

**DX10 OPERATING SYSTEM**



# ***Concepts and Facilities***

Part No. 946250-9701 \*F  
January 1985

**Volume I**

# **TEXAS INSTRUMENTS**

---

---

---

---

# LIST OF EFFECTIVE PAGES

INSERT LATEST CHANGED PAGES AND DISCARD SUPERSEDED PAGES

Note: The changes in the text are indicated by a change number at the bottom of the page and a vertical bar in the outer margin of the changed page. A change number at the bottom of the page but no change bar indicates either a deletion or a page layout change.

DX10 Operating System Concepts and Facilities; Volume I (946250-9701)

Original Issue ..... August 1977  
 Revision ..... March 1978  
 Revision ..... December 1979  
 Revision ..... April 1981  
 Revision ..... September 1982  
 Revision ..... September 1983  
 Change 1 ..... January 1985

Total number of pages in this publication is 144 consisting of the following:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
Cover .....	1	ix .....	0	3-5/3-6 .....	1
Effective Pages .....	1	x .....	1	4-1 .....	1
Eff. Pages Cont. ....	1	xi - xii .....	0	4-2 .....	0
iii - iv .....	1	1-1 - 1-2 .....	0	4-3 - 4-6 .....	1
v .....	0	2-1 - 2-6 .....	0	5-1 .....	0
vi .....	1	3-1 - 3-2 .....	0	5-2 .....	1
vii .....	0	3-3 .....	1	5-3 - 5-4 .....	0
viii .....	1	3-4 .....	0	6-1 - 6-6 .....	0

The computers, as well as the programs that TI has created to use with them, are tools that can help people better manage the information used in their business; but tools—including TI computers—cannot replace sound judgment nor make the manager's business decisions.

Consequently, TI cannot warrant that its systems are suitable for any specific customer application. The manager must rely on judgment of what is best for his or her business.

---

# LIST OF EFFECTIVE PAGES

---

INSERT LATEST CHANGED PAGES AND DISCARD SUPERSEDED PAGES

Note: The changes in the text are indicated by a change number at the bottom of the page and a vertical bar in the outer margin of the changed page. A change number at the bottom of the page but no change bar indicates either a deletion or a page layout change.

DX10 Operating System Concepts and Facilities, Volume 1 (946250-9701)

Continued:

Continued:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
7-1 - 7-2	.....0	Glossary-1 -			
8-1 - 8-2	.....0	Glossary-18	.....0		
9-1 - 9-3	.....0	Index-1 - Index-2	.....1		
9-4	.....1	Index-3/Index-4	.....0		
10-1 - 10-10	.....0	Master Index-1	.....0		
11-1	.....1	Master Index-2	.....1		
11-2 - 11-21	.....0	Master Index-3 -			
11-22 - 11-23	.....1	Master Index-6	.....0		
11-24 - 11-26	.....0	User's Response	.....1		
12-1 - 12-8	.....0	Business Reply	.....1		
13-1 - 13-2	.....0	Inside Cover	.....1		
A-1 - A-14	.....0	Cover	.....1		

---

# DX10 Software Manuals

## DX10 Operating System Manuals

DX10 Operating System Concepts and Facilities (Volume I) 946250-9701	DX10 Operating System Application Programming Guide (Volume III) 946250-9703	DX10 Operating System Systems Programming Guide (Volume V) 946250-9705	DX10 Operating System Release 3.7 System Design Document 939153-9701
DX10 Operating System Operations Guide (Volume II) 946250-9702	DX10 Operating System Text Editor Manual (Volume IV) 946250-9704	DX10 Operating System Error Reporting and Recovery Manual (Volume VI) 946250-9706	Link Editor Reference Manual 949617-9701

## Communications Manuals

DX10 3270 Interactive Communications Software (ICS) User's Guide  
2250954-9701

DX10 Remote Terminal Subsystem (RTS) System Generation and Programmer's Reference Manual  
2272054-9701

DX10 Remote Terminal Subsystem (RTS) Operator's Guide  
2272055-9701

DX10 3780/2780 Emulator Release 4 User's Guide  
2302663-9701

## Language Manuals

990/99000 Assembly Language Reference Manual  
2270509-9701

DX10 COBOL Programmer's Guide  
2270521-9701

COBOL Reference Manual  
2270518-9701

DX10 FORTRAN-78 Programmer's Guide  
2268679-9701

FORTTRAN-78 ISA Extensions Manual  
2268696-9701

FORTTRAN-78 Reference Manual  
2268681-9701

Report Program Generator (RPG II) Programmer's Guide  
939524-9701

TI BASIC Reference Manual  
2308769-9701

DX10 TI Pascal Programmer's Guide  
2270528-9701

TI Pascal Reference Manual  
2270519-9701

TI Pascal Configuration Processor Tutorial  
2250098-9701

## Miscellaneous Software Manuals

Operator's Guide, Business System 300 (International)  
2533318-9701

Operator's Guide, Business System 300 (Domestic)  
2533318-9702

ROM Loader User's Guide  
2270534-9701

Operator's Guide, Business System 300A  
2240275-9701

## Productivity Tools Manuals

TIFORM Reference Manual  
2234391-9701

DX10 Query-990 User's Guide  
2250466-9701

Model 990 Computer DX Sort/Merge User's Guide  
946252-9701

DX10 TIPE Texas Instruments Page Editor Reference Manual  
2302656-9701

Data Dictionary User's Guide  
2276582-9701

DX10 COBOL Program Generator User's Manual  
2308956-9701

# DX10 Hardware Manuals

## Miscellaneous Hardware Manuals

Model 990 Computer Communications System Installation and Operation 945409-9701

Model 990 Computer PROM Programming Module Installation and Operation 945258-9701

Model 990 Computer TILINE Coupler User's Guide 226988-9701

990 CRU/TILINE Expansion Installation and Operation 2272075-9701

Model 990/10 Computer System Hardware Reference Manual 945417-9701

ROM Loader User's Guide 2270534-9701

## Hard-Copy Terminal Manuals

Model 990 Computer Model 733 ASR/KSR Data Terminal Installation and Operation 945259-9701

Model 990 Computer Model 743 KSR Data Terminal Installation and Operation 943462-9701

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 2265938-9701

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 820 KSR Terminal Operator's Manual 2208225-9701

Model 840 RO Printer Terminal Operator's Manual 2203665-9701

Model 840 RO Printer Business System Series 2532270-9701

## Display Terminal Manuals

Model 931 Video Display Terminal Installation and Operation 2229228-0001

Model 990 Computer Model 940 Electronic Video Terminal (EVT) Installation and Operation Manual 2250368-9701

Model 990 Computer Model 911 Video Display Terminal Installation and Operation 945423-9701

Model 990 Computer Model 913 CRT Display Terminal Installation and Operation 943457-9701

## Printer Manuals

Model 990 Computer Models 306 and 588 Line Printers Installation and Operation 945261-9701

Model 781 RO Terminal Operator's Manual 2265935-9701

Model 990 Computer Model 810 Printer Installation and Operation 939460-9701

Operator's Guide, Model 810 Printer Business System Series 2228256-9701

Model 820 KSR Terminal Operator's Manual 2208225-9701

Model 990 Computer Model 820 KSR Data Terminal Installation and Operation 2250454-9701

Model 990 Computer Model 840 RO Printer Installation and Operation Manual 2302695-9701

Operator's Guide, Model 840 RO Printer Business System Series 2533270-9701

Model 850 Printer User's Manual 2219890-0001

Model 990 Computer Model 1230 and 2260 Line Printers Installation and Operation 946256-9701

Model 990 Computer Model LP300 and LP600 Line Printers Installation and Operation Manual 2250364-9701

Models LP300 and LP600 Line Printers Installation and Operation (Business System Series) 2302643-9701

Model 990 Computer Model LQ45 Letter Quality Printer System Installation and Operation Manual 2268695-9701

Model LQ55 Letter Quality Printer Installation and Operation 2234382-9701

## Storage Device Manuals

Model CD1400 Disk System Installation and Operation Manual 2272081-9701

Rectilinear CD1400 Disk System Installation and Operation (Business System Series) 2311346-9701

Model 990 Computer DS10 Cartridge Disc System Installation and Operation 946261-9701

Model 990 Computer Model DS25/DS50 Disc Systems Installation and Operation 946231-9701

Model 990 Computer Moving Head Disc System Installation and Operation 945260-9701

WD5000/WD500A Mass Storage System Installation and Operation (Business System Series) 2302688-9701

Model 990 Computer Model DS80 Disk System Installation and Operation Manual 2302629-9701

Model 990 Computer Model DS200 Disc System Installation and Operation 949615-9701

Model DS300 Disk System Installation and Operation Manual 2302631-9701

Operator's Guide Model WD5000/WD500A Disk Unit (Business System Series) 2533269-9701

Operator's Guide Model FD1000 Flexible Disk System with International Chassis Installation and Operation 2250698-9701

Model 990 Computer Model 979A Magnetic Tape System Installation and Operation 946229-9701

Model MT1600 Magnetic Tape System Installation and Operation 2302642-9701

Model 990 Computer Model 804 Card Reader Installation and Operation 945262-9701

WD900/MT3200 General Description Manual 2234398-9701

# Preface

---

This manual provides general background information about the DX10 operating system and describes its features, concepts and facilities. It also contains a glossary and a master index.

This manual is one of a set of six volumes that describes the operational characteristics and features of DX10. In addition to the six volumes, several support manuals are available for DX10 functions. Also each language supported by DX10 has its own associated manuals.

Become acquainted with these volumes and related DX10 manuals as necessary to prepare and execute application programs under DX10. The following paragraphs each contain a brief comment regarding the content of each volume. (The full titles and part numbers of all manuals associated with the DX10 operating system are provided in the frontispiece.) The six volumes are as follows:

*Concepts and Facilities (Volume I)* includes features, concepts, and general background information describing the DX10 operating system. It also contains a master subject index to help you find the information you need.

The *Operations Guide (Volume II)* contains information on how to perform an initial program load (IPL start procedure) and how to log on and operate a terminal. Additionally, this manual contains an introduction to your interface with DX10, the System Command Interpreter (SCI), and includes a complete description of the SCI commands required to operate DX10. (The Text Editor and Link Editor commands are not included in Volume II but can be found in their respective manuals. Debugger commands are in Volume III.)

The *Application Programming Guide (Volume III)* contains information required by the application programmer to prepare, modify, and execute application programs on DX10. Much of the material is relevant to both high-level language programmers as well as assembly language programmers, since it concerns program structure, program operation, file structure, and file I/O. The SCI programming language is included, since it is a major part of constructing applications under DX10. Complete descriptions for nonprivileged SVCs and the DX10 Debugger are included for assembly language programs.

The *Text Editor Manual (Volume IV)* includes operating instructions, examples, and exercises for the interactive Text Editor provided on DX10. The SCI commands and error messages related to the Text Editor are included.

The *Systems Programming Guide (Volume V)* includes information required by the system programmer to maintain or extend a computer system running under DX10. The disk build procedure required for building your initial system disk, and the system generation procedure and troubleshooting guide required for system start-up are located in this manual. The manual also includes support of nonstandard devices and the privileged SVCs available on DX10.

The *Error Reporting and Recovery Manual (Volume VI)* describes each error message you can receive while operating DX10 and gives suggested procedures for recovery. It documents task errors, command errors, SVC errors, SCI errors, and I/O errors, including those from disk and magnetic tape. Also included are sections on system crash analysis and system troubleshooting.

#### 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 optional Sort/Merge, Data Base Management System (DBMS), and COBOL Program Generator (CPG) packages.

# Contents

Paragraph	Title	Page
<b>1 — General Description</b>		
1.1	The Disk Executive Operating System (DX10) .....	1-1
1.2	DX10 Capabilities .....	1-1
1.3	DX10 Features .....	1-2
<b>2 — Documentation Overview</b>		
2.1	General Introduction .....	2-1
2.2	Volume I — Concepts and Facilities .....	2-2
2.2.1	Concepts Portion .....	2-2
2.2.2	Facilities Portion .....	2-2
2.3	Volume II — Operations Guide .....	2-2
2.4	Volume III — Application Programmer's Guide .....	2-3
2.5	Volume IV — Text Editor Manual .....	2-3
2.6	Volume V — Systems Programming Guide .....	2-3
2.7	Volume VI — Error Reporting and Recovery Messages .....	2-4
<b>3 — Supported System Hardware</b>		
3.1	Hardware Features of the 990/10A, 990/12LR, and Business System Computers ...	3-1
3.2	Required Hardware .....	3-1
3.3	Supported Hardware .....	3-3
<b>4 — Supported System Software</b>		
4.1	General Information .....	4-1
4.2	Program Development Tools .....	4-2
4.2.1	Text Editor .....	4-2
4.2.2	Macro Assembler .....	4-2
4.2.3	Link Editor .....	4-2
4.2.4	Debugger .....	4-2
4.3	High Level Languages .....	4-3
4.3.1	COBOL .....	4-3
4.3.2	RPG II .....	4-3
4.3.3	FORTRAN .....	4-3
4.3.4	BASIC .....	4-3
4.3.5	Pascal .....	4-3



Paragraph	Title	Page
4.4	Productivity Aids . . . . .	4-3
4.4.1	Sort/Merge . . . . .	4-4
4.4.2	TIFORM . . . . .	4-4
4.4.3	TIPE . . . . .	4-4
4.4.4	COBOL Program Generator (CPG) . . . . .	4-4
4.5	Data Management Tools . . . . .	4-4
4.5.1	DBMS (Data Base Management System) . . . . .	4-4
4.5.2	Query . . . . .	4-4
4.5.3	DD (Data Dictionary) . . . . .	4-5
4.6	Advanced Communications . . . . .	4-6
4.6.1	3780/2780 Emulator . . . . .	4-6
4.6.2	3270 Interactive Communications Software (ICS) . . . . .	4-6

## 5 — System Command Interface

5.1	Introduction to SCI . . . . .	5-1
5.1.1	Foreground and Background . . . . .	5-1
5.1.2	Interactive Mode . . . . .	5-1
5.1.3	Batch Stream Mode . . . . .	5-2
5.1.4	Writing Your Own Command Procedures . . . . .	5-3
5.1.5	Synonyms . . . . .	5-3
5.1.6	Available Commands . . . . .	5-3

## 6 — File Structures and Features

6.1	General Information . . . . .	6-1
6.2	File Structures . . . . .	6-1
6.2.1	Sequential Files . . . . .	6-1
6.2.2	Relative Record Files . . . . .	6-1
6.2.3	Key Indexed Files (KIFs) . . . . .	6-2
6.2.3.1	Key Values . . . . .	6-2
6.2.3.2	File Stability . . . . .	6-2
6.3	File Features . . . . .	6-2
6.3.1	File Applicability to Languages . . . . .	6-3
6.3.2	Delete and Write Protection . . . . .	6-3
6.3.3	File Access Privileges . . . . .	6-3
6.3.4	Record Locking . . . . .	6-3
6.3.5	Temporary Files . . . . .	6-3
6.3.6	Blocked Files . . . . .	6-4
6.3.7	Deferred or Immediate Write . . . . .	6-4
6.3.8	Blank Suppression and Adjustment . . . . .	6-4
6.3.9	Expandable Files . . . . .	6-4
6.4	Directory and File Arrangement . . . . .	6-4

Paragraph	Title	Page
<b>7 — System Generation</b>		
7.1	General Information .....	7-1
7.1.1	Communicating the System Configuration to DX10 .....	7-1
7.1.2	Assembling and Linking the Generated System .....	7-2
7.1.3	Patching the Generated System .....	7-2
7.1.4	Testing the Generated System .....	7-2
7.1.5	Installing the Generated System .....	7-2
<b>8 — Error Control</b>		
8.1	General Information .....	8-1
8.2	Error Reporting .....	8-1
8.3	System Crashes .....	8-1
8.4	System Log .....	8-2
8.5	Memory Mapping .....	8-2
8.6	End-Action Routines .....	8-2
8.7	Error Prevention .....	8-2
<b>9 — Disk Management and Organization</b>		
9.1	General Information .....	9-1
9.2	Disk Management .....	9-1
9.3	Disk Volume Content and Attributes .....	9-2
9.3.1	System Overhead Files .....	9-2
9.3.2	System Files .....	9-2
9.3.2	User Files .....	9-2
<b>10 — Device and File Services</b>		
10.1	General Information .....	10-1
10.2	Access Names .....	10-1
10.2.1	Device Names .....	10-1
10.2.2	Volume Names .....	10-2
10.2.3	File Pathnames .....	10-2
10.3	Logical Unit Numbers (LUNOs) .....	10-3
10.3.1	Scope of LUNO Assignments .....	10-3
10.4	Device Orientation .....	10-4
10.5	Device and File Operations .....	10-4
10.5.1	Step 1 — Disk Preparation .....	10-4
10.5.2	Step 2 — File Creation .....	10-4
10.5.3	Step 3 — LUNO Assignment .....	10-6
10.5.4	Step 4 — Normal I/O Operations .....	10-6

Paragraph	Title	Page
10.5.5	Step 7 — LUNO Release .....	10-6
10.5.6	Step 8 — Temporary File Deletion .....	10-6
10.5.7	Step 9 — Unloading Disk Volume .....	10-6
10.6	Perform I/O Operation Supervisor Call .....	10-7
10.7	Extended Video Display Terminal Support .....	10-8

## 11 — Application Programming Environment

11.1	General Information .....	11-1
11.2	Program Development Tools .....	11-3
11.2.1	Interactive Text Editor .....	11-3
11.2.2	Macro Assembler .....	11-3
11.2.3	Link Editor .....	11-4
11.2.4	Interactive Debugger .....	11-7
11.3	DX10 Advanced Communications .....	11-8
11.3.1	Communications Hardware Equipment .....	11-8
11.3.1.1	CI402 .....	11-10
11.3.1.2	CI403 .....	11-10
11.3.1.3	CI404 .....	11-10
11.3.1.4	CI421 .....	11-12
11.3.1.5	CI422 .....	11-12
11.3.1.6	CP501 .....	11-12
11.3.1.7	CP502 .....	11-14
11.3.2	3780/2780 Emulators Communications Software .....	11-14
11.3.3	3270 Emulator Interactive Communications Software .....	11-15
11.4	High Level Programming Languages .....	11-17
11.4.1	FORTRAN .....	11-17
11.4.2	COBOL .....	11-19
11.4.3	Pascal .....	11-19
11.4.4	BASIC .....	11-21
11.4.5	RPG II .....	11-21
11.5	Productivity Aids .....	11-22
11.5.1	TIFORM .....	11-22
11.5.2	Texas Instruments Page Editor (TIPE) Utility .....	11-22
11.5.3	Sort/Merge Utility .....	11-23
11.5.4	COBOL Program Generator (CPG) .....	11-23
11.6	Data Management Tools .....	11-25
11.6.1	Data Base Management System (DBMS) .....	11-25
11.6.1.1	Features of DBMS .....	11-25
11.6.1.2	DBMS User Interface .....	11-25
11.6.2	Query .....	11-25
11.6.2.1	Query Environment .....	11-26
11.6.2.2	Query Language .....	11-26
11.6.3	DD .....	11-26

Paragraph	Title	Page
<b>12 — Program Management</b>		
12.1	General Information .....	12-1
12.2	Tasks and Programs .....	12-1
12.3	Shared Procedures and Replicated Tasks .....	12-2
12.4	Sharing Data Among Tasks .....	12-2
12.5	Overlays .....	12-2
12.6	Sample Application of Tasks and Procedures .....	12-3
12.7	Task Activation .....	12-3
12.8	Priority Scheduling .....	12-5
12.8.1	Priority Levels .....	12-5
12.8.2	Scheduling Operation .....	12-6
12.9	Program Files .....	12-7
12.10	Program Identification .....	12-7
12.11	Task Sentry .....	12-8
12.12	Supervisor Calls (SVCs) .....	12-8

### 13 — Memory Management

13.1	General Information .....	13-1
13.2	Dynamic Allocation .....	13-1
13.3	Roll-In/Roll-Out .....	13-2
13.4	Memory-Resident Tasks .....	13-2

### Appendixes

Appendix	Title	Page
A	Keycap Cross-Reference .....	A-1

### Glossary

### Index

### Master Index

## Illustrations

---

Figure	Title	Page
3-1	A Texas Instruments Computer Installation .....	3-2
5-1	The Standard SCI Menu .....	5-2
6-1	Example Directory Structure for EMPLOY01 .....	6-6
9-1	Physical Layout of a Disk Volume .....	9-3
9-2	Files and Directory Structure .....	9-4
10-1	Derivation of a File Pathname .....	10-2
10-2	Device and File Operation Flowchart .....	10-5
10-3	Model 931 Video Display Terminal .....	10-9
10-4	Model 911 Video Display Terminal .....	10-9
10-5	Business System Terminal .....	10-10
11-1	Commercial Application Environment .....	11-2
11-2	Example Assembly Program Listing .....	11-5
11-3	Partial Listing Produced by the Link Editor .....	11-7
11-4	Interactive Debugging .....	11-8
11-5	Sample Business System 600 or 800 Asynchronous Communications Configuration .....	11-9
11-6	Sample Business System 300 Communications Configuration .....	11-11
11-7	Sample Business System 600 or 800 Synchronous Communications Configuration .....	11-13
11-8	Typical Application of 3780/2780 Emulator in Distributed Processing Environment .....	11-15
11-9	IBM 3270 Information Display System Configuration and ICS Configuration .....	11-16
11-10	Sort/Merge Process Showing Printouts of Results at Each Step .....	11-24
12-1	Task/Procedure Structure .....	12-4

## Tables

---

Table	Title	Page
3-1	DX10 Hardware Devices .....	3-3
10-1	Device and File Operations Available Through I/O Supervisor Calls .....	10-7
12-1	DX10 General-Purpose Operating System Supervisor Calls .....	12-8

# General Description

---

## 1.1 THE DISK EXECUTIVE OPERATING SYSTEM (DX10)

DX10 is a general-purpose, multitasking operating system designed to operate with the Texas Instruments 990/10A, 990/12LR, and Business System Series 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. DX10 is also a multiterminal system capable of making each of several users appear to have exclusive control of the system.

DX10 is an international operating system designed to meet the commercial requirements of the United States, most European countries, and Japan. DX10 supports several models of video display terminals (VDTs), most of which permit users to enter, view, and process data in their own language.

## 1.2 DX10 CAPABILITIES

DX10 requires a basic hardware configuration, but allows additional members of an extensive group of peripherals to be included in the configuration. Section 3 describes the available optional devices. During system generation, you can configure DX10 to support peripheral devices that are not members of the 990 family and devices that require realtime support. This capability requires that you also provide software control for these devices.

You can communicate with DX10 easily through the System Command Interpreter (SCI). SCI is designed to provide simple, convenient interaction between the user and DX10 in a conversational format. Through SCI you have access to complete control of DX10. SCI is flexible in its mode of communication. While SCI is convenient for interactive communication through a data terminal, SCI can be accessed in batch mode as well. Section 5 provides an in-depth discussion of SCI.

DX10 is capable of extensive file management. The built-in file structures include key indexed files, relative record files, and sequential files. A group of file control utilities exists for copying and modifying files, and controlling file parameters. Section 6 describes the file types and features.

### 1.3 DX10 FEATURES

DX10 offers a number of features that provide convenient use of your system's capabilities.

- Easy system generation for systems with custom device configurations. With proper preparation, peripheral devices that are not part of the 990 computer family can be interfaced through DX10.
- A macro assembler for translating assembly language programs into executable machine code.
- A text editor for entering source code or data into accessible files.
- Support of high-level languages, including FORTRAN, COBOL, Pascal, RPG II, and BASIC.
- A link editor and extended debugging facilities are provided to further support program development.

# Documentation Overview

---

## 2.1 GENERAL INTRODUCTION

The core of DX10 documentation is a six volume set consisting of these manuals:

- Volume I — Concepts and Facilities
- Volume II — Operations Guide
- Volume III — Application Programmer's Guide
- Volume IV — Text Editor Manual
- Volume V — Systems Programming Guide
- Volume VI — Error Reporting and Recovery Messages

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. You should consult all six manuals to become thoroughly familiar with all facets and capabilities of the operating system.

### NOTE

The names of keys in these manuals are generic key names. In some cases, the names on the key caps of the terminals match the generic key names, but in many cases they do not. Each manual in the DX10 set contains a table in Appendix A of key equivalents to identify the specific keys on the terminal you are using.

Other manuals are available in addition to the DX10 set. A user's guide to the Link Editor is provided as are user's guides and reference manuals for each available programming language. The preface to this volume includes an extensive list of related software and hardware manuals.



## **2.2 VOLUME I — CONCEPTS AND FACILITIES**

This volume provides general background information about the DX10 Operating System. To address readers with all levels of computer experience, Volume I consists of two parts: Concepts — a general introduction for readers of all levels; and Facilities — a more technically oriented introduction to DX10's features.

Volume I also contains a DX10 glossary, a word index for this volume only, and the master subject index for all six volumes.

### **2.2.1 Concepts Portion**

This first part of Volume I presents material that every owner or user of DX10 should know. Discussions in this part of the manual are general and frequently include references to more detailed discussions in other manuals.

Sections 3 and 4 describe the optional hardware and software that DX10 supports. Section 5 introduces you to the System Command Interpreter (SCI), the interactive interface between you and DX10. Section 6 discusses the features of the three file types available under DX10. Section 7 reports how, with DX10's flexible System Generation utility, you can generate a system image to match your computer's physical hardware configuration exactly.

### **2.2.2 Facilities Portion**

The sections in the Facilities half of Volume I, while still introductory material, have a more technical point of view and are concerned with the internal functions of DX10.

Section 8 of this manual discusses the error control features of DX10. Section 9 discusses the organization and management of disk storage resources. Section 10 presents a general overview of device and file services. Section 11, titled Application Programming Environment, discusses the various program development tools that DX10 provides and the optional software packages that DX10 supports. Section 12 describes the basic considerations of program management under DX10. Section 13 contains a discussion of DX10 memory management techniques.

## **2.3 VOLUME II — OPERATIONS GUIDE**

This volume supplies the information needed to operate a DX10-based system in a production environment. Volume II introduces you to the video display terminals (VDTs) supported by DX10. It introduces other data terminals and describes the method of operating DX10 from a terminal. Drawings that show the layout of the keyboards of each type of terminal are included. Volume II also describes system initialization from hardware power-up through Initial Program Load (IPL) and initialization of system internal values. Also included is a section on maintaining your system, including backup and recovery methods.

This volume also describes the functions and use of the System Command Interpreter (SCI). Following this introduction is a comprehensive description of the SCI commands used to direct the operating system.

## 2.4 VOLUME III — APPLICATION PROGRAMMER'S GUIDE

DX10 supports programming languages ranging from assembly code to high-level languages such as FORTRAN and Pascal. Volume III is directed to the needs of applications programmers who use these languages. This volume lays the groundwork for more sophisticated concepts with discussions of DX10 program management (including program mapping, program segmentation, priority scheduling, and code and data sharing), disk management and file support.

Other sections deal with manipulating assembly language programs with DX10, that is, installing, linking, and modifying assembly language programs. Section 8 provides instructions for using the Debugger to debug programs written in both assembly language and high-level languages.

Section 9 of this volume covers the SCI programming language. Knowledge of this material enables you to write your own command procedures and use SCI to its fullest extent to complement your applications and system programming requirements.

Sections 10 through 14 provide detailed information about Supervisor Calls (SVCs). SVCs are requests from a task for operating system services in the following functional groups:

- Device and file I/O
- Program control
- Memory control
- File Utilities

In addition to comprehensive descriptions of the available SVCs, these sections explain how to employ SVCs in assembly language programs.

## 2.5 VOLUME IV — TEXT EDITOR MANUAL

The Text Editor that DX10 provides is an interactive tool for modifying text files such as program source code and manuscripts. Volume IV is a guide to productive use of the Text Editor. This volume describes the set of SCI commands that direct the Text Editor and the control keys that direct entry into the edited text. Also included are examples and exercises using the Text Editor at a VDT and at a hard-copy data terminal.

## 2.6 VOLUME V — SYSTEMS PROGRAMMING GUIDE

Volume V discusses those topics concerning the maintenance and extension of DX10. This volume provides information on the following topics:

- Building and maintaining your system disk
- System Generation
- Privileged SVCs

- User-written SVC processors
- User-written Device Service Routines (DSRs)
- System DSRs
- System compatibility with earlier releases of DX10

Section 2 describes how to build your system disk from DX10 software provided on either disk or magnetic tape. Also included are instructions on making a backup copy of the system disk in case the primary copy is destroyed. This section contains other paragraphs concerning system files and the methods available to you for modifying their contents and, thus, the performance of your system.

Section 3 describes the system generation (sysgen) process, with instructions and examples. This section provides tips for optimizing your system response by selecting parameter values and options that suit your needs and use system resources thriftily. Section 4 provides Sysgen troubleshooting techniques in case you encounter difficulty in generating your system.

Although the standard DX10 package is usually sufficient in itself, you are able to install your own SVCs and DSRs to perform non-standard functions. Sections 5 and 6 teach you how to write your own SVCs and DSRs. Section 7 discusses privileged SVCs and defines the benefits and hazards of accessing them.

If you are upgrading a computer system using an early release of DX10 (earlier than release 3.0), you must convert your data files to the current format. Section 8 of this volume discusses this requirement and provides instructions for making the conversion.

