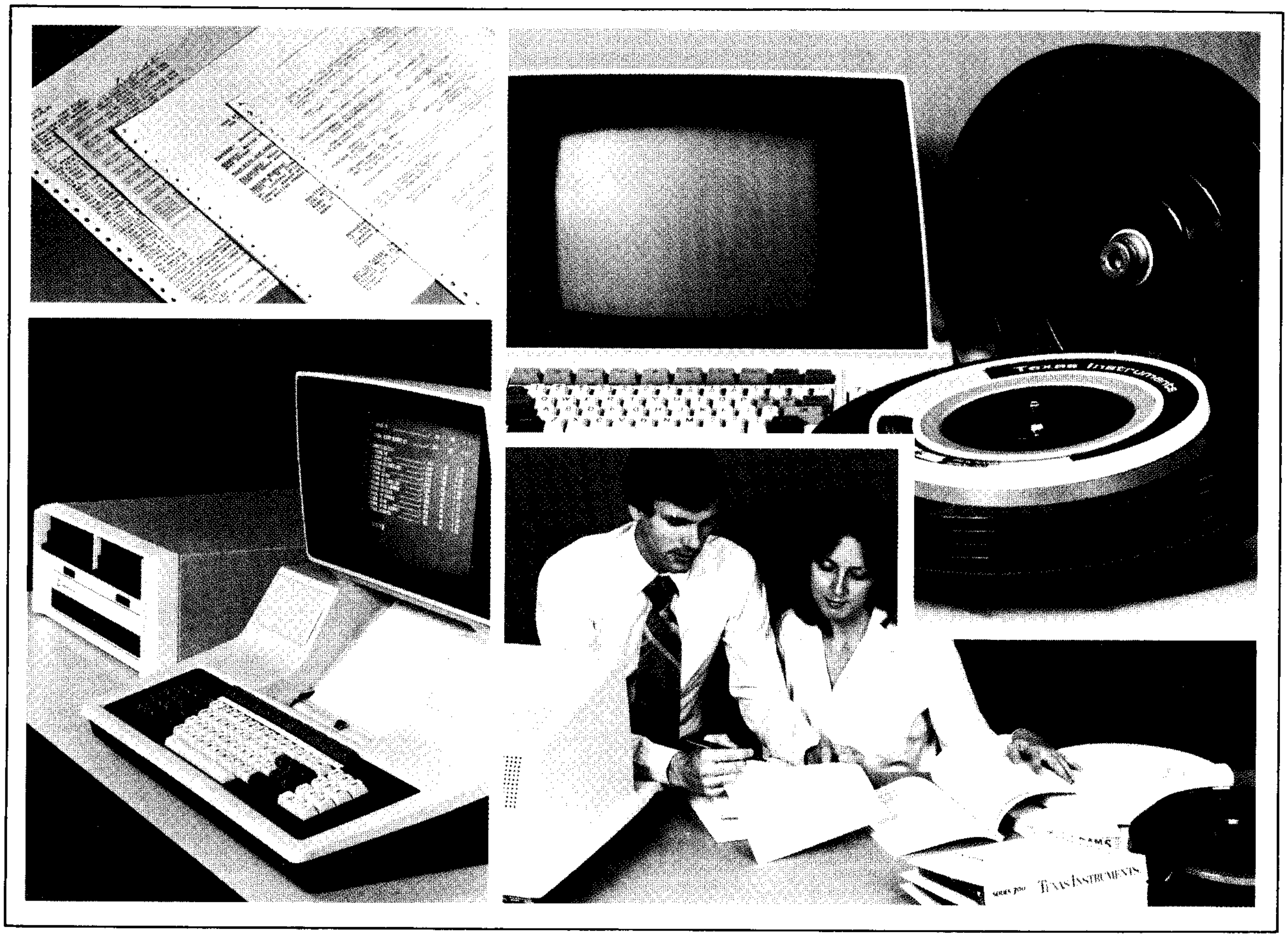
Model 990 Computer DX10 Operating System Concepts and Facilities Manual



Volume I

Part No. 946250-9701 *B
15 April 1981

TEXAS INSTRUMENTS

MANUAL REVISION HISTORY

Model 990 Computer DX10 Operating System Concepts and Facilities Manual, Volume I (946250-9701)

Original Issue	15 August 1977
Revised	15 March 1978
Change 1	15 October 1978
Revised	15 December 1979
Revised	15 April 1981

The total number of pages in this publication is 110.

PREFACE

The *Model 990 Computer DX10 Operating System Manuals* describe the features of the 990 Disk Executive, DX10. Each of the six DX10 operating system manuals includes a specific level of discussion related to a particular aspect of the DX10 operating system. All phases of system operation are treated throughout the manuals to enhance the development and application of programs. The manuals are organized with both the applications programmer and the production operation in mind and provide details to the system or application programmer to allow the extension and/or modification of DX10.

The user should consult all six manuals to become thoroughly familiar with all facets and capabilities of the operating system. Each manual serves a particular purpose and is designed to meet a specific goal. No single manual is intended to stand alone as a complete system tutorial. The titles and part numbers of the six manuals along with a brief comment regarding content and level of each are as follows:

Concepts and Facilities, Model 990 Computer DX10 Operating System — Includes features, concepts, and general background information describing the DX10 operating system.

Production Operation, Model 990 Computer DX10 Operating System — Describes the user interface to the DX10 operating system and application programs. It contains information required to start-up, query, control, and maintain DX10 and describes the steps needed to run application programs on Texas Instruments 990/10 and 990/12 computer systems under control of DX10. Accordingly, this manual describes the commands an operator may use to enter, access, execute, and control the execution of application programs. Also included is an introduction to terminal operation, details of log-on/log-off and operation of each specific user terminal.

Application Programming Guide, Model 990 Computer DX10 Operating System — Includes information required by the application programmer in the preparation or modification of application programs. It is primarily intended for assembly language programming under DX10 but also supplies a reference for high-level language programmers. Included is a discussion of program design and structure under DX10. It provides detailed information describing all calls for system services including input/output (I/O) processing. Information is provided describing the DX10 file structures and detailed information describing calls to DX10 for file I/O processing.

Developmental Operation Manual, Model 990 Computer DX10 Operating System — Includes operating instructions for the programmer who is creating new application programs. This volume describes software packages provided as a part of DX10 to support program development and includes a description of the Text Editor, the debug commands, and program installation/deletion functions.

Systems Programming Guide, Model 990 Computer DX10 Operating System — Includes information required by the systems or application programmer for extending or modifying DX10. It provides detailed discussion in such areas as system generation, support of nonstandard devices, and privileged supervisor service calls available within DX10. Also included are detailed instructions and descriptions of how to add commands to the System Command Interpreter (SCI) and how to customize DX10 for a particular configuration and application.

Error Reporting and Recovery, Model 990 Computer DX10 Operating System — Includes information describing error reporting within DX10. It documents task errors, command errors, supervisor call (SVC) errors, SCI errors, magnetic tape and other I/O errors. This manual documents all DX10 errors in cross-reference table form and includes a resolution of each message and suggested recovery techniques.

NOTE

Additional, in-depth descriptions related to specific languages including FORTRAN, COBOL, BASIC, RPG II, TI Pascal, Assembly Language, and Query are found in manuals dedicated to the appropriate programming language. A Link Editor manual is provided as a separate volume that describes the application of the link edit function in a DX10 environment. Separate manuals describe the use of an optional Sort/Merge package and the DBMS package.

The following documents contain additional information related to the DX10 operating system:

Title	Part Number
Model 990 Computer DX10 Operating System Production Operation Manual, Volume II	946250-9702
Model 990 Computer DX10 Operating System Application Programming Guide, Volume III	946250-9703
Model 990 Computer DX10 Operating System Developmental Operation Manual, Volume IV	946250-9704
Model 990 Computer DX10 Operating System Systems Programming Guide, Volume V	946250-9705
Model 990 Computer DX10 Operating System Error Reporting and Recovery Manual, Volume VI	946250-9706
Model 990 Computer DX10 Operating System Release 3.4, System Design Document	939153-9701
Model 990 Computer DX10 Operating System — Release 3 COBOL Programmer's Guide	946266-9701
Model 990 Computer FORTRAN Programmer's Reference Manual	946260-9701
Model 990 Computer Report Program Generator (RPG II) Programmer's Guide	939524-9701
Model 990 Computer TI 990 BASIC Reference Manual	2250304-9701
Model 990 Computer TI Pascal User's Manual	946290-9701
Model 990 Computer TIFORM User's Guide	2250374-9701
Model 990 Computer Model 911 CRT Display Terminal Installation and Operation	945423-9701
Model 990 Computer Model 913 CRT Display Terminal Installation and Operation	943457-9701
Model 733 Terminal User's Guide	945259-9701

Model 743 KSR Data Terminal Operating Instructions	984030-9701
Model 820 KSR Terminal Operator's Manual	2208225-9701
Model 781 RO Terminal Operator's Manual	2265935-9701
Model 783 KSR Terminal Operator's Manual	2265936-9701
Model 785 Communications Terminal Operator's Manual	2265937-9701
Model 787 Communications Terminal Operator's Manual	2265938-9701
Model 743 KSR Terminal Operator's Manual	984030-9701
Model 745 Portable Terminal Operator's Manual	984024-9701
Models 763/765 Operating Instructions	2203664-9701
Models 763/765 Memory Terminals Systems Manual	2203665-9701
TIBOL Programmer's Guide	2263354-9701
Model 990 Computer System 990/10 and /12 Assembly Language Reference Manual	2270509-9701
990/10 Minicomputer Hardware Reference Manual	945417-9701
DX10 3270 User's Guide	2250954-9701
Model 990 Computer DX10 Remote Terminal Subsystem (RTS) System Generation and Programmer's Reference Manual	2272034-9701
Model 990 Computer DX10 Remote Terminal Subsystem (RTS) Operator's Guide	2272055-9701
Model 990 Computer DX10 Remote Terminal Subsystem (RTS) Hardware Installation Manual	2272053-9701
DX10 3780/2780 Emulator Object Installation Manual	2250918-9701
Model 990 Computer DX3780/2780 Emulator User's Guide	946289-9701
Model 990 Computer 306 and 588 Line Printers Installation and Operation	945261-9701
Model 990 Computer PROM Programming Module Installation and Operation	945258-9701
Model 990 Computer Communications System Installation and Operation	945409-9701
Model 990 Computer Communications Systems Software	946236-9701

Model 990 Computer Terminal Executive Development System (TXDS) Programmer's Guide	946258-9701
Model 990 Computer Model FD800 Floppy Disk System with International Chassis Installation and Operation	2250697-9701
Model 990 Computer Model FD1000 Flexible Disk System with International Chassis Installation and Operation	2250698-9701
Model 990 Computer Model 733 ASR/KSR Data Terminal Installation and Operation	945259-9701
Model 990 Computer Model 804 Card Installation and Operation	945262-9701

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

DX10 GENERAL DESCRIPTION

DX10 is a general-purpose, multitasking operating system designed to operate with the Texas Instruments 990/10 and 990/12 minicomputers using the memory mapping feature. DX10 is a versatile disk-based operating system capable of supporting a wide range of commercial and industrial applications. It features a powerful file management package, which includes support for key indexed files. DX10 is interactively oriented, although a batch mode is also supported. It is also a multi-terminal system capable of making each of several users appear to have exclusive control of the system. DX10 may also be configured at System Generation (SYSGEN) time to support specialized real-time applications.

In addition to providing multi-terminal application support, DX10 features advanced program development support. A text editor is provided to enter source programs or data into the system. A Macro Assembler is provided for assembly language programs. Higher level language support includes FORTRAN, COBOL, RPGII, Pascal, and BASIC. A Link Editor and extended debugging facilities are provided to further support program development. A variety of utility programs, a data base management package, and a comprehensive Sort/Merge package are also supported.

DX10 is an international operating system designed to meet the commercial requirements of the United States, most European countries, and Japan. DX10 supports a complete range of international data terminals that permit users to enter, view, and process data in their own language. These terminals are currently available for the following countries: Austria, Belgium, Denmark, Finland, France, Germany, Japan, Norway, Sweden, the United Kingdom, and the United States.

SECTION 2

SYSTEM HARDWARE

2.1 HARDWARE FEATURES OF THE 990/10 AND 990/12

The 990/10 and 990/12 are advanced minicomputers of the Texas Instruments Model 990 computer family. They offer a memory mapping feature with memory capacities up to one million words. The instruction sets include both 16-bit word and 8-bit byte addressing capability.

2.2 REQUIRED HARDWARE

The DX10 general-purpose operating system requires certain minimum hardware consisting of the following items:

- The 990/10 or 990/12 CPU with the mapping option and a minimum random access memory of 64K 16-bit words.
- A system disk.
- A Video Display Terminal or Data Terminal.
- A means for disk backup.

Additional memory beyond 64K words is strongly recommended and is required to support additional terminals. Disk backup may be magnetic tape or a second disk drive.

Figure 2-1 shows a typical DX10 hardware installation .

2.3 SUPPORTED HARDWARE

2.3.1 MODEL DS10 DISK DRIVE. The Model DS10 disk is a dual platter, single-access moving-arm disk drive. A total storage of 10 megabytes is recorded on one 5-megabyte fixed non-removable platter and a 5-megabyte 5440 type disk cartridge. The disk format is 288 bytes per sector, 20 sectors per track, 816 tracks/platter, 2 platters (also 2 logical units) per disk. Single track seek time is 7.5 ms, average seek time is 35 ms, with average latency time of 12.5 ms. Transfer rate is 312K bytes per second.

2.3.2 MODEL DS31/32 DISK DRIVES. Models DS31/DS32 disks are moving-head disk units using 2315-type disk cartridges. The disk cartridge provides 406 total tracks on two heads, 203 cylinders; there are 24 sectors per track, 288 bytes per sector. Total storage at this format is 2,806,272 bytes per cartridge. The single track seek time is 15 milliseconds with a maximum seek time of 135 milliseconds and an average seek time of 70 milliseconds. The average latency time is 20 milliseconds. The transfer rate is 196,000 bytes per second.

2.3.3 MODEL DS25/DS50/DS200 DISK DRIVES. The Model DS25 and DS50 disk drives are physically-related, moving-head disk units utilizing the same type five-platter disk pack. The DS200 is a ten-platter pack. DS25 provides 408 cylinders or 2040 tracks, DS50 provides 815 cylinders or 4075 tracks, and DS200 provides 815 cylinders or 15,485 tracks. Tracks on each device contain 38 sectors with 288 bytes per sector. The DS25, DS50, and DS200 provide 22.33 million bytes, 44.60 million bytes, and 169.47 million bytes of storage, respectively. Single-track seek time is 6 milliseconds for the DS25 and DS50, and 7.5 milliseconds for the DS200. Maximum seek time is 55

(A)136679

Figure 2-1. 990 Computer System

2.3.4 CD1400/32 AND CD1400/96 DISKS. The CD1400/32 and CD1400/96 are moving-head disk drives that contain a removable unit and a fixed unit. The removable unit consists of one platter, and the fixed unit contains either one platter (CD1400/32) or three platters (CD1400/96). The CD1400/32 has a capacity of 32 million bytes, consisting of 16 million bytes on the removable unit and 16 million bytes on the fixed unit. The CD1400/96 has a capacity of 96 million bytes, consisting of 16 million bytes on the removable unit and 80 million bytes on the fixed unit. The disk surface for each disk is formatted into 821 tracks per surface, 64 sectors per track, and 256 bytes per sector. Using this format, the CD1400/32 and CD1400/96 contain a formatting capacity of 13,451,264 bytes on the removable unit plus 13,451,264 bytes (for the CD1400/32) or 67,256,320 bytes (for the CD1400/96) on the fixed unit. Single track seek time is 6 milliseconds, average seek time is 30 milliseconds, and maximum seek time is 55 milliseconds for each disk. The average latency time for the CD1400/32 and CD1400/96 disks is 8.33 milliseconds. The transfer rate is 1,210,000 bytes per second.

2.3.5 MODEL FD800 FLEXIBLE DISKETTE (FD) DRIVE. The FD800 flexible diskette drive (floppy disk) is a random-access drive that has a storage capacity of from 242,000 to 968,000 bytes with 77 tracks per drive and a transfer rate of 250,000 bits per second.

2.3.6 MODEL FD1000 DUAL-DENSITY DOUBLE-SIDED DISKETTE (FD) DRIVE. The FD1000 flexible diskette drive is a random-access drive with a storage capacity of 1,153,152 bytes with 77 tracks per drive and a transfer rate of 500,000 bits per second.

The storage media may be either a single-density or dual-density double-sided flexible diskette.

2.3.7 MODEL 913 VIDEO DISPLAY TERMINAL (VDT). The 913 VDT is an 80-character per line, 12-line VDT interactive device with 57 uppercase and 32 control characters. Display characters are formed by a 5 × 7 dot matrix.

2.3.8 MODEL 911 VIDEO DISPLAY TERMINAL (VDT). The 911 VDT is an 80-character per line, 12- or 24-line VDT interactive device. Display characters and a limited set of graphic characters are formed by a 5 × 7 dot matrix in either high or low intensity. The standard 911 VDT supports 128 upper- and lowercase characters plus 38 control characters. The 911 VDT is also available in various international versions, which meet the requirements of most Western European countries and Japan. In all of these versions, both the upper- and lowercase local character set is supported. The character codes generated and accepted by the VDTs are the internationally accepted Information Interchange codes for the countries involved. In the Japanese version of the 911 VDT, the intensity and graphics options are not supported.

2.3.9 MODEL 733 ASR/KSR HARD-COPY DATA TERMINALS. The Model 733 ASR/KSR terminals provide a 30-character per second print speed. The characters are generated character by character with an EIA line interface. The print head is a 5 × 7 dot matrix thermal type. The 733 ASR/KSR terminals are automatic send/receive and keyboard send/receive hard-copy terminals, respectively, with a full ASCII keyboard that contains 95 printable and 33 control characters. International versions of these terminals are available for most Western European countries and Japan.

2.3.10 MODELS 743 AND 745 PORTABLE KSR DATA TERMINALS. The Models 743 and 745 KSR terminals are keyboard send/receive hard-copy terminals with 50-character per second (743) and 30-character per second (745) print speeds. The characters are generated character by character with an EIA line interface. The print head is a 5 × 7, 35-element matrix thermal type. The Models 743 and 745 KSR terminals support ASCII standard and optional keyboards with 64 printable characters on the standard and 95 printable characters on the optional keyboard. The 743 and 745 KSR terminals support 33 control characters. The standard keyboard also includes an embedded 13-key numeric keypad. The Model 745 KSR terminal has a built-in acoustic coupler to provide remote capabilities. International versions of these terminals are also available.

2.3.11 MODEL 306 LINE PRINTER (LP). The Model 306 line printer is a 120-character per second, 80-character per line impact line printer. The characters are generated character by character with a 5 × 7 dot matrix print head. The Model 306 line printer incorporates a standard 64-ASCII character set (lowercase characters print as uppercase) and nine control codes.

2.3.12 MODEL 588 LINE PRINTER (LP). The Model 588 line printer is an 88-character per second, 132-character per line impact line printer. The characters are generated character by character with a 5 × 7 dot matrix print head. The Model 588 line printer incorporates a standard 64-ASCII character set (lowercase characters print as uppercase) and nine control codes.

2.3.13 MODEL 810 LINE PRINTER (LP). The Model 810 Receive-Only Printer is a remote office printer used in a data communication system. It is capable of printing 150 cps bidirectionally. The Model 810 printer has a 5 × 7 wire matrix print head allowing it to print 60 full, 132-character lines per minute. The Model 810 printer optionally provides a full ASCII character set (both upper-and lowercase letters) and Vertical Forms Control (VFC). International versions of the 810 printer are available for most Western European countries and Japan.

2.3.14 MODELS 2230/2260 LINE PRINTERS (LP). The Models 2230 and 2260 are medium-speed high-quality impact printers. They provide printing speeds of 300 and 600 lines per minute respectively, using a full 64-character ASCII character set in a 136-column format.

2.3.15 MODEL 979A MAGNETIC TAPE DRIVES (MT). The Model 979A magnetic tape drive is a serial-access 9-track magnetic tape transport. It features an IBM-compatible tape head and guide geometry with recording densities of 800 bpi (NRZI format) or optionally supports 1600 bpi, phase encoded (PE) format.

2.3.16 MODEL 804 CARD READER (CR). The Model 804 card reader is a column-oriented serial card reader. It is an 80-column, 400-card per minute card reader which uses a photoelectric-reflected light sensor that allows it to read both punched and mark-sense cards.

2.3.17 THE 3780/2780 AND 3270 COMMUNICATIONS PACKAGES. These two communications devices and their associated software emulate 3780 and 3270 communications terminals, using other terminals and line printers included in a DX10 system.

2.3.18 MODEL 915 REMOTE TERMINAL. The Model 915 Remote Terminal is an intelligent terminal designed specifically for use with the Texas Instruments DX10 Remote Terminal Subsystem (RTS) software package. RTS makes it possible for 915 remote terminals to be connected to a DX10-based computer system by means of communications lines. The 915 is most efficient for collecting and transmitting source data and is available with a standard numeric keypad for fast data entry. An 810 printer may be connected to a 915 remote terminal as an optional device.

2.3.19 MODEL 820 KSR DATA TERMINALS. The Model 820 KSR terminal is a keyboard send/receive hard-copy impact terminal with a 150-character per second print speed. Characters are placed in a 640-character buffer and printed either unidirectionally or bidirectionally for maximum efficiency. Characters are transmitted by an EIA line interface. The print head is a 9 × 7 dot matrix, which prints any of 128 upper- and lowercase ASCII characters. Up to five copies, plus one original, may be printed at the same time. International versions of the 820 KSR data terminal are available for most Western European countries and Japan.

2.3.20 MODELS 781 RO, 783/785/787 KSR DATA TERMINALS. The Model 781 RO terminal is a receive-only hard-copy terminal operating at 110 to 9600 bits per second. The Models 783, 785, and 787 KSR terminals are send/receive hard-copy terminals operating at 110 to 9600 bits per second. The Model 785 has an integrated modem that is compatible with the Bell 103, Bell 212A, and VADIC 3400. The Model 787 modems are registered with the FCC for use over public telephone lines without DAAs.

