

**DX10 OPERATING SYSTEM**



# ***Systems Programming Guide***

Part No. 946250-9705 \*G  
January 1985

**Volume V**

# **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INSERT LATEST CHANGED PAGES AND DISCARD SUPERSEDED PAGES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 Systems Programming Guide, Volume V (946250-9705)

Original Issue ..... August 1977  
Revision ..... March 1978  
Revision ..... October 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 406 consisting of the following:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
Cover	1	2-22	0	3-25 - 3-28	0
Effective Pages	1	2-23	1	3-29 - 3-33	1
Eff. Pages Cont.	1	2-24 - 2-25	0	3-34	0
iii - iv	1	2-26	1	3-35 - 3-36	1
v	0	2-26A/2-26B	1	3-37 - 3-39	0
vi	1	2-27 - 2-36	0	3-40	1
vii - xvi	0	3-1 - 3-2	0	3-41 - 3-42	0
1-1 - 1-4	0	3-3	1	3-43 - 3-44	1
2-1 - 2-20	0	3-4 - 3-23	0	3-44A/3-44B	1
2-21	1	3-24	1	3-45 - 3-64	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.

© 1977, 1978, 1979, 1981, 1982, 1983, 1985 Texas Instruments Incorporated.  
All Rights Reserved

Printed in U.S.A.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Texas Instruments Incorporated.

---

# 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INSERT LATEST CHANGED PAGES AND DISCARD SUPERSEDED PAGES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 Systems Programming Guide, Volume V (946250-9705)

Continued:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
4-1 - 4-8	.....0	7-36A/7-36B	.....1	H-1 - H-8	.....0
5-1	.....1	7-37 - 7-38	.....0	I-1 - I-2	.....0
5-2 - 5-134	.....0	8-1 - 8-6	.....0	I-3 - I-14	.....1
6-1	.....0	A-1 - A-14	.....0	J-1 - J-8	.....0
6-2	.....1	B-1 - B-4	.....0	Index-1 - Index-4	.....0
6-3 - 6-4	.....0	C-1 - C-2	.....0	User's Response	.....1
7-1 - 7-19	.....0	D-1/D-2	.....0	Business Reply	.....1
7-20	.....1	E-1 - E-2	.....0	Inside Cover	.....1
7-21 - 7-35	.....0	F-1 - F-24	.....0	Cover	.....1
7-36	.....1	G-1 - G-2	.....0		

---

# 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 Operations Guide (Volume II) 946250-9702	DX10 Operating System Error Reporting and Recovery Manual (Volume VI) 946250-9706
DX10 Operating System Release 3.7 System Design Document 939153-9701	Link Editor Reference Manual 949617-9701

Communications Manuals	Language Manuals	Miscellaneous Software Manuals
DX10 3270 Interactive Communications Software (ICS) User's Guide 2250554-9701	Report Program Generator (RPG II) Programmer's Guide 939524-9701	Operator's Guide, Business System 300 (International) 2533318-9701
DX10 Remote Terminal Subsystem (RTS) System Generation and Programmer's Reference Manual 2272054-9701	TI BASIC Reference Manual 2308769-9701	Operator's Guide, Business System 300 (Domestic) 2533318-9702
DX10 Remote Terminal Subsystem (RTS) Operator's Guide 2272055-9701	DX10 TI Pascal Programmer's Guide 2270528-9701	ROM Loader User's Guide 2270534-9701
DX10 3780/2780 Emulator Release 4 User's Guide 2302663-9701	COBOL Reference Manual 2270518-9701	Operator's Guide, Business System 300A 2240275-9701
	DX10 FORTRAN-78 Programmer's Guide 2268679-9701	
	FORTRAN-78 ISA Extensions Manual 2268696-9701	
	FORTRAN-78 Reference Manual 2268681-9701	
	TI Pascal Reference Manual 2270519-9701	
	TI Pascal Configuration Processor Tutorial 2250098-9701	

Productivity Tools Manuals		
DX10 Data Base Management System Programmer's Guide 2250425-9701	TIFORM Reference Manual 2234391-9701	Data Dictionary User's Guide 2276562-9701
DX10 Data Base Administrator User's Guide 2250426-9701	DX10 Query-990 User's Guide 2250466-9701	DX10 COBOL Program Generator User's Guide 2308956-9701
	Model 980 Computer DX Sort/Merge User's Guide 946252-9701	
	DX10 TIPE Texas Instruments Page Editor Reference Manual 2302656-9701	

# 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 2268688-9701

990 CR/UTILINE Expansion Installation and Operation 2272075-9701

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

ROM Loader User's Guide 2270534-9701

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 2203665-9701

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 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

## Hard-Copy Terminal Manuals

## Display Terminal Manuals

Model 931 Video Display Terminal Installation and Operation 2259228-0001

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 Model 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 2230 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 2265965-9701

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 Model DS10 Cartridge Disc System Installation and Operation 946261-9701

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

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

WD500WD500A Mass Storage System Installation and Operation (Business System Series) 2302688-9701

Model 990 Computer Model DS60 Disk System Installation and Operation Manual 2302629-9701

Model 990 Computer Model DS2000 Disc System Installation and Operation 949615-9701

Model DS300 Disk System Installation and Operation Manual 2302631-9701

Operator's Guide Disk Unit (Business System Series) 2533269-9701

Operator's Guide Model WD800WD800A Disk Unit (Business System Series) 2533319-9701

WD800WD800A Mass Storage System Installation and Operation 2306140-9701

Model 990 Computer Model FD900 Floppy Disc System Installation and Operation 945253-9701

Model 990 Computer Model MT1600 Magnetic Tape System Installation and Operation 2302642-9701

Operator's Guide Model WD800WD800A 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

990 TILINE Floppy Disc Installation and Operation 2261886-9701

# Preface

---

This manual provides detailed information required by the person responsible for maintaining a 990 computer system running under the DX10 operating system.

This manual is one of a set of six volumes that describe 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 five associated 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 supervisor calls (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 *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, DBMS, and CPG packages.