## **2.7 VOLUME VI — ERROR REPORTING AND RECOVERY MESSAGES**

Volume VI functions in two ways: as a reference manual to error messages, and as a troubleshooting guide to identify and rectify error conditions. Comprehensive tables provide cross-references between error codes, the error conditions that they indicate, and appropriate recovery action.

The following list presents the major topics covered in Volume VI.

- System loader errors
- SCI, command, Debugger, and Text Editor errors
- The system log

- Supervisor call errors
- System crashes
  - System crash codes
  - Instructions for the XANAL utility to study memory at the time of a crash
  - A troubleshooting guide that gives a step-by-step procedure for dealing with a system crash.



# Supported System Hardware

---

## 3.1 HARDWARE FEATURES OF THE 990/10A, 990/12LR, AND BUSINESS SYSTEM COMPUTERS

The 990/10A, 990/12LR, and Business System Series computer systems are high-speed, flexible, and powerful minicomputers of the Texas Instruments Model 990 computer family with the capability of handling a wide range of computer applications at low cost. The 990 computer family includes a wide variety of compatible terminals, printers, mass storage devices, and other peripheral devices to suit the needs and convenience of any computer user. The 990 systems allow you to increase the size of your system memory using a memory mapping feature that enables you to add more memory to a maximum capacity of two million bytes.

## 3.2 REQUIRED HARDWARE

DX10 requires certain minimum hardware consisting of the following items:

- The 990/10A, 990/12LR, or Business System CPU with the mapping option and a minimum random access memory of 128K bytes. It is recommended that at least 256K bytes of random access memory be used.
- A system disk with at least 4.7 megabytes of memory storage. This disk contains the DX10 software.
- A video display terminal (VDT) for communicating with the DX10 operating system.
- A means for disk backup — either a magnetic tape drive, a second disk drive, a flexible diskette drive, or a cartridge tape drive.

Additional memory beyond 128K bytes is strongly recommended and is required to support additional terminals.

Figure 3-1 shows a typical DX10 hardware installation.



2284757

Figure 3-1. A Texas Instruments Computer Installation

### 3.3 SUPPORTED HARDWARE

DX10 supports the full line of 990 hardware. Table 3-1 describes the mass storage devices and Input/Output terminals available to users of DX10.

**Table 3-1. DX10 Hardware Devices**

Device Type	Model Number	Description
Disk Drives	DS80	Moving-head disk drive with five-platter removable disk pack; provides 67 megabytes storage.
	DS300	Moving-head disk drive with 10-platter removable disk pack; provides 250 megabytes storage.
	CD1400/32	Moving-head disk drive with two platters, one fixed and one removable. Each platter provides 13 megabytes storage.
	CD1400/96	Moving-head disk drive with fixed and removable units. Fixed unit contains three platters that provide 67 megabytes storage. Removable unit is one platter with 13 megabytes storage.
	WD500A	5¼-inch Winchester disk systems with FD1000 backup. Two configurations are available: one fixed disk with 5 megabytes of formatted storage or two fixed disks with 17 megabytes.
	WD800/18 WD800/43	8-inch Winchester disk systems with cartridge tape backup. The WD800/18 has 18.5 megabytes of formatted storage, and the WD800/43 has 43.2 megabytes of formatted storage.
	WD800A/38 WD800A/69 WD800A/114	5¼-inch Winchester disk systems with cartridge tape backup. The WD800A/38 has 38.5 megabytes of formatted storage, the WD800A/69 has 69.5 megabytes of formatted storage, and the WD800A/114 has 114.6 megabytes of formatted storage.
	WD900/138 WD900/425	9-inch Winchester disk systems available with 138 or 425 megabytes of formatted storage.
Data Terminals	Business System Terminal with Keyboard	Combines a customized computer/cabinet/chassis and associated ports for the Business System 300 Series of computers. The Business System 300 supports Winchester-type mass storage devices, optional line printers, and data terminals. The video display presents 80 characters per line, 24 lines per screen (1920 characters). A 25th line provides selectable information display. The Business System Terminal supports a 128-displayable-ASCII-character set including true character descenders for lowercase. International versions are available for use in the following countries: Denmark, England, Finland, France, Germany, Japan, Norway, Sweden, and Spanish-speaking countries.



Table 3-1. DX10 Hardware Devices (Continued)

Device Type	Model Number	Description
	911 VDT	Video display terminal (VDT), displays 80 characters per line 24 lines per screen (1920 characters). The 911 VDT supports uppercase and lowercase characters and displays them in high or low intensity; control characters are also supported. The 911 VDT is available in international versions, supporting the local character set in both uppercase and lowercase. International versions are available for use in the following countries: Denmark, England, Finland, France, Germany, Japan, Norway, Sweden, Switzerland, Arab countries, and Spanish-speaking countries.
	931 VDT	VDT, displays 80 characters per line, 24 lines per screen (1920 characters). A 25th line is reserved for status information and diagnostic tests. The 931 VDT supports all 128 characters of the ASCII set. This includes 96 displayable alphanumeric characters and 32 special line drawing characters. Alphabetic characters are displayable in both uppercase and lowercase, with true descenders for lowercase. The 931 VDT also supports reverse images, true underlining, blinking characters, and high or low intensity. International versions are available for use in the following countries: Denmark, England, Finland, France, Germany, Norway, Sweden, Switzerland, and Spanish-speaking countries.
	Silent 700 Terminals	Portable keyboard send/receive hard-copy data terminals using a thermal dot-matrix printhead. Terminals support ASCII standard (64 printable characters) or optional (95 characters) keyboards. Standard keyboard includes numeric keypad. Some models include an acoustic coupler for remote capabilities.
Printers	810	The Model 810 printer with a 9 x 7 dot-matrix printhead prints 132 columns bidirectionally at 150 cps. Vertical forms control is supported. Optionally, the printer provides a full ASCII character set (both uppercase and lowercase characters) and vertical forms control.
	850	The Model 850 is a desktop-size, light-duty printer. It has a 9 x 9 dot-matrix printhead and prints at 150 cps with a maximum line width of 8 inches. It has true character descenders for lowercase.
	855	The Model 855 printer is a desktop-size printer with two modes of operation. The fast draft mode prints bidirectionally at 150 cps, with a 9 x 9 dot-matrix printhead. The correspondence mode (near letter quality) makes two passes per line at 35 cps, with a 32 x 18 dot-matrix printhead. Both modes have a maximum line width of 8 inches. The Model 855 uses interchangeable plug-in font modules.

Table 3-1. DX10 Hardware Devices (Continued)

Device Type	Model Number	Description
	LP300	The LP300 printer is a medium-speed line printer that prints 132-character lines at a rate of 300 lines per minute.
	LP600	The LP600 line printer is a high-speed line printer that prints 132-character lines at up to 600 lines per minute.
	LQ45	The LQ45 printer is a letter-quality printer using a daisywheel printing element to print at speeds up to 45 characters per second. The printer supports a full 96-character printwheel.
Other I/O Devices	MT1600 Magnetic Tape	Serial-access 9-track magnetic tape transport. Standard recording density of 800 bits per inch (bpi) (NRZI); optionally supports 1600 bpi (phase encoded).
	MT3200 Magnetic Tape	Streaming cache, serial-access, 9-track magnetic tape transport. Standard recording density of 1600 or 3200 bpi (phase-encoded).



# Supported System Software

---

## 4.1 GENERAL INFORMATION

A set of four program development tools is provided as an integral part of DX10.

- Text Editor
- Macro Assembler
- Link Editor
- Debugger

In addition to the intrinsic capabilities of DX10, you can purchase several powerful software packages to expand the scope of your computing power. DX10 supports the following groups of software packages:

- High-level languages
  - COBOL
  - RPG II
  - FORTRAN
  - BASIC
  - Pascal
- Productivity aids
  - Sort/Merge
  - TIFORM
  - TIPE
  - CPG
- Data Management
  - Query
  - DD (Data Dictionary)
  - DBMS (Data Base Management System)

- Advanced communication software packages
  - 3780/2780 Emulator
  - 3270 Interactive Communications Software

## 4.2 PROGRAM DEVELOPMENT TOOLS

DX10 includes the following four major program development tools:

- Text Editor
- Macro Assembler
- Link Editor
- Debugger

### 4.2.1 Text Editor

DX10 provides a powerful, interactive, full-screen text editor. Edit operations allow modification, insertion, and deletion of entire records or of character strings within records. You can direct edit operations through the keyboard of your data terminal using either special function edit keys or SCI commands. *DX10 Text Editor Manual*, Volume IV, is devoted solely to explanation of the Text Editor.

### 4.2.2 Macro Assembler

The DX10 Macro Assembler is the most powerful member of the 990 family assemblers. In addition to accepting the standard 990 assembly language instructions, the Macro Assembler is extended to include a macro facility, support for FORTRAN common segments, and conditional assembly. The macro facility provides character string manipulation, access to binary values in the symbol table, and support of macro definition libraries. A sequence of assembly language statements may be conditionally processed depending on the value of an assembler arithmetic or logical expression. *990/10 and 990/12 Assembly Language Reference Manual* contains a more extensive report concerning the features of the macro assembler.

### 4.2.3 Link Editor

The DX10 Link Editor accepts relocatable object code generated by the assembler or language compilers and combines the individual modules into a linked and loadable module. During this process, the Link Editor resolves references between modules to their correct values. You can direct the output of the Link Editor to an installable object file or directly into memory image form in a program file or system image file. Refer to the *Link Editor Reference Manual* for a more extensive discussion of the Link Editor.

### 4.2.4 Debugger

The Debugger is an interactive utility which aids you in diagnosing and correcting errors in programs under development. The Debugger provides for controlled execution of a program by allowing you to set breakpoints for the Debugger to begin and end program execution. The Debugger also provides commands by which you can monitor or modify the arithmetic unit registers, workspace registers, and memory. *Application Programmer's Guide*, Volume III, contains a detailed description of the Debugger functions and commands.

## 4.3 HIGH LEVEL LANGUAGES

DX10 supports sophisticated implementations of the most frequently used high-level languages. COBOL and RPG II are suitable for business applications. FORTRAN is a popular scientific language. BASIC is used for interactive applications and Pascal is well suited for development of either type of software.

You can find a more technical discussion of these high-level languages in Section 11 of this manual and in the *Application Programmer's Guide (Volume III)*.

### 4.3.1 COBOL

COBOL is the preferred programming language for commercial data processing environments. Using program statements with familiar business-like terminology, COBOL programs can perform data capture, data manipulation, data base management, file handling, computation, and report generation. The COBOL compiler supported by DX10 significantly exceeds the Level 1 requirements of ANSI Standard X3.23-1974. Documentation for COBOL in the DX10 environment is available in the *DX10 COBOL Programmer's Guide*.

### 4.3.2 RPG II

RPG II is an easy-to-use high level language for business data processing. RPG II is especially suited for applications requiring file maintenance or report generation. More information about RPG II is available in the *Report Program Generator (RPG II) Programmer's Guide*.

### 4.3.3 FORTRAN

FORTRAN is a high level language widely used for scientific and numeric problem solving. The TI FORTRAN-78 compiler conforms to and exceeds the ANSI standard X3.9-1978. Numerous extensions are provided for compatibility with the previous FORTRAN ANSI standard, X3.9-1966. The *DX10 FORTRAN-78 Programmer's Guide* contains extensive information about FORTRAN-78 in the DX10 environment.

### 4.3.4 BASIC

DX10 BASIC includes the popular features of ANSI standard BASIC and supplies valuable extensions as well. These extensions include integer arithmetic type, expanded string handling, calling ability, virtual arrays, and subprograms. DX10 BASIC commands and utilities simplify and speed program development in both scientific and business environments. Refer to the *TI BASIC Reference Manual* for more information about BASIC programming language.

### 4.3.5 Pascal

Pascal is a block-oriented procedural language that is particularly useful for systems programming because of its bit manipulation capabilities, recursive routines, self-documenting block structure, and efficient compilation and execution. For a more detailed discussion of Pascal in a DX10 environment, refer to the *TI Pascal User's Guide*.

## 4.4 PRODUCTIVITY AIDS

Sort/Merge, TIFORM, TIPE, and CPG are useful, time-saving data manipulation tools. A more technical discussion of these aids is included in Section 11.

#### 4.4.1 Sort/Merge

Sort/Merge is a fast, convenient utility for handling file sorting and merging operations. The comprehensive DX10 Sort/Merge package may be accessed in several ways. SCI commands access Sort/Merge interactively or in batch mode. BASIC, COBOL, FORTRAN and RPG II programs may interface with Sort/Merge by using the CALL statement. Refer to the *Sort/Merge User's Guide*.

#### 4.4.2 TIFORM

TIFORM is a dynamic software utility used for interactively managing forms, editing data for accuracy, and displaying stored data on a video display screen. To reduce costly program development time, TIFORM allows you to design and test forms interactively on a video display terminal rather than having to write a program. TIFORM may be accessed by COBOL, FORTRAN, and Pascal applications. The *TIFORM Reference Manual* contains more material relating to the use of TIFORM.

#### 4.4.3 TIPE

TIPE is a word-processing package for DX10 users. The TI Page Editor (TIPE) includes multi-terminal word processing functions for efficient creating, editing, and printing of letters and documents. Refer to the *TIPE User's Guide* for a more extensive discussion of TIPE.

#### 4.4.4 COBOL Program Generator (CPG)

The CPG is an easy-to-use programming system that generates complete, error-free COBOL source code. CPG generates both interactive and batch programs for business applications, such as data file creation and update, or generates reports from data files. Refer to the *DX10 COBOL Program Generator User's Guide* for a complete discussion of CPG.

### 4.5 DATA MANAGEMENT TOOLS

DBMS, DD, and Query facilitate the management and retrieval of data. See Section 11 for a more technical discussion of these data management tools.

#### 4.5.1 DBMS (Data Base Management System)

DBMS is a data base management system that lets you control your information assets in an organized fashion and retrieve information upon demand. You can modify your data base quickly and economically because there are no unnecessary costs or delays due to program rewrites. Information stored by DBMS can be accessed directly through Query or through COBOL, FORTRAN, or Pascal programs. In addition, you can protect sensitive information by using security passwords. For more information about DBMS, refer to *DX10 Data Base Management System Programmer's Guide* and *DX10 Data Base Management System Administrator User's Guide*.

#### 4.5.2 Query

Query is an interactive inquiry language designed for accessing DBMS data bases to generate reports on demand and to update DBMS files in a quick, cost-efficient manner. Not having to write a formal program to access data means that you spend less time accessing your data and more time making decisions. Query has optional step-by-step facilities to guide inexperienced users through inquiry procedures. The *Query User's Guide* provides more information about Query.

### **4.5.3 DD (Data Dictionary)**

DD is a data dictionary system that allows central definition and control of your data assets. Centralized control permits enforcement of data standards, a clearer view of the impact of proposed changes to a data base, and the reduction of data redundancy. DD supports the definition of data base structures in addition to conventional files (sequential, relative record, and key indexed files). Additionally, DD provides access to conventional files using English-like Query statements. Utilities are provided to convert COBOL record definitions to data dictionary definitions and vice versa. You can find more information about DD in the *Data Dictionary User's Guide*.

## **4.6 ADVANCED COMMUNICATIONS**

DX10 supports several software packages that allow your computer system to communicate with other computer systems, both Texas Instruments models and those of other manufacturers. These packages allow you to transfer data from one point to another quickly and easily, often using regular telephone lines. The following paragraphs discuss the software packages available in this group. A more technical discussion of these packages can be found in Section 11.

### **4.6.1 3780/2780 Emulator**

The 3780/2780 Emulator allows 990 computers to communicate from remote locations with IBM host computers or other computers equipped with the 3780/2780 Emulator. The 3780/2780 package can work unattended at night when phone rates are low to distribute data to other computer systems in your network. More information about the 3780/2780 Emulator can be found in the *DX 3780/2780 Emulator User's Guide*

### **4.6.2 3270 Interactive Communications Software (ICS)**

The 3270 ICS lets your 990 computer communicate interactively with an IBM host computer. The 3270 ICS can control as many as sixteen video display terminals and printers. You can also use 3270 ICS with COBOL, FORTRAN, and Pascal. The *DX10 3270 Interactive Communication Software (ICS) User's Guide* contains full documentation about the 3270 ICS.





# System Command Interface

---

## 5.1 INTRODUCTION TO SCI

You communicate with the DX10 operating system through the System Command Interpreter (SCI). You can direct SCI by entering any of over 200 brief SCI commands through the keyboard of an interactive data terminal. Communication with SCI is in a conversational manner; that is, if SCI needs more information before executing your command, it displays questioning messages or “prompts” on your VDT screen (or printout paper if you are communicating through a hard-copy data terminal). You respond to the prompts by entering the required information, called “parameters,” at your terminal.

SCI can reply to you through DX10-supported interactive devices including VDTs and hard copy data terminals. SCI operations on a VDT allow all parameter values to appear at once. When using a hard copy data terminal, SCI prompts you for parameter entries one line at a time.

You can direct SCI to process your commands in background, which allows you to issue commands to SCI from a batch command stream as well as to continue issuing commands directly from your terminal.

SCI gives you control over DX10 while requiring minimal effort from you. With a short SCI command, you can perform file operations requiring complex functions from DX10. You can add to the library of SCI commands by writing your own SCI command procedures. The SCI programming language is discussed in Volume III.

The remainder of this section discusses these SCI concepts briefly. For a full evaluation of SCI's power and flexibility, refer to Volume II.

### 5.1.1 Foreground and Background

An SCI command procedure can execute in foreground or background. At any one time, a terminal can execute two tasks, one foreground task and one background task. When you issue a command procedure that executes as a foreground task, SCI suspends its activity at your terminal and enters a waiting state until the task completes. No other activity can occur at that terminal until the foreground task finishes, allowing the foreground task to directly access your terminal if necessary. A command procedure executing as a background task executes independently of your terminal. You can issue an SCI command in foreground while the background task is executing.

### 5.1.2 Interactive Mode

Communication with SCI can be in one of two modes: interactive and batch. In the interactive mode, you and SCI communicate back and forth in foreground. Typically, you enter a command and SCI activates the appropriate command procedure, which executes in foreground mode. If the command procedure requires parameters in order to execute, SCI requests the information from you by sending prompts to your terminal. When the command procedure completes execution, SCI reactivates itself and you can issue another command.

When SCI is not processing a command, it displays a menu of commands on your data terminal screen. The standard DX10 menu is depicted in Figure 5-1. The DX10 menu presents a list of command groups. If you enter the name of one of the command groups, SCI displays a secondary menu (Each command group leads to a unique secondary menu.). This new menu presents a list of sub-groups of commands. Each level of the hierarchy of menus is more narrowly defined than the preceding level. The lower-most levels consist of lists of commands rather than command groups. With this system of menus, the new user of SCI can easily and quickly access a list of commands that are functionally related.

### 5.1.3 Batch Stream Mode

Commands issued from a batch command stream execute in background mode. To issue a command in a batch stream, you must enter a command procedure (in batch command stream format specified in Volume II) in a sequential file and initiate its execution with the Execute Batch (XB) command. Since a batch stream command procedure executes in background, it cannot interact with your terminal; consequently, all parameters must be supplied within the batch stream.

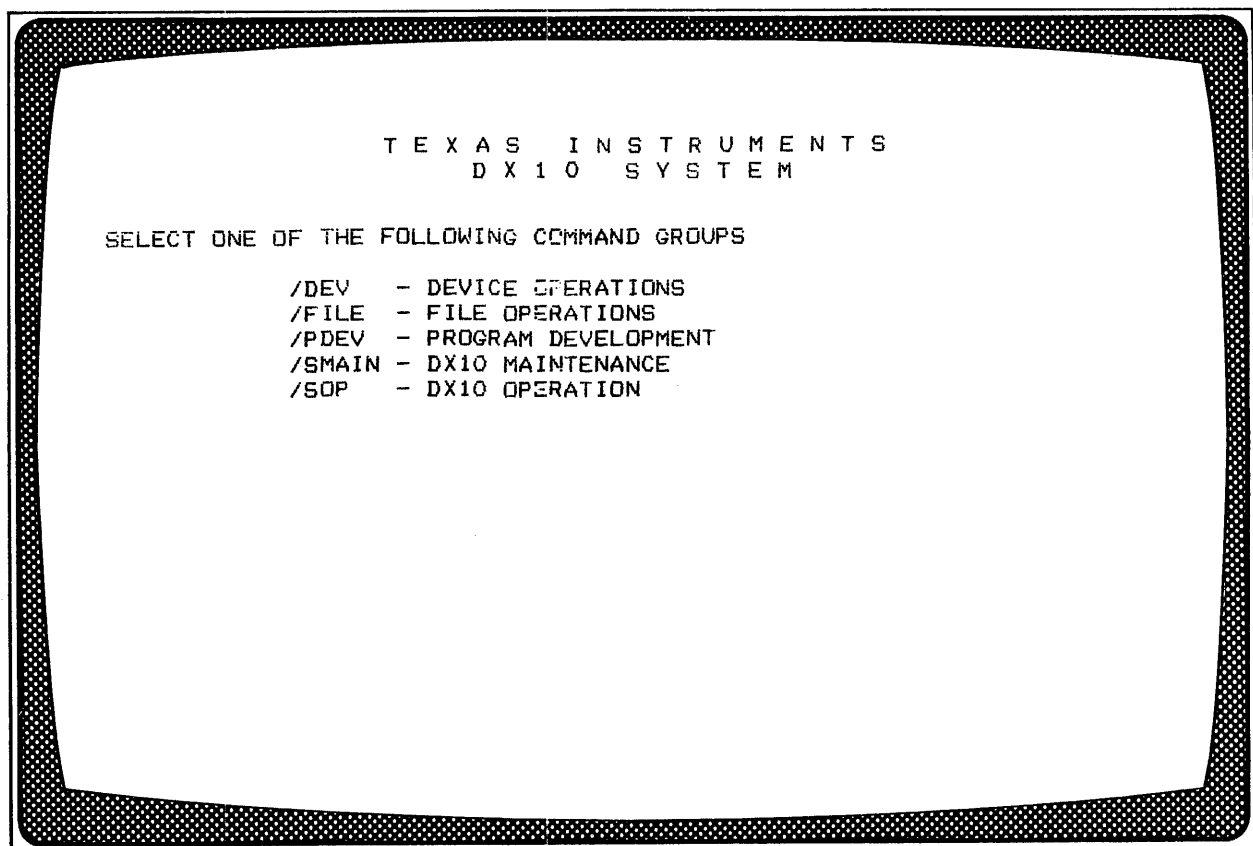


Figure 5-1. The Standard SCI Menu

#### 5.1.4 Writing Your Own Command Procedures

SCI is designed so that you can write your own command procedures. When you issue a command, SCI processes the contents of a file of the same name in the command library (the system directory, `.$$PROC`). This file contains the appropriate command procedure, a set of instructions written in the SCI command language. For instance, the SF (for Show File) command directs SCI to process the command procedure in the file `.$$PROC.SF`. With this SCI command language, you can write command procedures suited to your own needs, or rewrite existing command procedures. You can install new commands in either the system command library or in one or several of your own command libraries. For a description of the SCI command language and instructions on writing your own commands and command procedures, refer to Volume II.

#### 5.1.5 Synonyms

You can assign SCI variables called synonyms. A synonym is a relatively short string of alphanumeric characters that substitutes for a longer string. When you enter a synonym as a response to an SCI prompt (for instance, as a file pathname), SCI translates the synonym into its substituted value. SCI maintains a separate list of synonyms for each user. Each user can assign several synonyms and then use those synonyms without interfering with the synonym list of other users.

#### 5.1.6 Available Commands

DX10 has a comprehensive set of SCI commands that perform various utility operations. Many other commands are supplied for programmers to use as they develop applications under DX10. Refer to Volume II for extensive documentation of SCI commands. The following is a list of the activities served by one or more SCI commands.

- Log on and log off
- Set and display the time and date
- Analyze, initialize, install, and unload disk volumes
- Restore, backup, and copy disk directories
- Create and delete directories and files
- Support synonyms
- Allow file alias name(s)
- Change file names and protection
- View and list directories and files
- Copy files
- Assign, position, and release logical units
- Display I/O status
- Display task status

- Activate and control programs
- Activate and monitor batch status
- Maintain terminal status
- Install and delete programs
- Activate the system log
- Debug programs, including activities such as the following:
  - Set breakpoints
  - Dump or display memory
  - Perform decimal/hexadecimal arithmetic
  - Trace program execution interactively
- Control Text Editor
- Activate high-level language support
- Activate Link Editor
- Activate productivity aids
- Activate data management software
- Activate communication software

# File Structures and Features

---

## 6.1 GENERAL INFORMATION

Under DX10, data is stored on disk memory in a formal structure called a file. Within the general category of files, there are several particular types of files. Furthermore, an individual file can possess optional characteristics and capabilities to enhance its usefulness and efficiency. This section is divided into three parts: a discussion of the three major file types, a description of the featured characteristics of files, and an explanation of the hierarchy of files and directories in a volume and the pathnames that identify files. You can find more detailed information about using files in Volume III.

## 6.2 FILE STRUCTURES

DX10 supports three major file types: sequential, relative record, and key indexed. The following paragraphs describe these file types.

### 6.2.1 Sequential Files

Sequential files are useful for recording variable length data records in the same order in which they were received. Similarly, records must be read back in the same order in which they were recorded. The design of sequential files lends them to uses which require the rapid input and output of textual data, such as printing a report. Random access to sequential files is impractical since, to reach a given random record, all 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.

Several programs can read a sequential file concurrently at different positions in the file. However, only one program can write to a sequential file at a time. The current reading position of one program is retained while the file is logically assigned, even though the file is closed and reopened by other programs.

You can modify the content of textual data, such as a letter or program source code, in a sequential file using the Text Editor. When using the Text Editor, you can add and delete text in records of a sequential file without the constraint of sequential access. That is, you can freely move the cursor of your VDT about on a copy of a sequential file and modify the content of the file a letter at a time or even several records at a time. Refer to Volume IV for more information on the Text Editor.

### 6.2.2 Relative Record Files

A relative record file is a file in which all logical records have a fixed record length and each record can be randomly accessed by its unique record number. Relative record files are designed for rapid retrieval of a single record from the file. Records are accessed by supplying DX10 with the record number within the file. Such files are useful when the nature of the data lends itself to computation of a record number. Records can be accessed randomly by record number or DX10 can increment the caller's record number after each read or write so that sequential access is permitted.

### 6.2.3 Key Indexed Files (KIFs)

The most sophisticated file type supported by DX10 is the key indexed file (KIF). When accessing KIFs, programs access individual records by providing DX10 with any one of up to 14 keys by which the data is known. A key (in a KIF) is an identifying string of up to 100 characters. For example, with this file type you can construct a file of employee information so that the data record for any given employee is accessed by supplying the employee name, employee number, social security number, or any other designated key. Except for the primary 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. You must select one of the keys to be the primary key. All other keys are known as secondary keys. All records must include the primary key, but secondary keys are optional in any given record within the file.

**6.2.3.1 Key Values.** Key values for both primary and secondary keys are kept in indexes within the KIF. These indexes are 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 of a KIF, you can add, delete, or change a secondary key value if allowed by the attributes selected for that key. A primary key cannot be modified.

When you create a KIF, you assign the attributes of duplicability and modifiability to each key value. If a key is not duplicable, then no other record may use the same value in the same key. For instance, each value in a key designed to store social security numbers for a list of employees should be unique, but a key storing employees' marital status must be duplicable since the two possible values (single and married) are used repeatedly. Values in a key of social security numbers never change and should receive the attribute of nonmodifiability, but marital status can be modified and the corresponding key should be modifiable, too.

**6.2.3.2 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 replaces the master copy, guaranteeing the stability of the file.

## 6.3 FILE FEATURES

DX10 supports a variety of file features that add to the flexibility of the system and promote its overall usefulness. The file features include the following:

- File use from high-level languages
- Delete and write protection
- File access privileges
- Record locking
- Temporary files
- Blocked files

- Deferred or immediate write options
- Blank compression and adjustment
- Expandable files

### 6.3.1 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, providing indirect access to features not supported directly by the language syntax.

### 6.3.2 Delete and Write Protection

When a file is created, it is vulnerable to deletion and rewriting. You can modify a file's protection to prevent the destruction of its contents. If you declare a file to be delete-protected, no program or user can delete that file until the protection status is modified. You may wish to further protect the content of some files by write-protecting them. A write-protected file can only be read. Write-protected files are automatically delete-protected. Delete-protecting a directory does not prevent the deletion of files cataloged in that directory; it only guarantees the existence of the directory node.

### 6.3.3 File Access Privileges

An application program run under DX10 can request specific access privileges for any use of a file. A use may be defined as the entire file transaction from open through close. The following list describes the access privilege modes for files:

- Exclusive access — only the calling program can read or write to the file.
- Exclusive write access — only the calling program can write to the file; all programs may read the file.
- Shared access — the calling program and other programs share access to the file for read and write operations; shared access to a sequential file allows read and write only.
- Read only — the calling program cannot write to the file, but can read from it; other programs can both read and write to the file.

### 6.3.4 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.

### 6.3.5 Temporary Files

DX10 allows the creation and use of temporary files by tasks running under DX10. 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.



### 6.3.6 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 and so improves throughput of the system.

### 6.3.7 Deferred or Immediate Write

DX10 supports both immediate and deferred writing of logical record blocks to disk. The physical transfer to disk of logical record blocks 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 (for example, for data integrity in highly sensitive files), you may prefer that all writes to a file occur immediately upon request. DX10 supports this with the immediate write option.