These terminals use the full character set and have thermal printers with a 5 × 7 dot matrix character font. The printers operate at 120 characters per second.

2.3.21 MODELS 763/765 BUBBLE MEMORY TERMINALS. The Models 763/765 bubble memory terminals are send/receive hard-copy terminals with thermal printers. The printers have a 5 × 7 character font and print 30 characters per second. The Model 765 has a built-in acoustic coupler. Bits per second rates are operator selectable. The rates are 110, 200, or 300 bits per second on an internal port, or up to 9600 bits per second using an RS-232-C interface.

SECTION 3

DISK MANAGEMENT

DX10 can accommodate many uniquely-named data files on a disk cartridge. It provides the necessary management for allocation of disk space to the files. Disk space is allocated in units referred to as Allocatable Disk Units (ADUs). An ADU is an integral number of disk sectors, with it's size being disk-capacity dependent. Larger disks have larger ADUs, but an ADU is always smaller than a track, and on some disks they are as small as one sector.

The amount of space allocated to a file may be as small as one ADU or as large as all of the available space on the disk cartridge. The user may specify the maximum amount of space to be allocated to a file. More frequently, however, the user may specify that space is to be automatically allocated to the file as it is needed. Automatic allocation of disk space to files is supported by a sophisticated disk management strategy designed to meet two performance objectives. The first objective is to provide access to any physical record of the file using one disk access. In most cases this also applies to the logical record of a file as seen by the application program. Exceptions include key indexed files where several disk accesses are often required to process a logical record read or write. The strategy used provides an in-memory directory capable of cataloging several segments of disk space allocated to the file. The segment size is variable. The second objective is to provide for wide dynamic range of file size without incurring excessive allocation overhead. Allocation overhead under DX10 refers to: the time spent in the allocation function, any disk space wasted by allocating more disk space than is really used, and any memory space used to catalog allocated disk segments. The strategy here is to make allocations for the first segments allocated small or minimal in size, and to increase the allocation size as additional secondary segments are required. The first segment allocated is equal to the initial allocation. The next segment allocated is contiguous, if possible; otherwise, it is equal to the secondary allocation. The third segment is equal to two times the last allocation size. Subsequent allocations (up to a maximum of 16) also use the formula for segment allocation; that is, two times the last allocation size.

SECTION 4

DISK VOLUME CONTENT AND ATTRIBUTES

4.1 GENERAL

Under DX10, each disk volume contains system files, system overhead, and space reserved for user files. DX10 utilizes one disk drive from which the operating system is loaded. That disk (designated as the system disk) is also used by DX10 to perform its internal disk-based functions. Figure 4-1 shows the physical layout of a disk volume.

4.2 SYSTEM OVERHEAD AREA

Track 0 and part of track 1 are always allocated to system overhead on the system disk as well as secondary disk volumes as follows:

- VOLUME ID
  Each disk cartridge under DX10 is identified by a user-assigned name. User-assigned names are recorded with system data on the named disk cartridge. The assigned name is called the VOLUME ID. (Track 0)

- ALLOCATION MAP
  DX10 maintains a map of the currently allocated disk space for each volume in blocks called Allocatable Disk Units (ADUs) on the disk. (Track 0)

- BAD DISK ADU MAP
  DX10 maintains a map of unusable ADUs on the disk. The bad disk ADU map is initialized to reflect disk condition at disk initialization. The bad disk ADU map may be automatically updated by DX10 during system operation. (Track 0)

- SYSTEM INTERMEDIATE LOADER
  A system loader is optionally stored on disks. This loader reads a DX10 image into memory at Initial Program Load time (IPL). (Track 1)

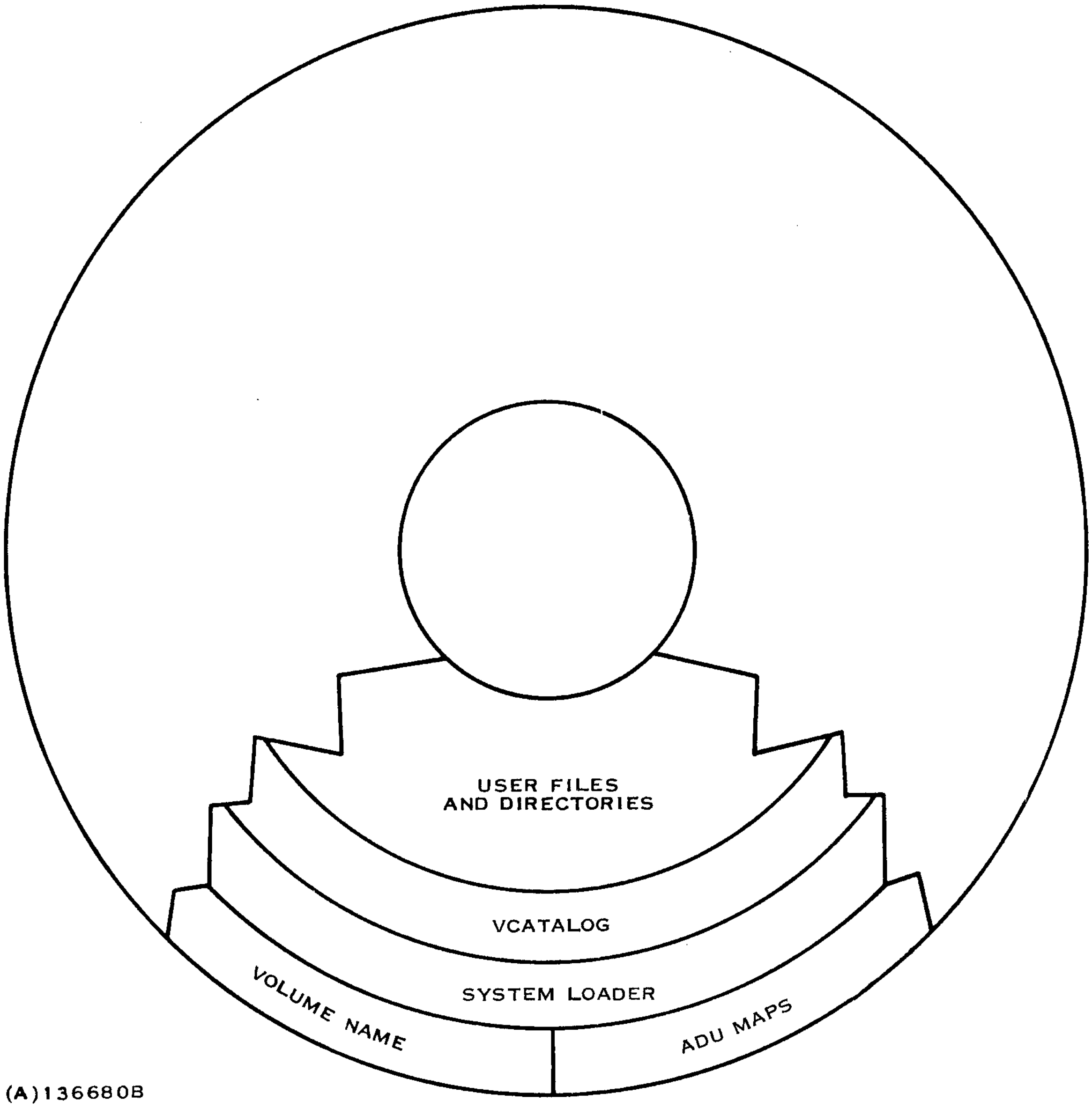
- VOLUME CATALOG (VCATALOG)
  Each disk volume has a specific file directory named VCATALOG where DX10 maintains a volume table of contents. The files described in VCATALOG may be data files or directory files as illustrated in figure 4-2. (Begins in Track 2).

4.3 DYNAMIC FILE AREA

The area on a disk volume other than system overhead is dynamically allocated as follows:

- SYSTEM FILES
  DX10 requires certain files to support DX10 internal disk based functions. System files follow a strict naming convention in order to minimize conflict with user-assigned file names. Each system file has a name beginning with "S$". The files currently in use by the DX10 operating system must reside on the system disk.

- USER FILES
  DX10 maintains user files on disk volumes. User files may be data files or directory files. Section V of this manual describes the types of data files supported by DX10. A representation of DX directory structures is shown in figure 4-2.



(A)136680B

Figure 4-1. Disk Volume Layout

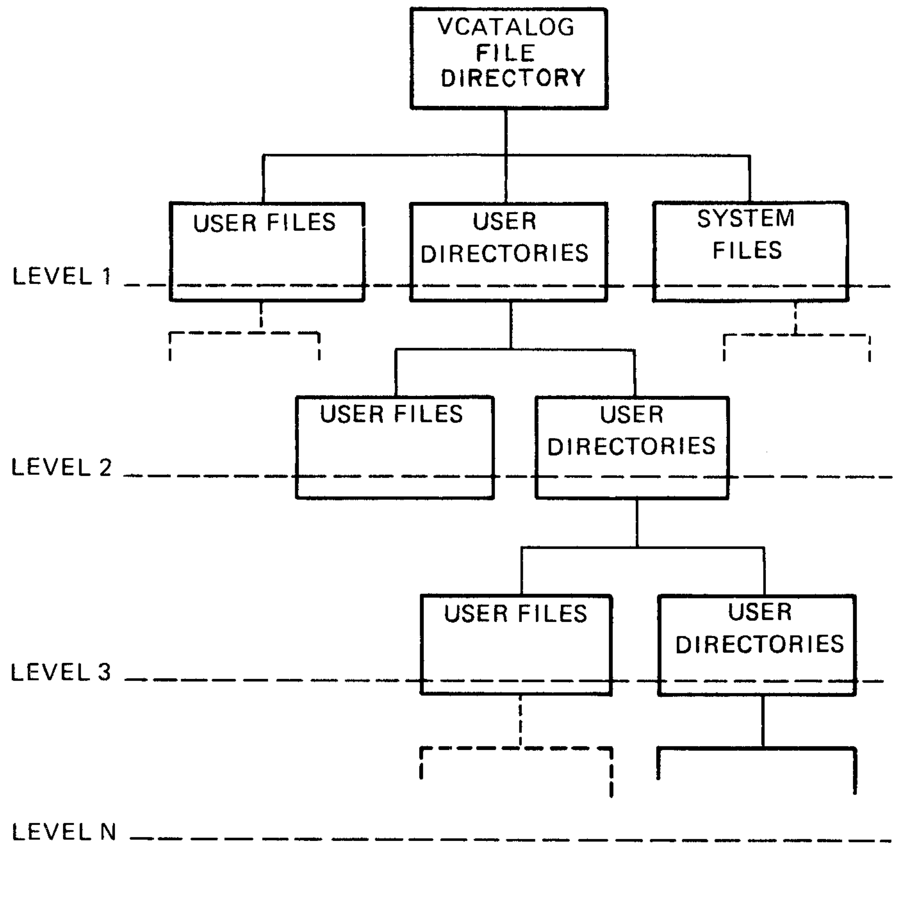


Figure 4-2. Files and Directory Structure

- PATHNAMES
  Any given file on a disk volume is referenced by its pathname. The pathname for a file is a concatenation of the volume name, names of the directory levels (excluding VCATALOG) leading to the file, and the file name itself. Components of the pathname are separated by periods. Some examples are:

  VOLONE.AGENCY.RECORDS
  MYDIRECT.MYDIRCTA.MYFILE
  VOLTWO.JOE
  EMPLOY01.USRA.PAYROLL
  EMPLOY01.USRB.PAYROLL
  EMPLOY01.USRB.CATALOGX.PAYROLL

  In the above examples, only JOE is directly cataloged in the VCATALOG directory of its volume,VOLTWO. Note that use of intervening directories allow distinct files of the same name to exist on the same volume. Absence of volume name implies the system volume.

Figure 4-3 illustrates the EMPLOY01 DIRECTORY example.

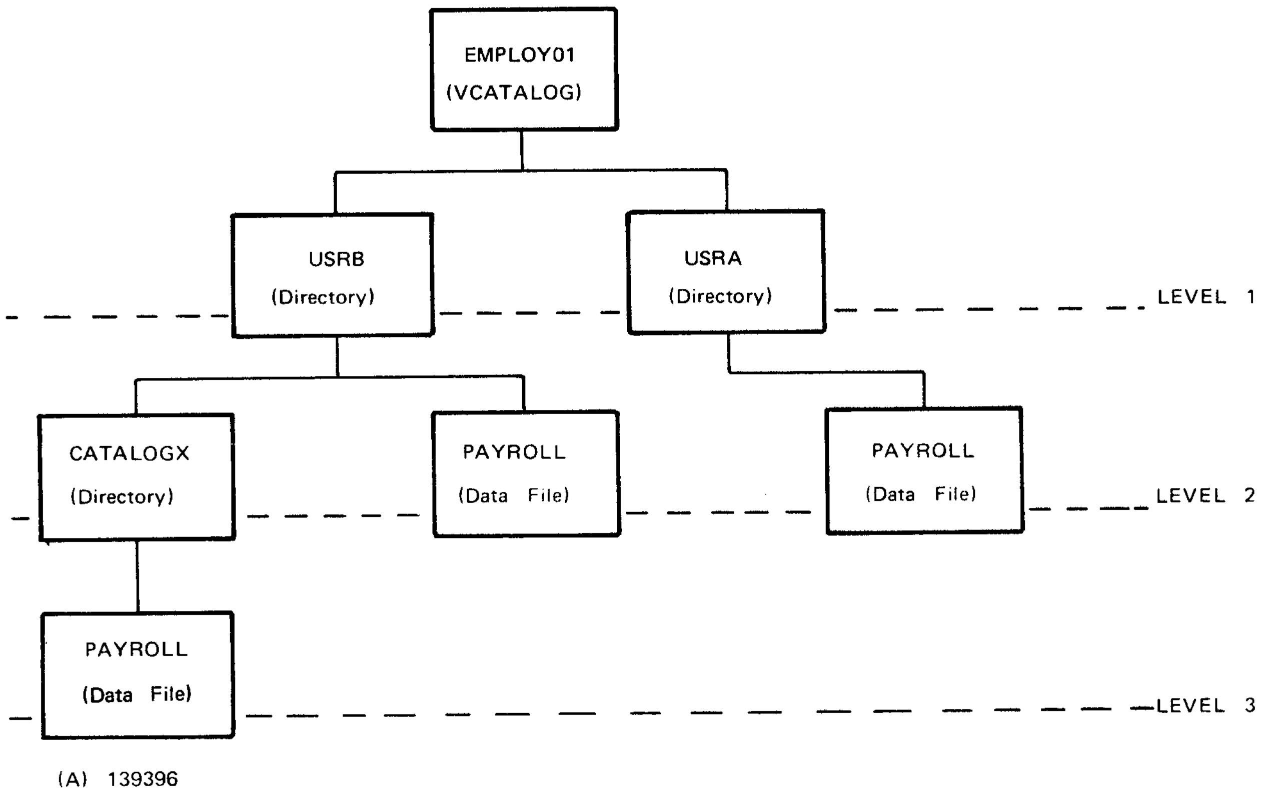


Figure 4-3. Example Directory Structure for EMPLOY01

SECTION 5

FILE TYPES

Three major file types for user data are supported by DX10. These are sequential, relative record, and key indexed files.

5.1 SEQUENTIAL FILES

Sequential files are useful for recording variable length data records in chronological order as received. Similarly, data must be read back in the same order. Random access to sequential files is impractical since, to acquire some given random record, all the intervening records must be processed. A pointer to the current file position is kept by DX10 for each active assignment to the file. As each record is read or written, the pointer is advanced. Sequential files have the property that no valid data may exist beyond the most recently written record. The only exception to this rule is that a limited rewrite of all records is supported. A sequential file may have *multiple end of file marks within the file*. Several programs may read a sequential file concurrently at different positions in the file. The current position is retained while the file is logically assigned even though the file is closed and reopened.

5.2 RELATIVE RECORD FILES

Relative Record Files are optimized for rapid random access. Fixed-length records are accessed by supplying DX10 the record number within the file. Such files are useful when the nature of the data lends itself to computation of a record number. DX10 increments the caller's record number after each read or write so that sequential access is permitted. One end-of-file record is maintained wherever last specified by a program.

5.3 KEY INDEXED FILES

The most sophisticated file type supported by DX10 is the key indexed file. In key indexed files variable length records are accessed by providing the operating system any one of up to 14 keys by which the data is known. A key is a string of up to 100 characters. For example, with this file type, an employee file may be constructed so that the data record for any given employee is accessed by supplying the employee's name, employee number, social security number, or any other designated key. Keys may be declared to overlay one another within the record. Although keys may be structured anywhere within the record, they must appear in the same relative position in all records in the file. One of the 14 possible keys must be selected as the primary key. All other keys are known as secondary keys. The primary key must be present in all records, but secondary keys may be optionally absent in any given record within the file.

5.3.1 KEY VALUES. Key values for both primary and secondary keys are kept in indexes within the file. These indexes are hierarchically structured for rapid random access while still allowing sequential access in the sorted order of any selected key. In a manner similar to that for sequential files, a pointer to the current position within the file is maintained by DX10 for each key. When updating any given record, the user of a key indexed file may add, delete, or change a key value.

5.3.2 FILE STABILITY. When a Read or Write operation is to be performed on a record in a key indexed file, DX10 makes a copy of the record before performing the operation. Thus, if a system failure occurs during the I/O operation, the prelogged record copy may replace the master copy, guaranteeing the stability of the file.

SECTION 6

FILE FEATURES

6.1 FILE APPLICABILITY TO LANGUAGES

The various file features and file types are all available to the assembly language programmer. High level languages may access any given feature depending on the syntax of the language. Assembly language programs can be written and called from high-level language programs, thus providing indirect access to features not supported directly by the language syntax.

6.2 DELETE AND WRITE PROTECTION

After each file is created it may be protected from accidental deletion from the volume. In some cases it may be desirable to further protect a file. DX10 permits file write protection – the data in a write-protected file may only be read. Files that are write-protected are automatically delete-protected.

6.3 FILE ACCESS PRIVILEGES

A DX10 program may request specific access privileges for any use of a disk file. A use may be defined as the entire file transaction from open through close. These privileges include:

1. EXCLUSIVE ACCESS: Only the calling program can access the file.
2. EXCLUSIVE WRITE ACCESS: Only the calling program can write to the file.
3. SHARED ACCESS: The calling program shares the file for read and write operations.
4. READ ONLY: The calling program is prohibited from writing to the file.

6.4 RECORD LOCKING

DX10 supports locking individual records within a file. This feature allows a program exclusive access to the locked record until that record is unlocked. An example of the use of record locking is locking an inventory record while updating the quantity in stock. The lockout prevents programs responding to other terminals from updating the same quantity before the first update is complete.

6.5 TEMPORARY FILES

DX10 allows the creation and use of temporary files. These files are unnamed and subject to subsequent deletion by the operating system. This feature allows a trial preparation of a file. If the prepared file is satisfactory, it may be RENAMED and designated permanent. If the prepared file is not RENAMED, it is purged by DX10 whenever the LUNO is released or when the task terminates.

6.6 BLOCKED FILES

Multiple logical records may be automatically combined by DX10 into larger physical records. These larger records are called file blocks or physical records. Blocking conserves disk space and reduces the number of physical transfers of data between memory and the disk.

6.7 DEFERRED OR IMMEDIATE WRITE

The physical transfer to disk of blocks of logical records is normally deferred by DX10 until the memory space held by the blocking buffer is required for some other purpose. This reduces the number of physical disk accesses since data may be recalled from memory. DX10 updates the image to the record on the disk before the file is closed. In some cases it may be desirable (e.g., for data integrity in highly sensitive files) that all writes to a file occur immediately upon request. DX10 supports this with the immediate write option.

6.8 BLANK SUPPRESSION AND ADJUSTMENT

For file types that support variable length records (all except relative record), extraneous blank characters may be removed from each record. Blank suppression is a feature in which strings of consecutive blanks within the record are encoded in a shortened form. Blank adjustment is the removal of trailing blanks on a write operation and replacement of them on a subsequent read operation. Blank adjustment is available to devices as well as files.

6.9 EXPANDABLE FILES

DX10 permits declaration of the initial file size at file creation. Unless otherwise specified, when the file exceeds this initial allocation, additional space is automatically allocated. In this way, files may continue to grow beyond their initial bounds. These secondary allocations to the file become increasingly larger as the file expands beyond its current extent.

SECTION 7

LOGICAL INPUT/OUTPUT

7.1 DEVICE NAMES AND ACCESS NAMES

The DX10 operating system supports the assignment of four character names to each peripheral device at system generation time. Certain default names must be used (except for special devices). Sample peripheral device names include: ST02 for a terminal, LP01 for a line printer, DS03 for a disk, and CS02 for a cassette unit, etc. Peripherals are identified by these names when operating under DX10. Files are identified by pathnames. A volume on which a file resides may be referenced either through the device name or volume name. Omitting the volume name altogether, and beginning a pathname with a period, implies the system disk. Samples of valid names for devices and files are as follows:

File Identifier	Meaning
CR01	Device name.
DS02.MYCAT.MYFILE	Device name, directory name, file name.
.MYCAT.MYFILE	System disk, directory name, file name.
CS02	Device name.
VOLID.MYCAT.MYFILE	Volume name, directory name, file name.

Generically, the names of devices and files are called access names. Access names may be either device names or file pathnames.

7.2 LOGICAL UNIT NUMBERS (LUNOS)

DX10 performs input/output to logical units instead of physical units making programs more flexible in the disposition of input and output. For example, a program may be written to accept input from LUNO 82. Prior to operating the program, an assignment is made associating LUNO 82 with some selected access name. In this way the program input may be assigned to ST03, while for another use of the program, input might be assigned to:

MYDISK.MYDIRCT.FILEX.

A LUNO assignment may be made by either a user or a program via a supervisor call. The choice of a LUNO assignment number may be specified by the user or optionally generated by DX10. LUNOs may range in value from 0 to FF_{16} (255).