# Contents

---

Paragraph	Title	Page
<b>1 — Introduction</b>		
1.1	General Information .....	1-1
1.2	Disk Build Procedure .....	1-2
1.3	System Generation .....	1-2
1.4	Writing Your Own Device Service Routine (DSR) .....	1-2
1.5	Designing and Writing an SVC Processor .....	1-3
1.6	Using Privileged SVCs .....	1-3
1.7	DX10 2.X Compatibility .....	1-3
<b>2 — Building Your System Disk</b>		
2.1	Introduction .....	2-1
2.2	Base System Configuration .....	2-2
2.3	Media .....	2-2
2.4	Computer Programmer Panel .....	2-2
2.5	Object Kits Supplied on Disk Cartridge .....	2-4
2.5.1	Overview of the Disk Build Procedure .....	2-4
2.5.2	Disk Build Procedure .....	2-5
2.6	Object Kits Supplied on Multiple Flexible Diskette Media .....	2-10
2.6.1	Disk Build Procedure with a WD500 in Standard Configuration .....	2-11
2.6.2	Variations to Disk Build Procedure when Changing Default Values .....	2-16
2.6.2.1	Specifying the Target Disk .....	2-17
2.6.2.2	Changing Volume Characteristics of the Target Disk .....	2-17
2.6.2.3	Specifying the Backup File .....	2-20
2.6.3	Variations to Disk Build Procedure for FD1000 .....	2-20
2.7	Object Kits Supplied on Tape Media .....	2-20
2.7.1	Variations to Tape Build Procedure when Changing Default Values .....	2-26
2.8	Duplication of a Built System .....	2-29
2.9	Backup and Restore .....	2-30
2.10	Building a Custom Diskette or Tape to Use with Disk Build .....	2-30
2.11	System Files .....	2-32
2.12	Modifying Disk Volume Information .....	2-34
2.13	Modifying Initial State Specifications .....	2-34
2.13.1	Initial States .....	2-35
2.13.2	Modification Strategy .....	2-35
2.13.3	Modifying the IS Command .....	2-36



Paragraph	Title	Page
<b>3 — System Generation</b>		
3.1	System Generation . . . . .	3-1
3.2	Preexecution Considerations . . . . .	3-2
3.2.1	Base System Configurations . . . . .	3-2
3.2.2	Determining CRU Addresses . . . . .	3-7
3.2.2.1	Arrangement of CRU Base Addresses in a Chassis . . . . .	3-7
3.2.2.2	CRU Base Addresses for a 17-Slot Chassis . . . . .	3-7
3.2.2.3	CRU Addresses in Expansion Chassis . . . . .	3-10
3.2.3	Interrupt Levels . . . . .	3-10
3.2.3.1	Interrupts for Devices in Expansion Chassis . . . . .	3-10
3.2.3.2	Devices Sharing Interrupts . . . . .	3-14
3.2.4	TILINE Addresses . . . . .	3-14
3.3	GEN990 System Generation Utility . . . . .	3-17
3.3.1	Files Created and Used by GEN990 . . . . .	3-17
3.3.2	GEN990 Modes of Operation . . . . .	3-18
3.3.2.1	Inquire Mode . . . . .	3-18
3.3.2.2	Command Mode . . . . .	3-18
3.3.3	GEN990 Commands . . . . .	3-18
3.3.3.1	BUILD Command . . . . .	3-20
3.3.3.2	CHANGE Command . . . . .	3-20
3.3.3.3	DELETE Command . . . . .	3-21
3.3.3.4	DVC Command . . . . .	3-22
3.3.3.5	INQUIRE and HELP Commands . . . . .	3-22
3.3.3.6	LIST Command . . . . .	3-22
3.3.3.7	STOP and BUILD Commands . . . . .	3-23
3.3.3.8	SVC Command . . . . .	3-23
3.3.3.9	GEN990 WHAT Command . . . . .	3-23
3.4	Executing GEN990 . . . . .	3-24
3.4.1	Defining the GEN990 Operating Prompts . . . . .	3-28
3.4.2	Defining the System Timing Parameters . . . . .	3-28
3.4.2.1	LINE Parameter . . . . .	3-29
3.4.2.2	TIME SLICING ENABLED Parameter . . . . .	3-29
3.4.2.3	TIME SLICE VALUE Parameter . . . . .	3-29
3.4.2.4	TASK SENTRY ENABLED Parameter . . . . .	3-29
3.4.2.5	TASK SENTRY VALUE Parameter . . . . .	3-29
3.4.3	TABLE Parameter . . . . .	3-29
3.4.4	Defining Optional Features . . . . .	3-31
3.4.4.1	COMMON Parameter . . . . .	3-31
3.4.4.2	INTERRUPT DECODER Parameter . . . . .	3-31
3.4.4.3	FILE MANAGEMENT TASKS Parameter . . . . .	3-32
3.4.4.4	CLOCK Parameter . . . . .	3-32
3.4.4.5	RESTART ID Parameter . . . . .	3-32
3.4.4.6	OVERLAYS Parameter . . . . .	3-32
3.4.4.7	ONLINE DIAGNOSTIC SUPPORT Parameter . . . . .	3-32
3.4.4.8	SYSLOG Parameter . . . . .	3-32
3.4.4.9	BUFFER MANAGEMENT Parameter . . . . .	3-33
3.4.4.10	I/O BUFFERS Parameter . . . . .	3-33