### 6.3.8 Blank Suppression and Adjustment

For file types that support variable length records (that is, all except relative record), you can instruct DX10 to remove extraneous blank characters 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.

### 6.3.9 Expandable Files

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

## 6.4 DIRECTORY AND FILE ARRANGEMENT

DX10 arranges each disk volume into a hierarchical structure. The highest level of access in this hierarchy is the name of the volume itself. Within the volume are directories and files. A directory is a specialized file which contains only other files and directories. A file can be stored directly under the disk volume or under a directory. To access a file, you must specify not just the file name, but the file's entire pathname of the overall volume and any intermediate directories. To construct the pathname, you must string together the names of the volume, any intermediate directories, and the file. Be sure to separate each name with a period (.). If the disk volume is the system disk, you can omit the volume name from the pathname and begin the pathname with a period. The following examples illustrate this concept.

### EXAMPLE

VOLONE — This is the name of a hypothetical disk volume. If you wanted to access this volume, you would refer to it by the pathname, VOLONE.

**EXAMPLE**

VOLONE.FILE1 — This is the pathname for a file named FILE1 created on volume named VOLONE. If you wanted to access the contents of FILE1, you would refer to the file by its pathname, VOLONE.FILE1.

**EXAMPLE**

VOLONE.DIR1.FILE1 — This pathname refers to the file FILE1 in the directory DIR1 on volume VOLONE.

**EXAMPLE**

EMPLOY01.USRA.PAYROLL  
EMPLOY01.USRB.PAYROLL  
EMPLOY01.USRB.CATALOGX.PAYROLL

Figure 6-1 shows the relationships between the files and directories on the volume, EMPLOY01. Notice that two or more files can have the same file name if they are located in different directories; that is, if they have different pathnames.

**EXAMPLE**

.FILE1 — This is a valid pathname for a file named FILE1 created on the system disk with no intermediate directories.

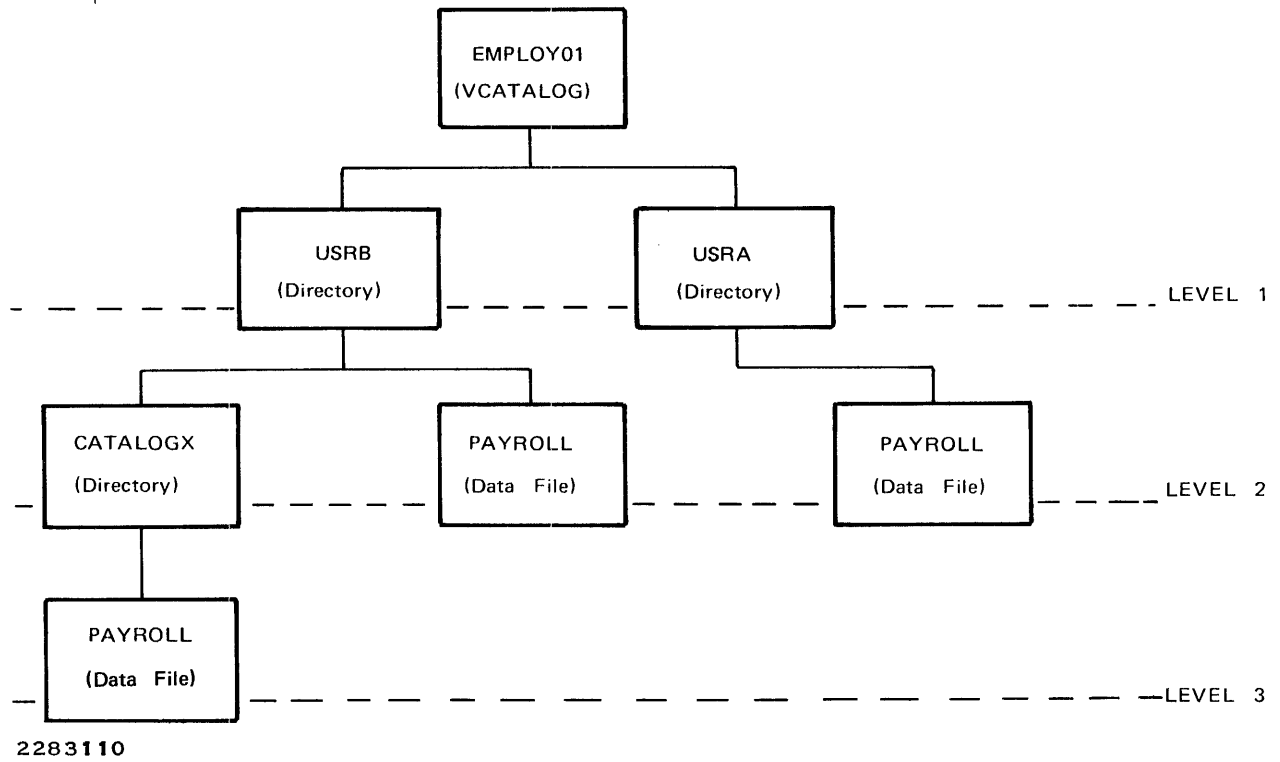


Figure 6-1. Example Directory Structure for EMPLOY01

# System Generation

---

## 7.1 GENERAL INFORMATION

When you first receive DX10 software, it has a built-in image of a basic computer system. That is, DX10 expects to oversee a computer with at least one VDT. A magnetic tape transport may also be required. For DX10 to support your particular system configuration, you must supply it with an accurate system image. This process is system generation, also called sysgen. This section provides a general description of system generation. You can find detailed instructions for performing a system generation in Volume V.

The DX10 software exists on a variety of media: disk cartridge, flexible diskette, magnetic tape, or cartridge tape. Before you can use DX10, it must be available on disk media. If your software is on a sequentially accessible medium, such as magnetic tape or tape cartridge, you must transfer DX10 to a randomly accessible medium. Instructions for making this transfer by means of the Disk Build utility are provided in Volume V.

Generating a system involves five stages:

- Communicating the system configuration to DX10 by issuing the Execute System Generation (XGEN) command
- Assembling and linking the system with the Assemble and Link Generated System (ALGS) command
- Patching the system using the Patch Generated System (PGS) command
- Testing the system with the Test Generated System (TGS) command
- Installing the generated system with the Install Generated System (IGS) command

### 7.1.1 Communicating the System Configuration to DX10

When you issue the XGEN command, the SCI activates the GEN990 utility. GEN990 is an interactive program. Through your terminal, GEN990 questions you about the physical makeup of your computer, such as, the number of disk drives present or the number and kinds of terminals and printers present. During this phase of System Generation, you can incorporate special device drivers, custom supervisor call processors, and an initialized system common module. Also at this time, you can select optional task management features, such as time slicing and the task sentry. GEN990 takes this information and processes it into files that can be further processed, in the next stage, assembling and linking the generated system.

### **7.1.2 Assembling and Linking the Generated System**

You direct DX10 to process the information gathered during the execution of the XGEN utility by issuing the ALGS command. After entering the letters ALGS at the terminal keyboard and responding to the five prompts identifying the system to be assembled and linked, wait 15 to 30 minutes while DX10 further processes the generated system. When the ALGS command has finished processing, continue to the next step of system generation, patching the generated system.

### **7.1.3 Patching the Generated System**

When you issue the Patch Generated System (PGS) command, DX10 applies the data from the patch file to your generated system. This process makes additional software implementations and updates to your generated system and enables IPL completion. After you have identified the system to be patched, the PGS utility proceeds without your further direction.

### **7.1.4 Testing the Generated System**

After you have patched your generated system, it should be tested before being installed as the primary system. Issue the TGS command and perform an IPL to test your system. This utility executes in a very short time and insures that your system functions correctly.

### **7.1.5 Installing the Generated System**

The final step in System Generation is installing the generated system. Until this point, the installed system has been a generalized utility system that can direct any variety of system configurations, but at less than optimum efficiency. Your custom generation, when installed, will provide efficient use of your systems resources.

At this point, issue the IGS command, and DX10 establishes your newly generated system as the primary system each time you perform an Initial Program Load (IPL).

# Error Control

---

## 8.1 GENERAL INFORMATION

DX10 incorporates several features to prevent and diagnose error conditions. The following list introduces the topics of error control discussed in this section:

- Error reporting
- System crashes
- System logs
- Memory mapping
- End action routines
- Error prevention

## 8.2 ERROR REPORTING

DX10 has the ability to detect errors caused by a variety of sources, both hardware and software. When an error is detected, DX10 issues a message reporting the error to the terminal or terminals that control the erring task or device. Volume VI contains a complete listing of error messages, provides the material necessary for interpreting the error messages, and recommends recovery techniques for each error.

## 8.3 SYSTEM CRASHES

In severe circumstances, a fatal system error occurs. For example, failure of the system disk on a critical operation is a fatal system error. When fatal system errors occur, the system itself comes to an abnormal termination. This condition is also called a system crash. In these circumstances, an error code is reported to the LEDs on the system control panel. Volume VI contains instructions for preserving a post-crash image of memory on the disk when a system crash occurs.

DX10 includes ANALZ, a system crash analysis utility. Volume VI includes instructions for executing ANALZ and for interpreting the output of ANALZ.

#### **8.4 SYSTEM LOG**

The system log maintains a record of device and task execution errors, status messages, and the time at which they occurred. Abnormally terminated tasks provide task error messages to the system log. Application programs can log additional messages by issuing the System Log supervisor call. DX10 provides a system command to start the system log. The system log consists of a pair of files. After one fills, log messages transfer to the second file.

#### **8.5 MEMORY MAPPING**

The 990 memory mapping feature prevents application programs from accidentally destroying any other application programs. Without memory mapping, even system functions would be vulnerable to corruption. When a task performs an illegal function (such as requesting memory outside its legal range or requesting an illegal instruction), DX10 abnormally terminates the task. In this case, DX10 sends an advisory message to the system log. This message contains code indicating the cause of abnormal termination. You should analyze this message to determine the appropriate recovery procedure.

#### **8.6 END-ACTION ROUTINES**

Any task can optionally include a sequence of instructions designated as an end-action routine. If you have selected this option and an abnormal termination occurs, 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.

#### **8.7 ERROR PREVENTION**

Use the following measures to avoid failures and minimize losses should they occur:

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

# Disk Management and Organization

---

## 9.1 GENERAL INFORMATION

The primary mass storage medium for DX10 systems is the magnetic disk. DX10 provides the necessary management for allocation of disk space to files. Allocatable Disk Units (ADUs) are the most basic logical unit into which DX10 divides disks. ADUs are grouped into files either by DX10 for its own use or by you for your purposes. This section discusses the management of disk memory space at the ADU level and the general structure of files in a disk volume.

## 9.2 DISK MANAGEMENT

DX10 provides the management for allocation of disk space to the files. Disk space is allocated in units referred to as ADUs. An ADU is a space on the disk surface measured in sectors (sectors being a permanent physical division of the disk surface). The specific size of an ADU varies depending on the model of disk being measured. Larger disks have larger ADUs than smaller disks, but an ADU is always smaller than a track. 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 the total space on the disk cartridge not occupied by system data. When you create a file, you can specify the maximum amount of space to be allocated to it. At this time, you can also designate the file to be expandable and DX10 will increase the file's allotment of space as the file fills up. DX10 supports automatic allocation of disk space to files with a sophisticated disk management strategy designed to meet two performance objectives:

- Provide access to any physical record of the file using one disk access
- Provide for wide dynamic range of file size without incurring excessive allocation overhead

The first objective also applies to the logical record of a file as seen by the application program. An exception is the case of key indexed files where several disk accesses are often required to process a logical record read or write.

Allocation overhead under DX10 refers to the following:

- The time spent in the allocation function
- Any disk space wasted by allocating more disk space than is really used
- Any memory space used to catalog allocated disk segments



DX10 creates the initial allocation according to the size specified during file creation (by you or a task). The allocation size increases as additional segments are required. The first and second additional segments are equal in size to the initial allocation. The third additional is equal to twice the size of the second allocation. Subsequent allocations (up to a maximum of 16) continue to be double the size of the previous allocation.

### **9.3 DISK VOLUME CONTENT AND ATTRIBUTES**

Each disk drive contains a single disk volume. One disk volume is designated to be the system disk. It contains the DX10 software and the system files. Every disk volume contains system overhead files and space reserved for user files. Figure 9-1 shows the physical layout of a disk volume.

#### **9.3.1 System Overhead Files**

Track 0 and part of Track 1 of each disk volume are always allocated to system overhead as follows:

- **Volume ID** — Each disk cartridge under DX10 is identified by a user-assigned name called the disk's Volume ID. DX10 records each disk's Volume ID on the first track of the disk (Track 0).
- **Allocation Map** — Also on Track 0 is a map (in ADUs) of the disk space currently in use.
- **Bad Disk ADU List** — DX10 maintains a list of unusable ADUs on the disk. When the disk is initialized, DX10 inspects it for unusable ADUs and writes the Bad Disk ADU List on Track 0.
- **System Intermediate Loader** — A system loader is optionally stored on disks. During an Initial Program Load (IPL), the bootstrap loader (stored in read-only memory on the CPU itself) accesses this loader program which then reads the DX10 software into memory.
- **Volume Catalog (VCATALOG)** — Each disk volume has a specific file directory beginning in Track 2 named VCATALOG which acts as table of contents for the volume. The files described in VCATALOG may be data files or directory files as illustrated in Figure 9-2.
- **Diagnostic file (S\$DIAG)** — The innermost cylinder of the disk is reserved for use with online diagnostics.

#### **9.3.2 System Files**

DX10 includes special system files to support itself. 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\$.

#### **9.3.3 User Files**

All of your own files are stored on disk volumes in the space not required for system overhead and system files. Section 6 of this manual describes the types of data files supported by DX10.

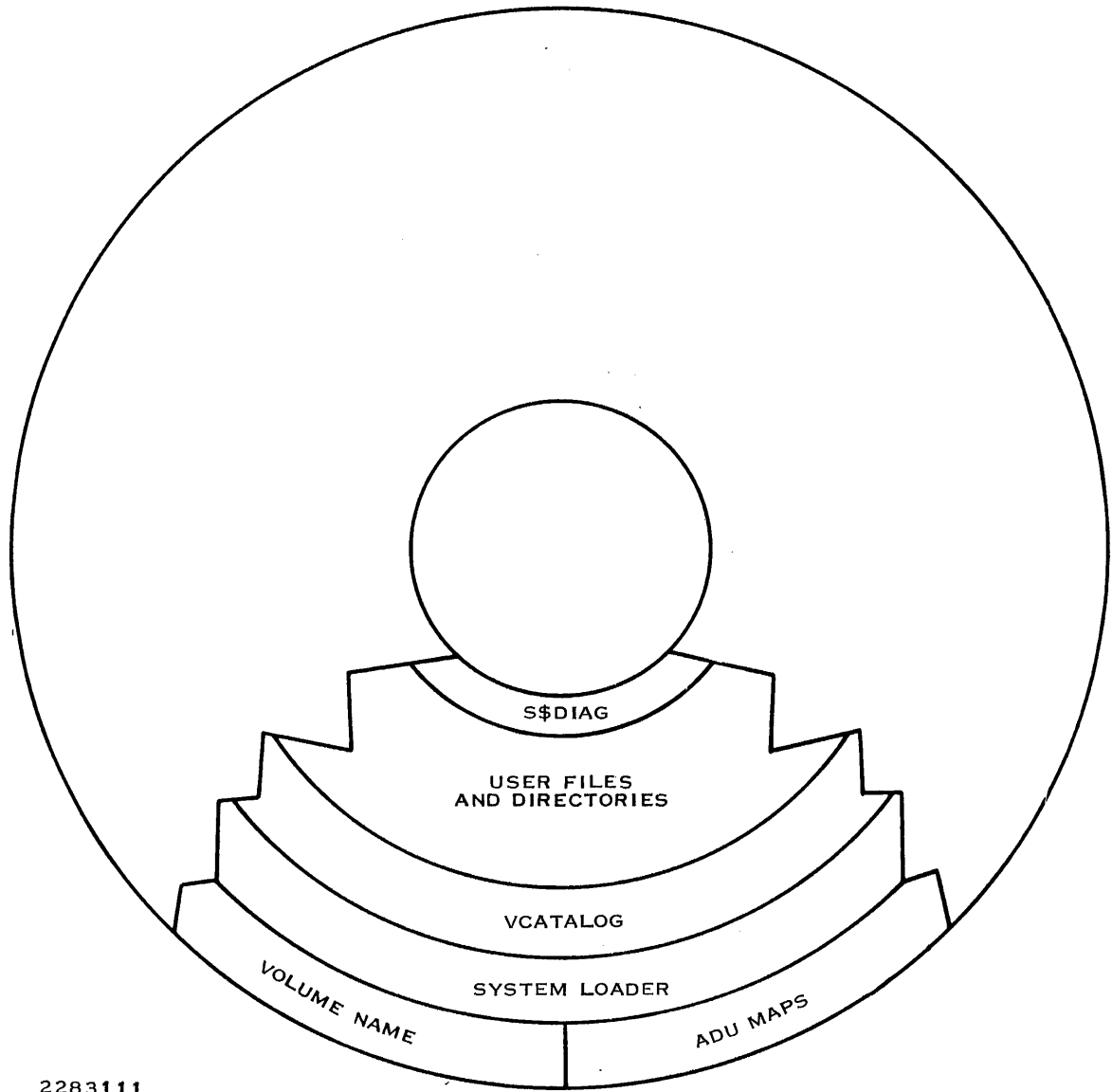
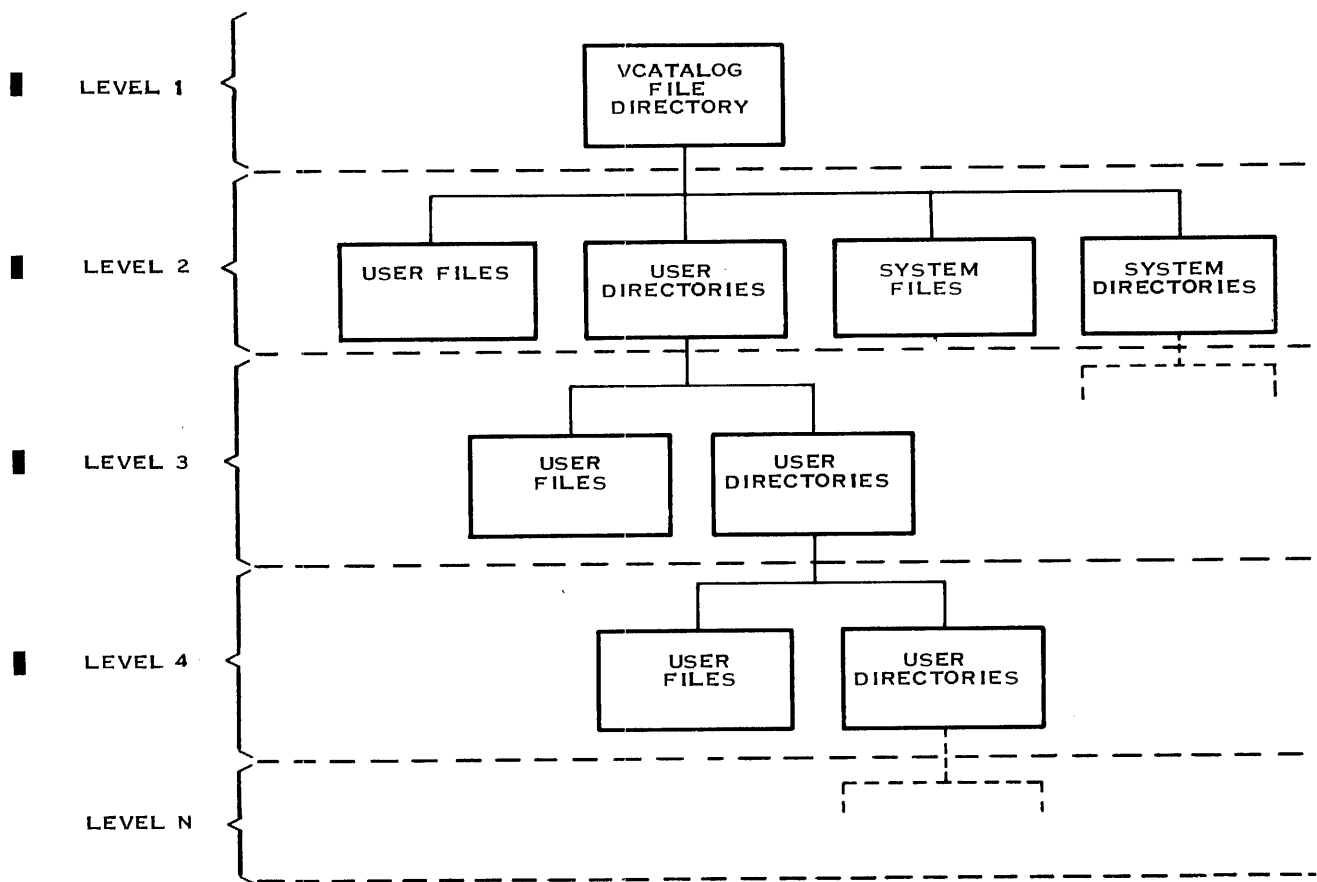


Figure 9-1. Physical Layout of a Disk Volume



2278899

Figure 9-2. Files and Directory Structure

# Device and File Services

---

## 10.1 GENERAL INFORMATION

This section discusses device and file services under the control of DX10. The section has three major divisions:

- Logical input/output
  - Access names for devices and files
  - Logical unit numbers (LUNOs)
  - Device orientation
- Device and file operations
- Extended video display terminal support

## 10.2 ACCESS NAMES

DX10 identifies peripheral devices and files with access names. Access names can be divided into three categories:

- Device names
- Volume names
- File pathnames

### 10.2.1 Device Names

DX10 identifies each peripheral device by a unique device name. Each device name is four characters long. The first two characters are alphabetic and identify the device type; the last two characters are digits with a range from 01 to 99 and identify a particular device in its serial order in the system image. DX10 assigns device names during system generation. The following list demonstrates the format for device names and includes all device type acronyms. In this list, xx substitutes for an integer from 01 to 99.

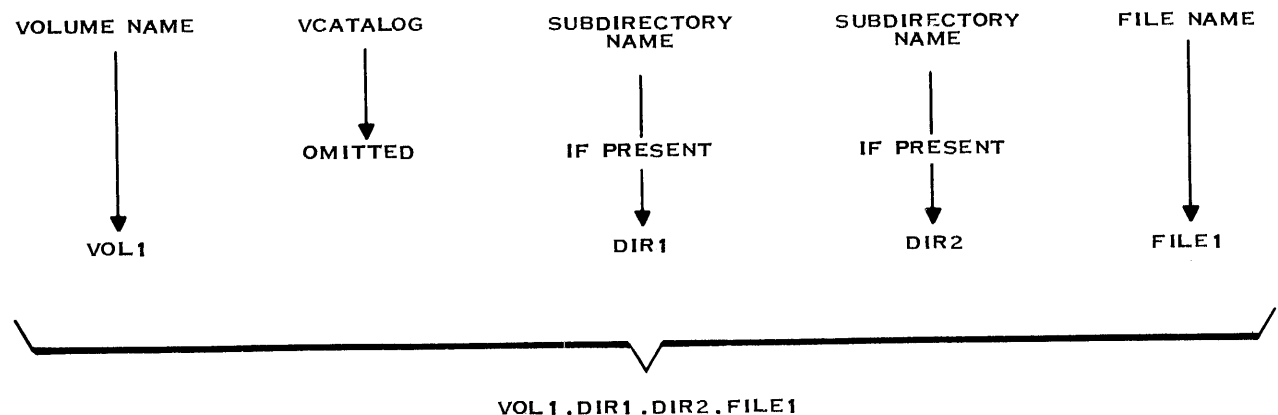
- CMxx — a communication device
- CRxx — a card reader
- CSxx — a cassette unit of a Model 733 ASR terminal
- DKxx — a flexible diskette drive
- DSxx — a disk drive
- EMxx — an AMPL Emulator
- LPxx — a line printer
- MTxx — a magnetic tape transport
- STxx — an interactive data terminal
- TMxx — an AMPL Trace Module

### 10.2.2 Volume Names

A volume is a logical device that represents a physical disk pack. You assign each volume a name when you initialize it (with either the Initialize Disk Surface (IDS) or Initialize New Volume (INV) commands). Refer to a disk by its volume name when logically installing or unloading it into a disk drive.

### 10.2.3 File Pathnames

Identify a file by its unique pathname. A pathname is a string of shorter names, beginning with an access name of the volume that stores the file, followed by the file names of the directories, if any, that you have created to catalog the file, and ending with the file name. Figure 10-1 demonstrates the derivation of the pathname of a file.



2283103

Figure 10-1. Derivation of a File Pathname

You can refer to the volume that a file resides on either by its logical volume name or, in some cases, by its physical device name. You can imply that the file is on the system disk by simply omitting an access name for the volume and beginning the pathname with a period. The following list contains valid pathnames:

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

Although more than one file can share a file name, each pathname is unique. The following examples demonstrate how several files can share the same file name, FILE1, and still have different pathnames.

```
VOL1.FILE1
VOL1.DIR1.FILE1
VOL1.DIR1.DIR1.FILE1
VOL1.DIR2.FILE1
VOL2.FILE1
```

### 10.3 LOGICAL UNIT NUMBERS (LUNOs)

Tasks executing under DX10 perform input/output to logical units instead of physical units. DX10 refers to a logical input/output unit by an assigned logical unit number (LUNO). LUNOs are assigned either by you, using either the SCI or a supervisor call in the code of a program, or by DX10. As many as 256 LUNOs can be assigned at one time, ranging from >0 to >FF.

The use of LUNOs allows DX10 to direct the input and output resources of a single program to any single system device or file.

For example, suppose you have written a program designed to send its output to LUNO 82. Before you execute this program, assign LUNO 82 (using the AL command) to the output device or file that you require, for example, ST03. In this example, the program sends its output to ST03. If you were to assign LUNO 82 to the line printer, LP02, and reexecute the same program, then the program would send its output to LP02.

#### 10.3.1 Scope of LUNO Assignments

DX10 arranges LUNOs in these three classes:

- Task local LUNOs — accessible only by the task that assigned the LUNO
- Station local LUNOs — accessible only by tasks associated with a given terminal
- Global LUNO — accessible by all stations and tasks

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 with which the program is associated (station local LUNO), and finally the LUNO assignments which are available in scope to all programs (global LUNO). The use of 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 may be in use by another program or user. Some global LUNOs are pre-assigned by the system, as listed in Volume V. Also, only one task at a time can open and use a global LUNO assigned to a file or file-oriented device. See Volumes III and V for more information on LUNOs.

#### 10.4 DEVICE ORIENTATION

During system generation, you can declare certain devices to be either file-oriented or record-oriented. File-oriented devices support exclusive access privileges only. That is, if a task opens a LUNO assigned to a file-oriented device, no other task may read from or write to that device. Record-oriented devices, on the other hand, support only shared-access privileges. Different tasks can read and write records from an open record-oriented device regardless of which task opened it.

Whether a particular device is record-oriented or file-oriented depends on the nature of the device and on what kind of organization it imposes on the data. The orientation can also depend on the intended use of the device; a line printer used for logging by several programs would be record-oriented while one used for printing reports should be file-oriented to prevent one report from interleaving its lines into another report.

Magnetic tape drives are usually file-oriented, although you might choose record-orientation if it is used for logging. Disk devices themselves are always record-oriented to allow access to several files, which are controlled by access privileges, not orientation. VDTs are usually record-oriented.

