating as a multipoint supervisory or tributary station. For a supervisory station, **CHANGELIST** is used to enable/disable polling of a particular station. Additionally, **CHANGELIST** is used to restore polling of non-responding units.
- The **CLOSE** Procedure is used to terminate access to a line.

TRACE FACILITY

ENVOY provides an aid in checkout of communications applications with the Trace Facility. The Trace Facility works with all line types except auto call units.

Trace Facility records line "events" in a Trace Table. Each Trace Table entry provides;

- Sequence Number
- Controller and Line Number
- Line State
- Event
- Miscellaneous Information
- Timestamp

LINE STATISTICS

Line statistics are maintained by ENVOY for each line from the time a line is opened until the line is closed. The statistics are printed on the operator console when a predetermined error threshold is exceeded, when a path switch occurs, or when the line is closed.

ON-LINE TESTING

ENVOYTST (Envoy Test) is used to verify the operation of the data communications process and the synchronous controller operating as the following line types:

- Point-to-point non-switched primary
- Point-to-point non-switched secondary
- Point-to-point switched primary
- Point-to-point switched secondary
- Multipoint supervisor
- Multipoint tributary

TANDEM COMPUTERS, INC. ■ 19333 Vallco Parkway ■ Cupertino, Calif. 95014 ■ (408) 996-6000

T16/8733-00