Paragraph	Title	Page
3.4.4.11	INTERTASK Parameter . . . . .	3-33
3.4.4.12	ITC MESSAGES Parameter . . . . .	3-33
3.4.4.13	KIF Parameter . . . . .	3-33
3.4.4.14	COUNTRY CODE Parameter . . . . .	3-34
3.4.4.15	POWERFAIL Parameter . . . . .	3-34
3.4.4.16	AUTO MEDIA CHANGE RECOVERY Parameter . . . . .	3-34
3.4.4.17	SCI BACKGROUND and SCI FOREGROUND Parameters . . . . .	3-34
3.4.4.18	BREAKPOINT Parameter . . . . .	3-35
3.4.4.19	PANEL DISPLAY(BAR CHART) Parameter . . . . .	3-35
3.4.4.20	CARD 1 and CARD 2 (Defining Expansion Chassis) . . . . .	3-35
3.4.5	Defining Devices . . . . .	3-35
3.4.5.1	Defining Devices in an Expansion Chassis . . . . .	3-38
3.4.5.2	Defining Disk and Magnetic Tape Drives . . . . .	3-40
3.4.5.3	Defining Communications Devices . . . . .	3-41
3.4.5.4	Defining Display Terminals . . . . .	3-42
3.4.5.5	Defining ASRs . . . . .	3-45
3.4.5.6	Defining KSRs (or Other Teleprinter Devices (TPDs)) . . . . .	3-46
3.4.5.7	Defining Line Printers . . . . .	3-49
3.4.5.8	Defining a Card Reader . . . . .	3-51
3.4.5.9	Defining Diskettes . . . . .	3-52
3.4.5.10	Defining Nonstandard Devices . . . . .	3-53
3.4.6	Defining SVCs . . . . .	3-55
3.4.7	Terminating GEN990 . . . . .	3-55
3.4.8	Optional Processor System Generation Parameters . . . . .	3-56
3.4.8.1	DX10 Remote Terminal Subsystem (RTS) . . . . .	3-56
3.4.8.2	Sort/Merge . . . . .	3-56
3.4.8.3	3780/2780 and 3270/ICS Emulators . . . . .	3-56
3.4.8.4	Bubble Memory Terminal Support . . . . .	3-56
3.5	Assembling and Linking the System . . . . .	3-57
3.5.1	Executing the ALGS Command . . . . .	3-57
3.5.2	Results of the ALGS Command . . . . .	3-58
3.5.2.1	Normal Termination . . . . .	3-58
3.5.2.2	Abnormal Termination . . . . .	3-59
3.5.3	ALGS Batch Stream . . . . .	3-60
3.6	Patching the System . . . . .	3-61
3.6.1	PGS Error Recovery . . . . .	3-61
3.6.2	Executing the Patch Synonym Assignment Program . . . . .	3-62
3.6.3	Batch Stream Error Counter Errors . . . . .	3-63
3.6.4	Clear Secret Synonyms . . . . .	3-63
3.7	Testing the System . . . . .	3-63
3.8	Installing the Generated System . . . . .	3-64

Paragraph	Title	Page
-----------	-------	------

## 4 — System Generation Troubleshooting

4.1	Introduction .....	4-1
4.2	Maintaining a System Logbook .....	4-1
4.3	Troubleshooting System Generation Problems .....	4-2
4.3.1	Crashes and Hangs .....	4-2
4.4	Other System Generation Problems .....	4-4
4.4.1	SVC Error > 05 .....	4-4
4.4.2	SVC Error > 0E .....	4-4
4.4.3	SVC Error > 89 .....	4-5
4.5	ALGS and PGS Problems .....	4-6
4.6	Building a New System Disk .....	4-7

## 5 — How to Write a Device Service Routine

5.1	Introduction .....	5-1
5.2	Device Service Routines (DSRs) .....	5-2
5.2.1	I/O Call Routine .....	5-2
5.2.1.1	Entry Point .....	5-2
5.2.1.2	Workspace .....	5-2
5.2.1.3	Processing .....	5-2
5.2.2	Shared Routines .....	5-5
5.2.3	Power Restored Routine .....	5-5
5.2.3.1	Entry Point .....	5-6
5.2.3.2	Workspace .....	5-6
5.2.3.3	Processing .....	5-6
5.2.4	I/O Abort Routine .....	5-6
5.2.4.1	Entry Point .....	5-6
5.2.4.2	Workspace .....	5-6
5.2.4.3	Processing .....	5-6
5.2.5	Interrupt Service Routine (ISR) .....	5-7
5.2.5.1	Entry Point .....	5-8
5.2.5.2	Workspace .....	5-9
5.2.5.3	Processing .....	5-9
5.3	Data Structures .....	5-12
5.3.1	Physical Device Table (PDT) .....	5-12
5.3.2	Keyboard Status Block (KSB) .....	5-20
5.3.3	Buffered I/O Request Block .....	5-25
5.3.4	Task Status Block (TSB) .....	5-25
5.3.5	Extension for Terminals with Keyboards (XTK) .....	5-32
5.3.6	Asynchronous DSR Local PDT Extension .....	5-36
5.3.7	Asynchronous DSR Long-Distance Device Extension .....	5-37
5.4	DSR Conventions and Techniques .....	5-39
5.4.1	Workspace for Keyboard ISRs .....	5-39
5.4.2	Decoding Interrupts .....	5-39

Paragraph	Title	Page
5.4.3	Reporting Errors to the System Log .....	5-39
5.4.4	Accessing the Data Buffer .....	5-40
5.4.4.1	Indirect Access .....	5-40
5.4.4.2	Direct Access .....	5-40
5.4.5	Reenter-Me Processing .....	5-41
5.4.6	DSR Priority Schedule .....	5-41
5.4.7	Bidding Tasks from a DSR .....	5-42
5.4.7.1	ISR Procedure for Bidding a Task .....	5-42
5.4.7.2	Reenter-Me Procedure for Bidding a Task .....	5-43
5.5	Asynchronous DSR Structure .....	5-44
5.5.1	Asynchronous DSR Design Overview .....	5-47
5.5.2	Terminal Service Routine (TSR) .....	5-50
5.5.3	Interrupt Service Routine (ISR) .....	5-51
5.5.4	Hardware Controller Service Routine (HSR) .....	5-52
5.5.5	Asynchronous Data Structure Allocation .....	5-53
5.6	HSR Common Subroutines .....	5-55
5.6.1	Power-Up Initialization .....	5-56
5.6.2	Write Output Signal or Function .....	5-57
5.6.3	Read Input Signal or Function .....	5-59
5.6.4	Enable/Disable Status Change Notification .....	5-59
5.6.5	Output a Character .....	5-60
5.6.6	Write Operational Parameters .....	5-61
5.6.6.1	Set Channel Speed (Baud Rate) .....	5-62
5.6.6.2	Set Data Character Format .....	5-63
5.6.7	Read Operational Parameters and Information .....	5-63
5.6.8	Request Time Interval Notification .....	5-64
5.6.9	Controller Interrupt Decoder .....	5-65
5.7	Common Routines .....	5-66
5.7.1	BRCALL — Branch Table Call Routine .....	5-66
5.7.1.1	Parameters .....	5-66
5.7.1.2	Calling Sequence .....	5-67
5.7.2	BZYCHK — Busy Check Routine .....	5-67
5.7.2.1	Parameters .....	5-67
5.7.2.2	Calling Sequence .....	5-67
5.7.3	ENDRCD — End-of-Record Routine .....	5-68
5.7.3.1	Parameters .....	5-68
5.7.3.2	Calling Sequence .....	5-68
5.7.4	KEYFUN — Keyboard Function Routine .....	5-68
5.7.4.1	Parameters .....	5-70
5.7.4.2	Calling Sequence .....	5-70
5.7.5	GETC — Get Character Routine .....	5-70
5.7.5.1	Parameters .....	5-71
5.7.5.2	Calling Sequence .....	5-71
5.7.6	JMCALL — Jump Table Call Routine .....	5-71
5.7.6.1	Parameters .....	5-71
5.7.6.2	Calling Sequence .....	5-72
5.7.7	PUTCBF — Put Character in Buffer Routine .....	5-72
5.7.7.1	Parameters .....	5-72