#### 10.5 DEVICE AND FILE OPERATIONS

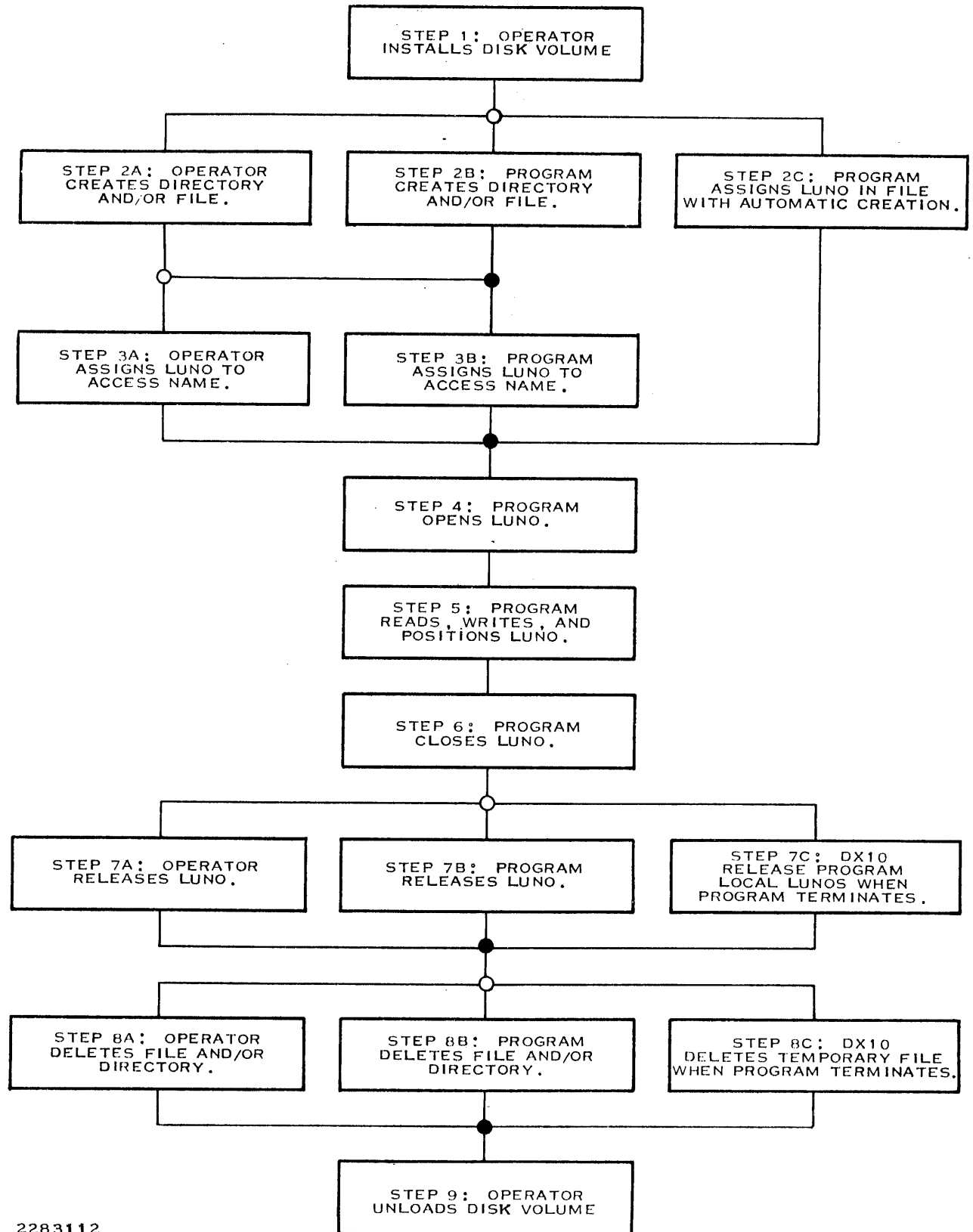
The flowchart in Figure 10-2 illustrates the progress of typical device and file operations. The following paragraphs comment on the flowchart.

##### 10.5.1 Step 1 — Disk Preparation

Steps 1, 2A, 2B and 2C apply only to disk files. Note that for any disk-resident file, the volume on which the file is stored must be installed first, both physically in disk drive, and logically by use of the Install Volume command. Disk volumes that have not previously been used must be initialized by use of the Initialize Disk Surface and Initialize New Volume commands.

##### 10.5.2 Step 2 — File Creation

Before a file can be accessed, it must be created under a directory on the disk volume. If the directory does not exist, it, too, must be created. Both directories and files can be created either by you, using SCI commands, or by a task, (assembly language task or high level language runtime task) using I/O service calls. DX10 can create files automatically with the Assign LUNO SVC using the autcreate option.



2283112

Figure 10-2. Device and File Operation Flowchart



### 10.5.3 Step 3 — LUNO Assignment

Since files or devices are accessed by tasks through a LUNO, the LUNO must be assigned to the appropriate access name (steps 3A and 3B). Either you or a task (possibly a high-level language run-time task) can request a LUNO assignment.

### 10.5.4 Step 4 — Normal I/O Operations

Steps 4, 5 and 6 constitute a typical I/O operation of a task. When a task requests a read or write operation, it may be designated as either an execute operation or an initiate operation. If the task is an execute operation, DX10 does not return to the requesting task until the operation is complete. If the task is an initiate operation, DX10 returns to the requesting task as soon as the operation has begun. 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 wait for the I/O operation to complete.

### 10.5.5 Step 7 — LUNO Release

When a particular LUNO assignment is no longer required, you should release that LUNO. You can release a global or station-local LUNO by issuing the appropriate SCI command, either Release Global LUNO or Release LUNO. A task can release a LUNO by issuing an appropriate SVC. DX10 automatically releases task-local LUNOs when the assigning task terminates. DX10 automatically releases station-local LUNOs when one of the following conditions occurs:

- The terminal user logs off
- All tasks initiated from the terminal complete
- All batch streams associated with the terminal complete

### 10.5.6 Step 8 — Temporary File Deletion

Any temporary files (described in Section 6) used by a task are automatically deleted from the disk volume by DX10 when the file's LUNO is released or when the task terminates. You can delete any other unnecessary files by issuing the SCI command, Delete File. A task can delete a file with an SVC, possibly through a language run-time routine.

### 10.5.7 Step 9 — Unloading Disk Volume

When you no longer need to access files on a given volume, logically unload the volume by issuing the Unload Volume SCI command. After you have logically unloaded the disk volume, you can physically remove it from the disk drive.

## 10.6 PERFORM I/O OPERATION SUPERVISOR CALL

The I/O SVC supports all task-initiated record transfer and file positioning operations on devices and files. The I/O SVC also controls utility operations such as file creation and LUNO assignment. A complete list of these operations is provided in Table 10-1. SVCs are discussed in greater detail in Volume III.

Additional SVCs provided with DX10 structure are designed to complement and support I/O operations. Most of these functions are available only through assembly language programs. Consult the appropriate high level language programmer's guide for instructions on accessing these facilities. These functions 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 keyboard event character (function key)

**Table 10-1. 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	

---

## 10.7 EXTENDED VIDEO DISPLAY TERMINAL SUPPORT

DX10 offers extended I/O support for video display terminals (VDTs). The Model 931 VDT is shown in Figure 10-3, the Model 911 VDT in Figure 10-4, and the Business System Terminal in Figure 10-5. Device-dependent I/O support provides access to many unique features available in these 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 Definitions** — 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, or blank.
- **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 Previous Character and Next Character keys, Insert Character or Delete Character keys, field boundary checking, and the Erase Field key for rekeying.
- **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 931 VDT, the 911 VDT, and the Business System Terminal provide graphics capability through special graphics character sets.



2284759

**Figure 10-3. Model 931 Video Display Terminal**



2283113

**Figure 10-4. Model 911 Video Display Terminal**



2283093

**Figure 10-5. Business System Terminal**

# Application Programming Environment

---

## 11.1 GENERAL INFORMATION

This section describes the software available for developing and enhancing your application programs. The section consists of five parts:

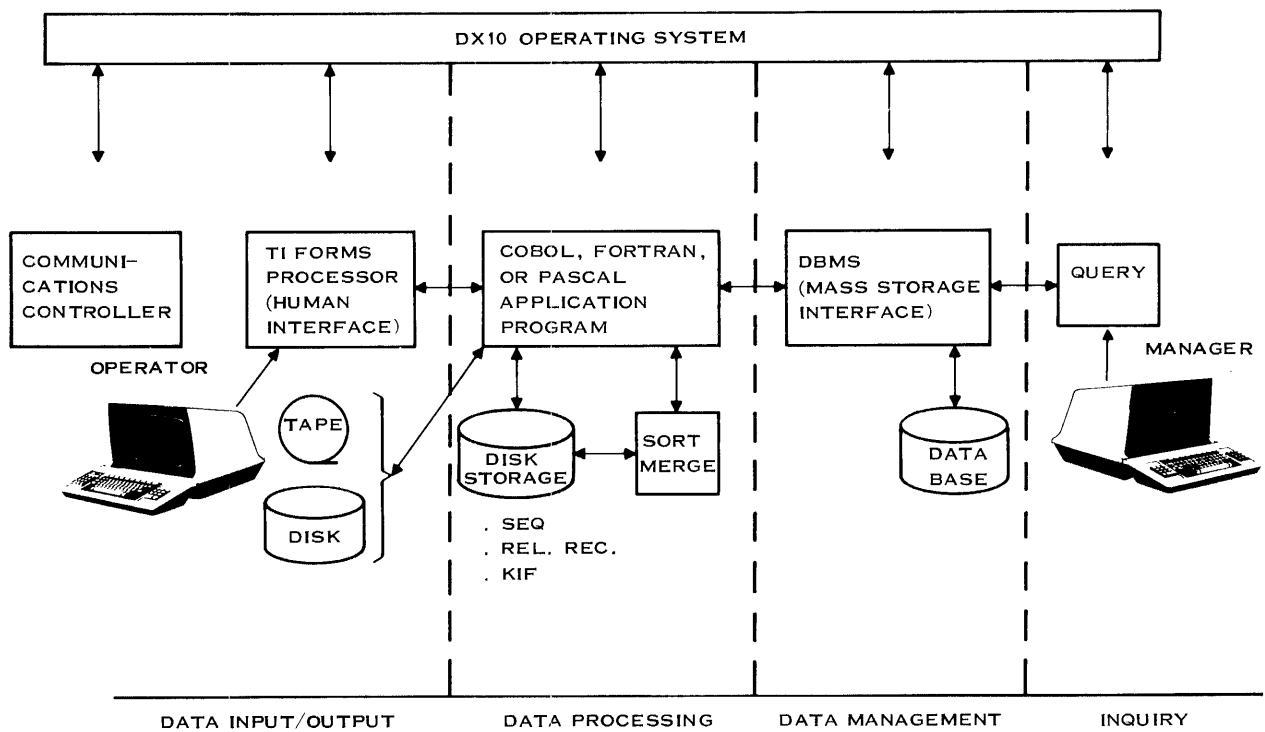
- Program development tools; DX10 provides these features to every purchaser.
  - Interactive Text Editor
  - Macro Assembler
  - Link Editor
  - Interactive Debugger

The remaining software packages are supported by DX10 but must be purchased separately.

- Advanced communications software
  - 3780/2780 Communications Software
  - 3270 Emulator Communications Software
- High-level languages
  - FORTRAN
  - COBOL
  - Pascal
  - BASIC
  - RPG II
- Productivity aids
  - TIFORM
  - Texas Instruments Page Editor (TIPE)
  - Sort/Merge
  - COBOL Program Generator (CPG)

- Data management tools
  - Data Base Management System (DBMS)
  - Interactive DBMS Retrieval (Query)
  - Data Dictionary (DD)

Each software package provides specialized features for use in the different functional areas of data processing. You can combine the various languages and utilities into a total information system best adapted to your individual requirements. Figure 11-1 illustrates the interrelationships of the system software tools available from Texas Instruments.



2282021

Figure 11-1. Commercial Application Environment

## 11.2 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

### 11.2.1 Interactive Text Editor

DX10 provides an interactive text editor for the creation and modification of a sequential file of textual data, such as program's source code or documentation. The Text Editor operates on files with lines up to 240 characters long. The Text Editor also provides a wrap-around mode in which the cursor automatically moves to the next line as you reach the right-hand margin.

To make full use of the Text Editor's capabilities, use the Text Editor from a video display terminal. The Text Editor can operate on an interactive hard-copy data terminal, but only displays one line at a time.

When you initiate the Text Editor on a text file, a copy of the text file appears on the VDT screen. You can modify any part of this copy by adding, deleting or replacing individual characters. You can make modifications by keying new characters into the copy, by using special edit function keys on your terminal, or by issuing Text Editor commands (these commands are a subset of SCI commands). You can send portions of your file to auxiliary files and insert auxiliary files into the file being edited by using Text Editor commands. Text Editor commands also allow copying or moving lines of text from one location of the edit copy to another. When you end an edit session, you can replace the original contents of the text file with the edited copy or send the edited copy to another file. For more details, refer to Volume IV.

### 11.2.2 Macro Assembler

The DX10 assembler implements the standard 990 assembly language instructions and includes a macro facility.

When implemented on a 990/10 computer, the standard 990 assembler language includes a set of 72 instructions that provide for the input, output, manipulation, and comparison of integer and ASCII character data. The 990/10 instruction set provides five modes of addressing memory. The 990/12 instruction set includes 72 instructions in addition to the 72 instructions of the 990/10 instruction set. The additional instructions provide the capability to manipulate stacks, lists, strings, individual memory bits, multiple-precision integers, and single- and double-precision real numbers; enable or disable interrupts; execute microcode diagnostics; load the writable control store; and convert one type of data to another.



The macro assembler offers the capability of defining macro-instructions (called macros). A macro is a user-defined set of assembler language source statements. Macro definitions assign a name to the macro and define its source statements. The macro name may then be used in the operation field of a program's source statement to cause the assembler to insert the predefined source statements and assemble them along with the other source statements of the program. The macro capability allows you to perform the following functions:

- Define macros to specify frequently used sequences of source code
- Define macros for problem-oriented sequences of instructions to provide a means of programming more meaningful to users who are not computer-oriented.

The *990/10 and 990/12 Assembly Language Reference Manual* provides a complete description of the macro assembler. An example of the macro assembler listing is found in Figure 11-2.

### 11.2.3 Link Editor

The Link Editor provides you with the means of combining separately generated object modules to form a single linked output. The Link Editor accepts modules that have been generated by the assembler, the COBOL compiler, the FORTRAN compiler, the Pascal compiler, or a previous partial link.

The major function of the Link Editor is to resolve external definitions and references in each of the individual unlinked or partially linked object modules. The Link Editor also provides for the design and use of overlays, which allow the user to design memory efficient programs. The Link Editor resolves references to these overlays.

Using control statements, you can specify the use of Link Editor options. These options include the following:

- Generation of a load map
- Search of a set of object libraries for referenced values
- Production of a partial link that leaves external references to be resolved at a later time. Figure 11-3 shows a partial listing produced by the Link Editor.

You can direct the output of the Link Editor to an installable object file or, as a memory image, to a program file or DX10 image file.

For a more detailed discussion of the Link Editor, refer to the *Link Editor Reference Manual*. Figure 11-3 displays a typical Link Editor listing output.

```

TSTSDS      SDSMAC 3.5.0 82.130   16:33:16 WEDNESDAY, MAR 16, 1983.      PAGE 0002

0001          IDT 'TSTSDS'
0003          *
0004          *****
0005          *
0006          *           SIMPLE EXAMPLE OF ASSEMBLY LANGUAGE PROGRAM
0007          *           TO CREATE, OPEN, WRITE, & CLOSE A FILE.
0008          *
0009          *****
0010          *
0011          DXOP SVC,15
0012          REF WSPACE
0013          0001 R1 EQU 1
0014          0002 R2 EQU 2
0015          *
0016          0000 0000 DATA WSPACE
0017          0002 0006' DATA TSTSDS
0018          0004 0000 DATA 0
0019          0006 TSTSDS EVEN
0020          *           CREATE FILE "DS01.TSTMSG"
0021          0006 2FE0 SVC @FUSCB
0022          0008 00F0'
0023          *
0024          *           ASSIGN LUND TO FILE
0024          000A DB20 MOVB @AL,@FUSOPC
0025          000C 0044'
0025          000E 00F2'
0025          0010 2FE0 SVC @FUSCB
0026          0012 00F0'
0027          *
0028          *           OPEN THE FILE
0028          0014 2FE0 SVC @SEQIRB
0029          0016 0114'
0030          *
0031          *           WRITE ASCII
0031          0018 0201 LI R1,ADRTBL
0032          001A 003C'
0032          001C C0B1 LOOP MOV *R1+,R2 R2 = @ OF NEXT LINE TO WRITE
0033          001E 130B JEQ DONE
0034          *
0035          0020 C802 MOV R2,@IRBDBA STORE BUFFER ADDRESS
0036          0022 011A'
0036          0024 DB20 MOVB @WA,@IRBOPC STORE 'WRITE ASCII' OP CODE
0037          0026 0045'
0037          0028 0116'
0037          002A 2FE0 SVC @SEQIRB WRITE OUT THE LINE
0038          002C 0114'
0038          002E 10F6 JMP LOOP LOOP
0039          *
0040          *           DONE
0041          0030 DONE
0042          0030 DB20 MOVB @CLWEDF,@IRBOPC
0043          0032 0046'
0043          0034 0116'
0044          *           CLOSE FILE AND WRITE E-O-F
0044          0036 2FE0 SVC @SEQIRB
0045          0038 0114'
0045          003A 10FF JMP $
0046          *
0047          *-----

```

Figure 11-2. Example Assembly Program Listing (Sheet 1 of 2)

```

0048          *
TSTSDS      SDSMAC 3.5.0 82.130 16:33:16 WEDNESDAY, MAR 16, 1983.
TSTSDS - SDS VERIFICATION PROGRAM                                PAGE 0003
0049          *                      DATA
0050 003C          EVEN
0051          *
0052 003C 0048' ADRTBL DATA LINE1
0053 003E 0098'      DATA LINE2
0054 0040 0048'      DATA LINE3
0055 0042 0000      DATA 0
0056          *
0057 0044 91 AL      BYTE >91
0058 0045 0B WA      BYTE 11
0059 0046 02 CLWEOF BYTE 02
0060          *
0061 0048          LINE1 EVEN
0062 0048 2A          TEXT '*****'
0063 0070 2A          TEXT '*****'
0064          *
0065 0098          LINE2 EVEN
0066 0098 2A          TEXT '** THIS IS THE OLD MESSAGE '
0067 00C0 20          TEXT ' **'
0068          *
0069          0048' LINE3 EQU LINE1
0070          *
0071 00E8 07 PATHNM  BYTE 7          LENGTH OF PATHNAME TO FOLLOW
0072 00E9 2E          TEXT '.TSTMSG'
0073          *
0074          *          FILE UTILITY SUPERVISOR CALL BLOCK
0075          *
0076 00F0 0000 FUSCB DATA >0000 0   SVC CODE/RETURN CODE
0077 00F2 90 FUSOPC BYTE >90 2   UTILITY OP CODE >90 = CREATE
0078 00F3 20 FUSLUN BYTE >20 3   LUND (WILL USE LUND >20)
0079 00F4          FUSRES BSS 8 4   NOT USED FOR FILE UTILITY SET TO 0
0080 00FC 0000 FUSKIF DATA 0 12  USED FOR KIF FILES SET TO 0
0081 00FE 0000 FUSPRD DATA 0 14  USED FOR PROGRAM FILES SET TO 0
0082 0100 008D FUSFLG DATA >8D 16  UTILITY FLAGS =>USE 18,19 SEQ FILE,
0083          *          EXPANDABLE, BLANK SUPPRESSED RECORD
0084 0102 0050 FUSLRL DATA 80 18  LOGICAL RECORD LENGTH
0085 0104 0000 FUSPRL DATA 0 20  PHYSICAL RECORD LENGTH(USE DEFAULT)
0086 0106 00EB' FUSPNA DATA PATHNM 22  PATHNAME POINTER
0087 0108 0000 FUSPAS DATA 0 24  PASSCODE POINTER(NOT IMPLEMENTED)
0088 010A 0000          DATA 0 26  PASSCODE POINTER(NOT IMPLEMENTED)
0089 010C 0000 FUSINA DATA 0 28  INITIAL ALLOCATION(DEFAULT)
0090 010E 0000          DATA 0 30
0091 0110 0000 FUSSNA DATA 0 32  SECONDARY ALLOCATION(DEFAULT)
0092 0112 0000          DATA 0 34
0093          *
0094          *
0095          *          IO REQUEST BLOCK FOR SEQUENTIAL IO
0096          *
0097 0114 0000 SEGIRB DATA >0000 00  SVC CODE/ERROR CODE
0098 0116 00 IRBOPC BYTE 0 02  IO SUB OP-CODE
0099 0117 20 IRBLUN BYTE >20 03  LUND
0100 0118 00 IRBSFL BYTE 0 04  SYSTEM FLAGS
0101 0119 00 IRBUFL BYTE 0 05  USER FLAGS ==> EXCLUSIVE WRITE SET
0102 011A 0000 IRBDBA DATA 0 06  DATA BUFFER ADDRESS
0103 011C 0000 IRBICC DATA 0 08  INPUT CHARACTER COUNT
0104 011E 0050 IRBOCC DATA 80 10  OUTPUT CHARACTER COUNT
0105 0120 0000 IRBRN1 DATA 0 12  RELATIVE RECORD NUMBER(NOT USED)
0106 0122 0000 IRBRN2 DATA 0 14  RELATIVE RECORD NUMBER(NOT USED)
0107          *

```

Figure 11-2. Example Assembly Program Listing (Sheet 2 of 2)

```

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

```

**Figure 11-3. Partial Listing Produced by the Link Editor**

#### 11.2.4 Interactive Debugger

The Debugger is an interactive symbolic debugging program for assembly language tasks running under DX10. It operates from either an interactive video display terminal (VDT) or an interactive hard-copy terminal. The Debugger allows you to display and modify CPU registers, workspace registers and memory, and to control execution of a task.

The Debugger can operate in one of two modes: run or simulation. In the run mode, a task may be halted and started at will. Also, you can set breakpoints for a more controlled execution of a task. In the simulation mode, a task's execution is analyzed between each instruction. You can set trap conditions that interrogate the program counter or memory content. In simulation mode, you can set conditional breakpoints dependent on the state of the task. Breakpoints designed to halt or continue task execution as required can 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.

For detailed instructions on using the Debugger, refer to Volume III. A simulated debugging session is depicted in Figure 11-4.

```

[] XHT
EXECUTE AND HALT TASK
PROGRAM FILE OR LUNG: .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

```

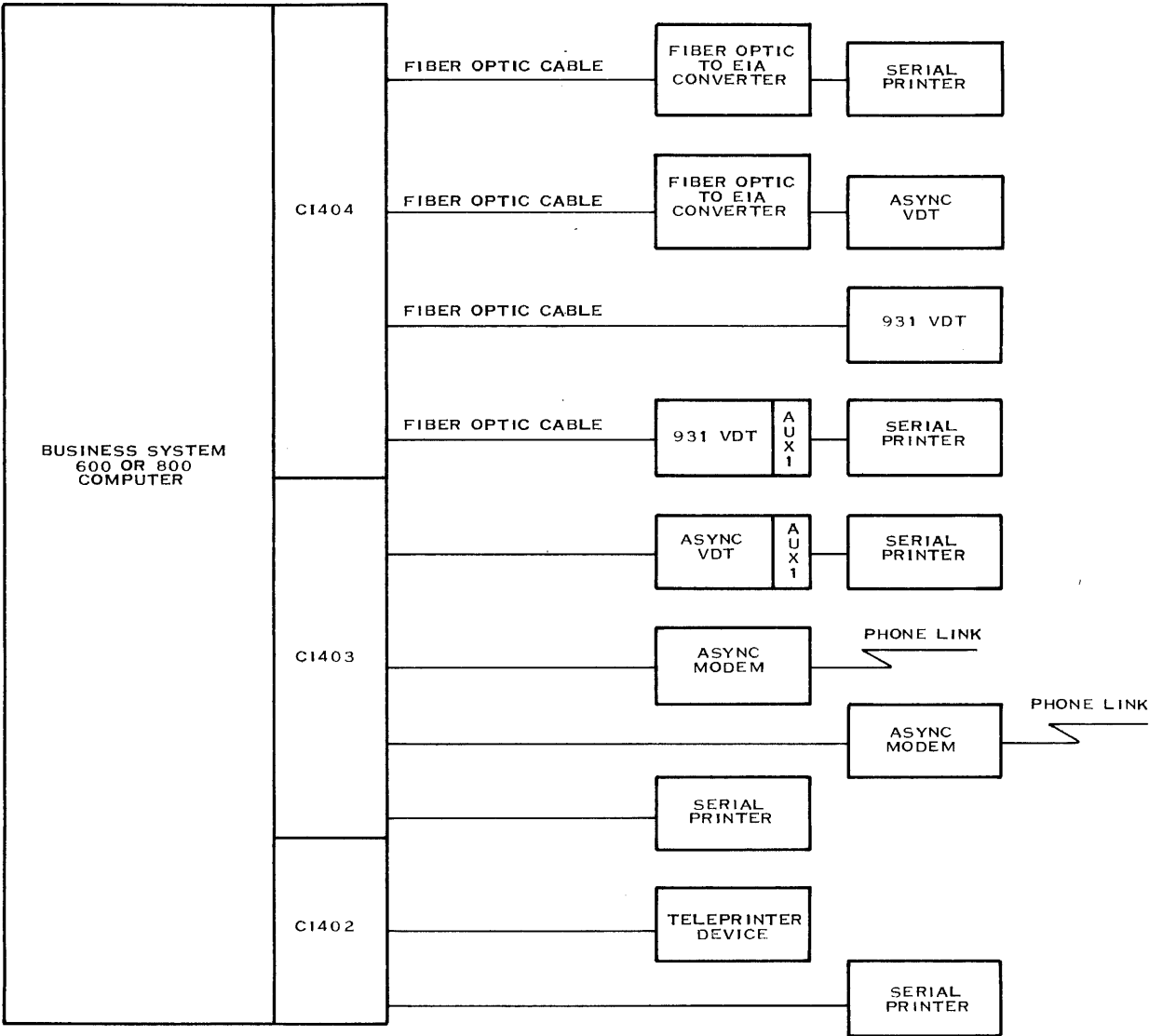
Figure 11-4. Interactive Debugging

### 11.3 DX10 ADVANCED COMMUNICATIONS

DX10 supports several methods of computer-to-computer communication using specially designed software packages and hardware devices. Depending on the application, you generate a custom DX10 system to meet your needs. The available software includes 3780/2780 emulator support and 3270 interactive communications software. The following paragraphs describe these software packages and the hardware controllers necessary for their implementation, as well as other controllers that can provide a wide variety of communication applications.

#### 11.3.1 Communications Hardware Equipment

The 990 communications controllers supported by DX10 include the CI402, CI403, CI404, CI421, CI422, CP501, and CP502. The following paragraphs discuss the communications controllers. Figures 11-5, 11-6, and 11-7 illustrate sample configurations using the controllers.



2284761

Figure 11-5. Sample Business System 600 or 800 Asynchronous Communications Configuration

**11.3.1.1 CI402.** The CI402 provides two independent, serial, asynchronous interfaces that allow a Business System 600 or 800 series computer to communicate with two asynchronous modems or other asynchronous devices compatible with RS-232-C or CCITT (V.24). The CI402 supports half-duplex or full-duplex communications at speeds from 75 bits per second (bps) through 19.2K bps on each channel. Character size is selectable from 5 to 8 data bits with programmable parity (odd, even, or none). Other features include a programmable timer and stop-bit selection. The CI402 supports local asynchronous peripherals via direct cables, or remote asynchronous peripherals via modems.

**11.3.1.2 CI403.** The CI403 is an asynchronous, buffered multiplexer that provides a TILINE slave interface between a Business System 600 or 800 series computer and four asynchronous serial channels. The four channels are RS-232-C and CCITT (V.24) compatible, programmable for speeds up to 19.2K bps, and support half-duplex or full-duplex communication. Character size is selectable from 5 to 8 data bits with programmable parity (odd, even, mark, space, or none). Other features include break detection/generation, a 250-millisecond timer, and stop-bit selection. The CI403 supports local asynchronous peripherals via direct cables, or remote asynchronous peripherals via modems.

**11.3.1.3 CI404.** The CI404 is an asynchronous, buffered multiplexer that provides a TILINE slave interface between a Business System 600 or 800 series computer and four fiber optic channels. The four channels are programmable for speeds up to 19.2K bps and support half-duplex or full-duplex communication. Character size is selectable from 5 to 8 data bits with programmable parity (odd, even, mark, space, or none). Other features include a 250-millisecond timer and stop-bit selection. The CI404 supports local fiber optic peripherals for distances up to 1 kilometer.

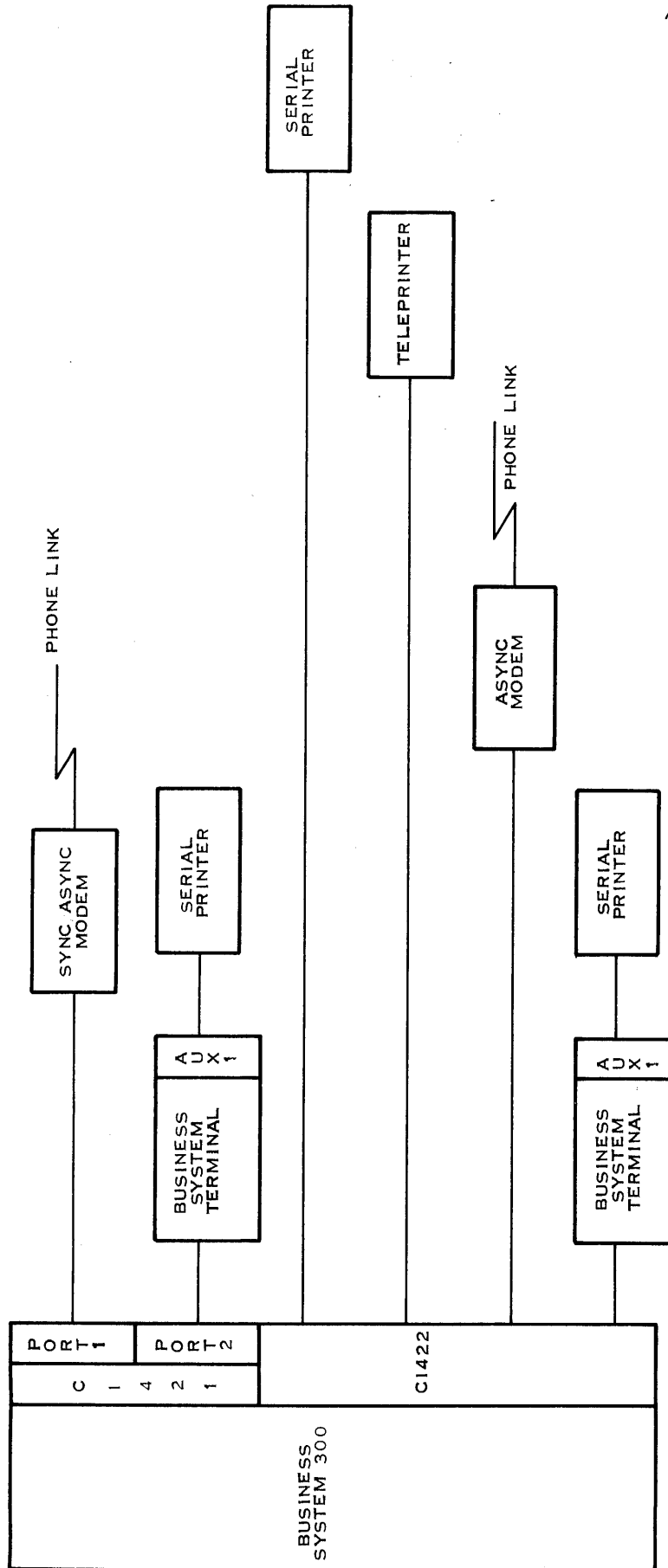


Figure 11-6. Sample Business System 300 Communications Configuration

2284762



**11.3.1.4 CI421.** The CI421 provides a CRU interface between a Business System 300 series computer and two RS-232-C or CCITT (V.24) compatible communications channels, one asynchronous and the other either synchronous or asynchronous. It provides full modem control and status capability for interfacing to remote sites.