7.3 SCOPE OF LOGICAL UNIT NUMBER ASSIGNMENTS

Logical unit numbers exist at all access levels. A logical unit number assignment may apply only to the program that made the assignment, to all programs running for a given terminal, or to all programs. DX10 attempts to map a LUNO to a device by first searching LUNO assignments local to the requesting program (Task Local LUNO), then LUNO assignments local to the terminal for which the program is running (Station Local LUNO), and finally the LUNO assignments which are available in scope to all programs (global LUNO). Task and station local LUNO assignments allow different programs and different users of the same program to have exclusive use of a LUNO even though the LUNO number may be in use by another program or user. Some global LUNOs are preassigned by the system, as shown in the appendix in the *Model 990 Computer DX10 Operating System Systems Programming Guide.*

SECTION 8

DEVICE AND FILE SERVICES

8.1 GENERAL

Figure 8-1 provides a scenario of typical device and file operations. Steps 1, 2A, 2B and 2C apply only to disk files. Note that for any disk file, the volume on which that file is stored must be installed. Therefore, the first step is installation of the appropriate disk volume, if it is not already installed. For disk volumes that have not previously been used, the operator must request that DX10 initialize the disk. Normally, however, the volume is already installed and step 1 is bypassed.

8.2 ASSIGN LUNO

If a file is to be accessed, it must have previously been created and its name recorded in a directory on the volume. If the desired directory does not already exist, it must be created. Both directories and files may be created either by System Command Interpreter (SCI) commands under operator control or by a task issuing I/O service calls. A file may be created automatically by DX10 in conjunction with an assign LUNO supervisor call using the auto-create option. Since files or devices are accessed by programs through a LUNO, the LUNO must be assigned to the appropriate access name (steps 3A and 3B). Either the operator or the program may request a LUNO assignment.

8.3 NORMAL I/O OPERATIONS

Steps 4, 5 and 6 of figure 8-1 constitute the normal input/output operations of a program. A complete list of operations available is provided in table 8-1. When read and write operations are requested, they may be categorized as either execute or initiate operations. If they are execute operations, DX10 does not return to the requesting program until the operation is complete. DX10 returns before the operation is complete for initiate operations. This option on all data transfers allows the program to elect whether to overlap computation and input/output in the user program (initiate I/O), or to allow other programs to run during the I/O operation.

8.4 FILE-ORIENTED DEVICES

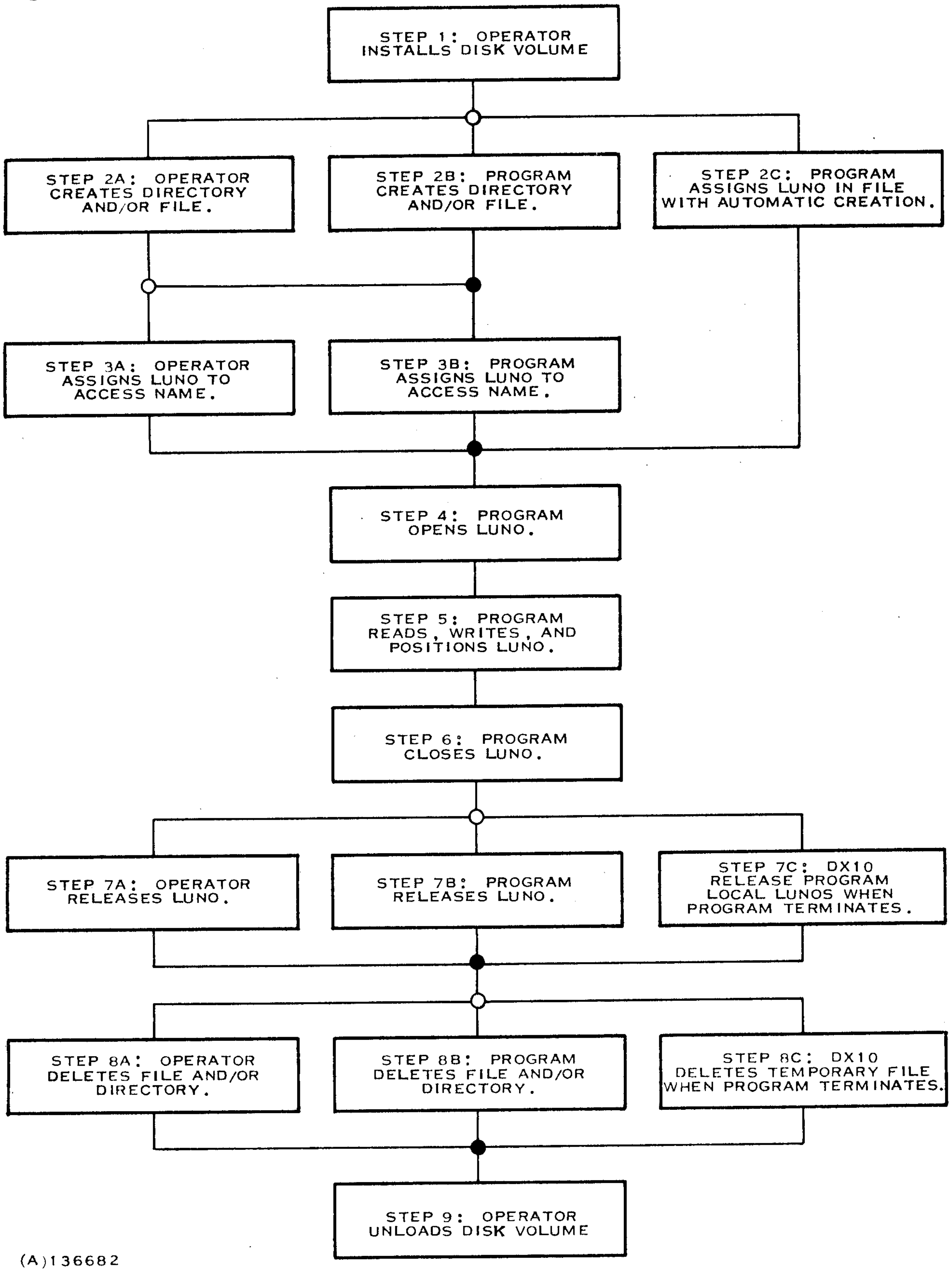
Certain devices may be declared with exclusive access privileges at system generation time. When a device is file-oriented, it becomes the property of the program that successfully performs an open operation to a LUNO assigned to the device; no other program is permitted to intersperse output or steal input records while the device is in use. The device becomes available to other programs when that program has terminated and only after that device is closed. File-oriented devices support exclusive access privileges only.

8.5 RECORD-ORIENTED DEVICES

The alternative to the file-oriented device is the record-oriented device. In this case, records may be freely interspersed between programs. Note that file and record orientation only applies to devices. Disk files are controlled by specifying access privileges at open time. Record-oriented devices support shared access privileges only.

8.6 DEVICE STATE

All devices have two possible states, online and offline. Normally, devices are in the online state, which makes the device available for LUNO assignment. On the other hand, a device in the offline state is not available. A device may be in the offline state for various reasons but typically because the device is under repair or is not physically connected to the computer.


STEP 1: OPERATOR INSTALLS DISK VOLUME
STEP 2A: OPERATOR CREATES DIRECTORY AND/OR FILE.
STEP 2B: PROGRAM CREATES DIRECTORY AND/OR FILE.
STEP 2C: PROGRAM ASSIGNS LUNO IN FILE WITH AUTOMATIC CREATION.
STEP 3A: OPERATOR ASSIGNS LUNO TO ACCESS NAME.
STEP 3B: PROGRAM ASSIGNS LUNO TO ACCESS NAME.
STEP 4: PROGRAM OPENS LUNO.
STEP 5: PROGRAM READS, WRITES, AND POSITIONS LUNO.
STEP 6: PROGRAM CLOSES LUNO.
STEP 7A: OPERATOR RELEASES LUNO.
STEP 7B: PROGRAM RELEASES LUNO.
STEP 7C: DX10 RELEASE PROGRAM LOCAL LUNOS WHEN PROGRAM TERMINATES.
STEP 8A: OPERATOR DELETES FILE AND/OR DIRECTORY.
STEP 8B: PROGRAM DELETES FILE AND/OR DIRECTORY.
STEP 8C: DX10 DELETES TEMPORARY FILE WHEN PROGRAM TERMINATES.
STEP 9: OPERATOR UNLOADS DISK VOLUME

(A)136682

Figure 8-1. Device and File Input/Output Scenario

8.7 RELEASE LUNO

When a LUNO assignment is no longer appropriate, it should be purged from active assignments retained by DX10. The release of a LUNO (steps 7A and 7B) can be made by either SCI operator commands or a task issuing programmed I/O supervisor calls. Task local LUNOs are automatically purged by DX10 when the task terminates (step 7C). Station local LUNOs are purged when:

- A terminal user has logged off.
- All programs initiated from the terminal have completed.
- Any batch stream associated with the terminal has completed.

8.8 TEMPORARY FILES

If temporary files are used by a task, they are automatically deleted from the disk volume when the last LUNO assigned to the file is released, or the task terminates (step 8C). Otherwise if it becomes desirable to delete a file, SCI operator commands and supervisor calls are available to do so (step 8A, 8B). When all files have been accessed and the volume is no longer in use, it may be unloaded through the use of an SCI operator-entered unload command (step 9).

The I/O Supervisor Call is provided to support all record transfer and file positioning operations in devices and files from a program. Additionally, utility operations such as file creation and LUNO assignment are available through the I/O Supervisor Call. A complete list of these operations is provided in table 8-1. Supervisor calls are discussed in greater detail in the *Model 990 Computer DX10 Operating System Application Programmer's Guide.*

Additional supervisor calls provided with DX10 structure are designed to complement and support I/O operations. These are listed below:

- Wait for previously initiated I/O to complete.
- Wait for any initiated I/Os to complete.
- Abort previously initiated I/O operation.
- Fetch special keyboard event character (function key).

Table 8-1. Device and File Operations Available Through I/O Supervisor Calls

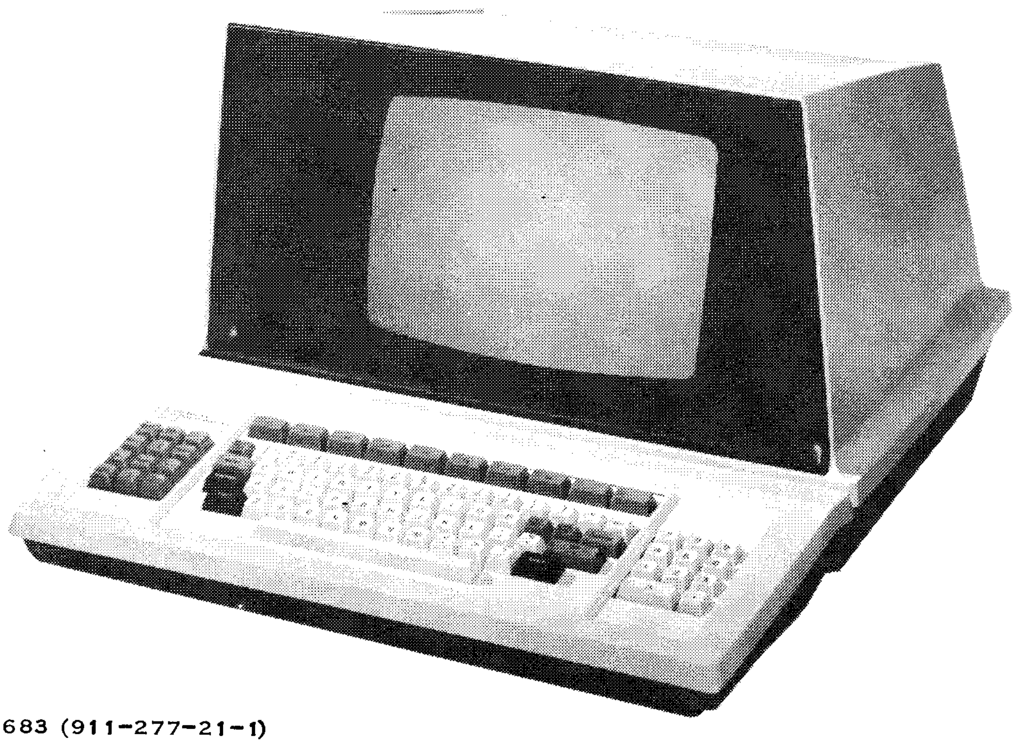
Assign LUNO	Rewind
Release LUNO	Unload
Fetch characteristics of device or file	Rewrite the record previously read
Verify legality of access name	Create a file
Open LUNO	Delete a file
Close LUNO	Establish immediate/deferred write mode
Close LUNO and write end-of-file	Change a file name
Open LUNO and rewind	Write protect/delete protect/unprotect a file
Forward space record	Add an alias name for a file
Backward space record	Delete an alias name for a file
Read record	Unlock a record
Write record	Key-indexed file operations
Read direct (used to acquire data with special formats)	Open extend
Write direct (used for special data formats)	Modify access privileges
Write end-of-file	

SECTION 9

EXTENDED VIDEO TERMINAL SUPPORT

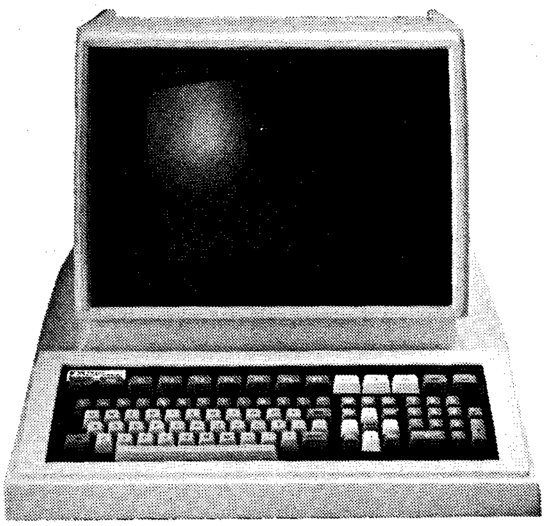
DX10 offers expanded I/O support for video terminals. The Model 911 VDT is shown in figure 9-1 and the Model 913 VDT in figure 9-2. Device-dependent I/O support provides access to many unique features available in either or both of these specific terminals as described below:

- Intensity Control – The programmer is able to control the high or low intensity of data displayed.
- Beep Control – Optional beep signals may be issued on read or write operations.
- Field Definition – The calling program is able to establish screen (row and column) limits on input fields.
- Default Data and Fill Character – Fields for data entry may be prefilled with default data. In addition, the field may be filled (beyond the default data if any) with an operator specified fill character such as underscore, period, blank, etc.
- Scrolling – Optionally the screen image may be moved (scrolled) upward or downward any specified number of records by the user.
- System Handled Editing – Within an input field, DX10 allows editing for left and right cursor keys, insert or delete character keys, field boundary checking, and field erase key for re-keying.
- Special Event Characters – Function keys and certain keystrokes are set aside and routed to a requesting task upon demand. Two event characters are buffered.
- Buffering Keystrokes — Up to "n" keystrokes are buffered by DX10 and saved pending a read request. The value "n" is defined during system generation, with a default value of six.
- Graphics — The 911 VDT provides graphics capability through special graphics character sets.



(A)136683 (911-277-21-1)

Figure 9-1. Model 911 Video Display Terminal



(A)136684 (914-775-4-2)

Figure 9-2. Model 913 Video Display Terminal

SECTION 10

PROGRAM MANAGEMENT

10.1 GENERAL

User programs that operate under control of the DX10 operating system may include a composite of data, procedures, and overlays as required. Programs are installed and stored in program files in memory image form. When a program is activated, the images of its program segments are loaded into any available memory areas. The 990 hardware memory mapping facility precludes the necessity to relocate program images. An active program may be rolled in and out of various locations in memory by DX10 several times during its execution to efficiently share memory and available CPU execution time. When in memory and active, a program competes with other programs for CPU execution time on a priority basis. When a program terminates, DX10 releases all program-owned resources including files, devices, and memory. This DX10 program structure is made possible by the 990 hardware memory mapping capability that allows three separately loaded program segments to be mapped into a single contiguous program address space.

10.2 TASKS

A specific activation of a program is referred to as a task. DX10 is able to concurrently share the memory, machine execution time, and peripheral resources of the system among several tasks. While one task is actually active (executing) others are suspended awaiting reactivation. In a typical mixture of tasks, most are awaiting execution pending completion of some input or output operation. While these tasks await input/output completion, other tasks (one at a time) access system resources and execute instructions.

10.3 SHARED PROCEDURES AND REPLICATED TASKS

It is frequently desirable to have several concurrent (replicated) executions of the same program. Such is the case in many multi-terminal environments or in industrial applications where several similar device types are controlled. An example might be a program serving bank tellers interacting with several tellers concurrently. Similarly one program may be used to control several hundred sewing machines at the same time. In many cases the procedural part of a program may be common to each of a number of concurrent executions, whereas, the data for each execution may be unique to that execution. Therefore, the user may develop differentiated functions while employing the same procedures. Under DX10, the shared procedural part is called a PROCEDURE while the unique part is called the TASK. A program operating under DX10 may consist of a task and one, two or no procedures. The procedures may be shared with other executing tasks. Sharing of procedures conserves memory usage by precluding the necessity to replicate the procedural part of a program. Conversely, the task portion is unique to each separate execution. DX10, therefore, provides an effective mechanism to replicate tasks and provide for multiple executions.

In cases where each concurrent program activation has the same initial state (data), only one program image need be stored on disk. A task may be uniquely replicated from a single image installed in a program file on disk for each activation. Replication of tasks, therefore, conserves disk space and time by avoiding the requirement of installing a copy of the same task with different IDs for each possible concurrent activation of a program.

10.4 SHARING DATA BETWEEN TASKS

Occasionally, it is desirable to share a block of data among two or more tasks. A convenient method for accomplishing this under DX10 is through the use of a shared procedure segment. A procedure may contain data that is shared among several tasks and/or procedural code. A shared procedure may be updated or modified (dirty procedure) by a program with the updated version shared by several tasks.

10.5 OVERLAYS

As programs become large, it may be desirable to partition them in such a way as to allow only a portion of the program to be resident in memory at a given time. The overlay support provided by DX10 provides the mechanism to establish disk-resident program modules. One initial portion of the program, called the root, becomes the memory-resident part. The remainder is divided into disk-resident overlays. Segmentation is performed by the Link Editor.

When an overlay module is required, the program initiates a supervisor call that loads it into memory. Alternatively, the Link Editor can include an automatic overlay manager. Overlay modules may be further segmented into a lower level of root and overlays. Multi-level overlay structures are supported by the Link Editor.

10.6 SAMPLE APPLICATION OF TASKS AND PROCEDURES

Figure 10-1 (task/procedure structure) shows three examples of task and procedures structures of increasing complexity. In the simplest case, a single task includes its own procedural part. In this case the program can be repeated, but its operation would typically be in a single user mode. An application program that reorganizes a specific file of data might be written in this way since two programs cannot reorganize the file at the same time.

In the second sample structure, a single procedure X is shared by tasks A, B, and C. Such a structure might be useful where tasks A, B, and C are each serving different terminals collecting data from three locations in a parts warehouse. Procedure X might be an inventory inquiry program. Note that tasks A, B, and C may be replicated copies of a single disk-resident image.

The third sample structure is an example of tasks operating with two procedures. A program written in COBOL operates in this manner. One procedure (X in the example) that may be shared by all stations is the COBOL run-time support package. This package is required in support of any application program written in COBOL. A second procedure is the procedural part of the COBOL program itself (Y and Z). Note that multiple terminals (tasks) may share the same COBOL procedure (Y in the sample). The task portion contains the data in use by the COBOL program (unique to each program usage) and any overlays required by the COBOL procedure. Overlays are placed in the task segment since different tasks (or program activations) may require different overlay modules at the same time.

10.7 TASK ACTIVATION BY A PROGRAM

Any task may request that some other task be activated. The result of this request is that both tasks become concurrently active. DX10 also supports the identification of a station with which the new task is to be associated. In this manner all the station-local LUNO assignments are available to the new task. Furthermore, the requesting task may specify that it be suspended until termination of the activated task. This provides a convenient mechanism for a master application program serving a station to activate subprocesses either in parallel with or instead of the master program. Note that in the latter case, when the subprocess completes, the master program resumes execution.

An additional method for initiating task execution is activation of the task via a Device Service Routine (DSR). A DSR is memory resident code which processes external interrupts for a specific device. Once the device signals the computer that an event has occurred, a special application task may be executed by the DSR to perform various control functions such as turning on fire extinguishers, sounding an alarm through a smoke detector, or activating a burglar alarm. For emergency situations, these application tasks can be installed to be memory resident with real-time priorities to facilitate their execution.

10.8 PRIORITY SCHEDULING

The DX10 Operating System requires that each task have a defined priority level. There are 132 priority levels:

Highest	0	DX10 internal use
	R1-R127	Real-time priorities
Lowest	1, 2, 3	Interactive and batch mode
	4	Floating priority

Level zero is intended for the most critical system functions and is reserved for DX10 internal use only.

Real-time priorities provide the user with the capability to supercede all but the most important system tasks. For applications that require expeditious access to the CPU, DX10 will delay some routine maintenance of system duties in an effort to schedule real-time tasks.

Priorities one, two, three, and four are designed to satisfy the requirements of most installations. Priority level one gives quick response for programs which interact with the user's terminal, while level two is adequate for programs requiring multiple-disk accesses. Priority four automatically switches priority levels between levels one and two as the program executes.

Priority level three is for batch execution not requiring user interaction. At this level tasks access the CPU only when no higher priority tasks (interactive, real time, or system) are waiting for execution.

DX10 always schedules the highest priority task waiting for execution. Scheduling occurs when one of the following conditions exists:

- An external interrupt bids up a task.
- The executing task suspends.
- The Task Sentry lowers the priority of the executing task (if Task Sentry is enabled).
- The time slice of the executing task expires (if time slicing is enabled).
- A task which was in a time delay becomes active.

When an event external to the computer occurs, an interrupt can be used to signal that a specific user program be scheduled for execution. The operating system then reschedules the tasks for execution because the newly-bid task may be the highest priority task in the system. If the current task suspends (relinquishes control of the CPU), the scheduler finds the next highest priority task and gives control of the CPU to that task.