Paragraph	Title	Page
5.7.7.2	Calling Sequence .....	5-72
5.7.8	PUTEBF — Put Event Character in Buffer Routine .....	5-72
5.7.8.1	Parameters .....	5-72
5.7.8.2	Calling Sequence .....	5-72
5.7.9	SETWPS — Set Interrupt Mask Routine .....	5-73
5.7.9.1	Parameters .....	5-73
5.7.9.2	Calling Sequence .....	5-73
5.7.10	BRSTAT — Branch Table Call with Statistics Routine .....	5-73
5.7.10.1	Parameters .....	5-73
5.7.10.2	Calling Sequence .....	5-73

## 6 — How to Write a Supervisor Call Processor

6.1	Introduction .....	6-1
6.2	How to Write an SVC Processor .....	6-1
6.2.1	SVC Call Block .....	6-1
6.2.2	SVC Structure .....	6-2
6.2.3	Example of User-Defined SVC .....	6-2
6.2.4	System Generation Requirements .....	6-4

## 7 — Privileged Supervisor Calls

7.1	General .....	7-1
7.2	Install Task (Code > 25) .....	7-1
7.2.1	Special Form of Install .....	7-5
7.2.2	Install Real-Time Task .....	7-5
7.3	Install Procedure (Code > 26) .....	7-5
7.4	Install Overlay (Code > 27) .....	7-8
7.5	Delete Task (Code > 28) .....	7-10
7.6	Delete Procedure (Code > 29) .....	7-11
7.7	Delete Overlay (Code >) .....	7-12
7.8	Kill Task (Code > 33) .....	7-12
7.9	Suspend Awaiting Queue Input (Code > 24) .....	7-13
7.10	Read/Write TSB (Code > 2C) .....	7-13
7.11	Read/Write Task (Code > 2D) .....	7-15
7.12	Get System Pointer Table Address (Code > 32) .....	7-17
7.13	Initialize Date and Time (Code > 3B) .....	7-18
7.14	Disk Manager (Code > 22) .....	7-19
7.15	Assign Space on Program File (Code > 37) .....	7-23
7.16	Initialize New Volume (Code > 38) .....	7-25
7.17	Install Disk Volume (Code > 20) .....	7-27
7.18	Unload Disk Volume (Code > 34) .....	7-29
7.19	Direct Disk I/O .....	7-29
7.19.1	Direct Disk I/O Call Block .....	7-31
7.19.2	Direct Disk I/O Opcodes .....	7-34
7.19.2.1	Read Format (Opcodes > 5 and > F) .....	7-34

Paragraph	Title	Page
7.19.2.2	Write Format (Opcode > 8) . . . . .	7-35
7.19.2.3	Store Registers (Opcode > E) . . . . .	7-35
7.19.2.4	Read by ADU (Opcode > 9) . . . . .	7-35
7.19.2.5	Read by Track (Opcode > A) . . . . .	7-36
7.19.2.6	Write by ADU (Opcode > B) . . . . .	7-36
7.19.2.7	Write by Track (Opcode > C) . . . . .	7-36
7.19.2.8	Write Interleaved Factor (Opcode > 12) . . . . .	7-36
7.19.3	Direct Diskette I/O . . . . .	7-36
7.19.3.1	Read Format (Opcodes > 5 and > F) . . . . .	7-36
7.19.3.2	Write Format (Opcode > 8) . . . . .	7-37
7.19.3.3	Read ASCII (Opcode > 9), Read Direct (Opcode > A) . . . . .	7-37
7.19.3.4	Write ASCII (Opcode > B), Write Direct (Opcode > C) . . . . .	7-37
7.19.3.5	Write Deleted Sector (Opcode > 10) . . . . .	7-37
7.19.3.6	Read Deleted Sector (Opcode > 11) . . . . .	7-38
7.19.3.7	Special Diskette Options . . . . .	7-38

## 8 — DX10 2.X Compatibility

8.1	Introduction . . . . .	8-1
8.2	DX10 2.X Disk Conversion . . . . .	8-1
8.2.1	Disk/Magnetic Tape Configuration . . . . .	8-1
8.2.2	Multiple Disk Configuration . . . . .	8-2
8.2.2.1	Operating Procedure . . . . .	8-2
8.2.2.2	Converted File Log . . . . .	8-3
8.3	DX10 2.X SVC Compatibility . . . . .	8-5
8.3.1	SVCs Not Supported . . . . .	8-6
8.3.2	SVCs Supported Only for Compatibility . . . . .	8-6
8.3.3	SVCs Still Used But Different . . . . .	8-6