The asynchronous channel supports half-duplex or full-duplex communications at speeds from 75 through 9600 bps. Character size is selectable from 5 to 8 data bits with programmable parity (odd, even, or none). Other features of the asynchronous channel include a programmable timer and stop-bit selection.

The synchronous channel can be programmed to support synchronous, asynchronous, or isochronous operation. It supports bit rates from 50 to 9600 baud by an internally generated clock signal. When programmed for synchronous operation, the channel supports character-oriented protocols such as Binary Synchronous Communication (BSC), or bit-oriented protocols such as Synchronous Data Link Control (SDLC).

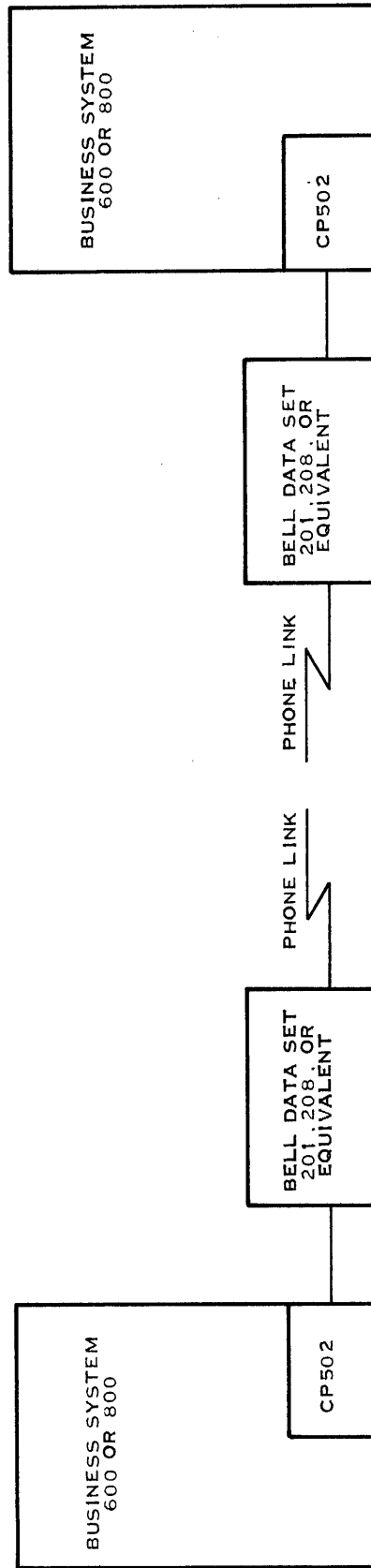
**11.3.1.5 CI422.** The CI422 provides a CRU interface between a Business System 300 series computer and four RS-232-C or CCITT (V.24) compatible communications channels. The CI422 supports half-duplex or full-duplex communications at speeds from 75 through 9600 bps. Character size is selectable from 5 to 8 data bits with programmable parity (odd, even, or none). Other features include a programmable timer and stop-bit selection.

**11.3.1.6 CP501.** The CP501 provides an interface between RS-232-C or RS-423 compatible devices and a Business System 600 or 800 series computer. It provides full modem control and status capability for interfacing to remote sites, and also supports power-isolated TI local line drivers and receivers. The CP501 supports half-duplex or full-duplex communications at speeds from 50 through 9600 bps. Character size is selectable from 5 to 8 data bits with programmable parity (odd, even, or none).

The CP501 can be programmed to support synchronous, asynchronous or isochronous operation. When programmed for synchronous operation, the CP501 supports bit-oriented protocols such as SDLC and Advanced Data Communications Control Procedures (ADCCP). Synchronous operation also supports BSC and Digital Data Communications Message Protocol (DDCMP).

The CP501 contains routines that provide complete support including message blocking, error detection, and cyclic redundancy checks for BSC, SDLC, and ADCCP protocols. The BSC protocol (Extended Binary-Coded Decimal Interchange Code, EBCDIC) supports IBM 3780/2780 emulation. The CP501 also supports download of character detect software.

Synchronous protocols can be externally clocked (NRZ) or internally clocked (NRZI), with bit rates up to 9600 baud.



2283036

Figure 11-7. Sample Business System 600 or 800 Synchronous Communications Configuration

**11.3.1.7 CP502.** The CP502 provides an interface between EIA (RS-232-C/CCITT V.24, RS-422) or X.21 compatible devices and a Business System 600 or 800 series computer. The EIA interface provides full modem control and status capability for interfacing to remote sites. The X.21 interface includes full DCE (Data Carrier Equipment) control and status capability for interfacing to DCE connected to an X.21 leased line or an X.21 switched circuit. The CP502 supports half-duplex or full-duplex communications at speeds from 50 through 9600 bps. Character size is selectable from 5 to 8 data bits with programmable parity (odd, even, or none).

The CP502 can be programmed to support synchronous, asynchronous or isochronous operation. When programmed for synchronous operation, the CP502 supports bit-oriented protocols such as SDLC and ADCCP. Synchronous operation also supports character-oriented protocols such as BSC or DDCMP, and non-sync protocols (bit stream).

The CP502 contains routines that support message blocking, error detection, and cyclic redundancy checks for BSC, SDLC, and ADCCP protocols. The BSC protocol (Extended Binary-Coded Decimal Interchange Code, EBCDIC) supports IBM 3780/2780 emulation. For switched X.21 operation, X.21 Call Establishment and Call Reception protocols are supported. The CP502 also supports download of character detect software.

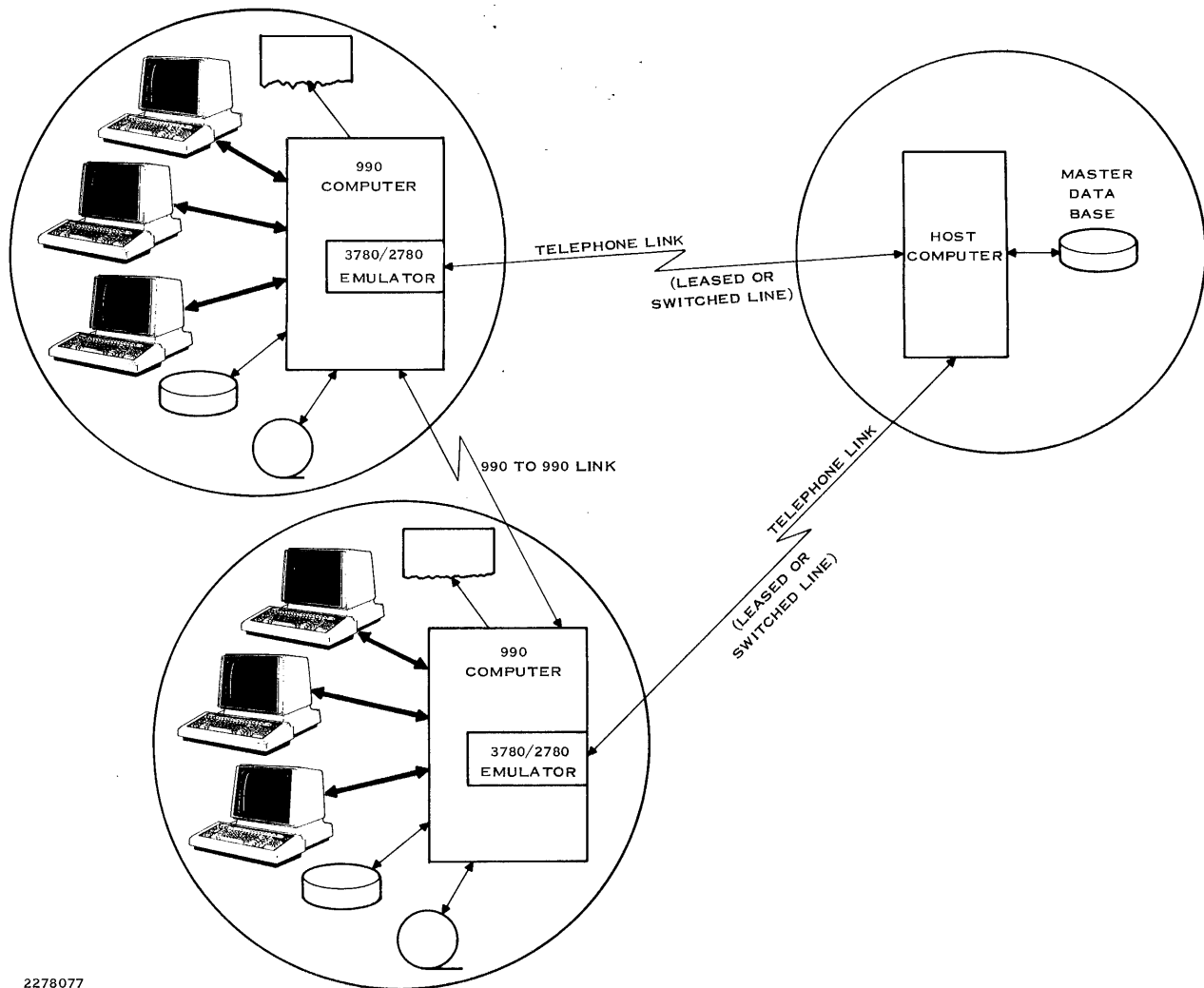
Synchronous protocols can be externally or internally clocked, NRZ (direct) or NRZI encoded, with bit rates up to 9600 baud.

### **11.3.2 3780/2780 Emulators Communications Software**

The 3780/2780 emulator 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 as shown in Figure 11-8.

The 3780/2780 emulator communications software simulates the operation of the IBM 3780 Data Communications Terminal and the IBM 2780 Data Transmission Terminal, respectively. 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 or input device can be the input source; likewise, any file or output device can be the output destination.

Using the 3780/2780 emulator, 990 computer systems can serve as satellite stations or as central stations in distributed-processing networks or can 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 operate in an unattended mode as a called station in a distributed network. The 3780/2780 emulator operates over switched communications lines, leased multipoint lines, or private communications lines. Communications require one of the synchronous controllers discussed earlier (CI421, CP501 or CP502), along with the appropriate modem.



2278077

Figure 11-8. Typical Application of 3780/2780 Emulator in Distributed Processing Environment

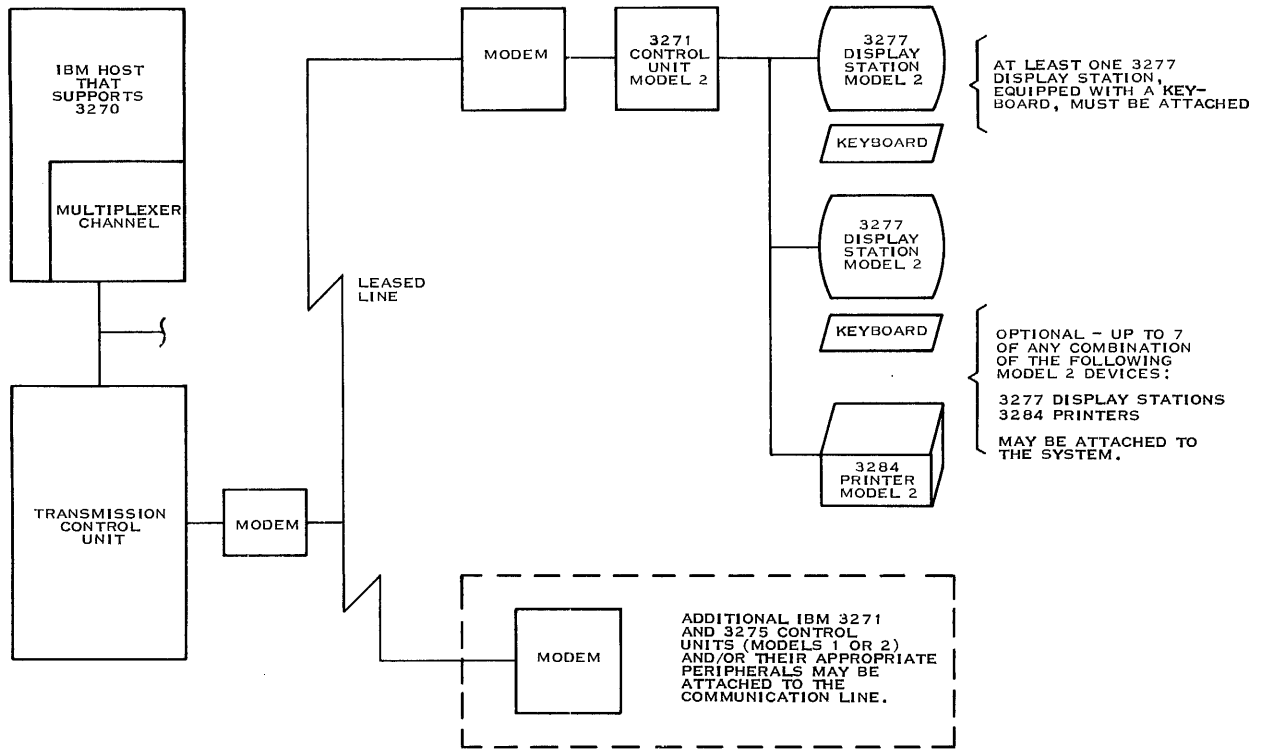
**11.3.3 3270 Emulator Interactive Communications Software**

The 3270 Interactive Communications Software (ICS) provides a means of connecting IBM main-frame computers to 990 computer systems. ICS allows interactive access to applications on the IBM that support the 3270 terminal family. The software provides interactive access through the 911 VDT, 931 VDT, or Business System Terminal. ICS also supports batch access through user-written COBOL, FORTRAN, Pascal, or 990 assembly language programs that control the linkable Programmed Station Control (PSC) emulator.

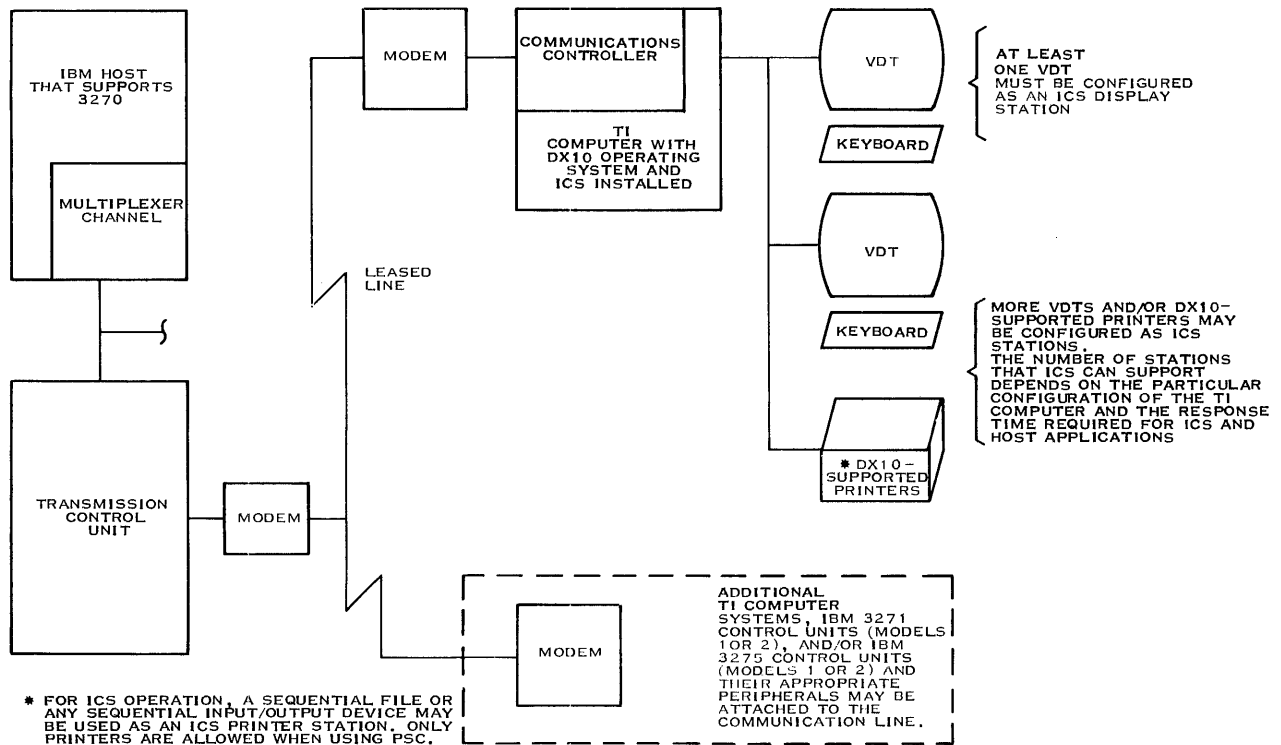
ICS operates over leased multipoint lines or private communications lines. Communications require one of the synchronous controllers discussed earlier (CI421, CP501 or CP502), along with the appropriate modem. When operating with ICS, a 990 computer can share a multipoint line with IBM 3270 terminals using BSC protocol.

ICS maintains comprehensive statistics about data-link and application performance to aid network trouble-shooting. Figure 11-9 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.

Application Programming Environment



IBM CONFIGURATION EMULATED WITH ICS



\* 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.

2280079

ICS CONFIGURATION

Figure 11-9. IBM 3270 Information Display System Configuration and ICS Configuration

## 11.4 HIGH-LEVEL PROGRAMMING LANGUAGES

DX10 supports several high-level languages:

- FORTRAN for mathematical and scientific applications
- COBOL for business environments
- Pascal for a variety of applications including system software development and scientific applications.
- BASIC for interactive scientific programming and business application programming
- RPG II for business applications requiring file maintenance or report generation.

### 11.4.1 FORTRAN

FORTRAN is a high-level computer language that allows complex problems to be stated in common mathematical expressions for input to the computer. The FORTRAN-78 software package provides for the processing of your source code in three major steps:

- Compilation, to translate the source code from FORTRAN programming language to machine code
- Linking, performed by the Link Editor, to join the compiled program with run-time support and called subroutines
- Execution

The FORTRAN-78 compiler translates FORTRAN language input code into computer machine code. The FORTRAN-78 compiler conforms to the American National Standards Institute (ANSI) standard FORTRAN 3.9-1978. 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.

Under your direction the Link Editor, an integral component of DX10, takes the modules of object code developed by the FORTRAN-78 compiler and links them together, resolving references between modules, to produce a single, executable module of code.

Texas Instruments has incorporated several useful attributes into the FORTRAN-78 compiler that provide for more effective coding and program development. These added features include:

- Overlapped I/O
- Free format source input
- Internal data manipulation statements
- 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.

The FORTRAN-78 function library includes all intrinsic functions and basic external functions defined in the ANSI standard. This library contains all run-time support to interface with the DX10 operating system. In addition, several useful routines such as a multiple-key indexed file handler are provided.

### 11.4.2 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 documentation and incorporates extensions to this subset to provide added capabilities. The compiler package employs the following ANSI standard COBOL modules at the level indicated.

#### Features

Level 1	Level 1 + *
Interprogram communications	Table handling
Library	Nucleus
Segmentation	Relative I/O
	Sequential I/O
	Indexed I/O

\* Selected features from level 2.

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

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

### 11.4.3 Pascal

Texas Instruments 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 use. Its readability makes the language especially useful when programs must be maintained by users other than the originator.

A common 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 useful for scientific or engineering applications that are usually written in FORTRAN or ALGOL. Its general-purpose structure is also useful for many business applications.

The minimum system required to support Pascal consists of the 990 CPU with 256K bytes (where K equals 1024) of memory, plus 10 megabytes of disk memory.



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 routine capability
- User-defined data structures that are adaptable to data used in application
- User-defined data types and type checking
- Excellent bit-manipulation capability

#### 11.4.4 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 BASIC, executable on all members of the 990 and Business System product lines
- An interactive reentrant editor/interpreter for creating and executing TI BASIC programs

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

#### 11.4.5 RPG II

RPG II (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, RPG II is especially suited for applications requiring file maintenance or report generation. A series of six basic specification formats is used to input the specific actions to be taken within the RPG II sequence execution.

Texas Instruments version of the RPG II language is closely compatible with the widely used IBM System/3 RPG II. Extensions of many of the System/3 features have been included in RPG II to provide more flexible programming. A utility program is provided with RPG II to copy System/3 or System/32 source programs or files from diskette to 990 disk files.

The RPG II package also includes an RPG II-oriented VDT text editor and a trace feature that prints each major step occurring in the execution of an RPG II program. An industry compatible sort/merge capability is provided by the optional Sort/Merge package. Communication of RPG II files is available through the optional DX10 3780 Emulator package.

The minimum system required to run RPG II consists of a 990 CPU with 128K bytes of memory and 10M bytes of disk memory.

The RPG II 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

## 11.5 PRODUCTIVITY AIDS

Texas Instruments supplies the TIFORM, TIPE, Sort/Merge, and CPG utility packages for application programming.

### 11.5.1 TIFORM

TIFORM 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 are both an interactive screen generator (formatter) 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, highlight, no display, etc.)
- Character and field level editing
- Up to 40 percent improvement in interactive application development time

### 11.5.2 Texas Instruments Page Editor (TIPE) Utility

The TIPE package offers word processing features for creating, editing, and printing pages of text that complement the data processing capabilities of the TI commercial computer systems. TIPE efficiently produces letters and documents, is easy to use, and requires minimal training. The TIPE package operates on standard 990 and Business System hardware, including VDTs and the 810 matrix printer. Letter-quality printing is also available, using the optional Model LQ45 printer.

The following functions are available with TIPE:

- Create a new TIPE document
- Edit an existing TIPE document
- File the document now being created/edited

- Disregard changes made since last Create or Edit Command
- Print a TIPE document
- Quit using the creation/editing program

### 11.5.3 Sort/Merge Utility

The DX10 system supports a comprehensive Sort/Merge package that can be accessed in several ways. SCI provides commands to access Sort/Merge in batch or interactive modes. COBOL, FORTRAN, BASIC, Pascal and assembly language programs can 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 input files. The sort process allows the following:

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

Figure 11-10 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.

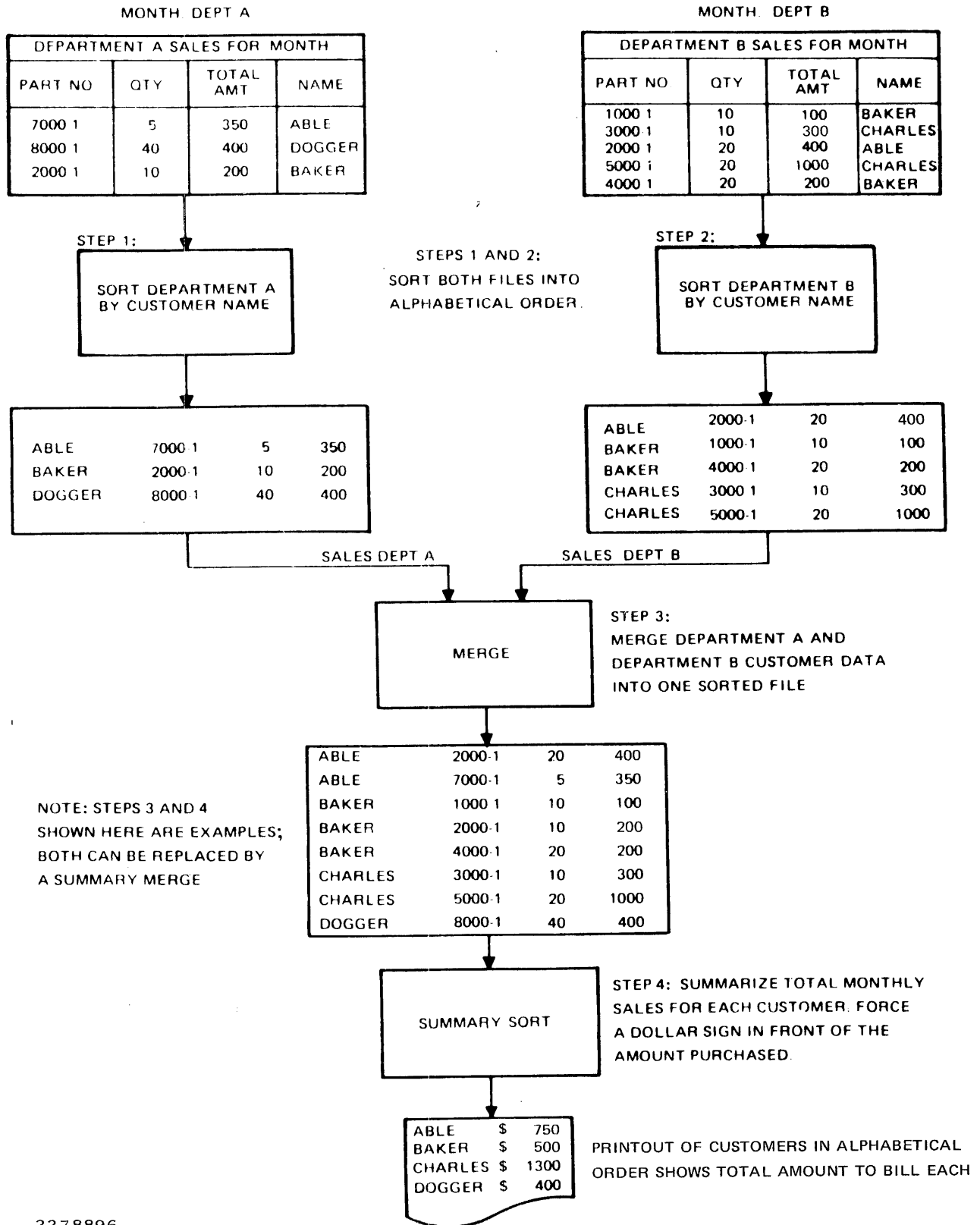
### 11.5.4 COBOL Program Generator (CPG)

CPG generates complete COBOL source programs that can be compiled and executed under DX10 or any system that supports RM™/COBOL. CPG reduces the time and effort normally associated with COBOL program development by performing routine coding that is tedious and time-consuming. Menus, fill-in-the-blank screens, prompts, and explanation screens guide you in using CPG.

CPG generates both batch and interactive programs. Batch programs can update files and prepare reports with no user interaction. Interactive programs accept data entry from formatted screens or modify data files by adding to, deleting from, or performing calculations on the current data fields in the file.

The CPG system provides two levels of usage: standard and advanced. The standard operation of CPG can be used by both experienced and inexperienced COBOL programmers to generate programs that perform routine tasks such as preparing reports and accepting data entry for file updating. The advanced operation of CPG can be used by experienced COBOL programmers to generate complex programs and define all program parameters.

RM is a trademark of Ryan-McFarland Corporation.



2278896

Figure 11-10. Sort/Merge Process Showing Printouts of Results at Each Step

## 11.6 DATA MANAGEMENT TOOLS

DX10 supports several related data management packages that streamline file handling:

- Data Base Management System (DBMS)
- Query
- Data Dictionary (DD)

### 11.6.1 Data Base Management System (DBMS)

The DBMS (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 allows you 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.

**11.6.1.1 Features of DBMS.** The independence of the data definitions from the application software allows modification of the data base without impact to existing programs. It 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.) The 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 provides optional password security for the most elementary data level; this provides control and protection of the data base from unauthorized access or tampering.

**11.6.1.2 DBMS User Interface.** The primary user interfaces to DBMS consist of the Data-Definition Language (DDL) and Data-Manipulation Language (DML).

DDL provides the means to completely describe the DBMS 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 data by supporting the reading and/or writing of DBMS data. DBMS 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 and specify a function to be performed, and the data element to be manipulated. DBMS processes the request and returns the results to the application program.

### 11.6.2 Query

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

**11.6.2.1 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.

**11.6.2.2 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.

### **11.6.3 DD**

The DD data dictionary allows you to define all the data used in an organization and store these definitions in a central location. This centralization aids in the enforcement of data standards, clarifies the impact of changes to the data, and limits data redundancy.