If the Task Sentry is enabled and has lowered the priority of the currently executing task, the tasks awaiting execution are rescheduled in case a task of higher priority is in the system.

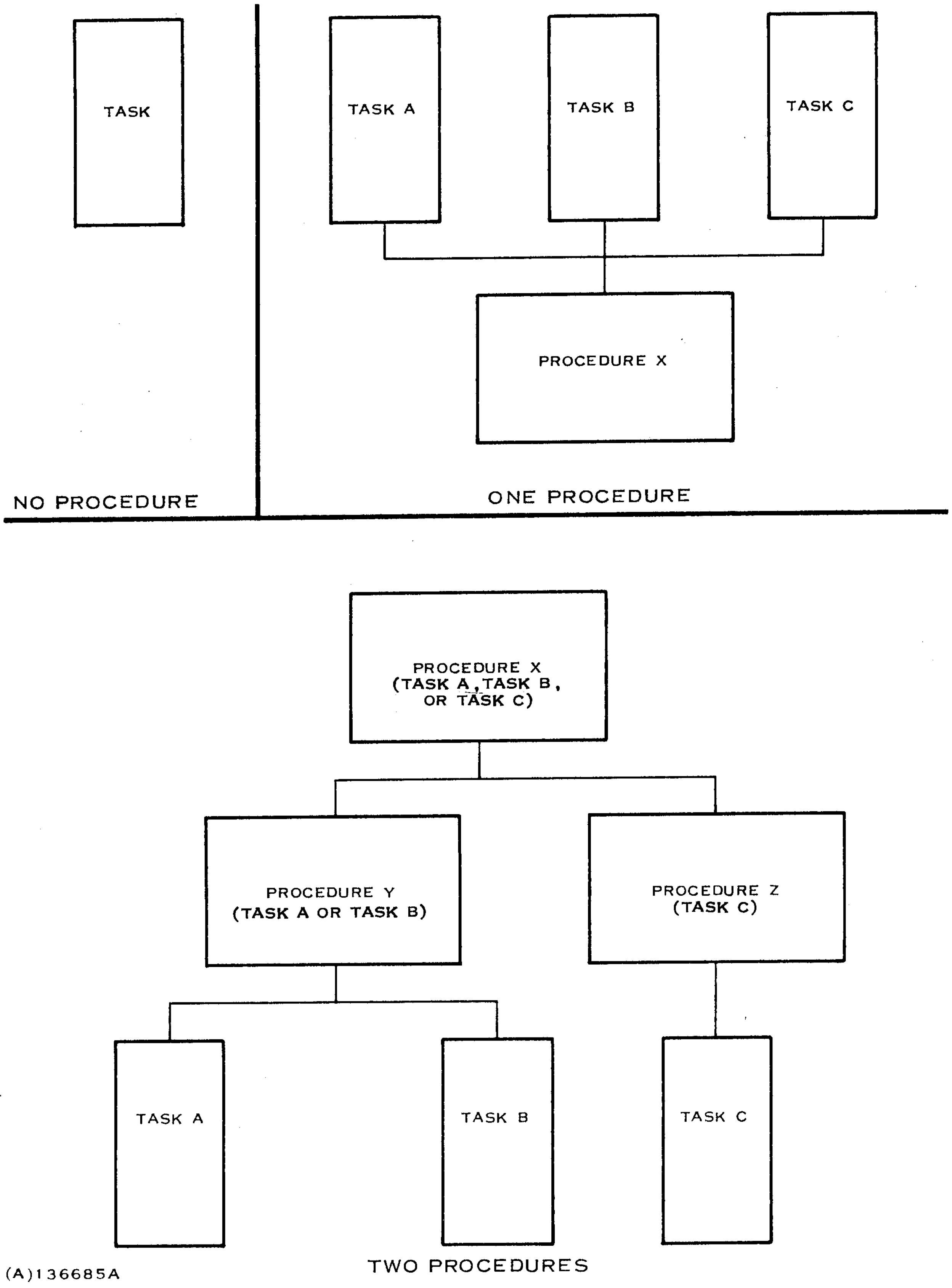


Figure 10-1. Procedure Segment

If time slicing is enabled, the active programs of the highest priority share the CPU in a round-robin fashion. After a task has executed for a fixed interval of time, DX10 reschedules the tasks in the system to allow another task to execute.

When a time delayed task is due to become active, the scheduler unsuspends the task and reschedules the task for execution.

NOTE

Task Sentry and time slicing are SYSGEN options.

10.9 TASKS WITH VARIABLE PRIORITIES

When a task is installed, it may be designated with the floating-priority level of four. In this event, the task is initialized to priority level one when first executed. The task's priority level is lowered to level two after a specified number of time slices (a SYSGEN parameter) when performing computations.

The task priority is set to one when doing terminal input and two for other I/O devices. The floating priority (managed by DX10) provides rapid response to input/output events and deemphasizes the program during periods of heavy CPU operation. For example, application programs which function interactively are normally installed with a floating-priority level designation.

10.10 PROGRAM FILES

All tasks, procedures, and overlays are installed in structures referred to as program files. These files are based on the expandable relative record file type, and contain program images in blocks corresponding to file records. The program file is structured such that an internal directory is maintained within the file itself. This internal directory contains pointers to each image on the file, as well as relevant information about the images. DX10 requires two disk accesses to load a disk-resident task, procedure or overlay. In some cases, a program image may not be stored contiguously on the disk and additional accesses are necessary.

One program file is designated as the system program file. The system program file initially contains only programs that constitute parts of the DX10 operating system. Other program files may be created to hold application programs. In most areas the capabilities of the system and user program files are identical. The differences are:

1. Memory-resident tasks and procedures must be installed on the single system program file.

2. Procedures installed on the single system program file may be used by any task. Procedures installed on a user program file may only be used by tasks installed on the same user program file.

3. Non-replicatable tasks installed on the system program file have the same runtime and installed ID. Those installed on user program files have a runtime ID allocated that differs from the installed ID. If a task is non-replicatable, concurrent executions of that task cannot occur.

10.11 PROGRAM IDENTIFICATION

Program parts stored in program files are retrievable by task, procedure or overlay numbers specified at install time. The program file internal directory also has the capability to transliterate between the program name and its number in the file, thus allowing retrieval by name. A program may be installed with or without the replicatable attribute. The activated image of a replicatable task is assigned a runtime identification number that differs from its installed number. Tasks that are not replicated and installed on the system program file have the same runtime and installed ID. Non-replicatable tasks installed on user program files are allocated a runtime ID that is different than the installed ID.

10.12 TASK SENTRY

Task Sentry is a SYSGEN option which guards against CPU lock out. DX10 always executes the highest priority task in the system; therefore, it is possible to lock out lower priority tasks for seconds at a time.

When a task remains compute bound at any priority for a specified number of 50 millisecond intervals, the Task Sentry will lower the priority of the task by one. This lowering process continues for as long as the task remains compute bound or until the task reaches priority three. When the task finally does suspend, the task sentry will restore the task to its proper priority.

SECTION 11

MEMORY MANAGEMENT

11.1 GENERAL

The DX10 Operating System takes advantage of the 990 Mapping Option to dynamically allocate memory to disk-resident task and procedure segments, as well as file blocking buffers. These allocated blocks may be released from memory and rolled to disk on an as-needed basis. The Roll-in/Roll-out mechanism depicted in figure 11-1 ensures efficient use of main memory and CPU time.

11.2 DYNAMIC ALLOCATION

DX10 places tasks and procedures in memory wherever space is available. The memory mapping feature of the 990 permits dynamic memory allocation. Therefore, a program may become dynamically segmented into a maximum of three disjointed areas of physical memory. Although an application program consisting of a task and two procedures may have the three segments physically separated in memory, the mapping feature causes the program to see a virtual environment where:

1. All three segments appear to be contiguous in memory with no gaps in addressability.

2. The first segment appears to begin at memory address 0.

3. The maximum addressable memory space for any one program is 32K words.

4. Each program segment must begin on a 32-byte boundary. The Link Editor locates procedures and tasks on such boundaries.

11.3 ROLL-IN/ROLL-OUT

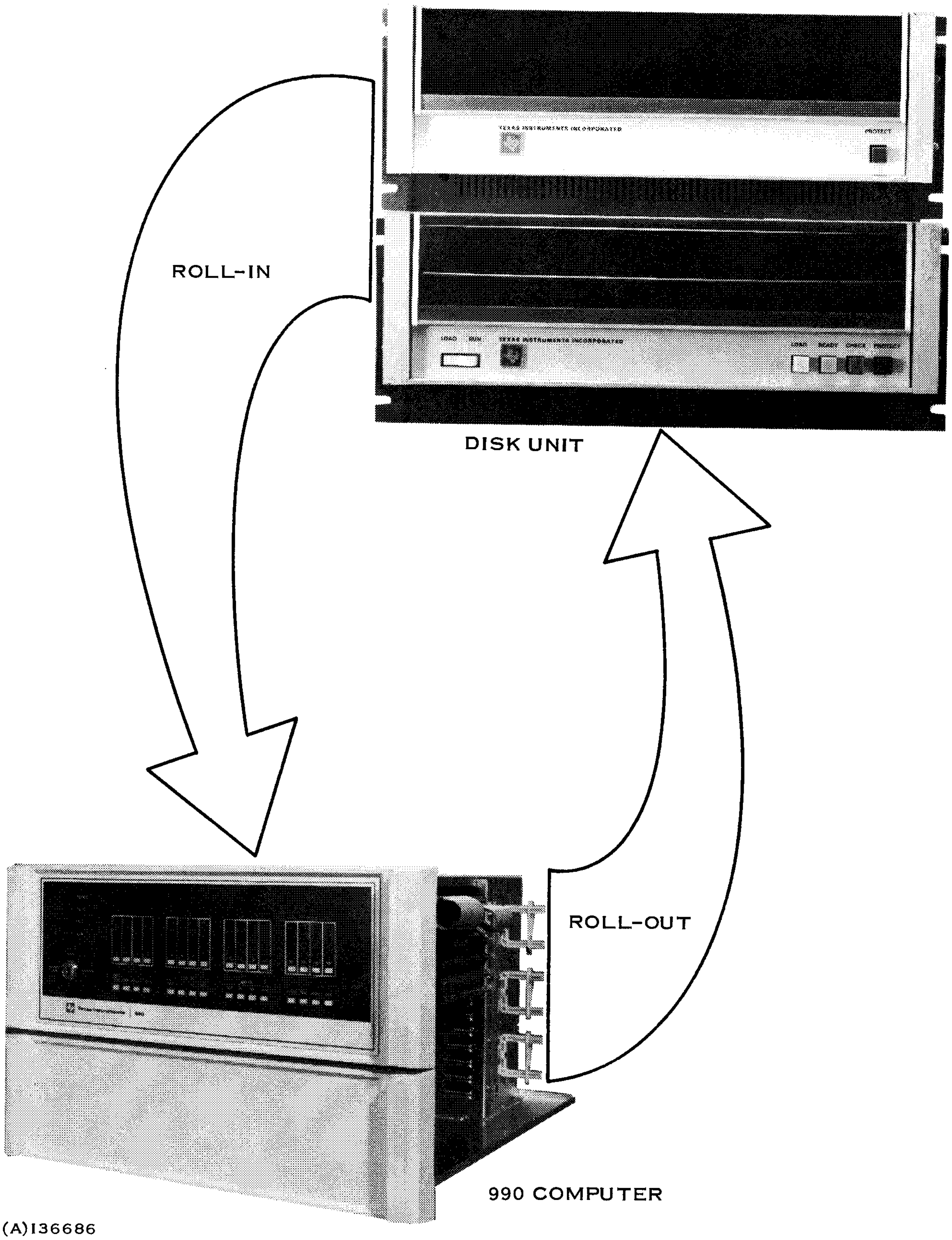
DX10 incorporates an algorithm that permits programs of high priority to preempt memory space from those of lesser or equal priority. Any program may preempt space from a suspended program. Whenever insufficient memory space is available to permit the operating system to execute a program, DX10 seeks out suitable lower priority or suspended task segments and dispatches the programs to disk. This process is called roll-out. Similarly, when the task and priority mix indicates, the rolled out program is rolled back in from the disk and execution resumed. The memory mapping feature permits a program segment to be restored into available physical memory space, possibly different than that which it had occupied at the time it was rolled out. The priority-oriented roll-in/roll-out mechanism permits high priority tasks guaranteed and immediate access to memory that allows the task to respond to users or other external events.

Shared procedures do not become eligible for roll-out unless all the tasks that use them are also rolled out. Similarly, tasks with disk or magnetic tape input or output transfers in progress are not eligible for roll-out until the transfer is complete.

File blocks are allocated in the dynamic memory area along with tasks and procedures. Any blocks of data retained by the operating system from recent disk transfers may be rolled out by DX10 if the memory space is needed. These blocks are written to their appropriate file location on the disk (if they have been updated) to acquire memory space.

11.4 MEMORY-RESIDENT TASKS

Programs are normally disk-resident in program files and loaded by DX10 each time they are activated. DX10 supports the designation of selected programs as memory-resident. Memory-



(A)136686

Figure 11-1. Dynamic Memory Management

resident programs are loaded when the system is loaded from the disk (i.e., at IPL time) and remain in memory even after program termination. All activations of memory-resident programs bypass disk accesses involved in the loading process. Any program designated as memory-resident must be installed in the system program file and cannot be installed in a user-created program file.

SECTION 12

SUPERVISOR CALLS

Programs request service from DX10 by issuing supervisor calls. A supervisor call is initiated by a 990 instruction (level 15 XOP) that transfers execution control to the operating system. Each supervisor call (SVC) includes a block of information containing the detailed parameters associated with the service requested.

Table 12-1 delineates all SVCs supported by DX10 that are usable by an application program. Certain other unlisted SVCs are available only for privileged tasks and are explained in the *Model 990 Computer DX10 Operating System Systems Programming Guide.*

Table 12-1. DX10 General-Purpose Operating System Supervisor Calls

- File and I/O Calls.
- Program Control Calls:
  - Execute a task.
  - Execute a task at a specified future time.
  - Reactivate a suspended task.
  - Load an overlay.
  - End of task.
  - Momentarily suspend a task.
  - Suspend a task for time period.
  - Change task priority level.
  - Determine status of task.
  - Retrieve input parameters.
  - Task identification services.
- Memory Control:
  - Expand the task's memory segment.
  - Contract the task's memory segment.
- Other Calls:
  - ASCII/Binary conversion services.
  - Inter-task communications.
  - Log a message.
  - Fetch time and Julian date.

SECTION 13

USER INTERFACE

13.1 GENERAL

DX10 provides both an effective conversational user interface and a mechanism for batch input through the System Command Interpreter (SCI). A brief sample of conversational operation is shown in figure 13-1. A similar example of batch input is shown in figure 13-2. All commands are interpreted by an activation of SCI, both in interactive and in batch mode.

13.2 INTERACTIVE OPERATION

DX10 features an interactive user interface for control of the system through the implementation of SCI. All entries keyed by an operator are meaningfully prompted. Fields are easily edited by the operator and are verified by the system. The number of prompts, and therefore time, can be conserved by an experienced operator since all arguments for a command can be entered before the system requests them by prompt. When a partial list of arguments is entered any non-defaulted arguments, or those not already supplied by the operator, are then solicited by the system.

The SCI package is supported on DX10 interactive devices including hard copy data and Video Display Terminals. For hard copy data terminals, the user is prompted for parameter entry one line at a time. Prompting may be avoided by entering all parameter values with the command. VDT mode operation allows all parameters to be prompted at once on the screen. A hierarchy of command menus is made available to all types of system terminals. The command menus provide each terminal on-line command prompts by logical grouping.

At each terminal, the user may have one foreground and one background program concurrently active. Foreground programs interact with users via the terminal. Background programs never interact directly with users at the terminal.

```
 SYSTEM COMMAND INTERPRETER - PLEASE LOG IN
   USER ID:    KOC018
   PASSCODE:
[] LTS
LIST TERMINAL STATUS
 TERMINAL NAME:
 OUTPUT ACCESS NAME:
 TERMINAL  USER   LI REQ'D  MODE  DEFAULT
   ST01    KOC018    YES     TTY    VDT
   ST02    DUM034    YES     VDT    VDT
   ST03              YES     VDT    VDT
   ST04    DJE006    YES     VDT    VDT
   ST05    KWD011    YES     VDT    VDT
   ST06    GPS021    YES     VDT    VDT
   ST07              YES     VDT    VDT
   ST08    DUM032    YES     VDT    VDT
   ST09    DUM030    YES     VDT    VDT
   ST10              YES     VDT    VDT
```

(A)136687

Figure 13-1. User/System Interaction in SCI

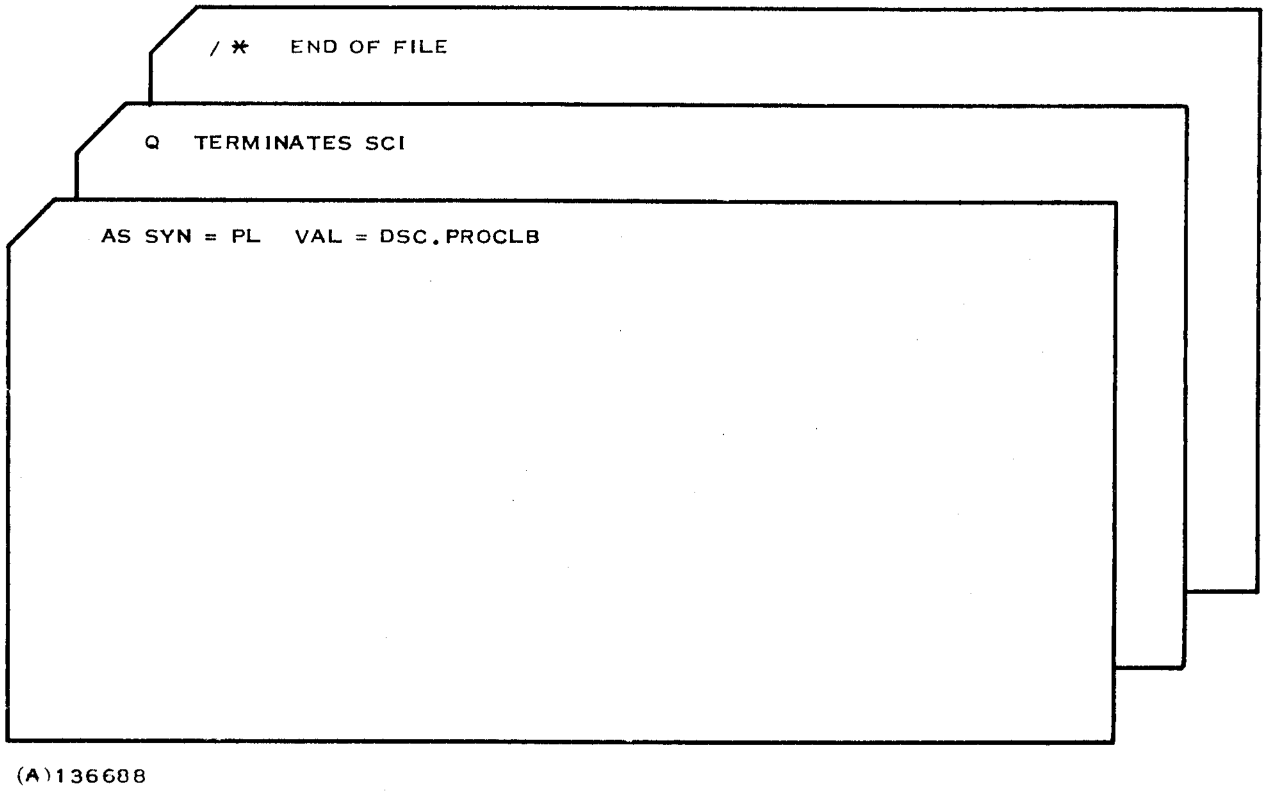


Figure 13-2. Card Input of Batch Commands

13.3 BATCH OPERATION

The single background program allowed at a terminal may be a copy of SCI. In this case, the System Command Interpreter is interpreting SCI commands in the background, and this is referred to as batch processing. Batch input is from any sequentially-oriented file device, but not the terminal itself. A user may initiate batch processing, query its status, and receive information concerning its normal or abnormal completion.

13.4 SYNONYMS

Long text strings commonly entered by an operator may be abbreviated through the use of synonyms. DX10 retains a list of synonyms for each user. When they are entered by the user, DX10 replaces the synonym with the actual text string.

13.5 FLEXIBLE COMMAND PROCEDURES

All potential operator commands are processed using a set of command specifications maintained on the system disk. The command specifications are coded in a special-purpose language. As commands are entered by the user, SCI interprets the specification language and performs prompting, data entry, and verification functions appropriate to the command. Statements within the language structure inform the system as to what data is to be collected and to what processing program that data is to be passed. This scheme facilitates the addition and modification of application specific operator commands.

13.6 COMMANDS AVAILABLE

DX10 offers a comprehensive list of SCI commands that perform various utility operations. Many other commands are supplied for the use of programmers as they develop applications under DX10. The *Model 990 Computer DX10 Operating System Production Operation* and the *Model 990 Computer DX10 Operating System Developmental Operation Manual* provide operating details for these commands. An abbreviated list of the functional areas served by one or more commands is given in table 13-1.

Table 13-1. Areas Served by DX10 SCI Commands

Log on and log off
Time and date setup and inquiry
Disk volume initialization, install, and unload
Disk directory backup, restore, and copy
Creating and deleting directories and files
Synonym support
File alias name
Changing filenames and protection
Viewing and listing directories and files
Copying directories and files
Logical unit assignment, positioning, and release
System I/O status display
System task status display
Program activation and control
Batch command input, activation, and status
Station control (user ID, terminal status, etc.)
Installing and deleting programs
Activation of the system log
Program debugging including such items as:

- Breakpoints
- Memory/Disk dump or display
- Decimal/Hexadecimal arithmetic aid
- Interactively controlled program trace

Text Edit Control
Assembler, COBOL, FORTRAN, RPGII, Pascal, and BASIC activation
Link Edit activation
Sort/Merge activation
Data base management activation
File transfers for bubble memory terminals

SECTION 14

APPLICATION PROGRAMMING ENVIRONMENT

14.1 PROGRAM DEVELOPMENT TOOLS

In addition to a comprehensive set of utilities that operate in conjunction with DX10, Texas Instruments provides four major program development tools including:

- Interactive Text Editor.
- Macro Assembler.
- Link Editor.
- Interactive Debugger.