## Appendixes

Appendix	Title	Page
A	Keycap Cross-Reference .....	A-1
B	System Task IDs .....	B-1
C	System Overlay IDs .....	C-1
D	System Procedure IDs .....	D-1
E	Global LUNOs .....	E-1
F	System Generation Examples .....	F-1
G	SVC Codes .....	G-1
H	Bidding a Task from a DSR .....	H-1
I	DX10 Memory Estimator .....	I-1
J	Example Patch File .....	J-1

## Index

## Illustrations

Figure	Title	Page
2-1	Model 990 Computer Programmer Panel .....	2-3
2-2	Control/Display Module (CDM) .....	2-4
3-1	Base System Configuration for 990A13 13-Slot Chassis, Business System 600/800 .....	3-4
3-2	Base System Configuration for 990 13-Slot Chassis .....	3-5
3-3	Base System Configuration for 990 17-Slot Chassis .....	3-6
3-4	CRU Base Addresses for 990 and 990A13 13-Slot Chassis .....	3-8
3-5	CRU Base Addresses, 17-Slot Chassis .....	3-9
3-6	CRU Base Addresses, 13-Slot Expansion Chassis #1 .....	3-11
3-7	990A13 Interrupt Assignments, 13-Slot Chassis .....	3-12
3-8	Interrupt Assignments, 13-Slot Chassis .....	3-13
3-9	Interrupt Assignments, 17-Slot Chassis .....	3-15
4-1	Memory Used by File Management .....	4-5
5-1	Flow of an I/O Call .....	5-3
5-2	DSR Structure .....	5-4
5-3	ISR Processing .....	5-9
5-4	PDT and DSR Configuration for Multiple Disk Drive Controller .....	5-11
5-5	PDT Structure .....	5-13
5-6	PDT Template .....	5-19
5-7	KSB Structure .....	5-21
5-8	KSB Template .....	5-24
5-9	Buffered I/O Request Block .....	5-25
5-10	TSB Structure .....	5-26
5-11	XTK Structure .....	5-32
5-12	XTK Template .....	5-35
5-13	Asynchronous DSR Local PDT Extension Structure .....	5-36
5-14	Asynchronous DSR Local PDT Extension Template .....	5-37
5-15	Asynchronous DSR Long-Distance Device Extension Structure .....	5-38
5-16	Asynchronous DSR Long-Distance Device Extension Template .....	5-38
5-17	Asynchronous DSR Structure .....	5-46
5-18	Asynchronous DSR Logic Flow .....	5-48
5-19	Interrupt Processing Flow .....	5-53
5-20	Asynchronous Data Structure Linkages .....	5-54
5-21	Keyboard Interrupt Processing .....	5-69
5-22	Example DSRs .....	5-74
6-1	SVC Routine Workspace .....	6-3
7-1	Addressing a Track .....	7-30
7-2	Store Registers and Read Format Data .....	7-35
7-3	Logical Track Addressing .....	7-38
8-1	Sample Log of Converted Files .....	8-4



## Tables

Table	Title	Page
2-1	DX10 Distribution Media and Part Numbers .....	2-2
2-2	Minimum Configuration for TAPBLD .....	2-21
3-1	Base System Configuration for DX10 on Disks .....	3-3
3-2	Base System Configuration for DX10 on Magnetic Tape .....	3-3
3-3	CRU Base Address Offsets for Expansion Chassis .....	3-10
3-4	Recommended TILINE Address Assignments .....	3-16
3-5	Legal Commands in GEN990 Command Mode .....	3-19
3-6	Legal Commands in GEN990 Inquire Mode .....	3-19
3-7	GEN990 Parameters .....	3-25
3-8	System Table Area Sizing Guideline .....	3-31
3-9	Description of Various Interface Boards .....	3-37
3-10	Relationship Between Peripherals and Interface Boards .....	3-37
3-11	XPS Command Prompt Responses .....	3-62
5-1	Example Interrupt Vector Table .....	5-8
5-2	PDT Values .....	5-14
5-3	KSB Values .....	5-22
5-4	TSB Values .....	5-27
5-5	Task State Codes .....	5-31
5-6	XTK Values .....	5-33
5-7	Asynchronous Device Support .....	5-44
5-8	HSR Object Modules .....	5-47
5-9	DSR/TSR Entry Points .....	5-51
5-10	HSR Baud Rate Codes .....	5-62
5-11	Controller Type Codes .....	5-64
5-12	Character Ranges Used by GETC .....	5-70
7-1	Disk Descriptions .....	7-20
7-2	Direct Disk I/O Opcodes .....	7-34
7-3	Diskette Direct Disk I/O Opcodes .....	7-37

# Introduction

---

## 1.1 GENERAL INFORMATION

This manual explains how to generate a customized DX10 operating system. You are presumed to be an experienced programmer with responsibility for maintaining and extending your operating system. This manual focuses on programming your operating system so that it runs most efficiently under general circumstances. Refer to *DX10 Application Programming Guide (Volume III)* for information about programming DX10 for specific applications.

The DX10 operating system includes some standard functions and offers a set of optional functions that are hardware dependent. During system generation, you specify which functions to incorporate in your system. The standard functions include:

- A multitasking, preemptive central processing unit (CPU) scheduler
- An internal clock interrupt and timer
- A supervisor call (SVC) preprocessor
- Some task management SVC routines
- Basic input/output (I/O) SVC routines

Optional functions that require additional memory include:

- Device support
- Standard SVCs
- Communications

Much of the DX10 software resides on disk and is loaded into memory only when needed. Program swapping, file management, multiple dynamic job management, task loading, and numerous SVCs are required to operate the system. The system generation procedure allows you to include optional features such as support for multiple-user software development, key indexed files (KIFs), and remote I/O.

## **1.2 DISK BUILD PROCEDURE**

The DX10 software is shipped on several different media: assorted types of rigid disks, flexible diskettes, and magnetic tape. If your DX10 software arrives on disk, you can simply install the disk in the system disk drive and load the software immediately. However, if your system includes a DS10 or CD1400 disk drive, you may want to transfer the system software to the fixed disk. Also, if your software arrives on flexible diskette or magnetic tape, you must transfer it to disk. DX10 includes a Disk Build utility to simplify this transferral.