The DD system consists of a dictionary file, a data librarian, utilities, and a data manager. The dictionary file contains the definitions of data in other files. (Note that the dictionary file does not contain the actual data from these other files.) The dictionary file controls and maintains key indexed, relative record, sequential, and data base files. You use the data librarian to enter information into the dictionary. The data librarian is responsible for the accuracy of all definitions in the dictionary file. The utilities generate detail, summary, and cross-reference reports. This is an effective means of managing file definitions in the dictionary file and understanding the relationships of the definitions to each other. The data manager provides the DD interface to Query. This allows total control and access to conventional (non-DBMS) files for inquiry processing.

DD can be used in a stand-alone manner or in combination with Query and DBMS for full data file control. Query does not require the data manager to access data base files.

# Program Management

---

## 12.1 GENERAL INFORMATION

DX10 provides the means for you to install and execute your own application programs. This section discusses program structure and the characteristics of program installation and execution.

User programs operating under control of the DX10 operating system can 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 Link Editor separates the program into segments of data and segments of procedure. DX10 loads the linked segments into any available memory areas. These areas in memory may be located together or they may be widely separated. The 990 hardware memory mapping facility handles the relocation of programs.

An active program may be rolled in and out of various memory locations several times by DX10 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 logically contiguous program address space.

DX10 provides services to tasks through SVCs. These common routines handle the complex functions of file and device I/O, task control, memory control and other service needs. Assembly language tasks have direct access to SVCs. High level language statements requiring SVCs are processed by the compiler or interpreter. Services that may be needed by a program written in a high level language but which are not available through the language can often be made available. Consult the appropriate language programmer's guide for instructions on accessing SVCs.

## 12.2 TASKS AND PROGRAMS

A program is an ordered collection of instructions that directs the activities of a computer. Under DX10, a task is a specific activation of a program. 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.



### **12.3 SHARED PROCEDURES AND REPLICATED TASKS**

DX10 provides an efficient task replicating mechanism that allows several concurrent executions of a single program. Such is the case in many multiterminal environments or in industrial applications where several similar device types are controlled. An example might be a bookkeeping program interacting with several bank tellers concurrently. Similarly, multiple copies of one program can control many 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 data 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 eliminating the necessity of replicating the procedural part of a program. Conversely, the task portion is unique to each separate execution.

In cases where each concurrent program activation has the same initial state (data), only one program image need be stored on disk. Each task needed may be replicated from a single image installed in a program file on disk for each activation. Replication of tasks 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.

### **12.4 SHARING DATA AMONG TASKS**

Sometimes two or more tasks need to share a block of data. 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. A program can update or modify data in a shared procedure with the sharing tasks using the updated version.

An alternative way to share data is by storing it in the system common.

### **12.5 OVERLAYS**

A large program may need to be partitioned 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 module of the program, called the root, remains in memory during the entire execution of the program. The Link Editor, under your control, divides the remainder of the program into disk-resident overlays.

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. The Link Editor, following your specifications, may further segment overlay modules into a lower level of root and overlays.

## 12.6 SAMPLE APPLICATION OF TASKS AND PROCEDURES

Figure 12-1 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 replicated, but the procedural part for each task occupies memory. Generally, choose this scheme when you judge that the task is small enough or is executed infrequently enough that its execution does not overload your system's memory.

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.

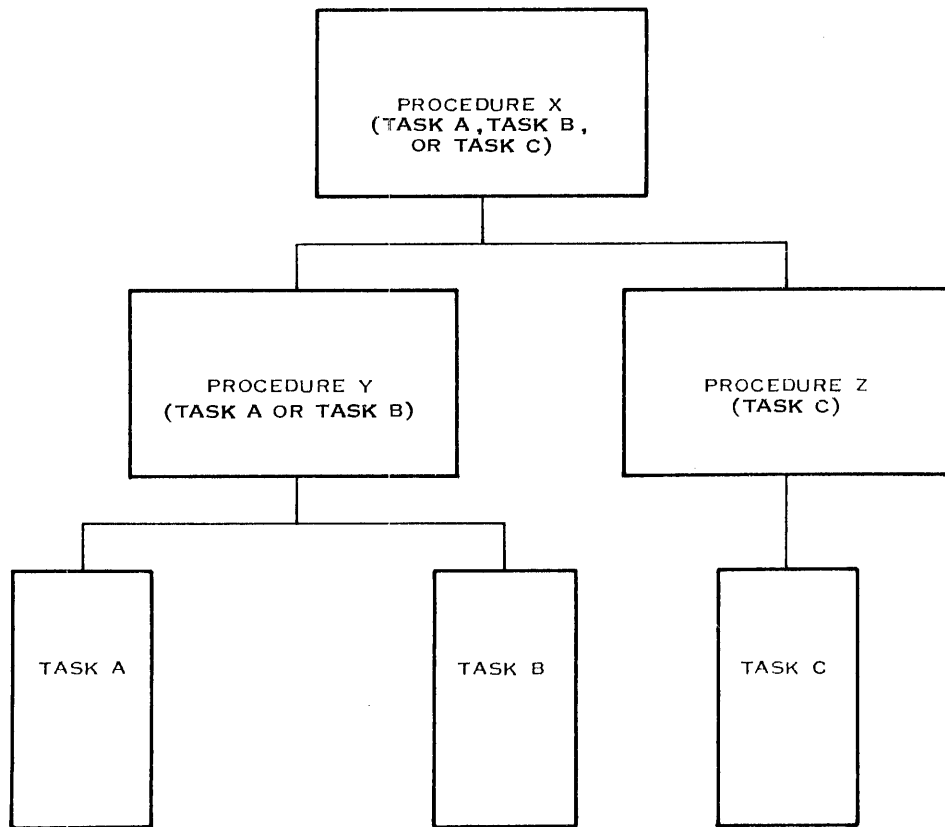
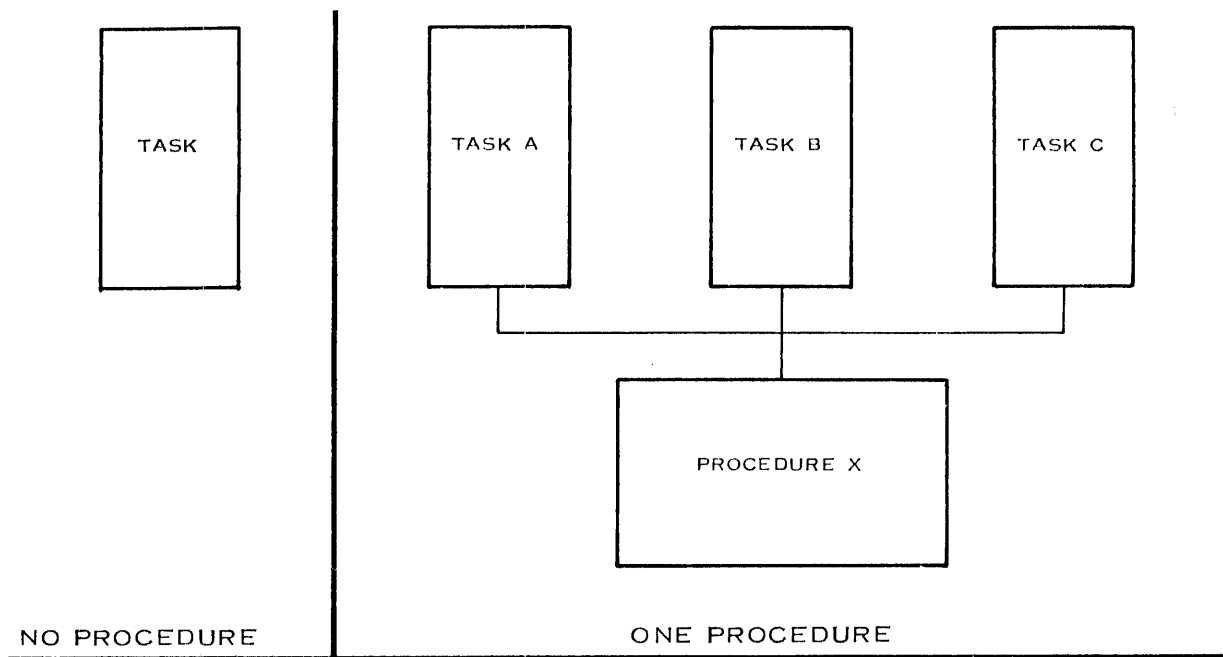
The third sample structure is an example of tasks operating with two procedures. A program written in COBOL may operate in this manner. One procedure (X in the example) that may be shared by all stations is the COBOL run-time support package. This package is required in support of any application program written in COBOL. A second procedure is the procedural part of the COBOL program itself (Y and Z). Note that multiple terminals, each with its own task, 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.

## 12.7 TASK ACTIVATION

Any task can activate another task. As a result, both tasks are active concurrently. DX10 identifies a task with the station that calls it. As a result of this station/task link, all the station-local LUNO assignments are available to the new task. Also, the requesting task may suspend itself until the activated task terminates. 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. When the subprocess completes in the former case, the master program resumes execution.

A frequent employment of this task-activated-task feature occurs when you execute a task through the SCI. Although the SCI appears to be the voice of DX10, it is actually the output of a task named SCI990. When you command the SCI to execute a task (using the XTS command), SCI990 activates the task and suspends itself until the called task is completed.

Additionally, a DSR can activate a task. 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.



2283117

TWO PROCEDURES

Figure 12-1. Task/Procedure Structure

## 12.8 PRIORITY SCHEDULING

DX10 provides for efficient sharing of CPU time by more than one active task through a system of priority scheduling. When you install a task, you are required to assign it a priority level. DX10 schedules tasks to use the CPU in order of their assigned priority levels. DX10 always schedules the waiting task with the highest priority to be executed next.

The following two paragraphs describe the available priority levels and the events that trigger scheduling.

### 12.8.1 Priority Levels

DX10 requires that each task have a defined priority level. There are 132 priority levels. The following list displays the assignable levels in order of their priority:

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

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

The assignment of real-time priority levels provides your tasks with the capability to supersede 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.

Assignment of priority levels one, two, three, and four satisfies 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.

Assign priority level three to batch executed tasks that do not require user interaction. Tasks at this level can access the CPU only when no higher priority tasks (interactive, real-time, or system) are waiting for execution.

The floating-priority level four provides rapid response to input/output events and de-emphasizes the task during periods of heavy CPU operation. Application programs that function interactively are typically installed with a level four priority assignment. When a task with level four priority begins to execute, DX10 initially sets it to priority level one. DX10 lowers the task's priority level to level two after a specified number of time slices when the task is performing computations. DX10 sets the task's priority level to one when the task is performing terminal input and to two when the task is communicating with other I/O devices.

### 12.8.2 Scheduling Operation

DX10 reschedules all waiting tasks 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).
- A time slice of the executing task expires (if time slicing is enabled).
- A task which was in a time delay becomes active.
- An I/O operation completes for the task waiting for it.

When a peripheral device sends an external interrupt to the CPU, the appropriate DSR may request that a program be scheduled for execution. DX10 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.

The task sentry monitors the length of time that a task runs. If a task continues execution without voluntarily suspending execution, the Task Sentry halts that task and lowers its priority. Note that you enable the Task Sentry and set the Task Sentry value during system generation. If the Task Sentry lowers the priority of the currently executing task, DX10 reschedules all waiting tasks in case a task of higher priority is in the system.

Time slicing allows the active tasks of the highest priority to share the CPU, each task receiving a slice of CPU time in turn. After a task has executed for its allotment of time, DX10 reschedules the tasks in the system to allow another task to execute. Note that you must enable time slicing and set the time slice value during system generation.

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

DX10 suspends tasks waiting for an I/O operation to complete. When the I/O operation completes, DX10 makes the task active and reschedules all waiting tasks in case the newly active task has the highest priority.

## 12.9 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 so 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.

DX10 has its own program file, created during system build at the factory. This system program file, `.$$PROGA`, initially contains only programs that constitute parts of the DX10 operating system. You may add other programs to `.$$PROGA` if you need them to be memory resident. (Refer to paragraph 13.4 for a discussion of memory resident programs.) You should create your own program files to store application programs. For the most part the capabilities of the system and user program files are identical. The following list describes the differences between the system program file, `.$$PROGA`, and your own program files:

- Memory-resident programs and procedures must be installed on the single system program file.
- Procedures installed on the single system program file, `.$$PROGA` 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.
- A nonreplicable program installed on the system program file uses the same number for its run-time ID and its installed ID. A nonreplicable program installed on a user program file uses a run-time ID that differs from its installed ID. If a program is nonreplicable, multiple executions of that program cannot occur concurrently.

## 12.10 PROGRAM IDENTIFICATION

You can retrieve a program or its parts by task number, procedure numbers or overlay numbers specified at install time. The program file internal directory can also call a program by its name as well as its number.

You can install a task with or without the replicatable attribute. The activated image of a replicatable task is assigned a run-time identification number that differs from its installed number. A nonreplicable task installed on the system program file uses the same number as its run-time ID as for its installed ID. A nonreplicable task installed on a user program file is allocated a run-time ID that is different from its installed ID.

## 12.11 TASK SENTRY

The Task Sentry is an option you can select during system generation that guards against CPU lockout. CPU lockout occurs when a task of the highest priority retains control over the CPU without suspending itself; in this case, tasks with a lower priority are effectively locked out of the CPU and the higher priority task is compute bound. Since DX10 always executes the highest priority task in the system, lower priority tasks can be locked out 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. The 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 restores the task to its proper priority.

## 12.12 SUPERVISOR CALLS (SVCs)

Assembly language application programs request service from DX10 by issuing SVCs. An SVC is initiated by a 990 instruction that transfers execution control to the operating system. Each 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. Volume III discusses each SVC and how to use them. Certain other unlisted SVCs are available only for privileged tasks and are explained in the Volume V.

**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.
    - Intertask communications.
    - Log a message.
    - Fetch time and Julian date.
-

# Memory Management

---

## 13.1 GENERAL INFORMATION

This section discusses three major topics concerning memory management by DX10:

- Dynamic allocation of memory
- Roll-in/roll-out
- Memory-resident tasks

DX10 uses the 990 mapping option to dynamically allocate memory to disk-resident task and procedure segments, as well as file blocking buffers. DX10 uses a mechanism called roll-in/roll-out to manage memory efficiently. When a program requires the use of a disk-resident task or procedure, DX10 rolls the program into dynamic memory. If a program with higher priority needs to use that memory space, DX10 rolls the lower-priority program back out to disk memory until memory space is available for it. In cases in which it is desirable, a program may be memory-resident and bypass the roll-in/roll-out mechanism.

## 13.2 DYNAMIC ALLOCATION

DX10 places tasks and procedures in memory wherever space is available. The memory mapping feature of the 990/10A and 990/12LR permits dynamic memory allocation. For the purpose of making the most efficient use of available memory, DX10 can execute a program with as many as three separate areas of physical memory. Although an application program consisting of a task and two procedures has three segments physically separated in memory, the mapping feature causes the program to see a virtual environment as described in the following:

- All three segments appear to be contiguous in memory with no gaps in addressability.
- The first segment appears to begin at memory address 0.
- The maximum addressable memory space for any one program is 64K bytes.
- Each program segment of a program must begin on a 32-byte boundary. The Link Editor locates procedures and tasks on such boundaries.



### 13.3 ROLL-IN/ROLL-OUT

DX10 memory management features permit 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 writes those programs to disk. This process is called roll-out. Similarly, when the task and priority levels indicate, DX10 rolls the program back in from the disk and execution resumes. The memory mapping feature permits a program segment to be restored into available physical memory space, possibly different from that which it had occupied at the time it was rolled out. In this way, the roll-in/roll-out mechanism guarantees high priority tasks immediate access to memory, allowing the task to respond rapidly 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.

DX10 allocates space in dynamic memory to file blocks as well as to tasks and procedures. DX10 writes to disk memory any file blocks left in memory after a task completes if that memory space is needed. DX10 writes these file blocks to their appropriate file location on the disk.

### 13.4 MEMORY-RESIDENT TASKS

Tasks are usually 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-resident tasks are loaded when the system is loaded from the disk (that is, during initial program load) and remain in memory even after execution termination. Since the task is memory-resident, it is never rolled out, saving operating system overhead by eliminating disk accesses otherwise required for roll-in/roll-out. Because they are never rolled out, memory-resident tasks reduce the size of available dynamic memory. You must install memory-resident tasks in the system program file (S\$PROGA). Memory-resident tasks can be installed in a user created program file, but will never be loaded by the system as memory-resident.

# Appendix A

## Keycap Cross-Reference

---

Generic keycap names that apply to all terminals are used for keys on keyboards throughout this manual. This appendix contains specific keyboard information to help you identify individual keys on any supported terminal. For instance, every terminal has an Attention key, but not all Attention keys look alike or have the same position on the keyboard. You can use the terminal information in this appendix to find the Attention key on any terminal.

The terminals supported are the 931 VDT, 911 VDT, 915 VDT, 940 EVT, the Business System terminal, and hard-copy terminals (including teleprinter devices). The 820 KSR has been used as a typical hard-copy terminal. The 915 VDT keyboard information is the same as that for the 911 VDT except where noted in the tables.

Appendix A contains three tables and keyboard drawings of the supported terminals.

Table A-1 lists the generic keycap names alphabetically and provides illustrations of the corresponding keycaps on each of the currently supported keyboards. When you need to press two keys to obtain a function, both keys are shown in the table. For example, on the 940 EVT the Attention key function is activated by pressing and holding down the Shift key while pressing the key labeled PREV FORM NEXT. Table A-1 shows the generic keycap name as Attention, and a corresponding illustration shows a key labeled SHIFT above a key named PREV FORM NEXT.

Function keys, such as F1, F2, and so on, are considered to be already generic and do not need further definition. However, a function key becomes generic when it does not appear on a certain keyboard but has an alternate key sequence. For that reason, the function keys are included in the table.













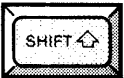






























Multiple key sequences and simultaneous keystrokes can also be described in generic keycap names that are applicable to all terminals. For example, you use a multiple key sequence and simultaneous keystrokes with the log-on function. You log on by *pressing the Attention key, then holding down the Shift key while you press the exclamation (!) key*. The same information in a table appears as *Attention!(Shift)!*.

Table A-2 shows some frequently used multiple key sequences.

Table A-3 lists the generic names for 911 keycap designations used in previous manuals. You can use this table to translate existing documentation into generic keycap documentation.

Figures A-1 through A-5 show diagrams of the 911 VDT, 915 VDT, 940 EVT, 931 VDT, and Business System terminal, respectively. Figure A-6 shows a diagram of the 820 KSR.

Table A-1. Generic Keycap Names

Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820 <sup>1</sup> KSR
Alternate Mode	None				None
Attention <sup>2</sup>		 			 
Back Tab	None	 	 	None	 
Command <sup>2</sup>					 
Control					
Delete Character					None
Enter					 
Erase Field					 




























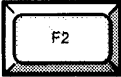











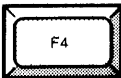





Notes:

<sup>1</sup>The 820 KSR terminal has been used as a typical hard-copy terminal with the TPD Device Service Routine (DSR). Keys on other TPD devices may be missing or have different functions.

<sup>2</sup>On a 915 VDT the Command Key has the label F9 and the Attention Key has the label F10.

2284734 (2/14)













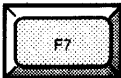





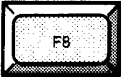





















Table A-1. Generic Keycap Names (Continued)

Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820' KSR
Erase Input					 
Exit			 	 	
Forward Tab	 			 	 
F1					 
F2					 
F3					 
F4					 

Notes:

The 820 KSR terminal has been used as a typical hard-copy terminal with the TPD Device Service Routine (DSR). Keys on other TPD devices may be missing or have different functions.














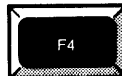






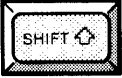









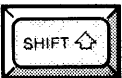


















**Table A-1. Generic Keycap Names (Continued)**

Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820 <sup>1</sup> KSR
F5					 
F6					 
F7					 
F8					 
F9	 			 	 
F10	 			 	 

**Notes:**

<sup>1</sup>The 820 KSR terminal has been used as a typical hard-copy terminal with the TPD Device Service Routine (DSR). Keys on other TPD devices may be missing or have different functions.

Table A-1. Generic Keycap Names (Continued)

Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820 <sup>1</sup> KSR
F11	 			 	 
F12	 			 	 
F13	 	 	 	 	 
F14	 	 	 	 	 
Home					 
Initialize Input		 			 

Notes:

<sup>1</sup>The 820 KSR terminal has been used as a typical hard-copy terminal with the TPD Device Service Routine (DSR). Keys on other TPD devices may be missing or have different functions.














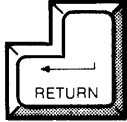
















Table A-1. Generic Keycap Names (Continued)

Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820 <sup>1</sup> KSR
Insert Character					None
Next Character	 or  				None
Next Field	 		 	 	None
Next Line					  or 
Previous Character	 or 				None
Previous Field		 			None

Notes:

<sup>1</sup>The 820 KSR terminal has been used as a typical hard-copy terminal with the TPD Device Service Routine (DSR). Keys on other TPD devices may be missing or have different functions.

Table A-1. Generic Keycap Names (Continued)

Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820 <sup>1</sup> KSR
Previous Line					 
Print					None
Repeat		See Note 3	See Note 3	See Note 3	None
Return					
Shift					
Skip					None
Uppercase Lock					

Notes:

<sup>1</sup>The 820 KSR terminal has been used as a typical hard-copy terminal with the TPD Device Service Routine (DSR). Keys on other TPD devices may be missing or have different functions.

<sup>2</sup>The keyboard is typamatic, and no repeat key is needed.

228 4734 (7/14)



**Table A-2. Frequently Used Key Sequences**

<b>Function</b>	<b>Key Sequence</b>
Log-on	Attention/(Shift)!
Hard-break	Attention/(Control)x
Hold	Attention
Resume	Any key

**Table A-3. 911 Keycap Name Equivalents**

<b>911 Phrase</b>	<b>Generic Name</b>
Blank gray	Initialize Input
Blank orange	Attention
Down arrow	Next Line
Escape	Exit
Left arrow	Previous Character
Right arrow	Next Character
Up arrow	Previous Line

2284734 (8/14)

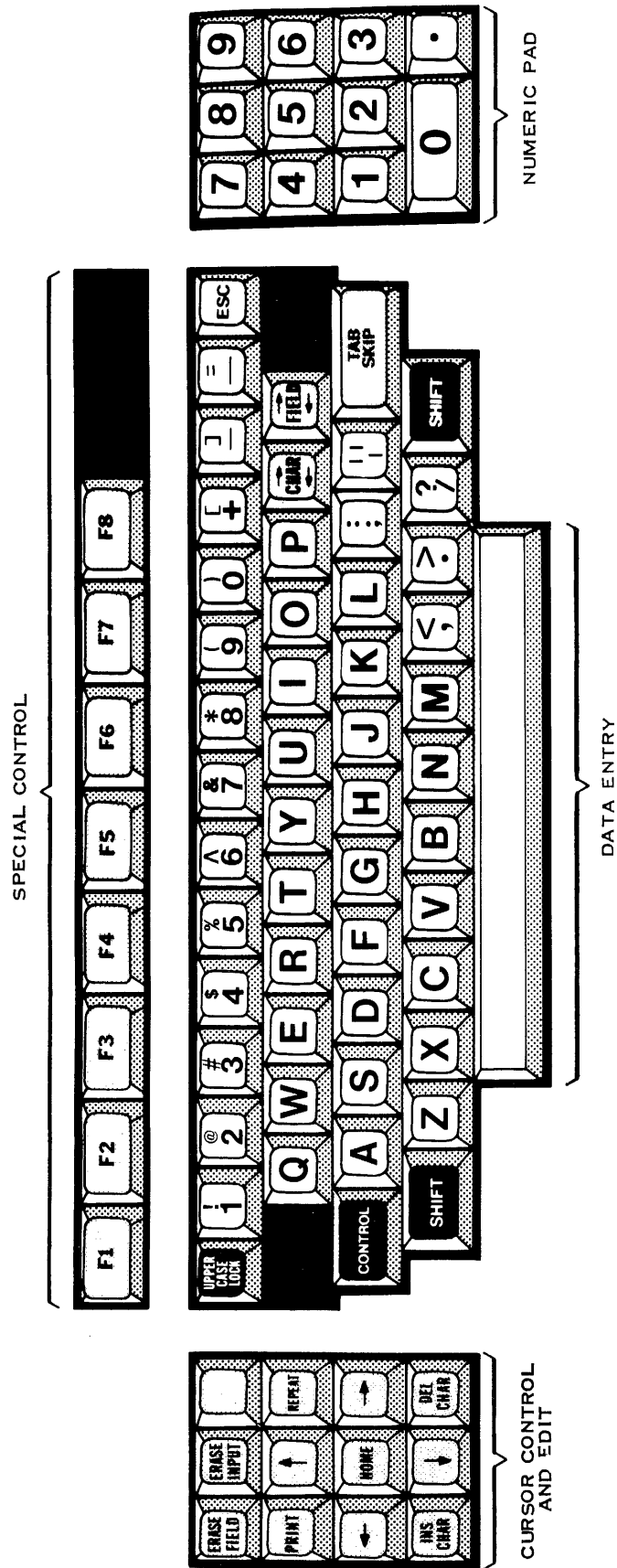


Figure A-1. 911 VDT Standard Keyboard Layout

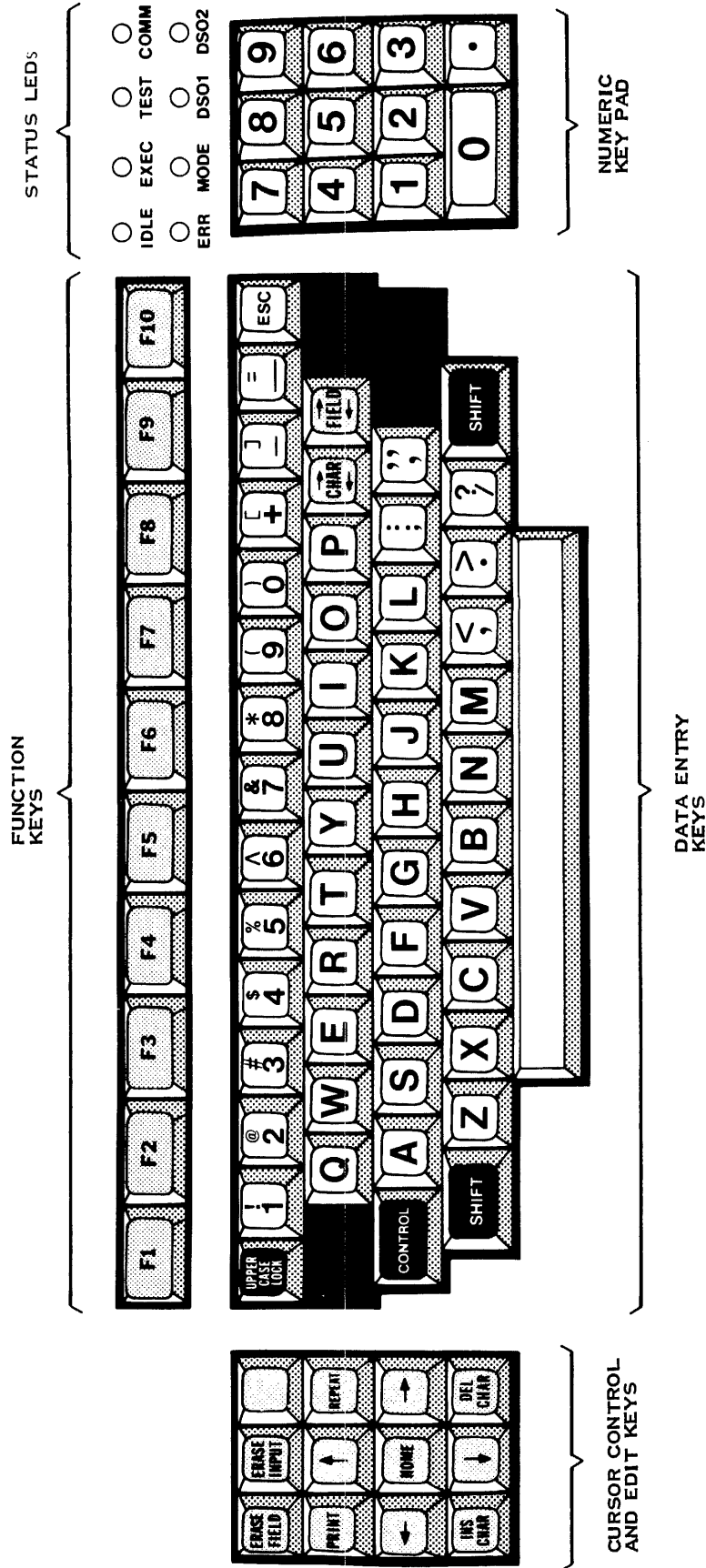


Figure A-2. 915 VDT Standard Keyboard Layout

2284734 (10/14)