14.1.1 INTERACTIVE TEXT EDITOR. The DX10 system provides an interactive text editor that operates from either a hard copy interactive device or from a video display terminal. Edit operations are initiated via edit keys directly, or by command. When operations are intiated by commands, parameters are necessary and are prompted by DX10. Edit operations allow modification, insertion and deletion of entire records or of character strings within records. Data may be copied from files other than the primary input to become a part of the new output. Changes may be made at any position within the edited file and positioning may be either forward or backward. The user may position to a given character string or line number within the file. The output file is only created upon normal termination of the text edit session. The text editor uses the file rename operation when the user wants to replace the primary input file with the edited output. Figure 14-1 shows a text edit session.

14.1.2 MACROASSEMBLER. The DX10 system assembler is the most powerful of the 990 family assemblers. In addition to accepting the standard 990 assembly language instructions, the assembler is extended to include a macrofacility, support for FORTRAN common segments, and conditional assembly. The macrofacility provides character string manipulation, access to binary values in the symbol table, and supports macrodefinition libraries. The relocatable object code produced by the assembler may be partitioned in segments. Common blocks, program segments and data segments are assigned separate location counters. The system Link Editor collects segments of the same type into contiguous memory areas. A sequence of assembly language statements may be conditionally processed depending on the value of an assembler arithmetic or logical expression. Directives are provided to specify the amount of information in the assembly listing. Figure 14-2 shows an example assembly listing.

14.1.3 LINK EDITOR. The DX10 system Link Editor accepts relocatable object code generated by the assembler, FORTRAN compiler or COBOL compiler, and combines the individual modules into a linked load module. References between the modules are resolved to their correct values. Common blocks, program segments and data segments are collected and each segment type assigned to its own contiguous area of memory.

The Link Editor accepts control statements that may specify the use of shared procedures and the use of overlay structures. Available options include the generation of a load map, search a set of object libraries to automatically resolve unresolved values, and production of a partial link that leaves external references to be resolved at a later time. An automatic overlay load option is available where applications require overlays. Output of the Link Editor may be an installable object file, or may be written directly as memory image into a program file or DX10 image file. In the latter case, the install phase of a program development cycle is avoided. Figure 14-3 shows a partial listing produced by the DX10 Link Editor.

```
[] XE
INITIATE TEXT EDITOR
 FILE ACCESS NAME: .TISOURCE.TSTSDS
    1         TITL 'TSTSDS - SDS VERIFICATION PROGRAM'

[] RS SI=DSC,C=DS01
 NUMBER OF OCCURRENCES:    1
 START COLUMN:    1
 END COLUMN:    80
   19*                            CREATE FILE "DS01.TSTMSG"

[] RS SI=AND WRITE,C=WITH
   42*                            CLOSE FILE WITH E-O-F

[] RS SI=CLOSE FILE WITH E-O-F ,C=CLOSE FILE WITH E-O-F S
                                                           AND TERMINATE
   42*                            CLOSE FILE WITH E-O-F AND TERMINATE

   43         SVC   @PRB

   44         JMP   $
       SVC   @EOP
[] SL L=44
   44         SVC   @EOP

[] FS
FIND STRING
 OCCURRENCE NUMBER: 1
 START COLUMN: 1
 END COLUMN: 80
 STRING:    PRBRCN
   95 PRBRCN DATA 0              10    RECORD NUMBER
```

Figure 14-1. Text Editing in TTY Mode

```
TSTSDS
                                                              PAGE 0001

                  TITL 'TSTSDS - SDS VERIFICATION PROGRAM'
                  IDT  'TSTSDS'
           *
           ************************************************************
           *
           *              THIS IS THE SOURCE TEXT FOR THE SDS
           *                    VERIFICATION PROGRAM
           *
           *
           *
           ************************************************************
           *
                  DXOP SVC,15
                  REF  WSPACE
     0001  R1     EQU  1
     0002  R2     EQU  2
           *
0000 02E0 TSTSDS LWPI WSPACE
0002 0000
           *                        CREATE FILE "DS01.TSTMSG"
0004 2FE0         SVC  @FUSCB
0006 00E8'
           *
           *                        ASSIGN LUNO TO FILE
0008 D820         MOVB @AL,@FUSOPC
000A 0044'
000C 00EA'
000E 2FE0         SVC  @FUSCB
0010 00E8'
           *
           *                        OPEN THE FILE
0012 2FE0         SVC  @PRB
0014 00FE'
           *
           *                        WRITE ASCII
0016 0201         LI   R1,ADRTBL
0018 003C'
001A C0B1 LOOP    MOV  *R1+,R2           R2 = @ OF NEXT LINE TO WRITE
001C 1308         JEQ  DONE
           *
001E C802         MOV  R2,@PRBBAD        STORE BUFFER ADDRESS
0020 0104'
0022 D820         MOVB @WA,@PRBOPC       STORE 'WRITE ASCII' OP CODE
0024 0045'
0026 0100'
0028 2FE0         SVC  @PRB              WRITE OUT THE LINE
002A 00FE'
002C 10F6         JMP  LOOP              LOOP
           *
           *                        DONE
002E      DONE
002E D820         MOVB @CLWEOF,@PRBOPC
0030 0046'
0032 0100'
           *                        CLOSE FILE WITH E-O-F AND TERMINATE
0034 2FE0         SVC  @PRB
0036 00FE'
0038 2FE0         SVC  @EOP
003A 010E'
           *
```

(A)136691

Figure 14-2. Example Assembly Program Listing

```
TI 990/10                         07/19/77  15:04:58                          PAGE   3


PHASE 0, TSTSDS   ORIGIN = 0000  LENGTH = 0130

MODULE      NO    ORIGIN     LENGTH      TYPE        DATE         TIME        CREATOR

TSTSDS      1      0000       010F      INCLUDE    06/19/77     14:56:56     SDSMAC
WSPACE      2      0110       0020      INCLUDE    06/19/77     14:57:13     SDSMAC


                                 D E F I N I T I O N S

 NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO

 WSPACE    0110   2



****   LINKING COMPLETED
```

(A)136692

Figure 14-3. Partial Listing Produced by the Link Editor

14.1.4 INTERACTIVE DEBUGGER. The debugger is an interactive symbolic debugging program (figure 14-4) for assembly language tasks running under DX10. It operates from either an interactive video display terminal (VDT) or an interactive hard copy terminal such as an ASR 733 Terminal. The debugger allows display and modification of CPU registers, workspace registers and memory, and provides for controlled execution of a task. In the run mode, a task may be halted and started. New breakpoints may be set to halt the task when that breakpoint is encountered, thus allowing program execution in *near* real time. In the simulation mode, a task's execution is analyzed between each instruction. Trap conditions that interrogate the program counter or memory content may be specified. Breakpoints designed to halt, or continue, task execution as required may be conditional on a given number of accesses within a specified range of Program Counter (PC) values, memory locations, or CRU addresses. Breakpoints may be incurred on given Status Register (SR) values or supervisor calls. A task is debugged in its own address space and, therefore, may be a full 32K words in length having any desired task structure.

```
 [] XHT
 EXECUTE AND HALT TASK
  PROGRAM FILE OR LUNO:    .S$PROGA
  TASK NAME OR ID:    TSTSDS
  PARM1: 0
  PARM2: 0
  STATION ID: ME
 RUNTIME TASK ID = >2C
 [] XD
 INITIATE DEBUG MODE
  RUN ID:    02C
  SYMBOL TABLE OBJECT FILE:    .TISOURCE.TSTSDSO
 RUN ID = 2C    WP = 02E0    PC = 0110   <PC> = 0000    ST = 018F    STATE = 06
 [] MIR R=02C
  WP: >02E0
  PC: >0110   0
  ST: >018F
 [] AB
 ASSIGN BREAKPOINTS
  RUN ID: 02C
  ADDRESS(ES):    TSTSDS.TSTSDS
 [] AT
 ACTIVATE TASK
  RUN ID: 02C
 TASK STATE: 06
 [] PB
    LM
 LIST MEMORY
  RUN ID: 02C
  STARTING ADDRESS:    0
  NUMBER OF BYTES:    010
  LISTING ACCESS NAME:
 0000     2FCF   0110   2FE0   00E8   D820   0044   00EA   2FE0    /. .. /. .. .   .D .. /.
 [] PB
 PROCEED FROM BREAKPOINT
  RUN ID: 02C
  DESTINATION ADDRESS(ES):    0E
```

(A)136693

Figure 14-4. Interactive Debugging

14.2 DX10 COMMUNICATIONS, LANGUAGES, AND UTILITIES

Figure 14-5 illustrates the interrelationships of the system software tools available from Texas Instruments. Each software package provides specialized features for use in the different functional areas of data processing. Users may combine the various languages and utilities into a total information system best adapted to their individual requirements.

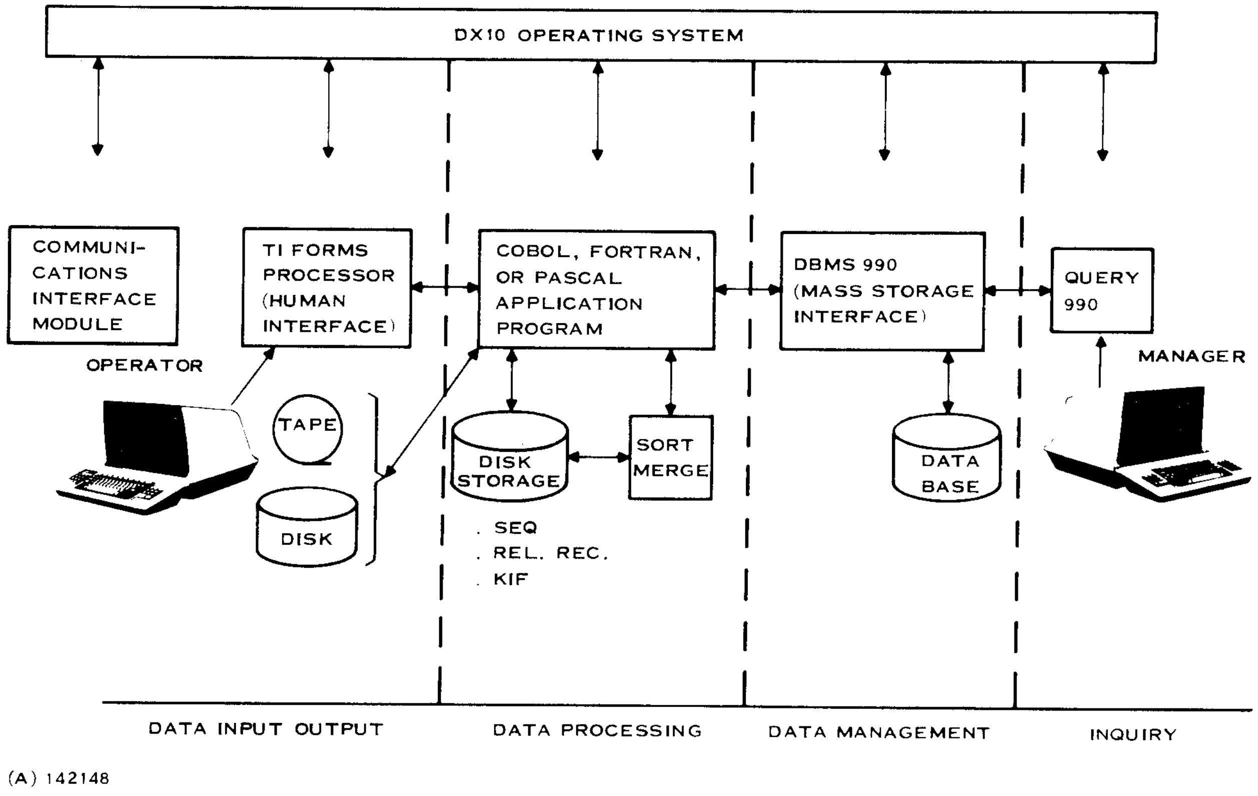


Figure 14-5. DS990 Commercial Application Environment

14.2.1 DX10 COMMUNICATIONS. Several communications methods are available using DX10 and other 990 software packages and communications modules. Depending on the application, the user may generate a custom DX10 system to meet his needs. The available software ranges from 3780/2780 emulator support to basic device support which must be complemented with user-supplied software.

14.2.1.1 Communications Equipment. The 990 communications modules available for the DS990 system are shown in figure 14-6. They include the 990 communications interface module, a choice of an asynchronous or synchronous modem, and an accessory auto-call unit. The communications interface module can be used with Bell data sets, which include modems and data-access arrangements.

The 990 communications interface module provides an RS-232-C interface with full modem control signals for synchronous and asynchronous modems. Baud rates of 75, 110, 150, 200, 300, 1200, 2400, 4800, and 9600 meet almost any communication requirement. Character size is selected from 5 to 9 bits with programmable parity (odd, even, or none). Other features include a line-break detection/generation, 250-millisecond timer, programmable SYN, DLE stripping, false-start-bit detection, stop-bit selection (1, 1-1/2, or 2 stop bits), and programmable self-test.

The 990 communications interface module requires a half slot in the 990 chassis and interfaces with the CRU bus. The *Model 990 Computer Communications System Installation and Operation Manual* also covers the 990 asynchronous modem and the 990 synchronous modem.

The 990 asynchronous-modem kit provides a Bell-202-equivalent (1200-baud) modem with auto-answer, capable of a full-duplex operation over a four-wire private line or a half-duplex operation over a DDD network. The modem provides a loop-back for the test. The module requires a half slot in the 990 chassis and interfaces with a 990 communications interface module (not included) via the top-edge connector and cable. The modem must be adjacent to the 990 communications interface module in the chassis.

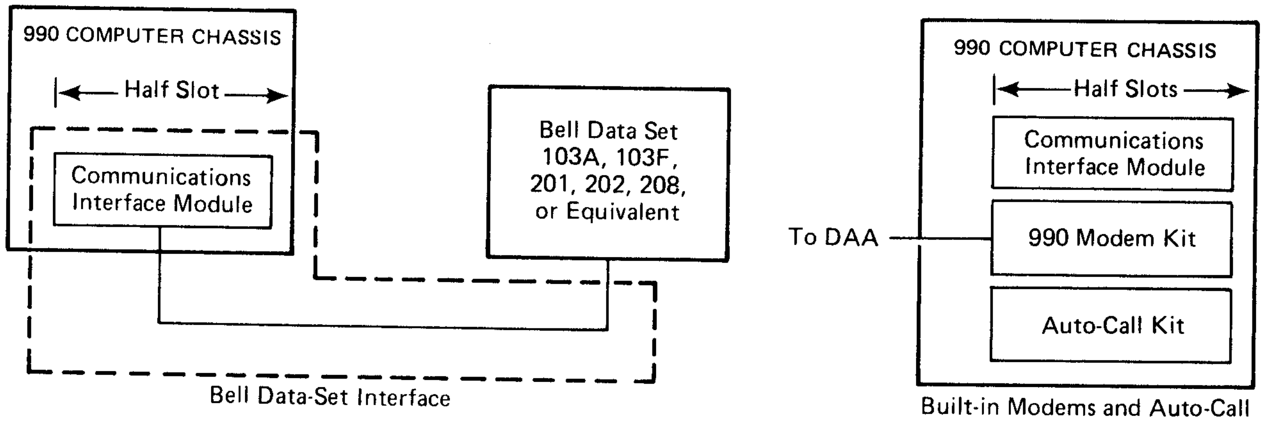

990 COMPUTER CHASSIS
Half Slot
Communications Interface Module
Bell Data Set 103A, 103F, 201, 202, 208, or Equivalent
Bell Data-Set Interface
990 COMPUTER CHASSIS
Half Slots
Communications Interface Module
To DAA
990 Modem Kit
Auto-Call Kit
Built-in Modems and Auto-Call

Figure 14-6. 990 Communications Modules

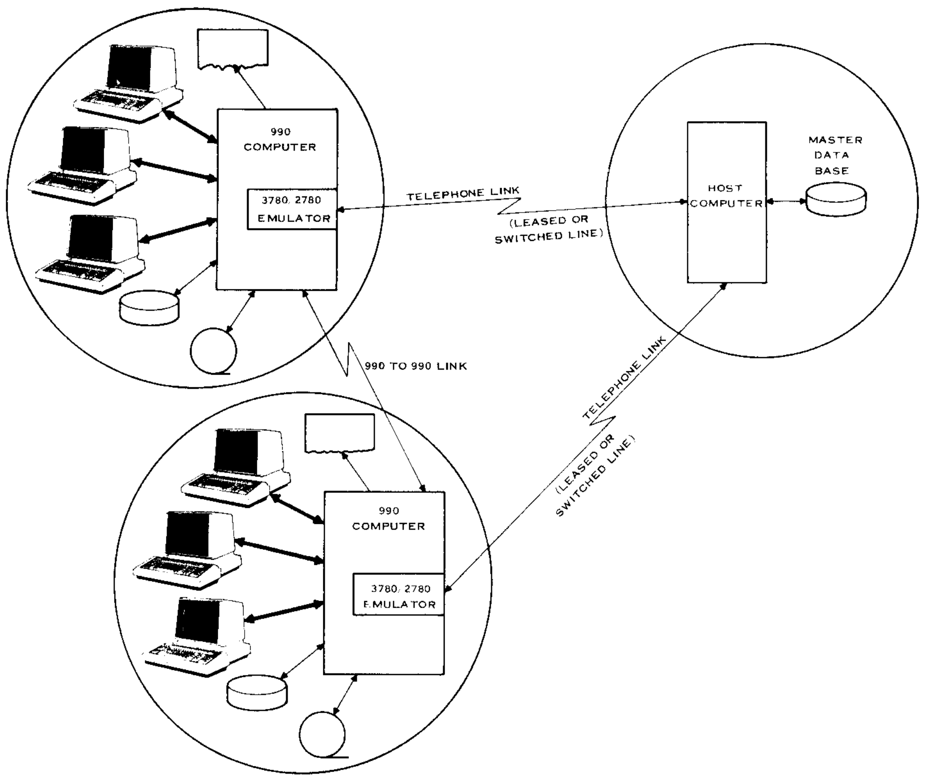
The 990 synchronous-modem kit is similar to the 990 asynchronous-modem kit, except this kit provides a Bell-201C-equivalent (2400-baud) modem for synchronous communication. The modem provides an internal clock and loop-back for a self-test.

The auto-call kit provides for CPU calling via dial pulse or tone signals to telephone-switching circuitry. The CPU test of an auto-call module is provided by access to internal states. The module plugs into a half slot in the 990 chassis and interfaces with a synchronous or asynchronous modem by a top-edge cable. The auto-call module must be adjacent to a modem in the chassis.

14.2.1.2 3780/2780 Emulators Communications Software. The 3780/2780 emulators communications software packages provide the 990 family of computers with a means of remote-job-entry (RJE) communications with an IBM 360/370 host computer or another 3780/2780 emulator-equipped 990 computer.

Communications consist of exchanging data files between master and slave stations over leased point-to-point or switched telephone lines (figure 14-7).

Using the 3780/2780 emulators, 990 computer systems can serve as satellite and/or central stations in distributed-processing networks or can be used to handle remote-job or batch-data entry for processing by a host. Remote stations can be dialed manually or automatically with an optional auto-call unit and internal modem. They can also be operated in an unattended mode as a called station in a distributed network.



2278077

Figure 14-7. Typical Application of 3780/2780 Emulator in Distributed Processing Environment

Texas Instruments 3780/2780 emulators communications software emulate the operation of the IBM 3780 Data Communications Terminal and the IBM 2780 Data Transmission Terminal, respectively. However, unlike the IBM devices, the source and destination of the transferred files are not restricted to the card reader/punch and line printer. Any file, input device, or output device available to the user's system can be used for input or output.

The DX10 3780/2780 emulators are available for operation under the DX10 system software on DS990 systems.

14.2.1.3 DX10 Remote Terminal Subsystem. The DX10 Remote Terminal Subsystem (RTS) software is a communications and remote-device-control software package that can be included in the Texas Instruments DX10 operating system. RTS enables a DX10-based DS990 host computer system (Model 4 or larger) to communicate and interact with 915 and Remote Terminal Controller (RTC) remote terminals as if they were local to the host installation.

An 810 printer may be connected to each 915 remote terminal as an optional device. Usually, the 915 terminals are located at sites physically remote from the DX10-based computer system. Together, a 915 terminal and a printer are referred to as a remote station. The DX10-based computer system is referred to as the host computer.