## **1.3 SYSTEM GENERATION**

Section 3 on system generation describes the base system configurations that DX10 supports and explains how to generate a logical configuration that reflects the physical configuration of your system. If you want to include extra devices or instructions, or exclude unnecessary features, you can customize your operating system during the process known as system generation.

Customizing your operating system allows you to tailor it to fit specific application needs. During system generation, prompts appear on the screen of your video display terminal (VDT) that direct you to perform any of the following actions:

- Add or delete devices to the hardware configuration
- Include or exclude optional DX10 capabilities
- Include user-defined SVCs
- Generate specialized systems for specific programming environments, which can be used as operating conditions change
- Change various DX10 operating system parameters

GEN990, the System Generation utility, interactively prompts you for information about the system to be built. After the interactive session, GEN990 builds the files necessary to complete system generation. You then direct a system utility to assemble and link the new or modified system. To complete system generation, you test the new system. If the new system contains errors, you can return control to the old system and repeat the system generation process.

Section 4 explains how to handle errors that may occur during system generation.

## **1.4 WRITING YOUR OWN DEVICE SERVICE ROUTINE (DSR)**

When you are quite familiar with programming for the DX10 operating system, you may decide to include nonstandard devices in the system. You must write a DSR to detect and service each nonstandard device. During system generation, you integrate these DSRs with the supplied software. Section 5 describes how to write your own DSR and incorporate it in the system during system generation.

## **1.5 DESIGNING AND WRITING AN SVC PROCESSOR**

You can write new SVCs and SVC processing routines to provide services for DX10 that the standard SVCs do not provide. Section 6 describes how to do so, and how to incorporate the new SVCs in the system during system generation.

## **1.6 USING PRIVILEGED SVCs**

Section 7 describes certain privileged SVCs that are used almost exclusively for enhancing the capabilities of the operating system itself. These SVCs have a special privilege level because their misuse can either destroy the system by improperly and irrevocably modifying the disk, or hoard the system by killing other tasks. Section 7 describes these SVCs and how to use them.

## **1.7 DX10 2.X COMPATIBILITY**

Section 8 describes how to transfer data files from a DX10 2.X disk to a disk from a 3.0 or later release of DX10.



# Building Your System Disk

---

## 2.1 INTRODUCTION

Conventional maintenance of a system disk consists of two parts. First, you build the system disk by installing the software kit containing the system components. Next, you make a backup copy of the system disk as a security precaution.

Section 2 begins by explaining how to build your system disk. The four procedures for Disk Build are as follows:

- From the software kit shipped on disk cartridge
- From the software kit shipped on flexible diskette
- From the software kit shipped on tape (cartridge or reel)
- From a DX10 system that is already running

The first three procedures build a new DX10 operating system from the software kit shipped from the factory; the procedure you use depends on the shipping media. The fourth procedure is for the convenience of users who already have a DX10 system, but would like to make copies without having to build the disk again.

The system is shipped on media compatible with your hardware; however, installation frequently includes transferring the data from the media on which it was shipped (for example, flexible diskette) to other media (a fixed disk).

The section concludes by explaining procedures for customizing the initial state of your system. That state can be set to modify the allowed activities at any given terminal. You can modify the activities more at log-on time by using an M\$00 command procedure. (Refer to the *DX10 Operations Guide (Volume II)*.)

Once you install and back up the system, you can generate a customized operating system as described in Section 3.

### NOTE

Throughout this manual, the names of keys are generic key names. In some cases, the names on the keycaps of the terminals match the generic key names, but in many cases they do not. Appendix A contains a table of key equivalents to identify the specific keys on the terminal you are using. Drawings that show the layout of the keyboard of each type of terminal are also included.

## 2.2 BASE SYSTEM CONFIGURATION

The DX10 software kit ships on disk, flexible diskettes, or tape; within the constraints of your hardware, you can select the shipping medium. The installation procedure depends on your media. Each software kit is designed for use with a standard system configuration.

### CAUTION

**Your configuration must agree with the base system configuration to this extent: if you have a device in the system at a CRU or TILINE address listed in the base system configuration, it must be the specified device type at the interrupt listed, or the system enters an infinite loop when it tries to process an interrupt from that device. Remove conflicting devices or avoid their use if a loop condition does not occur until you attempt to use the device.**

**If you have modified your hardware configuration, return enough of it to the base system configuration shown in Table 3-1 or 3-2 to fulfill the above rule.**

## 2.3 MEDIA

Table 2-1 lists the master DX10 distribution media kits and their part numbers. Some steps in the installation procedure differ depending on the media type you have. Be sure to note the differences that apply to your media type.

**Table 2-1. DX10 Distribution Media and Part Numbers**

Medium	Part Number
Disk	939151-1601
Magnetic tape (reel or cartridge)	939151-1301, 939151-1302 (two tapes required)
8-inch double-sided, double-density (DSDD) flexible diskette	939151-1602

## 2.4 COMPUTER PROGRAMMER PANEL

Figure 2-1 shows switch and component locations on the 990 computer programmer panel described in the following paragraphs. Figure 2-2 shows the control/display module (CDM) for the 990A13, which the Business System 600 and 800 use. The Business System 300 does not require a programmer panel.