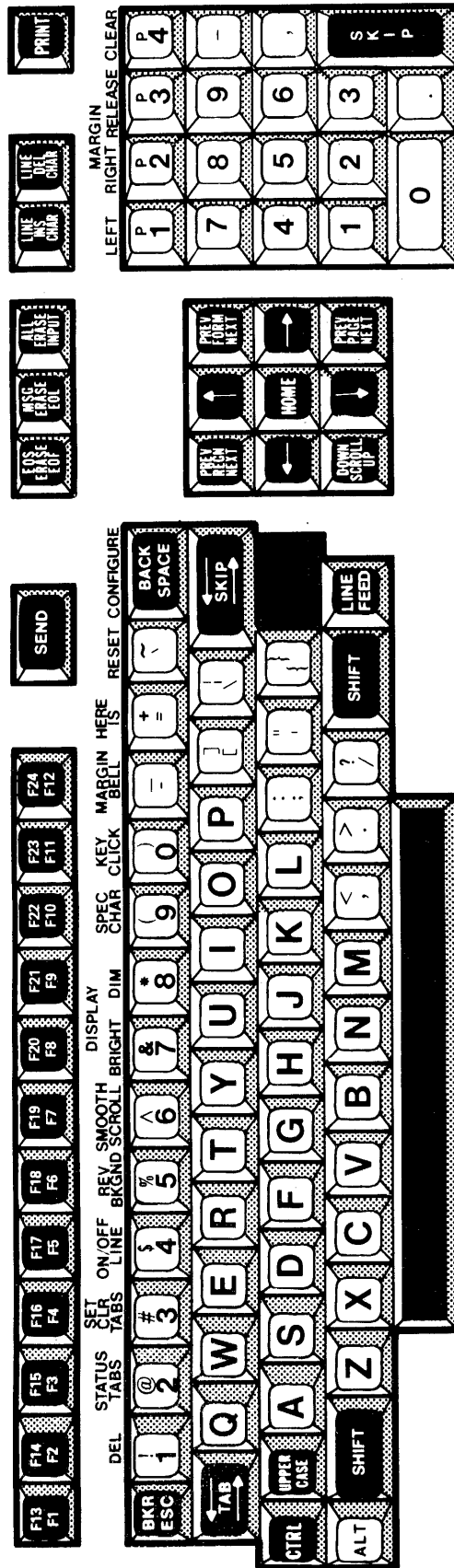
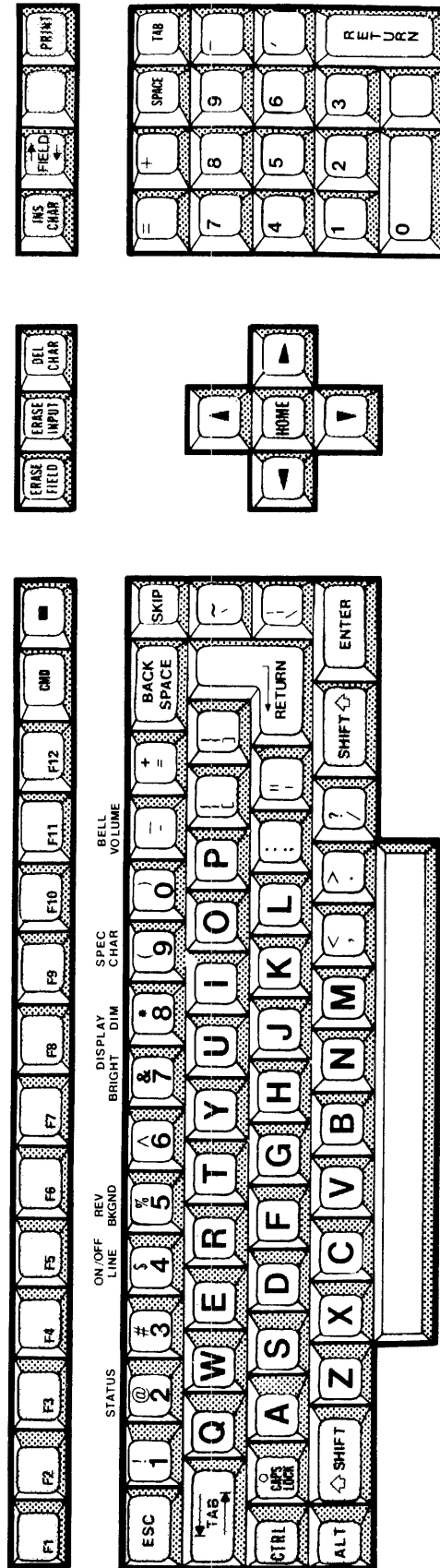


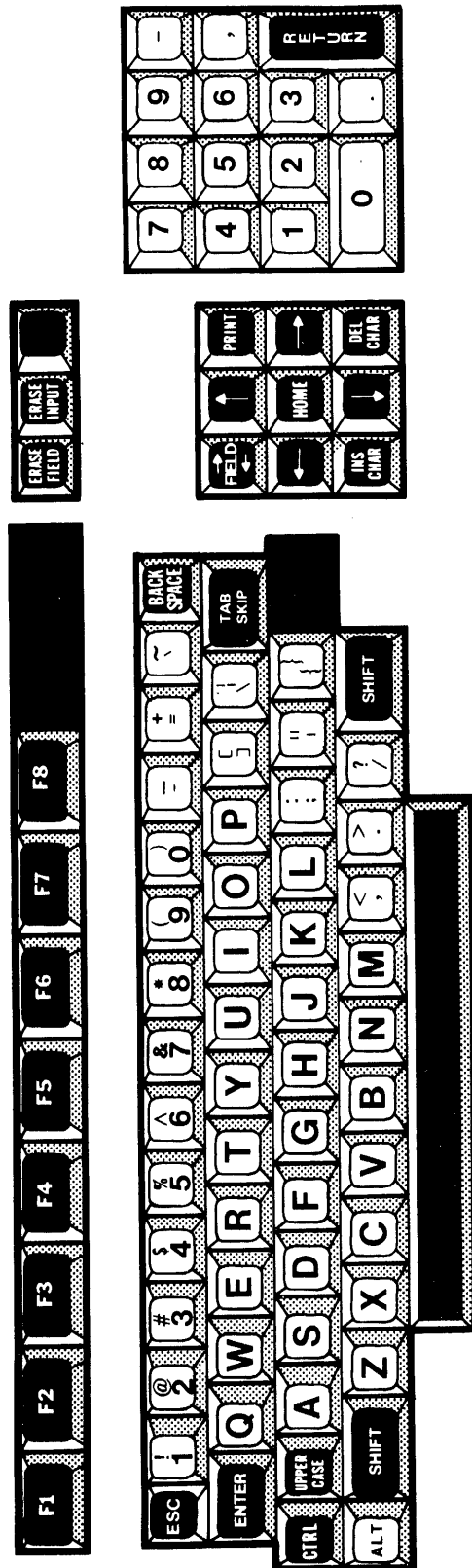
Figure A-3. 940 EVT Standard Keyboard Layout

2284734 (11/14)



2284734 (12/14)

Figure A-4. 931 VDT Standard Keyboard Layout



2284734 (13/14)

Figure A-5. Business System Terminal Standard Keyboard Layout

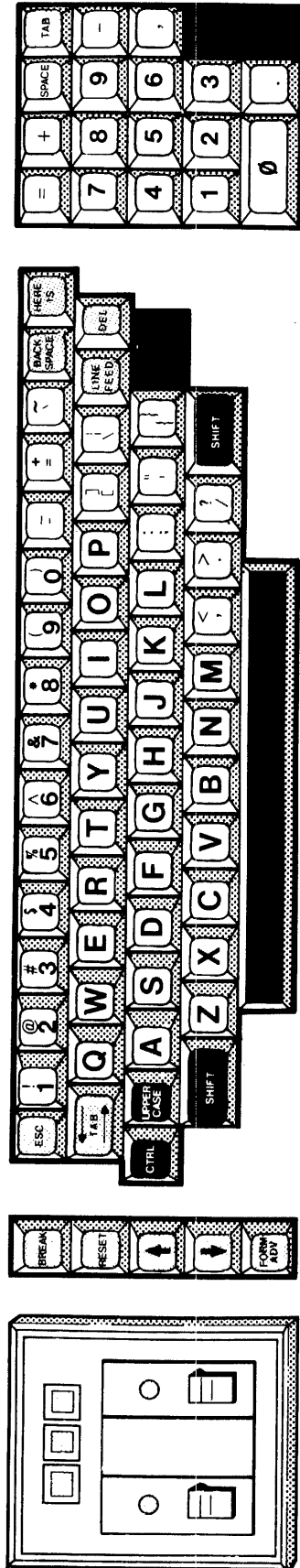


Figure A-6. 820 KSR Standard Keyboard Layout

2284734 (14/14)

# Glossary

---

The terms defined in this glossary are used to describe the structure and operation of DX10. All members of the DX10 family of manuals contain some of these terms. The terms are arranged alphabetically.

**Access Name** — A name used to access an I/O resource. An access name can be a device or volume name, or a file pathname.

**Access Privileges** — An attribute of a task that defines the task's mode of access to a particular file. The task is said to have one of the following types of access privileges:

- Shared access (more than one task can read and write)
- Read only access (this task will only read)
- Exclusive write access (while others can read, only this task can both read and write)
- Exclusive all access (only this task can read or write)

**Active State** — The state of a task that is ready to use CPU time. An active task has all of its required resources except for CPU time. See Executing Task.

**Address** — A group of characters that specifies the location of an area of memory. The operating system uses addresses to correctly access data and instructions. For example, an address can be 4 hexadecimal digits that represent the location of a byte of memory.

**Address Space** — The memory that can be addressed by an addressing scheme. The memory that a task can address differs from the memory that the computer can address. A task can address 65,504 bytes while the computer can address 2048K bytes. This difference is resolved by the mapping hardware and DX10.

**ADU** — See Allocatable Disk Unit

**Alias** — An alternate for a file pathname component.

**Allocatable Disk Unit (ADU)** — The smallest disk space that can be allocated for file creation or expansion. This space varies from 1 to n sectors depending on the type of disk drive.

**Application Program File** — A program file on which users install applications programs that are not memory resident.

**Array** — A logical data structure consisting of one or more ordered sets of elements of the same type (such as integers or literal strings.)



ASR — See Automatic Send/Receive

Asynchronous Transmission — Transmission in which the receiving device is not dependent upon the time relationship of the sending device. Start and stop bits frame each character to maintain necessary control of timing.

Automatic Send/Receive (ASR) — A line-oriented terminal device with attached cassette tape drives or paper tape drives.

Autocreate — The process performed by the operating system when the autocreate bit is set in the Assign LUNO supervisor call block, specifying that the operating system is to create the file to which the LUNO is being assigned if that file does not already exist.

Background Mode — A type of system use in which a task executes without interacting with the terminal. The user can do other processing in the foreground while a background activity proceeds. The initiated background task receives as its environment a copy of the current SCI environment and proceeds to run simultaneously with SCI. SCI commands entered in a batch stream always execute in background mode.

Base System — The initial system image shipped to the customer. The base system is a minimum system capable of performing system generation, and supports VDTs and multiple disk and tape drives.

Batch Mode — A mode of execution in which a task executes independent of a terminal. Under DX10, SCI commands can be executed from any terminal, sequential file, or sequential device. When the input device is other than a terminal, that is, 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.

Beet — A 32-byte block of memory starting at a memory address that is exactly divisible by 32.

Bid — To place a task in the active state (ready for execution), requesting the system bring the task into memory, if necessary, and place the task on the active list at the appropriate priority.

Bit — A binary digit. In the binary numbering system, each bit has a value of either one or zero.

Blank Adjusted — An attribute of records in a file, denoting the padding of records with trailing blanks on input, or the removal of trailing blanks on output. This attribute, when desired, is specified at the time of the I/O operation.

Blank Suppressed — An attribute of a file indicating that each occurrence of a string of consecutive blanks is replaced by a code that represents the number of blanks. Consecutive blanks are not actually stored. This attribute of a sequential file, when desired, is specified during file creation. It is used to reduce the disk space required for a file.

Blocked Data Transfer — The movement of data through a system-supplied buffer as an intermediary between a disk and the user buffer. This employs packing of one or more logical user records into a single physical record and reduces the number of physical accesses to the hardware storage device.

Blocked File — A file in which a physical record includes more than one logical record.

- Bootstrap** — A technique for loading the first few instructions of a routine into memory, and then using these instructions to load the rest of the routine. For DX10, this bootstrap sequence begins in a ROM loader after the initial program load sequence is keyed. Also see Initial Program Load.
- Boundary Error** — An error in which a task tries to access memory outside its allowed address space. The error causes task termination.
- Bounded File** — A file which cannot grow beyond its initial allocation.
- Breakpoint** — A place in a routine at which execution halts. Under DX10 a breakpoint is specified by XOP 15,15. Except in the controlled mode of the Debugger, breakpoints do not automatically return control to a monitor program.
- 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.
- Byte** — A group of bits operated on as a unit. The Business System computers use bytes consisting of eight bits.
- Cache Memory** — A portion of memory that operates much faster than primary memory and in which the controller stores frequently used data from primary memory.
- Call Block** — 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. Also see Supervisor Call Block.
- 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).
- Central Processing Unit (CPU)** — The arithmetic and logic unit of a computer.
- 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 directive to perform an action. The SCI is provided to interpret commands and to initiate utilities as needed to perform the desired functions.
- Command Mode** — A processor mode in which the processor interprets input as commands rather than as data. The Text Editor and system generation utilities have a command mode.
- Command Procedure** — A program in the SCI language that implements a command.
- Command Processor** — A task bid by a command procedure to perform an SCI command or part of an SCI command.

**Common** — The System Common memory area; 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.

**Communications Register Unit (CRU)** — The general-purpose, command-driven I/O mechanism of the Business System computer. The CRU is used to control I/O to low-speed peripherals such as video display terminals, cassette drives, EIA devices, card readers, line printers, communication devices, and ASR and KSR devices.

**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. This is the mode enabled when creating a new file.

**Concurrent Tasks** — Tasks which are simultaneously active. Only one task may be executing at a given instant. See also Active Task and Executing Task.

**Configuration File** — A configuration file is a file describing a DX10 System created by the system generation utility.

**Context Switch** — A transfer of control (to a program or subroutine) that 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. The *990/10 and 990/12 Assembly Language Reference Manual* discusses context switching fully.

**Country Code** — An indicator carried within system data structures to indicate the country, or language character collating sequence, that was specified for support during system generation.

**CPU** — See Central Processing Unit

**CPU-Bound** — A characteristic of a program that spends more execution time in CPU operations than I/O operations. Also see I/O-bound.

**Crash Code** — A numeric code displayed on the programmer panel indicator lights (if present) showing that the operating system has detected an uncorrectable system error and has aborted itself (crashed). Most Business System computers do not have a programmer panel.

**CRU** — See Communications Register Unit

**Currency** — A block of memory that contains information about the current record position while processing a file.

**Customer Representative** — System software support personnel; the person or agency responsible for providing support to you for software related problems. If you purchased your software directly from Texas Instruments, then you have access to the Customer Support Line for help with software problems. If you bought your system (or software) from an intermediate dealer or an Original Equipment Manufacturer, then that dealer is your customer representative.

Texas Instruments is not responsible for error conditions that may degrade DX10 operation in systems where modifications have been made to its software by the user or intermediate vendors.

**Cylinder** — The group of tracks on the recording surfaces of a multi-platter disk that are accessed when the disk heads are in a particular location.

**Deadlock** — A system state in which two or more programs are stalled contending for resources in such a way that none can continue unless others release resources they already hold and none can release the resources they already hold. Also see Thrashing.

**Dealer** — The company that sold your computer or software to you. See Customer Representative.

**Debug** — To remove flaws from a program. Debug aids under DX10 include extensive error detection capabilities and the interactive Debugger.

**Default Value** — A value supplied by the operating system if a user enters a null response to a field prompt. Also see Initial Value.

**Deferred Write** — A file attribute that causes logical records to be written to disk after they are buffered in memory. File I/O normally uses a deferred write operation to place logical records, being written to disk files, into an area of memory that corresponds to a physical record. For disk files, deferring disk writes can dramatically increase system throughput. When you create a file, deferred write is the default value. Also see Immediate write.

**Device** — Physical equipment such as a card reader or line printer to which DX10 allows input or output.

**Device Name** — A unique four-character pathname for a physical peripheral device. 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 (DSR)** — A routine that communicates directly with an I/O Device. As an intrinsic part of the operating system, it services interrupts, and performs necessary input and output operations.

**Diagnostics** — Information, messages, and routines that provide assistance in eliminating errors and clarifying special conditions. See Online Diagnostics.

**Directory** — A relative record file that contains the information necessary to locate other files and describe the characteristics of those files. It does not contain user data.

**Disk-Resident** — An attribute of a task that resides on disk when it is in a terminated state or is not yet used. Also see Memory-Resident.

**Disk Volume** — A disk cartridge that has been named and initialized; a disk volume may be installed on a disk drive and be known to the system by the volume name.

**DSR** — See Device Service Routine.

**Drive** — A peripheral device that holds and operates a disk volume, magnetic tape reel, a cassette, or other similar medium.

**Edit Mode** — A mode in the Text Editor in which lines may be entered in a text file.

**End Action** — A routine specified by a task, that is to be executed before aborting the task if a fatal error occurs or if the task is killed. 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 on a file or device.

**Error Code** — A numeric code returned when an error is detected. Depending upon the cause of the error, this code can be returned in a supervisor call block, displayed to a terminal, or displayed on the programmer panel.

**Event Character** — Characters generated by certain keys (event keys) on the keyboard that have special significance to executing tasks.

**EVT** — Electronic Video Terminal. A type of VDT.

**Executing Task** — The task that has control of the CPU.

**Expandable File** — A file that can be extended beyond its initial size.

**Fatal Error** — An error in a task that causes the task to be terminated. Examples of fatal errors include boundary errors, memory parity errors, use of illegal instructions, and other non-recoverable situations.

**Field Prompt** — A word or phrase displayed to the user by an SCI command procedure in order to gather information needed by the procedure. Also see “prompt.”

**Field Service** — Repair and maintenance service at your computer installation provided by your dealer.

**File** — A named, organized collection of data records. Disk files can be organized in a sequential, relative record, or key indexed format. Special forms of relative record files include directory files, program files, and image files.

**File Attribute** — A declared characteristic of a file that limits the types of operations 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 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 and no other task can access it until the requesting task releases it.
- Foreground** — The mode in which an executing task interacts with the terminal. After SCI bids a foreground task, SCI is suspended and releases access to the terminal and the foreground terminal local file. The task that is bid shares the environment in use by SCI. Also see Background Mode.
- Front Panel** — See Programmer Panel.
- Function Key** — A key on a terminal, such as the F4 key, that causes the system to perform a pre-defined function for the user.
- Generate Mode** — A mode in the system generation program in which a new system configuration is described.
- Global LUNO** — A logical unit number that is available to any task or station.
- Hard-Copy Terminals** — An I/O device that prints its output onto paper. Also see Teleprinter.
- Hashing** — A technique for mathematically processing key words or filenames to produce numbers, usually record numbers.
- Hexadecimal** — A numbering system that uses a radix of 16. Hexadecimal values are indicated to SCI by a leading right angle bracket, ">", or by a leading zero, "0."
- Hollerith Code** — An alphanumeric coding system for punched cards.
- Image File** — A special form of relative record file that usually 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** — A file attribute that forces output to a disk file to occur immediately. Ordinarily, file I/O is buffered in memory and writing is deferred until the memory space is required.
- Initial Program Load (IPL)** — The operation of pressing a series of buttons on the Business System computer chassis that trigger the work of the ROM loader in performing the bootstrap sequence when the computer is first powered up. The term is also used to refer to the entire sequence of placing the initial program load program in memory and loading the operating system.
- Initial Value** — A value shown next to a field prompt from an SCI command procedure. You can choose to use this value as the response to the prompt or can replace it with another response. If the initial value is replaced by a null value, the system will use a default value for the field prompt, if one exists. The initial value is specified by the command procedure to be a constant value or to be the current value of some synonym. Also see Default Value.
- Initialization, Disk** — The process of writing required system overhead tracks and formatting a disk prior to using the disk under DX10.

**Initiate I/O** — An I/O call that causes I/O to be started but does not cause the requesting task to be suspended while the I/O is in progress. This mode of I/O is specified by a flag in the SVC block.

**Inquiry Mode** — A mode in the system generation program in which a new system configuration is described through specific prompts.

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

**Installed ID** — A unique identification number from >1 to >FF assigned to a task, procedure, or overlay that is installed in a program file. The operating system uses this ID to locate that module in the program file.

**Interrupt** — A coded signal that can transfer CPU control; thus enabling the CPU to service devices, power up/down procedures, and error conditions. The Business System computers support interrupts with 16 levels of priority. The interrupt that is currently executing prevents interrupts of a lower priority from interrupting the CPU.

**Interrupt Service Routine** — A system module that directs the system to take action according to the interrupt received. Examples include: power up or down, error instructions, and clock operations.

**I/O** — An abbreviation for input/output operations. Input/output operations involve transferring information between the CPU and peripheral devices.

**I/O-Bound** — A property that describes a task when it spends more time waiting for I/O operations than executing CPU operations. See CPU-bound.

**IPL** — See Initial Program Load.

**Isochronous Transmission** — The process of transmitting and receiving asynchronous data (with start and stop bits) using a clocking connection between transmitter and receiver (synchronous mode).

**K** — An abbreviation for 1024, often used when referring to a number of bytes in memory.

**Key** — A character field within a record in a key indexed file, used to determine the order of the record in relation to other records in the file.

**Keyboard Send/Receive (KSR)** — A line-oriented terminal device, such as a teletypewriter.

**Keyboard Status Block (KSB)** — A block of memory used as a workspace by an interrupt service routine for a keyboard device.

**Key Indexed File (KIF)** — A file in which records may be accessed by the value of a character string called a key. Each record can have up to 14 unique keys, with access through each key independent of the other keys.

**Key Value** — A particular character string being used in a key field.

KIF — See Key Indexed File.

Least Recently Used Strategy — A strategy used by the operating system to choose task segments to be rolled out of memory. The segment that has been inactive the longest is considered the least likely candidate for immediate future use and is thus the first chosen for rolling to disk.

Link Control File — A file created by the user that contains instructions for the Link Editor.

Link Editor — A system utility that takes related object modules and links them together into a single object module.

Linked Object File — A file created by the Link Editor that contains one or more program object modules that have been linked together to produce linked object modules.

Load — To copy a task or segment 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 65,504 bytes.

Logical Device Table (LDT) — A memory structure in the system that contains a LUNO and pointers to system tables that correspond to the resource to which the LUNO is assigned.

Logical Record — A logical division of data in a file that can be transferred 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 — See Logical Unit Number.

Log-Off — The action that ends a work session. Using the System Command Interpreter, log-off is accomplished by typing a Q when the command prompt appears. (Q calls the QUIT procedure.)

Log-On — The action that begins a work session. It identifies a user to the system so he can gain access to it. Refer to Volume II for instructions for logging-on.

M — An abbreviation for 1,048,576, when referring to bytes of memory or disk storage.

Mark-Sense Card — A data-entry card designed so that a user can specify its data by marking it with a special conducting-lead pencil.

Memory — A device designed to store data.



**Memory Mapping** — A hardware feature of the 990 computer controlled by DX10, which allows the 990 to contain 2048K bytes of physical memory and use whichever segments of physical memory it decides for the segments comprising a task's logical address space. In this way, memory mapping allows control of system resources during roll-in and roll-out. Memory mapping is not available on some models of the Business System series.

**Memory-Resident** — The attribute assigned to a task indicating that the operating system loads this task into physical memory at the next initial program load and retains the task in memory even when it is in a terminated state.

**Modem** — A modulator-demodulator; a device that interfaces between the digital signals of a computer and the analog signals of data transmission wires.

**Module** — A set of computer program instructions treated as a unit by an assembler, compiler, link editor, or other similar processor.

**Multiprogramming** — A mode of operation in which more than one computer program can be in memory and queued for execution. This differs from multiprocessing, in which two or more programs can execute simultaneously.

**Multitasking** — Another term for multiprogramming. Several tasks may be in memory simultaneously, each allotted execution time in turn.

**Natural Load Address** — The address, relative to a task's address, at which an overlay is loaded.

**Nonblank Suppressed** — The file characteristic that indicates consecutive blanks in a record are stored as consecutive blanks rather than being stored in a compressed format.

**Nonexpandable File** — A file that cannot grow beyond its initial size.

**Object Code** — Machine language code together with load control code; object code is usually created by an assembler or compiler.

**Object File** — A file that contains one or more program object modules comprised of object code, usually created by an assembler or compiler.

**Online Diagnostics** — Nonprivileged diagnostic routines that operate concurrently with program execution. See diagnostics.

**Overlay** — 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.

**Password** — A character string supplied by a user for validation of access and security privileges, primarily for log-on.

**Pathname** — A character string that indicates a path to a resource such a file or device. For a file, the pathname components include an optional volume name, 0 to 24 directory names, and a final component identifying the file. A pathname should not exceed 48 characters in total length.

PC — See Program Counter.

PDT — See Physical Device Table.

Peripheral Device — Any equipment in a computer system that is separate from the CPU and can provide information, communication, and capabilities to the system. Examples include: disk units, VDTs, printers, and communication equipment.

Physical Address Space — The addressable memory of the computer. This term also refers to the range of memory addresses available to the computer although the memory may not actually be implemented in a particular configuration.

Physical Device Table (PDT) — A data structure associated with a device and used by device service routines.

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

Port — A physical interface between a processing unit and a peripheral device.

Powerup — An initial application of electric current to a computer or peripheral device.

Priority, Floating — One of four priority levels assigned to tasks to order their execution. DX10 supports four levels of priority (0, 1, 2, 3) for tasks, zero priority being the highest and 3 the lowest. 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 by DX10 to 1, 2 or 3, depending on the type of activity of the task (I/O, Terminal I/O, and computation are considered).

Privileged — Attribute of tasks installed on a program file. Nonprivileged tasks are prohibited from performing certain reserved functions while privileged tasks may execute any code and any supervisor call.

Procedure Segment — A segment of a task that can be shared with any other tasks; usually consists of executable code.

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

Program — The instructions and data to perform a particular function; a program consists of one or more segments that define a logical address space for a set of instructions and data. Also see Task.

Program Counter — 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 and segments in memory image form. A program file contains system generated information that enables more than one program to be stored in the file.

**Programmer Panel** — A peripheral device mounted on the front of the computer providing an elementary interface to the computer. Most Business System computers do not have a programmer panel.

**Prompt** — See Field Prompt

**Queue** — A 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 removed for processing (first-out).

**Ready Task** — A task that is ready to execute, that is, queued in memory for a given priority level awaiting its turn for execution.

**Record Locking** — A procedure used to restrict access to a record in a file. A locked record cannot be accessed until it is 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 program that may be shared among several users in a multitasking environment without conflict of data among the users.

**Relative Record File** — A directly addressable file in which the records are numbered from the beginning of the file. The length of records in the file is a fixed number specified during file creation. Any record can be easily accessed after calculating the record's displacement from the beginning of the file. The calculation is based only on the record number and the fixed record length attribute of the file.

**Relocatable** — An attribute of an overlay that allows that overlay to be loaded at an address other than its natural load address. Addresses within the overlay are resolved at load time.

**Resource** — A logical or physical commodity such as a peripheral device, file, or memory, that can be allocated to a program. An I/O resource is any such commodity other than computer memory.

**Resource Contention** — The situation in which two or more tasks attempt to use the same resource at the same time.

**Replicable Task** — An installation attribute for a task, indicating that multiple copies of the task can be simultaneously in memory. Each of these copies is a duplicate of a single master copy on a program file. Utility tasks such as the FORTRAN compiler or SCI are frequently assigned this attribute.

**Roll-In** — To load a task into memory.

**Roll-Out** — To move a task or procedure segment from memory to disk so that its memory space can be used by another task.

- ROM Loader** — A program in Read-Only Memory (ROM) on the system memory interface board at a dedicated memory address that is triggered by the IPL sequence to initiate loading of the operating system into memory.
- Run-Time ID** — An identification number unique to an active task for the duration of its execution.
- Scheduler** — The part of the operating system that decides which task is to receive execution time.
- SCI** — See System Command Interpreter.
- Scrolling** — Moving the contents of the screen of a video display terminal up or down in order to view a file that contains more data than the screen can display at one time. This is accomplished by using the Previous Line key, Next Line key, or some other special key.
- Secondary Allocation** — A block of disk space automatically allocated by the system to an expandable file that 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.
- Segment** — A piece of software occupying a single block of memory, or an in-memory image on disk.
- Sequential File** — A variable record length file which must be accessed in successive order, that is, the order in which the records are written to the file.
- Shareable** — An attribute of a procedure segment that specifies that the procedure segment can be shared concurrently by more than one task.
- Source File** — A file containing one or more program source modules (source code or statements) that is usually created using the Text Editor.
- ST** — See Status Register.
- STA** — See System Table Area.
- Station** — An interactive terminal with an associated Keyboard Status Block. Physically, a station may be a VDT or a hard-copy device such as the 733 ASR terminal.
- Status Register (ST)** — A hardware register on the computer that holds condition codes set by preceding operations, fault flags, mode control information, and the interrupt mask.
- 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 (SVC)** — A user task request for operating system services. 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** — A list of necessary parameter values for a supervisor call. The size and content of the supervisor call block depends on the particular call being made.

**Suspended Task** — A task that was temporarily removed from the active list and from execution as a result of a supervisor call. When reactivated, a suspended task executes from the point of suspension.

**SVC** — See Supervisor Call

**Synchronous Transmission** — Transmission in which data character bits are transmitted at a fixed rate with equal time intervals between characters.

**Synonym** — A text string that functions as an alternative for another string, under SCI. Usually a synonym is shorter than the text it replaces and is more convenient to use.

**Sysgen** — An acronym for System Generation. Also see System Generation.

**System Command Interpreter (SCI)** — 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** — The language in which commands to SCI are written. It has a detailed syntax defined in Volume III.