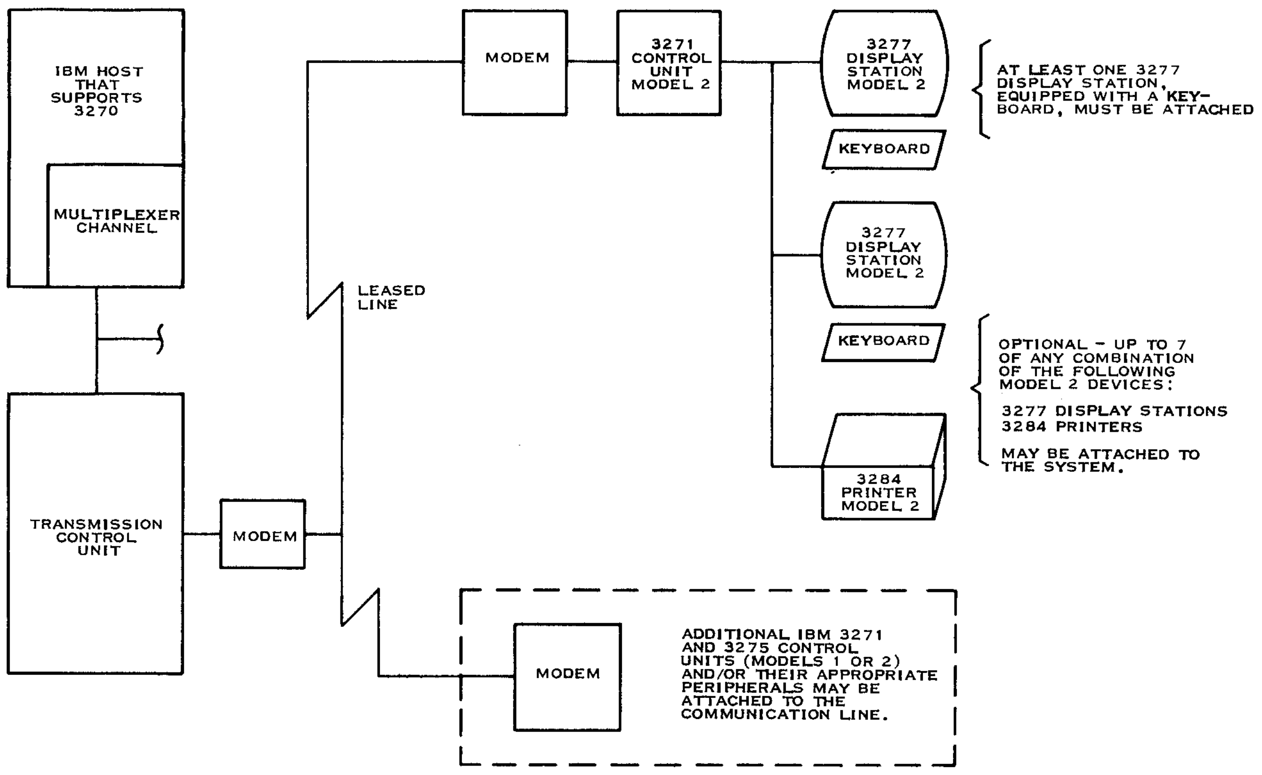
Several remote stations can be connected to the same communications line, and several lines can be connected to the host computer. This permits distributed terminal networks to be built and connected to the host computer. The remote stations are used to enter data into the host computer, to inquire about data being processed at the host computer, and to receive output from the host computer. From a user's standpoint, once the line connection is made between the host computer and a remote station, that station interfaces with the host computer's DX10 operating system as if it were locally connected to the host computer.

DX10 RTS is described in detail in the *DX10 Remote Terminal Subsystem (RTS) for Model 915 Remote Terminal Installation and Operation Manual.*

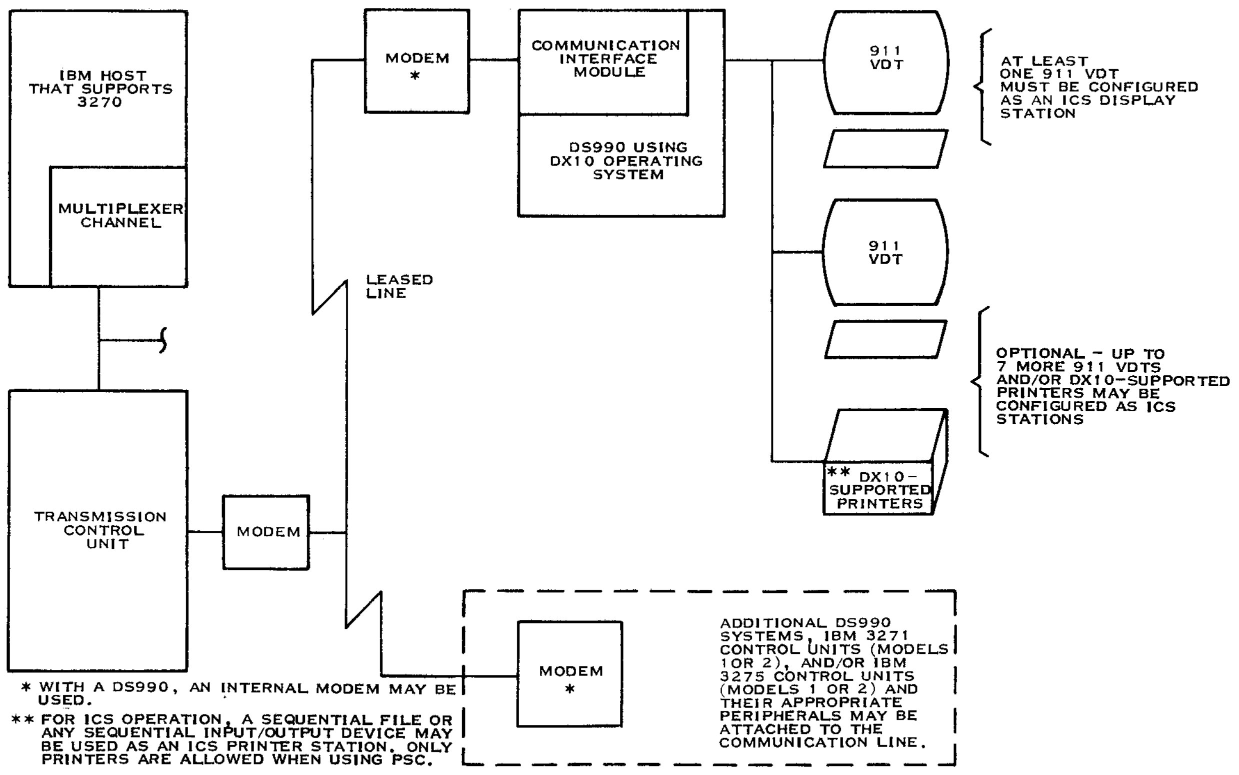
14.2.1.4 3270 Emulator Communications Software. The DX10 3270 interactive communications software (ICS) provides a DS990 system (Models 4 and above) operating with the DX10 operating system (3.2 or later) the capability of emulating the operation of a subset of the IBM 3270 Information Display System. This emulation allows a 911 VDT or DX10-supported printer to receive data from an IBM host application program. The 911 VDT, as an ICS display station, reads, modifies, or adds to received data or generates new data and transmits all modified or generated information to the host application program. The DX10-supported printer, as an ICS printer station, prints data transmitted by the host application program and prints data displayed at the ICS display station. In addition to the emulation, ICS provides a special capability, programmed station control (PSC), which allows a user-written program to interact with an IBM host as an ICS station or to operate an ICS station as an ICS station operator.

The DX10 system attaches remotely to the IBM host using leased lines and uses binary synchronous communication (BSC) procedures to control the data link. Figure 14-8 shows the IBM 3270 Information Display System configuration being emulated by ICS. ICS can communicate with any IBM host system that supports the IBM 3271 Model 2 Control Unit, IBM 3277 Model 2 Display Station, and IBM 3284 Model 2 Printer.

14.2.1.5 Bubble Memory Terminal Support. The Bubble Memory Terminal Support (BMTS-990) software is a set of utilities providing improved communications between a DX10-based system and Texas Instruments hard-copy terminal products: Models 743, 745, 781, 783, 785, 787, 820, 825, 763, and 765. The BMTS software includes the ability to call remote terminals from the DX10 system, monitor incoming calls from remote terminals, disconnect a communications link, and list and


IBM HOST THAT SUPPORTS 3270
MULTIPLEXER CHANNEL
TRANSMISSION CONTROL UNIT
MODEM
LEASED LINE
MODEM
3271 CONTROL UNIT MODEL 2
3277 DISPLAY STATION MODEL 2
KEYBOARD
AT LEAST ONE 3277 DISPLAY STATION, EQUIPPED WITH A KEY-BOARD, MUST BE ATTACHED
3277 DISPLAY STATION MODEL 2
KEYBOARD
3284 PRINTER MODEL 2
OPTIONAL - UP TO 7 OF ANY COMBINATION OF THE FOLLOWING MODEL 2 DEVICES:
3277 DISPLAY STATIONS
3284 PRINTERS
MAY BE ATTACHED TO THE SYSTEM.
MODEM
ADDITIONAL IBM 3271 AND 3275 CONTROL UNITS (MODELS 1 OR 2) AND/OR THEIR APPROPRIATE PERIPHERALS MAY BE ATTACHED TO THE COMMUNICATION LINE.

IBM CONFIGURATION EMULATED WITH ICS


IBM HOST THAT SUPPORTS 3270
MULTIPLEXER CHANNEL
TRANSMISSION CONTROL UNIT
MODEM
LEASED LINE
MODEM *
COMMUNICATION INTERFACE MODULE
DS990 USING DX10 OPERATING SYSTEM
911 VDT
AT LEAST ONE 911 VDT MUST BE CONFIGURED AS AN ICS DISPLAY STATION
911 VDT
** DX10-SUPPORTED PRINTERS
OPTIONAL - UP TO 7 MORE 911 VDTS AND/OR DX10-SUPPORTED PRINTERS MAY BE CONFIGURED AS ICS STATIONS
MODEM *
ADDITIONAL DS990 SYSTEMS, IBM 3271 CONTROL UNITS (MODELS 1 OR 2), AND/OR IBM 3275 CONTROL UNITS (MODELS 1 OR 2) AND THEIR APPROPRIATE PERIPHERALS MAY BE ATTACHED TO THE COMMUNICATION LINE.

* WITH A DS990, AN INTERNAL MODEM MAY BE USED.

** FOR ICS OPERATION, A SEQUENTIAL FILE OR ANY SEQUENTIAL INPUT/OUTPUT DEVICE MAY BE USED AS AN ICS PRINTER STATION. ONLY PRINTERS ARE ALLOWED WHEN USING PSC.

ICS CONFIGURATION

(B) 141563

Figure 14-8. IBM 3270 Information Display System Configuration and ICS Configuration

modify communications port characteristics. Because of the additional capabilities, the bubble memory terminals have additional utilities associated with them. These utilities allow remote terminal control, remote memory terminal file management, and file transfer between the DX10 system and the remote memory terminal. The Bubble Memory Terminal Support software is normally used in conjunction with the standard utility tasks that provide such services as auto-call unit (ACU) dialing and disconnect and port characteristic modification.

14.2.2 HIGH-LEVEL PROGRAMMING LANGUAGES. Texas Instrument provides a choice of high-level programming languages to users of the disk system software: FORTRAN IV for mathematical and scientific applications, COBOL for business environments, and Pascal is supported for a variety of applications including system software development and scientific applications.

Texas Instruments also supports a multiuser BASIC language for interactive scientific programming and business application programming, and the RPGII language for business applications requiring file maintenance or report generation.

14.2.2.1 FORTRAN IV. FORTRAN is an easy-to-use, high-level computer language that allows complex problems to be stated in common mathematical expressions for input to the computer. Before programs written in this high-level language are usable by the computer, they must first be processed by a FORTRAN compiler program. An extensive link-edit is required to join the compiled program with runtime support and called subroutines.

The FORTRAN compiler for the Texas Instruments Model 990 computer translates FORTRAN language input code into Model 990 computer machine code. The FORTRAN compiler conforms to the American National Standards Institute (ANSI) standard FORTRAN, or FORTRAN IV. The compiler also incorporates the extensions recommended by the Instrument Society of America in their document ISA-S-61.1, 1975 and in their document ISA-61.2, 1976.

Texas Instruments has incorporated several useful attributes into the FORTRAN compiler that provide for more effective coding and program development. These added features include:

- Direct disk I/O.
- Overlapped I/O.
- Free format source input.
- Internal data manipulation statements.
- Literal character strings represented in quotes.
- Variable names of any length.
- Double-word (32-bit) integer data type.
- Implicit variable typing.
- General integer expressions in subscripts.
- Data statement array names.
- Mixed mode expressions.
- Hollerith and hexadecimal constants and assignments.

- Scaled binary data types.
- Copy directives
- Accept and display directives for interfacing with a VDT

Optionally, the compiler can generate a cross reference listing for each variable in the program, provide a debug module that references specific line numbers in the source program input during execution, provide a list of generated object code in readable form, provide conditional compilation, allow free format, and generate assembly language source code.

A FORTRAN function library is provided that includes all intrinsic functions and the basic external functions defined in the ANSI standard. This library contains all runtime support to interface with the DX10 operating system. In addition, several generally useful routines such as a random number generator are provided. Additional libraries are provided to allow execution of FORTRAN programs under the TX990 operating system or in a standalone mode. These are particularly effective for DX10 users who are cross supporting smaller 990 configurations.

14.2.2.2 COBOL. COBOL is a high-level computer language that allows problems to be stated in words and syntax similar to the English language. COBOL consists of a set of English words and symbols that the programmer may use to define the problem and create a program to solve that problem. Because of its similarity to English, programs written in COBOL are nearly self documenting, and the time required to train a new programmer in the language is greatly reduced.

The COBOL compiler conforms to the ANSI COBOL subset as defined in ANSI document X3.23-1974, and incorporates extensions to this subset to provide added capabilities. The compiler package employs the following ANSI 74 standard COBOL modules at the level indicated.

Features

Level 1	**Level 1+***
Inter-Program Communications	Table Handling
Library	Nucleus
Segmentation	Relative I/O
	Sequential I/O
	Indexed I/O

The debug and accept/display modules are nonstandard and designed for ease of use on Video Display Terminals.

COBOL programs may be executed directly with SCI Execute COBOL commands, or they may be link-edited and installed in DX10 program files.

14.2.2.3 Pascal. Texas Instruments version of Pascal for the 990 computer is a general-purpose language well suited for a variety of applications. Originally designed as a language for teaching a systematic concept of programming, Pascal is straightforward to learn and to use. Its readability makes the language especially useful when programs must be maintained by users other than the original author.

A popular application of Pascal is the development of system software. The Pascal compiler is itself written in Pascal as are a number of other 990-system software modules. Pascal is also ideal for scientific or engineering applications that traditionally are written in FORTRAN or ALGOL. Its general-purpose structure is even useful for many business problems, although Pascal is seldom found in this application.

The minimum system for Pascal is the 990 processor with 128K-byte memory, 10M-byte disk drive, 911 VDT terminal, single-bay desk enclosure, and DX10 multiuser disk-system software license. Expansion capabilities can provide large amounts of memory, multiple 50M-byte disk drives, and a variety of additional standard 990 peripherals.

The Pascal system consists of five major components:

- Nester utility
- Configuration processor
- Pascal compiler
- Pascal run-time library
- Reverse assembler.

The nester utility generates source code indented on a standard format to improve readability. The configuration processor supports the separate compilation of nested program modules. The Pascal compiler with optimizing features produces linkable object modules. The Pascal run-time library provides operating-system interface. The reverse assembler optionally produces assembly-language source files or listings.

The object license for the Pascal software provides a complete package including all of the software components with the exception of the link editor, which is included with the license for the DX10 operating system. The DX10 system provides a powerful multiuser environment for Pascal.

Some of the more significant features of Pascal include:

- Block-structured format that directly supports structured programming concepts
- Stack allocation of variables for each routine
- Recursive routines
- User-defined data structures that are adaptable to data used in application
- User-defined data types and type checking
- Excellent bit-manipulation capability.

14.2.2.4 BASIC. BASIC is an easily understood programming language that is applicable to scientific and business problem solving. BASIC is an interactive language designed for several different users, each working from a separate terminal. The computer provides feedback to each command entered on a terminal and points out any programming errors by displaying diagnostic messages during execution time.

The BASIC language provided is a greatly extended implementation of ANSI Minimal BASIC. The extensions include support of multiple file organizations, closed subroutines, character strings and VDT screen handling. The system includes the following components:

- A compatible language, TI 990 BASIC, which can execute on all members of the DS990 product line.

- An interactive reentrant editor/interpreter for creating and executing TI 990 BASIC programs.

DX990 BASIC is directed to the user who is most concerned with program development time. The powerful editor and fast response of the system almost eliminate turnaround time. The 13-digit precision provides the accuracy needed for scientific calculations, and the decimal representation of noninterger variables eliminates the roundoff problems of most other BASICs when used for commercial applications. Language compatibility with smaller members of the DS990 family permits development of BASIC programs on large systems, to be subsequently used on smaller systems. DX990 BASIC provides key indexed files and temporary files in addition to the relative and sequential files provided on the smaller TI 990 BASIC systems.

14.2.2.5 RPGII. RPGII (Report Program Generator, version II) is an easy-to-use, high-level language for business data processing. Based upon a predetermined sequence that reads a record, processes the data, and outputs the results, RPGII is especially suited for applications requiring file maintenance or report generation. A series of six basic specification formats are used to input the specific actions to be taken within the RPGII sequence of execution.

Texas Instruments version of the RPGII language is closely compatible with the widely used IBM System/3 RPGII. Extensions of many of the System/3 features have been included in RPGII to provide more flexible programming. A utility program is provided with RPGII to copy System/3 or System/32 source programs or files from diskette to DS990 disk files.

The RPGII package also includes an RPGII-oriented VDT text editor and a trace feature that prints each major step occurring during execution of an RPGII program. A System/3-compatible sort/merge capability is provided by the optional Sort/Merge package, and communication of RPGII files is available through the optional DX10 3780 Emulator package.

Texas Instruments version of RPGII is especially suited for users with rapidly growing applications. The minimum system for RPGII is the 990 processor with 128K-byte memory, 10M-byte disk drive, 911 VDT terminal, single-bay desk enclosure, and DX10 disk-system software license. Expansion capabilities can provide large amounts of memory, multiple 50M-byte disk drives, and a variety of additional standard 990 peripherals.

The RPGII compiler has the following significant features:

- Efficient one-pass compiler
- Run-time trace that speeds the checking of the program
- Right- or left-hand sign handling
- ASCII or EBCDIC internal character set
- Capability to produce more than 500 unique diagnostic messages
- Alphabetic summary listing of all fields, labels, arrays, and tables
- Listing of all indicators specified in a program.

14.2.3 DX10 UTILITIES. Texas Instruments supplies the TIFORM and Sort/Merge utility packages for application programming.

14.2.3.1 TIFORM 990 Utility. TIFORM 990 is a utility package for controlling the interactive interface to an application. It provides convenient control of complex screen formats for COBOL, FORTRAN, and Pascal applications. Included in the package is both an interactive screen drawer and a screen description language compiler. Through these two tools, TIFORM isolates the description of the screen format from the application's procedural code, allowing applications to become independent of the terminal. TIFORM also provides:

- All available terminal features (blink, dim, hilite, no display, etc.)
- Character and field level editing
- Up to 40 percent improvement in interactive application development time.

14.2.3.2 Texas Instruments Page Editor (TIPE-990) Utility. The TIPE-990 package offers word-processing features for creating, editing, and printing pages of text to complement the data processing capabilities of the TI 990 commercial computer systems. TIPE-990 efficiently produces letters and documents, is easy to use, and requires minimal training. The TIPE-990 package operates on standard DS990 hardware, including the 911 VDT and the 810 matrix printer. Letter-quality printing is also available, using the optional Model LQ45 printer.

The following functions are available with TIPE-990:

- Create a new TIPE-990 document
- Edit an existing TIPE-990 document
- File the document now being created/edited
- Disregard changes made since last Create or Edit Command
- Print a TIPE-990 document
- Quit using the creation/editing program

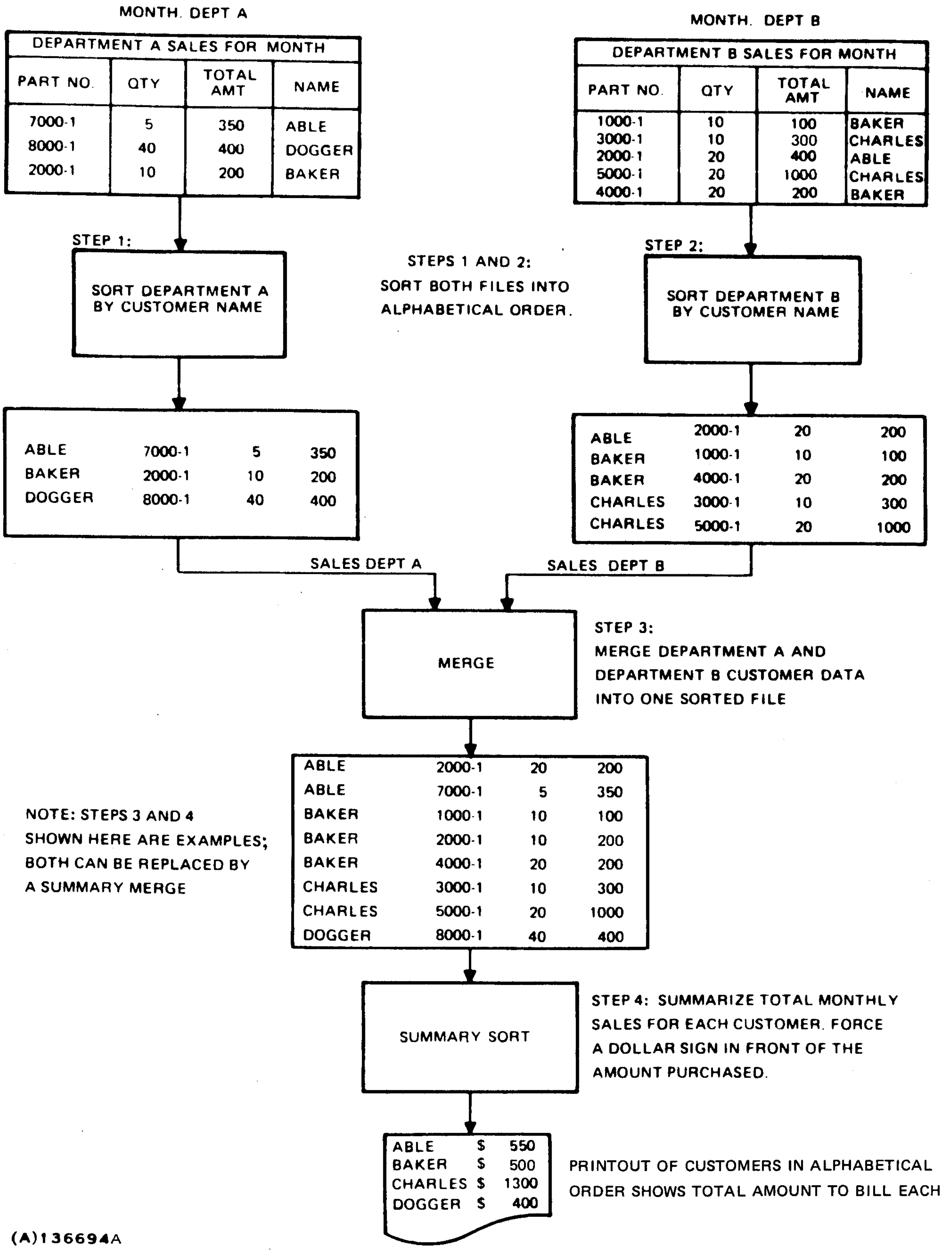
14.2.3.3 Sort/Merge Utility. The DX10 system supports a comprehensive Sort/Merge package that may be accessed in several ways. SCI provides commands to access Sort/Merge in batch or interactive mode. COBOL, FORTRAN, DX10 BASIC, and Pascal programs may interface to Sort/Merge by using the CALL statement. Both sort and merge support the following features:

- Record selection.
- Reformatting on input.
- Summarizing on output.