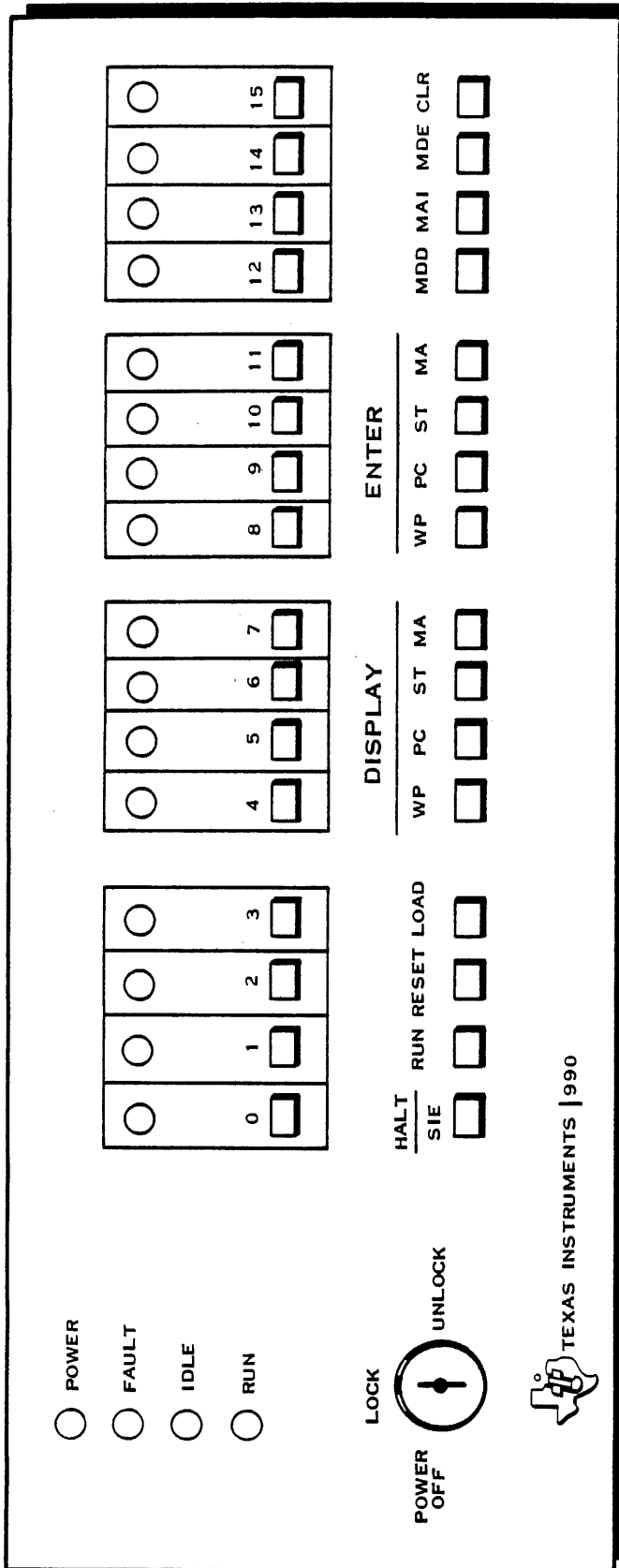
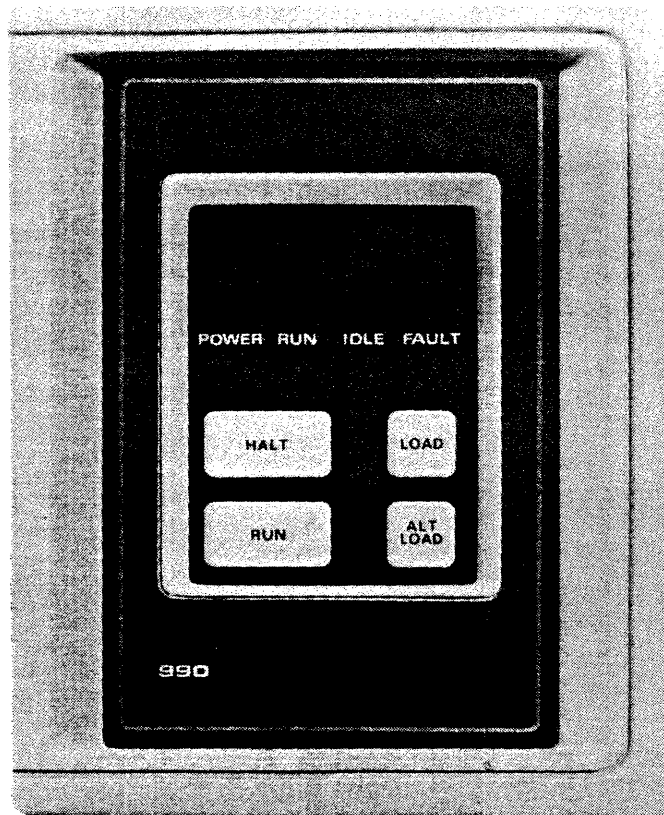


Figure 2-1. Model 990 Computer Programmer Panel

2283042





2283233

**Figure 2-2. Control/Display Module (CDM)**

## **2.5 OBJECT KITS SUPPLIED ON DISK CARTRIDGE**

Paragraph 2.5.1 presents an overview of the operation of building a system disk from a disk cartridge. Detailed instructions for this operation are in paragraph 2.5.2. The system supplied by TI will operate on the hardware configuration given in Table 3-1.

### **2.5.1 Overview of the Disk Build Procedure**

1. Load and initialize the DX10 system that comes on the master DX10 disk cartridge. This procedure is the initial program load (IPL).
2. Format and initialize a new system disk. If necessary, identify and specify known bad tracks on that disk to the operating system.
3. Issue the Copy Volume (CV) command to copy the contents of the DX10 master cartridge to the new system disk.
4. Remove the master disk.

### 2.5.2 Disk Build Procedure

To build a system disk on a base system configuration, perform the following steps:

1. Mount the master DX10 disk cartridge in disk drive unit 1 and a blank disk cartridge in disk unit 0. Remove write protection from disk unit 1 and enable write protection on unit 0.

#### NOTE

The two platters of the DS10, CD1400/32, and CD1400/96 disk drives are treated as two separate disk units. Mount the master DX10 disk cartridge as the removable platter. The nonremovable platter (unit 0) is where the new system will be built.

2. If you are using a Business System 300, turn off power to the computer. Then, turn power back on and continue at step 7 of this list.

If you are using a Business System 600 or 800, press the HALT switch and then the LOAD switch on the front panel and continue with step 7 of this list.

If you are using equipment with a programmer panel (most 990 equipment), continue with step 3. If you are using equipment with a CDM (most 990A13-based equipment), press ALT LOAD and continue with step 8.