**System Disk** — The disk (or other randomly accessible medium) on which the DX10 software and the system files reside.

**System File** — A disk file used internally by the operating system. All system filenames begin with “S\$”.

**System Generation** — An interactive process wherein a new version of DX10 is configured to match a particular hardware installation. A set of static data structures is created that represents the configuration. Also, certain parameters supplied at system generation allow the fine tuning of system performance. Also see Command Mode and Inquiry Mode.

**System Log Reporting** — The part of the system that reports hardware and task errors to a specified logging device and to a file. Any task can also write a message to the system log using the System Log SVC.

**System Operator** — The person who controls system start and restart, places information media into the input devices, removes the output, and performs other related functions.

**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.

**System Task** — A task that maps into the root portion of the operating system.

**System Time Unit** — A standard operating system measurement of 50 milliseconds.

**Task** — A particular activation of a program under DX10. Each activation of a program is a separate task. The address space of the task can consist of one or more segments. Segments can be overlaid with other segments during task execution, with a limit of three segments comprising the address space at one time. Also see Procedure Segment, and Task Segment.

**Task Local LUNO** — A logical unit number that can be used only by the task that assigns it.

**Task Segment** — The part of a program that contains the initial portion of the program (transfer vector, optional data, and optional program code). Also see Program, Task, and Procedure Segment.

**Task Sentry** — A part of DX10 that monitors task activity to prevent CPU lockout by lowering the priority of a CPU-bound task at set intervals. You can enable or disable the Task Sentry during System Generation.

**Task State** — The current disposition of a task. For example, a task can be in execution (executing state), suspended for any of a number of reasons (with separate states to describe these), or any of several other states.

**Teleprinter Device (TPD)** — A type of device used for input and/or output. The TPD prints a hard copy of the output. If used for input, the device also has a keyboard. This category of device includes printers and hard-copy terminals.

**Temporary File** — A file that DX10 deletes automatically after the LUNO assigned to the file is released.

**TCA** — See Terminal Communications Area.

**TCA File** — A file of images of users' TCAs. Also see Terminal Communications Area.

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

**Terminal** — Any interactive user device. Also see Station.

**Terminal Communications Area (TCA)** — An area of memory set aside for each user ID that contains user ID information and synonym names and values.

**Terminal Local File (TLF)** — A file associated with a background task or a foreground task when using the System Command Interpreter. This file is used for communicating information from a command processor to the user.

**Terminated Task** — A task that has been removed from execution and from the active list. Termination occurs when the task completes normally, when the system detects a fatal error on the part of the task, or when a user directs the operating system to kill the task. When reactivated from the terminated state, a task executes from its beginning.

**Textual Errors** — Errors that are registered by a displayed or printed text string that indicates the type and perhaps cause of the error.

**Thrashing** — The state of the system in which the overhead required to resolve resource contention occupies most of the system's time.

**TILINE\*** — The 990 asynchronous 50-megabit per second memory bus, used by the CPU, memory, and disk and tape controllers.

**TILINE Peripheral Control Space** — A range of TILINE addresses reserved for TILINE device controller interfacing.

**Time-Out** — Expiration of the time period during which a device must respond to remain an active device.

**Time Slice** — A unit of execution time allotted to a task. 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. The number of clock interrupts per time slice is set during system generation.

**TLF** — See Terminal Local File.

**TPCS** — See TILINE Peripheral Control Space.

**TPD** — See Teleprinter Device.

**Transfer Vector** — A pair of memory addresses in two consecutive words of memory. The first word contains the address of a 16 word area of memory, called a workspace. The second word contains the address of a subroutine entry point. Also see Context Switch.

**TTY** — A line-oriented terminal, such as a teletypewriter.

**TTY Mode** — Line-oriented I/O access to a terminal. A VDT can be accessed in either TTY or VDT mode by the System Command Interpreter.

**Unblocked File** — A file in which each physical record contains one logical record. During data transfer to or from such a file, no internal buffering is required.

**Unload** — Both the logical action of preparing a disk cartridge for removal and the act of physically removing the cartridge or a tape reel or cartridge.

**User ID** — A six-character identifier that identifies a user to the SCI.

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

**VDT** — A video display terminal.

**VDT Mode** — A screen-oriented mode, with the ability to read and write fields at any position on the screen. Video Display Terminals under DX10 can be used as hard-copy emulators (TTY mode) or can be used in the VDT mode to utilize their full power and intrinsic speed.

\* TILINE is a trademark of Texas Instruments Incorporated.

**Volume** — A logical device, such as a disk pack or flexible diskette, that can be uniquely identified by name, and that can store one or more logical files.

**Volume Information** — Data stored in track 0, sector 0 of every initialized disk volume, describing system information and the address of VCATALOG.

**Volume Name** — A character string that identifies a volume. It is used when installing and unloading the volume and can be used as part of the pathname for files contained within it. Disk volume names can have up to eight alphanumeric characters, but must begin with a letter.

**WCS** — See Writable Control Storage.

**Word** — A group of binary digits that can be operated on as a single unit. The Business System computers have 16 binary digits in a word.

**Workspace** — A 16-word area of memory addressed as work space register 0 through 15. The active work space is 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.

**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.

**WP** — See Workspace Pointer.

**Writable** — An attribute of a segment that specifies the segment can be modified when in memory. (This attribute is not available on some models of the Business System series.)

**Writable Control Store (WCS)** — A portion of computer memory that is separate from primary computer memory. WCS can contain microcode instructions that perform the operations of assembly language instructions.

**XOP** — Extended operation; a 990 Assembly Language instruction which generates a software interrupt.



# Index

---

This index lists key topics of this manual and specifies where each topic appears, as follows:

- Sections — Section references appear as *Section n*, where *n* represents the section number.
- Appendixes — Appendix references appear as *Appendix Y*, where *Y* represents the appendix letter.
- Paragraphs — Paragraph references appear as alphanumeric characters separated by decimal points. The first character refers to the section or appendix containing the paragraph, and any other numbers indicate the sequence of the paragraph within the section or appendix. For example:
  - 3.5.2 refers to Section 3, paragraph 5.2.
  - A.2 refers to Appendix A, paragraph 2.
- Figures — Figure references appear as *Fn-x* or *FY-x*, where *n* represents the section and *Y* represents the appendix containing the figure; *x* represents the number of the figure within the section or appendix. For example:
  - *F2-7* refers to the seventh figure in Section 2.
  - *FG-1* refers to the first figure in Appendix G.
- Tables — Table references appear as *Tn-x* or *TY-x*, where *n* represents the section and *Y* represents the appendix containing the table; *x* represents the number of the table within the section or appendix. For example:
  - *T3-10* refers to the tenth table in Section 3.
  - *TB-4* refers to the fourth table in Appendix B.
- See and See also references — *See* and *See also* direct you to other entries in the index. For example:
  - Logical Unit Number ..... See LUNO
  - Device ..... See also individual device names or numbers

Page numbers that correspond to these index references appear in the Table of Contents.

- Access Name ..... 10.2
- Access Privileges, File ..... 6.3.3
- Activation, Task ..... 12.7
- Allocatable Disk Space ..... 9.2
- Allocation, Memory ..... 13.2
- ANALZ ..... 8.3
- Assembler ..... 11.2.2
- Asynchronous Controllers ..... 11.3.1
  
- Background Execution, SCI ..... 5.1.1
- Backup, System ..... 3.2
- BASIC Language ..... 4.3.4, 11.4.4
- Batch Stream Mode, SCI ..... 5.1.3
- Blank Suppression and Adjustment, File ..... 6.3.8
- Blocked, File ..... 6.3.6
- Business System Hardware ..... 3.1
  
- Calls, Supervisor ..... See SVC
- COBOL Language ..... 4.3.1, 11.4.2
- COBOL Program Generator ..... 4.4.4, 11.5.4
- Command Interpreter, System ..... See SCI
- Command Procedures, SCI ..... 5.1.4
- Commands, SCI ..... 5.1.6
- Communications ..... 11.3
  - Controllers ..... 11.3.1
  - Hardware ..... 11.3.1
  - Software:
    - 3270 Emulator Interactive ..... 11.3.3
    - 3780/2780 Emulator ..... 11.3.2
- Contents, Disk ..... 9.3
- Controllers, Communications ..... 11.3.1
- Crash, System ..... 8.3
  
- Data Base Management
  - System ..... 4.5.1, 11.6.1
- Data Dictionary ..... 4.5.3, 11.6.3
- Data Management Tools ..... 4.5
- DBMS ..... 4.5.1, 11.6.1
- DD ..... 4.5.3, 11.6.3
- Debugger ..... 4.2.4, 11.2.4
- Deferred Write ..... 6.3.7
- Delete Protection ..... 6.3.2
- Device:
  - Name ..... 10.2.1
  - Orientation ..... 10.4
  - Service Routine ..... See DSR
- Devices, Hardware ..... T3-1
- Disk:
  - Contents ..... 9.3
  - Management ..... 9.2
  - System ..... 3.2
- DSR ..... 2.6, 12.7
  
- Emulator:
  - Communications Software, 3780/2780 ..... 11.3.2
  - Interactive Communications Software, 3270 ..... 11.3.3
- End Action Routines ..... 8.6
  
- Error:
  - Prevention ..... 8.7
  - Reporting ..... 8.2
- Expandable File ..... 6.3.9
  
- Features, File ..... 6.3
- File:
  - Access Privileges ..... 6.3.3
  - Blank Suppression and Adjustment ..... 6.3.8
  - Blocked ..... 6.3.6
  - Expandable ..... 6.3.9
  - Features ..... 6.3
  - Key Index ..... 6.2.3
  - Pathname ..... 10.2.3
  - Program ..... 12.9
  - Protection ..... 6.3.2
  - Record Locking ..... 6.3.4
  - Relative Record ..... 6.2.2
  - Sequential ..... 6.2.1
  - Stability ..... 6.2.3.2
  - Structures ..... 6.2
  - Temporary ..... 6.3.5
- Files, System Overhead ..... 9.3.1
- Foreground Execution, SCI ..... 5.1.1
- FORTRAN Language ..... 4.3.3, 11.4.1
  
- Generation, System ..... 7.1
- Generic Keycap Names ..... Appendix A, TA-1
  
- Hardware:
  - Business System ..... 3.1
  - Communications ..... 11.3.1
  - Devices ..... T3-1
  - Required ..... 3.2
  - 990/10A ..... 3.1
  - 990/12LR ..... 3.1
- High Level Languages ..... 2.4, 4.3, 11.4
  
- Identification, Program ..... 12.10
- Immediate Write ..... 6.3.7
- Interactive Communications Software, 3270 Emulator ..... 11.3.3
- Interactive Mode, SCI ..... 5.1.2
  
- Key ..... 6.2.3
- Key Index File ..... 6.2.3
- Key Sequences ..... TA-2
- Key Values ..... 6.2.3.1
- Keycap Names, Generic ..... Appendix A, TA-1
  
- Language:
  - BASIC ..... 4.3.4, 11.4.4
  - COBOL ..... 4.3.1, 11.4.2
  - FORTRAN ..... 4.3.3, 11.4.1
  - Pascal ..... 4.3.5, 11.4.3
  - RPG II ..... 4.3.2, 11.4.5
- Languages, High Level ..... 2.4, 4.3, 11.4
- Link Editor ..... 4.2.3, 11.2.3
- Lockout ..... 12.11

- Log, System ..... 8.4
- Logical Unit Number ..... 10.3
- LUNO ..... 10.3
- Macro Assembler ..... 4.2.2, 11.2.2
- Management:
  - Disk ..... 9.2
  - Memory ..... 13.1
- Mapping, Memory ..... 8.5, 13.2
- Memory:
  - Allocation ..... 13.2
  - Management ..... 13.1
  - Mapping ..... 8.5, 13.2
- Memory-Resident Task ..... 13.4
- Menu, SCI ..... 5.1.2
- Name:
  - Access ..... 10.2
  - Device ..... 10.2.1
  - Volume ..... 10.2.2
- Orientation, Device ..... 10.4
- Overhead Files, System ..... 9.3.1
- Overlay ..... 12.5
- Parameters, SCI ..... 5.1
- Pascal Language ..... 4.3.5, 11.4.3
- Pathname, File ..... 10.2.3
- Prelogging ..... 6.2.3.2
- Prevention, Error ..... 8.7
- Priority, Task ..... 12.8
- Procedure, Shared ..... 12.3, 12.4
- Productivity Aids ..... 4.4
- Program ..... 12.2
- Program Development Tools ..... 4.2
- Program:
  - File ..... 12.9
  - Identification ..... 12.10
- Prompts, SCI ..... 5.1
- Protection:
  - Delete ..... 6.3.2
  - File ..... 6.3.2
  - Write ..... 6.3.2
- Query ..... 4.5.2, 11.6.2
- Record Locking, File ..... 6.3.4
- Relative Record File ..... 6.2.2
- Replicated Task ..... 12.3
- Reporting, Error ..... 8.2
- Required, Hardware ..... 3.2
- Roll-In/Roll-Out ..... 13.3
- RPG II Language ..... 4.3.2, 11.4.5
- Scheduling, Task ..... 12.8
- SCI ..... 2.3, 5.1
  - Background Execution ..... 5.1.1
  - Batch Stream Mode ..... 5.1.3
- Command Procedures ..... 5.1.4
- Commands ..... 5.1.6
- Foreground Execution ..... 5.1.1
- Interactive Mode ..... 5.1.2
- Menu ..... 5.1.2
- Parameters ..... 5.1
- Prompts ..... 5.1
- Synonyms ..... 5.1.5
- Segment ..... 12.2, 12.3
- Sequential File ..... 6.2.1
- Service Routine, Device ..... See DSR
- Shared Procedure ..... 12.3, 12.4
- Software:
  - 3270 Emulator Interactive
    - Communications ..... 11.3.3
  - 3780/2780 Emulator
    - Communications ..... 11.3.2
- Sort/Merge ..... 4.4.1, 11.5.3
- Stability, File ..... 6.2.3.2
- Structures, File ..... 6.2
- Supervisor Calls ..... See SVC
- SVC ..... 2.4, 12.12
- Synchronous Controllers ..... 11.3.1
- Synonyms, SCI ..... 5.1.5
- System:
  - Backup ..... 3.2
  - Command Interpreter ..... See SCI
  - Crash ..... 8.3
  - Disk ..... 3.2
  - Generation ..... 7.1
  - Log ..... 8.4
  - Overhead Files ..... 9.3.1
- Task ..... 12.2
  - Activation ..... 12.7
  - Memory-Resident ..... 13.4
  - Priority ..... 12.8
  - Replicated ..... 12.3
  - Scheduling ..... 12.8
  - Task Sentry ..... 12.11
  - Temporary File ..... 6.3.5
- Text Editor ..... 2.5, 4.2.1, 11.2.1
- TIFORM ..... 4.4.2, 11.5.1
- Time Slicing ..... 12.8.2
- TIPE ..... 4.4.3, 11.5.2
- Values, Key ..... 6.2.3.1
- Volume Name ..... 10.2.2
- Write:
  - Deferred ..... 6.3.7
  - Immediate ..... 6.3.7
  - Protection ..... 6.3.2
- 3270 Emulator Interactive
  - Communications Software ..... 11.3.3
- 3780/2780 Emulator
  - Communications Software ..... 11.3.2
- 990/10A Hardware ..... 3.1
- 990/12LR Hardware ..... 3.1



# Master Index

---

This index is a master index to the major subjects covered in the DX10 operating system manuals. The subject listings are arranged in alphabetical order. Subject headings that start with numbers appear at the end of the index. Each listing contains a subject heading and the volume numbers of the DX10 operating system manuals in which the subject appears.

Use the master index to locate the volume(s) that give information about the subject that you wish to find. Then use the index of the volume(s) listed to find references to sections, appendixes, paragraphs, tables, or figures. If you want to find the page numbers corresponding to the index references, refer to the table of contents of the volume that contains the desired information.

Plural subject headings are used for general information, for tables, indexes or lists, or for groups or classes within a subject heading. Singular subject headings are used for all other entries.

The volume numbers refer to the DX10 operating system manuals as follows:

<b>Volume No.</b>	<b>Manual</b>
I	<i>DX10 Operating System Concepts and Facilities (Volume I)</i>
II	<i>DX10 Operating System Operations Guide (Volume II)</i>
III	<i>DX10 Operating System Application Programming Guide (Volume III)</i>
IV	<i>DX10 Operating System Text Editor Manual (Volume IV)</i>
V	<i>DX10 Operating System Systems Programming Guide (Volume V)</i>
VI	<i>DX10 Operating System Error Reporting and Recovery Manual (Volume VI)</i>

You can find a complete list of the System Command Interpreter (SCI) commands in the *DX10 Operating System Operations Guide (Volume II)*. Use that manual to locate full details about specific SCI commands. Details about the SCI programming language are contained in the *DX10 Operating System Application Programming Guide (Volume III)*.

You can find a complete list and discussion of the nonprivileged Supervisor Calls (SVCs) in *DX10 Operating System Application Programming Guide (Volume III)* and a list and discussion of the privileged SVCs in the *DX10 Operating System Systems Programming Guide (Volume V)*.

SCI commands and SVCs are not listed individually in this index.

Subject	Volume Number	Subject	Volume Number
Absolute Address	VI	Privileged SVC	V
Access Names	I	System Crash	VI
Access Privileges, File	I, III	Command:	
ACNM	III	For individual commands, see name or	
Activating:		mnemonic of command in the appropriate	
SCI	II, IV	volume listed under Commands	
Task	I	Entry Control Keys	II
Text Editor	IV	Error	VI
Add Error Code	II	Files	II
Add Error Message	VI	Format	II, III
Address	VI	Interpreter, System	See SCI
Absolute	VI	Library, SCI	III
Beet	VI	List, SCI	II
Block	VI	Mode, GEN990	V
CRU	V	Procedure	I, III
Relative	VI	Processor	III
TILINE	V	Commands:	
Address-Dependent Procedure	III	Debugger	III
Address-Independent Procedure	III	GEN990	V
Allocatable Disk Space	I	Privileged	II
ANALZ Utility	I, II, VI	SCI	II
ANSI Carriage Control Characters	II	Text Editor	IV
Assemble and Link the System	V	XANAL	VI
Assemble Task	III	Communication:	
Assembler	I, III	Intertask (ITC)	III
Assembly Language	I, III, V	Hardware Equipment	I
Assembly Language Program Segments	III	Software	I
Assign LUNO	II, III	Control/Display Module (CDM)	II, V, VI
Assistance	VI	Control Files	II
Asynchronous DSR	V	Conversion, DX10 to IBM	See II
Local PDT Extension	V	Conversion, DX10 to TX990,	
Long-Distance Device Extension	V	TX990 to DX10	II
Background	I, III	Country Codes	II, III
Backup Procedure	II	Crash:	
Base System Configuration	V	Code	VI
BASIC	I	Error	VI
Batch Mode	I, II, III	File	VI
Beet Address	VI	System	V, VI
Block, SVC Call	III, V	CRU Address	V
Boards, Interface	V	Customer Representative	VI
Boot (System)	See IPL	Data:	
Bubble Memory Terminal	II	Entry	IV
Support System	V	Keys	III
Buffered I/O Request Block	V	Locate	VI
Business System Terminal	II	Management Tools	I
Call Block, SVC	III, V	Protection	III
Call, Supervisor	See SVC	Shared	III
CDM	II, V, VI	Terminals	II
Change Disk Pack	II	Deactivating, SCI	II
Change Error Message	II	Deadlock	VI
Change System Generation Parameters	V	Dealer	VI
COBOL	I	Debug:	
COBOL Program Generator	I	Commands	III
Code:		Error	VI
Country	II, III	Modes	III
Error	II, VI	Debugger	I, III
Nonprivileged SVC	III	Default Values	II, III

Subject	Volume Number	Subject	Volume Number
Device	II	I/O SVCs	III
I/O SVCs	III	Key Indexed	I, III
Queues	II, VI	Management	II
Service Routine (DSR)	I, IV	News	II, III
Table, Physical (PDT)	V	Pathname	I, III
Directory Management	See also File Management, II	Program	I, III
Disk:		Relative Record	I, III
Build Utility	II, V	Sequential	I, III
Contents	I	SVC	III
File Organization	III	System	II, V
I/O Direct	V	Terminal Local (TLF)	II
Management	I, III	Types	I, II, III
Structure	III	Forcing a System Crash	VI
System	I	Foreground Execution, SCI	I, III
Disk-Resident Procedures and Tasks	III	Format, Command	II
DSR	I, V	Format, Disk	II
Asynchronous	V	Format, Dump	VI
Dump Format	VI	FORTRAN	I
Dump, Memory	VI	Function Keys (Keyboard)	III, IV
DX10	See System	FUTIL	III, VI
DX10 to TX990 Conversion	II		
DX10 2.X Conversion	II, V	Generic Key Names	Appendix A of all Manuals
Edit:		GEN990	V
Control Functions	IV	Hard Break Key Sequence	Appendix A of all Manuals, III
Mode	IV	Hardware:	
Editor, Text	I, IV	Features	I, II
Electronic Video Terminal	See 940 EVT	Power Down	II
Emulator, 3270	I, V	Power Up	II, VI
Emulator, 3780/2780	I, V	Service Routine (HSR)	V
Equivalents, Keyboard	Appendix A of all Manuals	HSR	V
Error	VI	Subroutines	V
Code	II, III, VI	IBM Conversion Utility	II
Messages	II, III, VI	Initial Program Load (IPL)	II, V
Prevention	I	Initial Values	II
Reporting	I, VI	Initialization, System	II, V
Types	VI	Initialize New Volume	II, V
EVT, 940	II, IV	Input/Output	See I/O
Execution:		Install Task	II, III
Program	III	Instructions, XOP	III
Task	III	Interactive Mode	I
Expanded:		Interface:	
Error Message	VI	Boards	V
Message	I	Routines	III
Expansion Chassis	V	Interleaving Factor	V
Expert Mode	II	Intermittent Error	VI
Extended Operation (XOP)	III	Internal Error	VI
Extension for Terminals		International Considerations	III
with Keyboards (XTK)	V	Interpreter, System Command	See SCI
File:		Interrupt Service Routine	See ISR
Command	II	Interrupt, Terminal	III
Control	II	Interrupts	V
Conversion, TX/DX	II	Intertask Communication (ITC)	III
Crash	VI	I/O	III
Features	I	I/O Errors	VI
GEN990	V	I/O SVC Subopcodes	III

Subject	Volume Number
IPL	II, VI
ISR	V
Asynchronous	V
ITC	III
Key Indexed Files (KIF)	I, II, III
Keyboard:	
Equivalents	Appendix A of all Manuals
Layouts	II
Status Block (KSB)	V
Key Names, Generic	Appendix A of all Manuals
Keys (File)	III
Keys (Terminal)	II, III, IV
KIF	I, II, III
KSB	V
Language:	
Assembly	III, V
Assembly, Program Segments	III
High-Level	I
SCI	III
Library, SCI Procedure	III
Line Control, Remote Terminal	II
Link:	
Editor	I, II
Task	III
List, SCI Command	II
Loader Error	VI
Loader, System	II
Loading Your System	VI
Local File, Terminal	II
Log-Off	III
Log-On	II, III
Logging, Transaction	III
Logical Records	III
Logical Unit Number	See LUNO
LP\$X	II, III
LUNO	I, II, III
Macro Assembler	I, III
Maintenance, System	II
Management:	
Disk	I, II
File	II
Memory	I, III
Map Files	VI
Mapping:	
Memory	I, III, VI
Program	III
Memory:	
Bubble	II, V
Estimator	V
Management	I, III
Mapping	I, III, VI
Memory-Resident:	
Procedure	III
Task	I, III
Menu	I, II, III

Subject	Volume Number
Message Handling	II, VI
Mode:	
Debug	III
Expert	II
TTY	II
VDT	II
Name:	
Access	I
Device	I, II
Volume	I, II
News File	II, III
Opcodes	See Subopcodes
Operating System	See System
Overlay	I, III
Parameters:	
Change System Generation	V
GEN990	V
SCI	I, III
Pascal	I
Patch the System	V
Pathname	I, II
PDT	V
Physical:	
Device Table (PDT)	V
Records	II, III
Power Up, Hardware	II, VI
Power Down, Hardware	II
Primitives	III
Priority	I, III
Privileged:	
Commands	II, III
SVCs	V
Procedures	III
Reentrant	III
Shared	I, III
Processor Interface Subroutines	III
Program	III
Example	III
File	I, III
Identification	I
Segments, Assembly Language	III
Programmer Panel	II, VI
Prompt:	
Field	III
Response	II
Prompts	I, II
Text Editor	II, IV
Protection:	
Delete	I
File	I
Write	I
Query	I
Queue	III
Device	II, VI
FUTIL	VI



Subject	Volume Number	Subject	Volume Number
Records:		Subopcodes	III
Key Indexed File	III	Subroutines	See also Routines, III, V
Logical	III	Supervisor Call	See SVC, SVCs
Physical	III	Suspend SCI	II
Recovery	See also Backup, II	SVC	I, III, V
Reentrant Procedures	III	Call Block	III, V
Relative Address	VI	Error	V, VI
Relative Record File	I, III	Structure	I, III, V
Remote Terminal Line Control	II	SVCs:	
Remote Terminal Subsystem (RTS)	II, V	Device I/O	III
Replicable Task	I, III	File I/O	III
Roll-In/Roll-Out	I, III	Nonprivileged	III
Routines	See also Subroutines	Privileged	V
HSR Subroutines	V	Program Support	III
Interface	III, V	Using	III
Processor Interface Subroutines	III	Writing	V
SCI	III	Synonyms	I, II
S\$ Subroutines	III	System:	
User Subroutines	III	Command Interpreter	See SCI
RPG II	I	Crash	See Crash
RTS	II, V	Disk	I, II, V
		Files	II, V
Scheduling Task	I, III	Generation	I, V
SCI	I, II, III	Initialization	II, V
Activating	II, IV	Loader Error	VI
Command Entry	II	Log	I, VI
Command Format	III	Error	VI
Command Library	II, III	Queue	VI
Command List	II	Logbook	V
Command Procedure	III	Maintenance	II
Command Processor	III	Patching	V
Commands	II	Pointer Table Address	V
Deactivating	II	Structure	V, VI
Errors	VI	S\$ Subroutines	III
Features	III		
Interface Routines	III	Tape Build Utility	II, V
Language	III	Task	II, III
Menu	II	Attributes	III
Parameters	I, III	Execution	See III
Primitives	III	Scheduling	III
Procedures	III	State Codes	II
Subroutines	III	Status Block (TSB)	II
Synonyms	III	Terminating	III
Secondary Loader Program	II	Teleprinter Characteristics	II
Segments	III	Temporary File	I
Self-Tests	VI	Terminal:	
Sequential File	I, III	Bubble Memory	II, V
Service Routine:		Interrupt	III
Device (DSR)	I, IV	Local File (TLF)	II
Terminal (TSR)	V	Service Routine (TSR)	V
Shared Procedures	I, III	Subsystem, Remote (RTS)	II, V
Sort/Merge	I, V	Terminals	See also model number of terminal, II
Starting the System	II		
Status Block:		Terminating:	
Keyboard (KSB)	V	System	II
Task (TSB)	III, V	Task	III
String	III	Text Editor	IV
Structures, File	I		

Subject	Volume Number
Text Editor .....	I, IV
Commands .....	IV
Error .....	VI
Exercises .....	IV
TIFORM .....	I
TILINE Address .....	V
Time Slicing .....	I
TIPE .....	I
TLF .....	II
Transaction Logging .....	III
Transfer Vector .....	III
Troubleshooting Guide .....	VI
Troubleshooting, System Generation .....	V
TSB .....	III, V
TSR .....	V
TTY Mode .....	II
TX990 to DX10 Conversion .....	II
User:	
Memory .....	VI
Program Files .....	III
Subroutines .....	III
Tasks .....	III
Utility:	
ANALZ .....	VI
Disk Build .....	II, V
File Conversion, TX/DX .....	II
Remote Terminal .....	II
System Generation .....	V
Tape Build .....	II, V

Subject	Volume Number
Values, Default .....	II
Values, Initial .....	II
Values, Key .....	I
VDT .....	II, IV
VDT Mode .....	II
Video Display Terminal .....	II, IV
Volume Names .....	I, II
Volumes .....	II
Write Protected Files .....	I
Writing, SVC .....	V
XOP Instructions .....	III
XTK .....	V
2.X Conversion, DX10 .....	II, V
3270 Emulator .....	I, V
3780/2780 Emulator .....	I, V
733 ASR/KSR Data Terminal .....	II, IV
703/707 KSR Data Terminals .....	I, II
743/745 KSR Data Terminals .....	I, II
763/765 Bubble Memory Terminals .....	II
781 RO Data Terminal .....	II
783/785/787 KSR Data Terminals .....	II
820 KSR/RO Data Terminal .....	II
840 KSR Teleprinter Device .....	II, IV
911 VDT .....	II, V
931 VDT .....	II, V
940 EVT .....	II, IV



FOLD



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

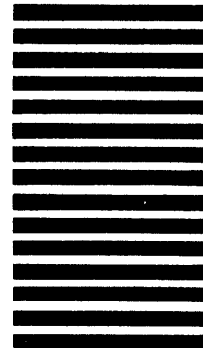
**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 7284 DALLAS, TX

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED  
DATA SYSTEMS GROUP

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



FOLD