Ascending key order, descending key order, or an alternate collating sequence may be specified. Any number of keys may be specified as long as their total length is less than 256 characters. The merge process supports up to five (5) input files. The sort process allows the following:

- Key sort (Tag-Along).
- Summary sort (Summary Tag-Along).
- Address Only sort.

Figure 14-9 shows an example of the Sort/Merge process with printouts of results at each step. Additional information describing Sort/Merge is found in the Sort/Merge manual.


MONTH. DEPT A
MONTH. DEPT B
STEP 1:
SORT DEPARTMENT A BY CUSTOMER NAME
STEPS 1 AND 2: SORT BOTH FILES INTO ALPHABETICAL ORDER.
STEP 2:
SORT DEPARTMENT B BY CUSTOMER NAME
SALES DEPT A
SALES DEPT B
MERGE
STEP 3: MERGE DEPARTMENT A AND DEPARTMENT B CUSTOMER DATA INTO ONE SORTED FILE
NOTE: STEPS 3 AND 4 SHOWN HERE ARE EXAMPLES; BOTH CAN BE REPLACED BY A SUMMARY MERGE
SUMMARY SORT
STEP 4: SUMMARIZE TOTAL MONTHLY SALES FOR EACH CUSTOMER. FORCE A DOLLAR SIGN IN FRONT OF THE AMOUNT PURCHASED.
PRINTOUT OF CUSTOMERS IN ALPHABETICAL ORDER SHOWS TOTAL AMOUNT TO BILL EACH
(A)136694A

DEPARTMENT A SALES FOR MONTH

PART NO.	QTY	TOTAL AMT	NAME
7000-1	5	350	ABLE
8000-1	40	400	DOGGER
2000-1	10	200	BAKER

DEPARTMENT B SALES FOR MONTH

PART NO.	QTY	TOTAL AMT	NAME
1000-1	10	100	BAKER
3000-1	10	300	CHARLES
2000-1	20	400	ABLE
5000-1	20	1000	CHARLES
4000-1	20	200	BAKER

ABLE	7000-1	5	350
BAKER	2000-1	10	200
DOGGER	8000-1	40	400

ABLE	2000-1	20	200
BAKER	1000-1	10	100
BAKER	4000-1	20	200
CHARLES	3000-1	10	300
CHARLES	5000-1	20	1000

ABLE	2000-1	20	200
ABLE	7000-1	5	350
BAKER	1000-1	10	100
BAKER	2000-1	10	200
BAKER	4000-1	20	200
CHARLES	3000-1	10	300
CHARLES	5000-1	20	1000
DOGGER	8000-1	40	400

ABLE	$	550
BAKER	$	500
CHARLES	$	1300
DOGGER	$	400

Figure 14-9. Sort/Merge Process Showing Printouts of Results at Each Step

14.3 DATA BASE MANAGEMENT SYSTEM (DBMS 990)

The DBMS 990 (Data Base Management System) is designed for minicomputer data-base applications. Specifically, this system handles applications with fast data-access requirements which need to be accessed in a logical format that can be easily equated with physical documents or records used in daily business transactions. The DBMS 990 allows the user to define and access a centralized, integrated data base using logical format without the physical data-access requirements imposed by conventional file-management software. Physical considerations such as access method, record size, blocking, and relative-field positions are resolved when the data base is initially defined. Thus, the user can concentrate fully on the logical data structures needed for interface.

14.3.1 FEATURES OF DBMS 990. The independence of the data definitions from the application software allows modification of the data base without impact to existing programs. It also provides a single, centralized copy of the data for all application subsystems. (Conventional file management provides fragmented and multiple copies of data held in a wide variety of files with each used by only one application.) This centralized copy results in more efficient data storage on disk, uniform processing of data requests, and centralized control of the Data Base Maintenance function. In addition, DBMS 990 provides optional password security for the most elementary data level; this provides control and protection of the data base from unauthorized access or tampering.

14.3.2 DBMS 990 USER INTERFACE. The primary user interfaces to DBMS 990 consist of the data-definition language (DDL) and data-manipulation language (DML).

DDL provides the means to completely describe the DBMS 990 data base and its associated data elements. The DDL logical data-base definition source is compiled by the DDL compiler, and the output is stored with its associated data on disk.

DML provides the user the means to manipulate DBMS 990 data by supporting the reading and/or writing of DBMS 990 data. DBMS 990 data can be accessed by imbedding the appropriate DML syntax in a COBOL, Pascal, or FORTRAN application program. The call construct of the language is used to call DBMS 990 and specify a function to be performed and the data element to be manipulated. DBMS 990 processes the request and returns the results to the application program.

14.3.3 QUERY 990. Query 990 is a general-purpose data base application that interactively retrieves DBMS 990 files through the standard DML interface. Query 990 allows a user who is familiar with a DBMS 990 file structure but unfamiliar with programming languages or the DML language to obtain complex formatted reports.

14.3.3.1 Query Environment. The Query processor produces a report or data file by accepting and executing a Query language statement. Two ways to build and execute a Query statement are as follows:

- The user may invoke the Query processor directly and pass to it a Query statement file (interactively or in batch mode) built through either the Query editor or the system Text Editor.

- The user may build a Query language statement by executing the Guided Query utility, which constructs the statement by prompting the user with questions to determine the contents and format of the report.

14.3.3.2 Query Language. The Query language is an English-like nonprocedural language with statements composed of several clauses. The clauses allow the user to specify the contents and format of each line as well as complex conditions that a data base record or line must meet to be qualified for output. Totals, counts, or averages may be performed on output fields, and default columnar headings and user-defined headings are supported.

By using the Query language, a complex report may be specified in a few lines, while an application program to obtain the same report may take several hundred lines.

SECTION 15

SYSTEM GENERATION

A disk cartridge supplied by Texas Instruments that contains a base DX10 system is used to generate a custom DX10 system. Alternatively, a DX10 disk may be built from magnetic tape. The generalized system supports only the required devices and is generated with parameters that are operational in a wide variety of hardware configurations. The disk contains the required and optional piece parts of DX10 that are combined during system generation to produce the target custom DX10. A custom DX10 supports the specific devices and resources of a particular hardware configuration. Custom system generation allows the user to include those portions of the system that a specific application requires, and to omit unused portions that occupy disk and/or memory space.

Custom system generation consists of executing the SYSGEN program which may be run interactively or in batch. The former is the preferred method. This program is used to obtain user specifications for the custom DX10. It outputs the source code to a system data module that is assembled for the custom system. The SYSGEN program also generates the command stream for the link editing operation. After the system is configured, the user executes a command to run a batch stream that assembles the generated source module and links it with the required system modules and any optional user-supplied modules. Link edit output is directed to a DX10 program file. The new system may be specified as a temporary primary system. When it has been loaded and executed properly, the user may designate the new system as the permanent primary image. Otherwise, the user may IPL the previous system while modifying the custom DX10 to correct any problems.

The SYSGEN process allows a user to incorporate special device drivers, custom supervisor call or extended operation (XOP) processors, and an initialized system common module. Also, task management features, such as the time-slicing option and time-slice value, may be selected at SYSGEN time in order to configure DX10 for the user's application.

SECTION 16

SYSTEM LOG

DX10 provides support for the system log. Information logged by DX10 includes device errors and task errors (see Figure 16-1). Tasks that are abnormally ended provide task error messages to the system log. Application programs may specify additional messages to be logged by issuing the appropriate supervisor call. A system command is provided to start the system log at which time output to disk files and/or output to a hardcopy device is specified. A pair of files is supported such that after one fills, data logging is transferred to the second file. Optionally, a user-specified task may be executed to read the file just filled. All logged messages include the time of occurrence. A complete description of system log records may be found in the *Model 990 Computer DX10 Operating System Error Reporting and Recovery.*

```
202:1004+DS01 ERR=01 IID=02 L=05 A=0007 0008 0009 000A 000B 000C 000D 000E
                ST04 RID=03 S=06 B=000F 0010 0011 0012 0013 0014 0015 0016
202:1005+DS01 ERR=01 IID=02 L=05 P=0007 0008 0009 000A 000B 000C
                ST04 RID=03 S=06
202:1006     THIS IS A SHORT MESSAGE
202:1007+    THIS MESSAGE IS MORE THAN 67 CHARACTERS LONG AND WILL BE SPLI
          T BETWEEN MORE THAN ONE LINE
202:1008 MEM  BIT=01 ROW=02 CORRECT=Y BASE=1F000 MEM=96KB TYPE=0 TPCS=FB04
202:1009 MEM  BIT=01 ROW=02 CORRECT=Y BASE=1F000 MEM=64KB TYPE=1 TPCS=FB18
202:1010 MEM  BIT=00 ROW=00 CORRECT=N BASE=00000 MEM=00KB TYPE=0 TPCS=0000
202:1011 MEMC BANK=A PARITY: A=G, B=G BASE=1F000 MEM=64KB EVEN=Y TPCS=FB10
202:1012 STAT DEV=DS04 READS=1F59 WRITES=2183 OTHER=0E7A ERRORS=03A1
202:1013 TASK ERR=01 IID=02 RID=03 ST04 WP=0123 PC=4567 ST=89AB
202:1014 **** LOG STARTED *************************************************
202:1015 **** LOG FILE .S$LG1 FULL
202:1016 **** LOG FILE .S$LG2 FULL
202:1017 **** LOG MESSAGE(S) LOST
202:1018 **** DEVICE MISSED LOG MESSAGE, I/O STATUS=00
202:1019 **** FILE MISSED LOG MESSAGE, I/O STATUS=00 ON LOG FILE .S$SLG1
202:1020 **** DEVICE LOGGING DISABLED
202:1021 **** FILE LOGGING DISABLED
202:1022 **** ALL LOGGING DISABLED
202:1023 **** SYSTEM CRASH OCCURRED ***************************************
```

Figure 16-1. Example System Log Output

SECTION 17

ERROR CONTROL

The DX10 Operating System incorporates several features dealing with error control. When failures are detected by DX10 during I/O data transfers, the attempted transfer may be retried. After several attempts, a code indicating the device failure is returned to the originating program. For example, the application program is expected to handle the situation where the line printer runs out of paper. The failure will be recorded in a system log.

Error codes are returned to programs that issue illegal supervisor calls. In some cases, additional codes are returned that convey information concerning potential errors. Every application program operating under DX10 should be examined for all generated error codes and appropriate action taken for recovery.

The 990 mapping feature protects DX10 from destruction by errant application programs. Application programs are protected from errant interaction in a similar fashion (except where they are overtly sharing a procedure). When application programs can no longer execute due to some illegal function (referencing memory outside their legal range, illegal instruction, etc.), they are abnormally terminated by DX10. In these cases, an advisory message is directed to the system log. The message displays a code indicating the cause of abnormal termination. These messages should be analyzed to determine the appropriate recovery procedure.

Any task may optionally include a sequence of instructions designated as an end action routine. If this option is selected and an abnormal termination situation arises, DX10 returns to the task one final time at the entrance of the end action routine in that task. The application programmer must provide code in the end action routine to analyze the termination code returned by DX10 and to take appropriate recovery steps.

In some severe circumstances, a fatal system error occurs. Failure of the system disk on a critical operation is a fatal system error. Where fatal system errors occur, the system itself comes to an abnormal termination (system crash). In these rare circumstances, an error code is provided to the front panel. DX10 provides an operating procedure that preserves a post mortem image of memory on the disk when a system crash occurs.

As in most cases, preventive procedures should be employed to avoid failures and to minimize losses should they occur. Some recommended preventive measures include:

- Back up disk files regularly.
- Schedule regular equipment preventive maintenance.
- Always analyze error codes of both user and/or program errors.
- Build self-checking into application programs.
- Design application procedures and programs to allow graceful degradation wherever possible.

APPENDIX A

GLOSSARY

Access Name — Device name or file pathname that identifies a physical source or destination for I/O operations when assigning a LUNO to a device or file. An access name may have a maximum length of 48 characters.

Access Privileges — Files may be shared by several tasks, shared for reading with writing exclusive to a single task, or exclusively owned by a single task. The task is said to have one of the following types of access privileges:

1. Shared access,
2. Read only access,
3. Exclusive write access, or
4. Exclusive all (read and write) access.

Active – As applied to a task, active means queued in memory for a given priority level and ready to execute in turn.

Address Space – The memory addressable by an addressing scheme is called the address space of that scheme. The memory which may be addressed by a task differs from the memory which may physically be addressed by the 990. This difference is resolved by the mapping hardware and DX10.

Alias – An alternate for a file pathname component. The alias may be used in place of the pathname component. It is convenient to use aliases to force special processing in some instances. Aliases are especially useful when link editing.

Allocable Disk Unit – The smallest disk space which may be allocated for file creation or expansion. At present, this space varies from 1 to 11 sectors depending on the type of disk drive.

Autocreate – A bit in the supervisor call block for the assign LUNO I/O call that when set, specifies that DX10 is to automatically create the file to which the LUNO is being assigned if it does not already exist.

Background Mode — Under DX10, foreground/background partition indicates terminal access rather than priority of execution. Background tasks and foreground tasks may execute at the same priority level. However, background tasks only output to a terminal when output is specifically requested or when all foreground tasks have terminated.

Base System — The initial system image shipped to the customer. The base system is a minimum system capable of performing system generation and supports a 911 VDT, 913 VDT, or teleprinter devices.

Batch Mode — Under DX10, SCI commands can be executed from any terminal, sequential file, or sequential device. When the input device is other than a terminal, i.e., it is a sequential file or device, SCI is said to be running in Batch Mode. In this mode, all parameters must be supplied in 'KEYWORD = value' format.

Bid – To place a task in the active status by requesting the system enter that task on the active list with the given priority.

Blank Suppressed — A file is blank suppressed if strings of consecutive blanks within the file (each occurrence) are replaced with a code which represents the number of blanks (i.e., consecutive strings of blanks are not stored literally).

Blank Adjusted – The record attribute denoting the padding of a record with blanks on input or the removal of the blanks on output.

Blocked Disk Transfer — The movement of data through a system-supplied buffer as an intermediary between the disk and the user buffer. This employs packing of one or more logical user records into a single physical record.

Boot – Short form of bootstrap. This is another name for the program which performs IPL.

Bounded File – A file which cannot grow beyond its initial allocation.

Break Point – A place in a routine at which execution halts. Under DX10 a break point is specified by XOP 15,15. Except in the controlled mode, breakpoints do not automatically return control to a monitor program.

Boundary Error – A task which attempts memory access outside its assigned range causes a fatal error called a boundary error.

Buffer – Storage used to compensate for differences in the rate of data flow, time of occurrence of events, repacking of data, or transmitting data from one location to another.

Call Block – A call block is a list of parameters passed to various DX10 supervisor calls. The call block contains all information needed by the supervisor call and may be quite extensive in some calls, for example, I/O.

Cache Memory — Memory providing rapid access to frequently used information.

Coded Error – An error for which the type is indicated by a numeric code. The numeric code may be returned on a call, displayed, or posted in a special area (usually byte 1 for most supervisor calls).

Collating Sequence — A method of specifying an order for a collection of character strings. For example, the usual collating sequence for English is alphabetical order. The natural collating sequence for DX10 is the numerical order sequence of the ASCII character codes.

Command – A command is an operator entry indicating to DX10 the next operation to be performed. Commands from interactive devices are interpreted by SCI. SCI then selects the tasks in DX10 that perform the desired action.

Command Mode – A processor mode in which the processor interprets input as commands rather than as data. The Text Editor and System Generation programs have a command mode.

Command Procedure – Under DX10, commands have a defined format and syntax which uniquely defines their action to DX10. Commands are defined by command procedure entered in a command procedure file supplied when the command is generated.

Command Processor – A DX10 task bid by a command procedure to perform an SCI command or part of an SCI command.

Common – The System Common memory area is an area of memory accessible to all tasks through the get common data address supervisor call. The size of system common memory area is a system parameter supplied at system generation. The system common memory area is accessed as one of the memory segments of every task that uses it.

Compose Mode – A mode of operation in the Text Editor in which keyed data is entered into the file being edited without special commands to cause entry.

Concurrent Tasks – Two tasks which are simultaneously *active* are said to be concurrent. Only one task may be *executing* at a given instant.

Configuration File – A configuration file is a file describing a DX10 System created by the system generation program.

Context Switch – Context switch is a transfer of control that uses a two-word transfer vector to define the new environment. A context switch consists of the following operations:

- The first word of a transfer vector is placed in the WP register.
- The second word of the same transfer vector is placed in the PC.
- The previous contents of the WP register is stored in WR 13 of the new workspace.
- The previous contents of the PC is stored in WR 14 of the new workspace.
- The contents of the ST register is stored in WR 15 of the new workspace.

A context switch transfers control to a program or subroutine, and activates a workspace associated with the program or subroutine as control passes to the new PC contents. The context switch stores the program environment, that is, the workspace address, the address of the next instruction in sequence, and the program status. A return instruction (RTWP) restores the environment by returning the stored data to the WP register, the PC, and the ST register. Context switches transfer control to subroutines when an interrupt occurs, when an extended operation instruction is executed, or when a branch and load workspace pointer instruction is executed.

Crash Code – When DX10 detects an uncorrectable system error, DX10 aborts itself and displays a code on the programmer panel indicator lights which indicates the type of error. The code displayed is called a crash code.

CRU – COMMUNICATIONS REGISTER UNIT (CRU). The CRU is the 990 general-purpose, command-driven I/O. The CRU is the "bit picking" I/O originally designed for process control operations and introduced with the Model 960 Computers. The CRU has proven to be a very successful computer I/O system because of the flexibility which adapts to every command-driven application.

Currency — Records in key indexed files are accessible in the sort order of any key. For this purpose, DX10 maintains a pointer called currency, which points to the current record position in sort order.

Deadlock – Deadlock occurs when contention for resources is so severe that the computer is essentially stalled. See Thrashing. DX10 is constructed to prevent deadlock.

Debug – Debug means to remove flaws from a program. Debug aids under DX10 include extensive error detection capabilities.

Default Value — Certain parameters under DX10 take default values. If a parameter value is not entered, then DX10 assumes the default value for that parameter. This is not the same as an initial value. See the paragraph entitled "Type of Response Expected" in Section 3 of the *Model 990 Computer DX10 Operating System Production Operation* for information on default values.

Deferred Write – Ordinarily, I/O Operations occur immediately when requested. However, for disk files, to speed processing, reads and writes of logical records may actually go to an area in memory that corresponds to the physical record block, to be written to disk at some later time. This deferring of disk writes may drastically increase the system throughput.

DSEG – An assembler directive which specifies that the following assembly language code is "data" (i.e., it is to be placed in the task segment).

Device – A device is physical equipment such as a card reader or line printer to which DX10 allows input or output.

Device Name – A device name is a unique four-character string by which a device is known to DX10. The first two letters are alphabetic and indicate generic class. The second two characters are numeric and indicate sequence number of the device.

Device Service Routine – A Device Service Routine (DSR) is that part of DX10 that communicates directly with an I/O Device. It services interrupts, and performs the desired input and output operations.

Directory – A directory is a file that contains the names of and pointers to other files. It does not contain data.

Disk — Refers to a hard disk medium, which may be installed on a DS10, DS25, CD1400/32, DS31, DS32, DS50, DS200, and CD1400/96 disk drive; or a dual-density double-sided medium, which is installed on an FD1000 disk drive.

Disk-Resident – A task is said to be disk-resident if it is rollable, i.e., if while active it may reside on disk when not actually executing. This is opposed to a memory-resident task that is resident in memory and never rolled.

Disk Volume — A disk cartridge that has been named and initialized; it may be installed by name on a disk drive. Disk volume names must not be the same as disk device names (i.e., DS01, DS02, etc.).

Diskette — Refers to a single-sided, single-density media, which is installed on an FD800 or an FD1000 diskette drive.

DSR — See Device Service Routine.

Drive A peripheral device that holds and operates a disk volume, or magnetic tape reel.

Edit Mode – A mode in the text editor in which lines may be entered anywhere in a text file, but only one line at a time.

End Action – If a task commits a fatal error, DX10 executes a routine specified by the task before aborting the task. This routine is called the *End-Action routine*, and the task is said to have taken end action.

End-Of-File – A marker or other identification that denotes the end of a sequential file of data.

End-Of-Medium – A mark or other identification that denotes the end of available storage space for a file of data.

End-Of-Record – The character of a record that marks the end of a record; the characters that denote the end of record vary from device to device.

Event Character – Under DX10, certain keys on the keyboard take precedence over data keys and have special significance to executing tasks. These keys are called Event Keys, and the characters they generate are called event characters.

Executing – A task is executing if it has control of the processor and resources of the computer.

Expandable File – A file is said to be expandable if it may grow beyond its initial size.

File Attribute — A declared characteristic of a file that limits the types of operation performed on a file. For example, delete-protected is an attribute and files with this attribute cannot be deleted until the attribute is removed.

File-Oriented – A device or file is said to be file-oriented if it is reserved through an OPEN call in order to perform I/O to the device. Typically, the device remains assigned to a requesting task until explicitly released.

Foreground — Under DX10, foreground/background partition indicates terminal access rather than priority of execution. Foreground tasks have immediate terminal access while background tasks only report when requested via the Show Background Status (SBS) command or WAIT command.

Function Key — A key on a terminal such as the ENTER key, that causes the system to perform some predefined function for the user.

Generate Mode – A mode in the system generation program in which a new system configuration is described.

Global LUNO — Global LUNO is a LUNO whose scope is not limited to a task or station. Any task may use a global LUNO unless it is in use by another task.

Hard-Copy Terminals — Same as Teleprinter Devices.

Hashing – Hashing is a technique for mathematically processing key words or file names to produce numbers, usually record numbers.

Image File – An Image File contains the memory image of a program. It differs from a program file in that there is no directory and only one program.

Immediate Write Attribute – Ordinarily, file I/O is buffered in memory and deferred until the memory space is required. File I/O for files with the immediate write attribute occurs immediately and is not deferred.

Initial Program Load — When the 990 computer is first powered up, it has no program in memory. The first step in executing DX10 is to place the initial program in memory. This procedure is called Initial Program Load (IPL).

Initialization, Disk – The process of writing required system overhead tracks and formatting a disk prior to using the disk under DX10.

Initial Value — Initial Value is similar to default value; however, initial values may be changed and, once changed, stay the same until changed again. Default values do not change. See the paragraph entitled "Type of Response Expected" in Section 3 of the *Model 990 Computer DX10 Operating System Production Operation* manual for information on initial values.

Initiate I/O – An Initiate I/O call causes I/O to be started but does not wait for the completion of that I/O event.

Install – When applied to programs, *To Install* means to load a task, procedure, or overlay on a program file. When applied to volumes, *Install* means to direct the system to reference an initialized volume on a given drive by the supplied name.

Installed ID – When a task, procedure, or overlay is installed on a program file it is assigned a unique identification number from 01 to FF_{16}, which DX10 uses to locate that module on the program file.

IPL — See Initial Program Load.

Key – A key is a character field within a record in a file denoting, at least partially, the structure of the file.

Key Indexed Files — A key indexed file is a file in which records may be accessed by the value of a character string called a key. Under DX10, each record may have up to 14 unique keys, thus DX10 access through each key is independent of the other keys. Key indexed files under DX10 are called key indexed files.

Key Value – Whereas a key is a character field, a key value is a particular instance of a literal value of the character string in that field.

Keyboard Status Block – A Keyboard Status Block is a data structure internal to DX10 used to respond to keyboard interrupts and which serves as an anchor for station LUNOs.

Least Recently Used Strategy – When considering candidates for Rollout, DX10 uses the least recently used strategy. The task which has been inactive the longest is considered the least likely candidate for immediate future use and is thus the first candidate for rollout.

Link Editor – The Link Editor is a DX10 processor that takes related object modules and links them together into a single object module.

Loaded – As applied to a task, this means having been copied from an external storage medium into the memory of the computer in preparation for execution.

Logical Address Space – The memory accessible to a task. The maximum extent of any logical address space is 32K words.

Logical Device Table – A table in DX10 that contains the LUNO and references system tables which correspond to the device or file assigned to the LUNO.

Logical Records – A logical division of data in a file that can be input or output with a single supervisor call.

Logical Record Length – The length (in bytes) of records in a file. This length does not necessarily correspond to any physical division of the disk.

Logical Unit Number (LUNO) — A number specified in an I/O operation which represents a file or device.

LUNO — Logical Unit Number.

Log-On — The process whereby the user station gains access to DX10 services.

Log-Out — The process whereby the user relinquishes access to DX10 services.

Memory Mapping – Memory Mapping is a hardware feature of the 990/10 and 990/12 controlled by the DX10 Software, which allows the 990 to address a million words of memory, even though the standard address format of 16 bits could only address, at most, 64K bytes.

Memory-Resident – A task is memory-resident if it is always in memory, even when not executing.

Multi-Programming – Multi-Programming is a process wherein more than one computer program may be in memory, queued for execution. This is opposed to multi-processing in which more than one program may be currently executing.

Multi-Tasking — Multi-Tasking is the name for DX10 multi-programming. Several tasks may be in memory simultaneously, each in turn allotted execution time, however, only one task may be executing at a given instant.

Multi-Terminal – DX10 supports concurrent use of more than one terminal, thus it is a multi-terminal system.

Non-Expandable File – The file is said to be Non-Expandable if it cannot grow beyond its initial size.

Object Code – Machine Language code together with load control code in any of several loadable formats.

Overlay – An overlay is a part of a task that resides on disk until explicitly requested. When requested, the overlay replaces part of the task previously in memory. Using overlays can reduce the amount of memory required by a task to the amount required for the largest segment requiring memory at one time.

Pathname – A pathname is a name for a file under DX10. It consists of a volume name (optional) followed by any number of directory names and then by a file name. All names are separated by periods. The volume name is 1 to 8 characters in length and may be omitted, directory names are 1 to 8 characters in length and may not be omitted, file name is 1 to 8 characters in length and must be present in a pathname. The maximum length of the pathname cannot be greater than 48 characters.

Physical Address Space – The addressable memory of the computer. This term also refers to the range of memory addresses available to the computer even if the memory is not actually implemented in a particular configuration.

Physical Record Length — The size (in bytes) of blocks of data transferred to and from a disk. Often, a physical record includes several logical records.

Priority, Floating – DX10 supports four levels of priority (0, 1, 2, 3) for tasks, zero priority being the highest and 3 the lowest. System tasks operate on 0 priority. Tasks may be executed with a constant priority or with a floating priority. Floating priority is specified as priority level 4, but the actual priority of the task is dynamically adjusted to 1, 2, or 3 depending on the type of activity of the task (I/O, Terminal I/O, and computation are considered).

Privileged – Tasks in DX10 are either privileged or nonprivileged. Nonprivileged tasks are prohibited from performing certain reserved functions while privileged tasks may execute any code and any supervisor call.

Procedure Segment – Procedure Segment of a task is that part of a task which is usually executable code, but in any case may be shared with any other tasks.

Procedure Library – A directory of files containing command procedure definitions.

Program Counter – Program Counter is a hardware register that indicates to the computer the next instruction to be executed.

Program File – A special form of relative record file used to contain executable programs.

Programmer's Panel – A peripheral device mounted on the front of the 990/10 computer providing an elementary interface to the computer.

Prompt — When an SCI command is executed, the user is prompted for values upon which the command operates. Each line that describes the input value needed is termed a prompt.

PSEG — An assembler directive which specifies that the following assembly language code is procedure code (i.e., it is to be placed in a procedure segment).

Queue — First-In, First-Out waiting list. A queue is a data structure consisting of a list of objects to be processed where the order of the list is chronological. The first object entered on the list (First-In) is the first object processed (First-Out). Further, objects, when processed, are removed from the list.

Record Locking — Within a file, records may be locked. A locked record may not be accessed until unlocked. This process is useful when controlling file access by several contending tasks.

Record Oriented – An I/O Device is said to be record oriented if access to the device is granted to users on a record basis. Usually, a record corresponds to a single I/O operation.

Reentrant Program – A reentrant program is a program which may be shared among several users in a multi-task environment without conflict of data.

Relative Record File – A Relative Record File is a directly addressable file in which the records are numbered from the beginning of the file. Access to any record is obtained by calculating displacement from the front of the file based on the record number and the record length, which must be fixed.

Resource Contention – When two or more tasks attempt to use the same resource. Resources include peripherals, memory, files, and CPU time. DX10 allocates resources to avoid contention.

Random File – Random File is another name for relative record file.

Replicatable Task – If specified at task installation, multiple copies of a task may be simultaneously in memory. This frequently occurs for utility tasks such as the FORTRAN Compiler or SCI.

Roll-In – When a task on disk is scheduled for execution it is rolled in from disk to memory, i.e., it is placed in memory so that it may begin execution.

Roll-Out – If a task is scheduled for execution but a lower priority task currently occupies the memory, then the lower priority task is copied onto the disk in a location called the system roll file. The lower priority task is rolled out. This makes room for the higher priority task currently scheduled to execute to be placed in memory.

ROM Loader – The 990 computer is totally software driven. Without a program in memory the 990 cannot be loaded. To circumvent this problem, there is a program in Read Only Memory (ROM) on the system interface board at a dedicated memory address which controls the program panel and performs the first part of IPL. Part of this program is called the ROM Loader.

Runtime ID – An identification number assigned by DX10 to an executing task for the duration of its execution.

Scrolling — Certain displayable files are larger than the display screen of a Video Display Terminal. To view the entire file, the user may show the next page or previous page of information by using one of the function keys. This process is called scrolling.

Secondary Allocation – A secondary allocation is a block of disk space automatically allocated by the system to a file which requires more space than its present allocation. The size of the secondary allocation is specified when the file is created, but increases with each subsequent secondary allocation.

Segmentation – DX10 allows tasks to be installed in separate pieces called segments. Up to three segments may be used for each task. Further, tasks may share segments. The process of separating a task into segments is called segmentation.

Sequential File – A Sequential File is a variable record length file which must be accessed in chronological order, the order in which the records are written to the file.

Station – A Station is an interactive terminal with an associated copy of the System Command Interpreter. Physically, a station may be a Video Display Terminal or a hard copy device such as the 733 ASR terminal.

Subdirectory – A directory which is pointed to by another directory. Every directory on a disk except VCATALOG is a subdirectory. If directory A contains the name "B" and a pointer to directory B then B is said to be a subdirectory of A.

Supervisor Call – All user task requests for DX10 Services are made by supervisor calls. A Supervisor Call is an XOP 15 instruction which performs a context switch into DX10 which then interprets the call and performs the desired service (if legal).

Supervisor Call Block – Associated with each Supervisor Call is a list of necessary parameter values for that call. This list is called a Supervisor Call Block. The size of the Supervisor Call Block depends on the Supervisor Call.

Suspended – As applied to a task, suspended means that the task was temporarily removed from the active list and from execution as a result of a Supervisor Call or during an I/O Operation.

Synonym – A synonym is an alternative name for a text string. Usually a synonym is shorter than the object it replaces and is more convenient to use.

System Command Interpreter — The System Command Interpreter (SCI) is the user interface to DX10. It prompts the user, accepts inputs, interprets them as commands, and directs activities in DX10 to satisfy those commands.

System Command Interpreter Language — This is a language in which commands to System Command Interpreter are written. It has a detailed syntax defined in the *Model 990 Computer DX10 Operating System Programming Guide.*

System File — A system file is a file used internally by the DX10 operating system. All system file names begin with "S$".

System Generation – System Generation is an interactive process wherein a new version of DX10 is configured to match a particular hardware installation. Also, certain parameters supplied at system generation allow the fine tuning of system performance.

System Log — System Log is the part of DX10 that reports hardware and task errors to a file and (optionally) to a terminal or printer. A user may also output a message to the System Log with a supervisor call.

Scheduler – The Scheduler is a part of DX10 which decides which task is to receive memory space and execution time.

System Table Area – Memory reserved for DX10 and used for system data structures and buffers. The size of the system table area is specified during system generation.

Tape Volume – Software name for a reel of tape.

Task – A program which executes under DX10 is a task. A task consists, logically, of a data division (data) and a procedure division (instructions). The data and procedure divisions may or may not be separately assembled, and may or may not have been separately loaded. When the data and procedure divisions are separately installed, the data division is called the task and the procedure division is called the procedure. A task may have one or two procedures and procedures may be shared with other tasks.

Task Local LUNO – A Task Local LUNO is a LUNO which may only be used by the task which assigns it. It is not available to any other user.

Task Sentry — A part of DX10 that monitors task activity to prevent CPU lock out by lowering the priority of a CPU-bound task at set intervals. The Task Sentry may be enabled or disabled at SYSGEN.

TCA File — A file of images of users' TCAs. A Terminal Communications Area (TCA) is an area of memory within SCI that contains user ID information and synonym names and values.

Teleprinter — One of the following hard-copy terminals: 733, 743, 745, 763, 765, 781, 783, 785, 787, 820, or 840.

Terminal – Terminal is any interactive user device. This means the same as station.

Terminal Local File – Terminal Local File is a file associated with the terminal which contains information to be displayed at the terminal when a given process terminates.

Terminated – As applies to a task, terminated means that the task was removed from execution and from the active list either at normal completion or at an abnormal termination initiated by the operator or by DX10 when a fatal error is detected.

Textual Errors – Errors which are indicated by a displayed or printed text string which indicates the type and perhaps cause of the error.

Time Slice – DX10 allocates execution time in small segments called time slices. The length of a time slice is determined from the clock interrupt rate (100 or 120 Hz) and the number of clock interrupts per time slice.

Thrashing – Thrashing occurs when the overhead required to resolve resource contention occupies most of the computer's time. DX10 is constructed to minimize thrashing.

TILINE* – The 990's Asynchronous 50 mega bit/sec memory bus, used by the CPU, memory, and disk and tape controllers.

TILINE Peripheral Control Space – A range of TILINE addresses reserved for TILINE (DMA) device controller interfacing.

Unload – Refers to both the software action preparing a disk cartridge or tape reel for removal and the act of physically removing the cartridge or reel.

User ID — A six-character identifier associated with a user used by SCI to select a TCA image at log-on. The first character must be alphabetic and the last character must be numeric.

VCATALOG – VCATALOG is the highest level directory on each disk volume which specifies (either directly or indirectly) all files on the volume.

VDT Mode – Video Display Terminals under DX10 may run as hardcopy emulators (TTY mode) or may use the full power and speed intrinsic to their nature (VDT mode).

Volume Name – On each disk volume, in a reserved location is a unique name for that volume which is used as the first part of a pathname to reference files contained in the volume. This name is defined when the volume is initialized.

Warm Start — To perform the initial program load (IPL).

WCS — See Writable Control Storage.

Weighting Factor — A count of time slices for priority level. When a number of time slices specified as a weighting factor for priority level has been used by tasks at that level, a task at a lower priority level receives control for a single time slice.

Workspace – A 16-word area of memory addressed as work space register 0 through 15. The active work space as defined by the contents of the workspace pointer.

Workspace Pointer – The hardware register that contains the address of work space register 0, and indicates the currently active work space.

*TILINE is a trademark of Texas Instruments Incorporated

Workspace Register – A memory word accessible to an instruction of the computer as a general purpose register. It may be used as an accumulator, a data register, an index register, or an address register.

Writable Control Storage (WCS) — In the Model 990/12 Computer, an area set aside that contains microcode that is executable by user tasks via an XOP instruction within the user program. WCS, with some restrictions, is available in the current release of DX10.

XOP -- XOP is a 990 Assembly Language instruction which generates an internal interrupt.

ALPHABETICAL INDEX

INTRODUCTION

This index lists the commands and concepts covered in the DX10 manuals. It presents the entries in alphabetical order with numerical items last. Each entry indicates where the command or concept receives major coverage in the DX10 manuals. These are the reference types used:

- Paragraphs — These references point to a discussion in the main body of this manual. They follow the same format as the paragraph numbers shown in the Table of Contents: section number (S), then top-level paragraph number (P), then lower-level paragraph numbers (p and q) as needed. A reference to an entire section uses only the section number (S).

  S. P. p. q
  Section S

- Appendixes — These references guide the reader to information in one of the appendixes in the back of this manual. Appendix references look like paragraph references, except that appendixes are lettered instead of numbered: appendix letter (X), then paragraph numbers (P and p) as needed. A reference to an entire appendix uses only the appendix letter.

  X. P. p
  Appendix X

- Figures — Figure references guide the reader to figures mentioned in the List of Illustrations at the front of this manual. The reference begins with the letter F and gives the figure number: the number of the section (S) or letter of the appendix (X) that contains the figure, a dash (-), and a sequence number (n).

  FS-n
  FX-n

- Tables — These entries refer to tables found in this manual. They consist of the letter T followed by the table number.

  TS-n
  TX-n

- Manuals — These entries refer to other DX10 manuals in the set. The indexes in those books indicate the exact location of the information needed.

PROD OPER	Production Operation Manual, Volume II
APPL PROG	Application Programming Guide, Volume III
DEV OPER	Developmental Operation Manual, Volume IV
SYS PROG	Systems Programming Guide, Volume V
ERROR	Error Reporting and Recovery Manual, Volume VI

USER'S RESPONSE SHEET

Manual Title: Model 990 Computer DX10 Operating System Concepts and Facilities Manual, Volume I (946250-9701)

Manual Date: 15 April 1981 **Date of This Letter:** ____

User's Name: ____ **Telephone:** ____

Company: ____ **Office/Department:** ____

Street Address: ____

City/State/Zip Code: ____

Please list any discrepancy found in this manual by page, paragraph, figure, or table number in the following space. If there are any other suggestions that you wish to make, feel free to include them. Thank you.

Location in Manual **Comment/Suggestion**

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES (LOCATED ON REVERSE SIDE), TAPE AND MAIL

CUT ALONG LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7284 DALLAS, TX

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED
DIGITAL SYSTEMS GROUP

ATTN: TECHNICAL PUBLICATIONS
P.O. Box 2909 M/S 2146
Austin, Texas 78769

FOLD

Texas Instruments U.S. District Sales and Service Offices

(A complete listing of U.S. offices is available from the district office nearest your location)

California
831 S. Douglas Street
El Segundo, California 90245
(213) 973-2571

100 California Street
Suite 480
San Francisco, California 94111
(415) 781-9470

776 Palomar Avenue
P.O. Box 9064
Sunnyvale, California 94086
(408) 732-1840*

3186 Airway
Suite J
Costa Mesa, California 92626
(714) 540-7311

Colorado
9725 East Hampden Avenue
Suite 301
Denver, Colorado 80231
(303) 751-1780

Florida
1850 Lee Road
Suite 115
Winter Park, Florida 32789
(305) 644-3535

Georgia
3300 Northeast Expressway
Building 9
Atlanta, Georgia 30341
(404) 458-7791

Illinois
515 West Algonquin Road
Arlington Heights, Illinois 60005
(312) 640-2900
(800) 942-0609*

Massachusetts
504 Totten Pond Road
Waltham, Massachusetts 02154
(617) 890-7400

Michigan
24293 Telegraph Road
Southfield, Michigan 48034
(313) 353-0830
(800) 572-8740*

Minnesota
7625 Parklawn Avenue
Minneapolis, Minnesota 55435
(612) 830-1600

Missouri
2368 Schuetz
St. Louis, Missouri 63141
(314) 569-0801*

New Jersey
1245 Westfield Avenue
Clark, New Jersey 07066
(201) 574-9800

Ohio
4124 Linden Avenue
Dayton, Ohio 45432
(513) 258-3877

Pennsylvania
420 Rouser Road
Coraopolis, Pennsylvania 15108
(412) 771-8550

Texas
8001 Stemmons Expressway
P.O. Box 226080
M/S 3108
Dallas, Texas 75266
(214) 689-4460

13510 North Central Expressway
P.O. Box 225214
M/S 393
Dallas, Texas 75265
(214) 238-3881

9000 Southwest Freeway, Suite 400
Houston, Texas 77074
(713) 776-6577

8585 Commerce Drive, Suite 518
Houston, Texas 77036
(713) 776-6531
(713) 776-6553*

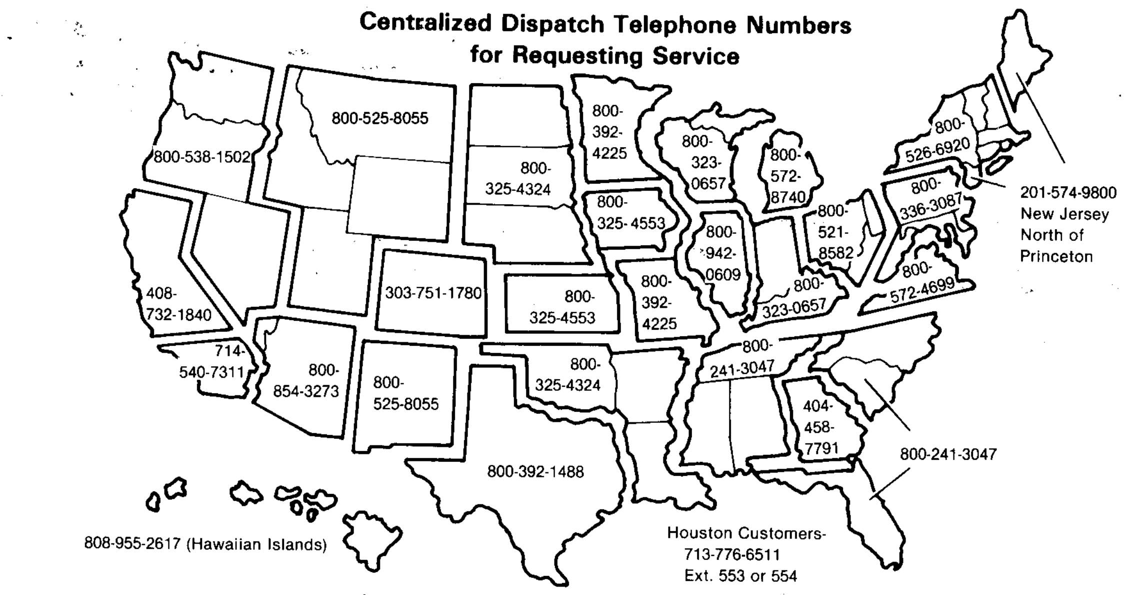
Virginia
1745 Jefferson Davis Highway
Crystal Square 4, Suite 600
Arlington, Virginia 22202
(703) 553-2200

Wisconsin
205 Bishops Way
Suite 214
Brookfield, Wisconsin 53005
(414) 784-1323

*Service telephone number

TI-CARE*

Centralized Dispatch Telephone Numbers for Requesting Service



Houston Customers-
713-776-6511
Ext. 553 or 554

Dallas Customers-
214-238-3881

Installation for Computer Systems
800-231-2807
713-937-1200 (Texas only, collect)

*Service mark of Texas Instruments

The **TI Customer Support Line** is available to answer our customers' complex technical questions. The extensive experience of a selected group of TI senior engineers and systems analysts is made available directly to our customers. The **TI Customer Support Line** telephone number is **(512) 250-7407.**

TEXAS INSTRUMENTS
INCORPORATED
DIGITAL SYSTEMS GROUP
POST OFFICE BOX 2909 AUSTIN, TEXAS