3. Press the CLR switch and enter > 0084 on the programmer panel data switches.
4. Press MA under ENTER and then press the CLR switch.
5. Enter > 0400 on the programmer panel data switches and press the MDE switch.
6. Press the LOAD switch. The computer loads and executes the disk IPL program from the master disk.
7. Remove write protection from disk unit 0.
8. Press the Attention key, release it, and hold down the Shift key while you press the exclamation point (!) key to activate the System Command Interpreter (SCI). (Refer to Appendix A for an explanation of keycap names.) When the message WARNING SYSTEM NOT INITIALIZED appears, return to SCI by pressing the Return key and enter the Initialize System (IS) command. The prompts are as follows:

[ ] IS

INITIALIZE DATE AND TIME

YEAR:  
MONTH:  
DAY:  
HOUR:  
MINUTE:

Respond to the prompts as follows:

**YEAR**

Enter a 2- or 4-digit integer that represents the year. For example, you can enter 1982 or 82.

**MONTH**

Enter a 1- or 2-digit integer that represents the month. For example, you can enter 1 or 01 to indicate January.

**DAY**

Enter a 1- or 2-digit integer that represents the current day.

**HOUR**

Enter a 1- or 2-digit integer that represents the current hour (according to the 24-hour clock). For example, 2 PM would be 14.

**MINUTE**

Enter a 1- or 2-digit integer that represents the current minute.

9. After you initialize the system, the following message and prompts appear. Respond to the first two prompts by entering DUMY. Accept the default values for the next three prompts that appear by pressing the Return key.

```
INITIALIZE SYSTEM LOG
      ATTENTION DEVICE:  DUMY
      LOGGING DEVICE:    DUMY
SYSTEM LOG PROCESSING TASK ID: 05E
      ANALYSIS OUTPUT PRINTER: LP01
      USER LOG PROCESSOR TASK ID: 0
```

The warmstart procedure is now complete. The system displays the following message:

**WARMSTART PROCEDURE COMPLETE:**

**NOTE**

In the following step, you must enter the Initialize Disk Surface (IDS) command, rather than the Initialize New Volume (INV) command, if the diagnostic surface analysis or IDS has not previously been performed on the disk. If you specify INV when IDS is required, DX10 will issue a message stating the need to use IDS.

10. Press the Return key to get to SCI command mode. Enter the INV command. The INV command initializes a disk — that is, deletes all files from the file structure and builds the system overhead data. Initialization also allows you to assign the disk a volume name and an interleaving factor. For a description of the INV command, refer to the *DX10 Operations Guide (Volume II)*. The following prompts appear:

```

INITIALIZE NEW VOLUME
                UNIT NAME:
                VOLUME NAME:
        NUMBER OF VCATALOG ENTRIES:
        DEFAULT PHYSICAL RECORD SIZE:
        HARDWARE INTERLEAVING FACTOR:
                FORCE CLEARING OF DISK?:  NO
                USED AS SYSTEM DISK?:    YES
                LISTING ACCESS NAME:
                EXECUTION MODE (F,B):    FOREGROUND

```

Respond to the prompts as indicated:

#### UNIT NAME

The device name of the disk drive containing the target volume. Since the system disk did not load from the normal system disk drive (TILINE\* address > F800 and unit select 0), the device names of the system disk drive and the drive from which the software kit loaded are opposite from those shown in the supported configuration (Table 3-1 or 3-2). In this description, DS02 references unit 0, and DS01 references unit 1. Enter DS02.

#### VOLUME NAME

The name of the system disk being built (for example, VOL1). The name must be alphanumeric — that is, from one to eight characters long, the first character of which must be alphabetic.

#### NUMBER OF VCATALOG ENTRIES

The number of entries in VCATALOG, the system-generated, top-level directory of each disk. The number of VCATALOG entries determines the amount of overhead available for directory and file structure. Press the Return key and DX10 will determine the appropriate value for the disk.

#### DEFAULT PHYSICAL RECORD SIZE

The physical record size to be used for all subsequent files created on the disk. Press the Return key without entering a number and DX10 will determine the appropriate value. For more information on default physical record size, see Volume III.

#### HARDWARE INTERLEAVING FACTOR

The number of disk sectors occurring between consecutive accessed sections. This is a major factor in determining access efficiency for files on flexible diskette and 5.25-inch Winchester disks. Press the Return key and the system determines the appropriate interleaving factor for the disk type.

\* Registered trademark of Texas Instruments Incorporated.

**FORCE CLEARING OF DISK?**

Accept the default value of NO by pressing the Return key unless the disk contains highly sensitive information that must be destroyed.

**USED AS SYSTEM DISK?**

Accept the default value of YES by pressing the Return key; the initialization process installs a bootstrap loader on the disk.

**LISTING ACCESS NAME**

Accept the default value, which is your terminal, or specify a file pathname or printer device name.

**EXECUTION MODE (F,B)**

The INV command procedure can execute in either foreground or background mode. A task executing in foreground mode executes more quickly than in background when other activity is occurring on the system, but no other activity can take place at the executing terminal until the foreground task is finished. A task executing in background mode leaves the terminal free to execute a foreground task. The default value is FOREGROUND.

**BAD TRACK ACCESS NAME**

This is a hidden prompt that you can use in batch mode or in expert mode. Enter the pathname of the file that is to supply bad track information. The file must contain the list of bad tracks in the following format:

HEAD, CYLINDER;  
or  
HEAD, CYLINDER; HEAD, CYLINDER; etc.

Enter the list of known bad tracks in the requested format (for example: 2, 235; 0, 15;). All entries, including the last, must end with a semicolon. All numbers entered are assumed to be decimal. Note that unlike the IDS command, for which you can enter ME in response to this prompt, you can only supply the list of bad tracks in a file for the INV command.

Specifying the list of bad tracks for a DS25 requires special consideration. The bad tracks labeled on the side of the disk pack are labeled as if the pack were formatted as a DS50. For a DS25, however, the system uses half of the cylinders that it uses for a DS50. Therefore, for a DS25, ignore all bad tracks listed for odd numbered cylinders. For each bad track on the disk label with an even numbered cylinder, divide the cylinder number (not the head number) by two, and enter the result for the cylinder number of the bad track.

When you enter the last prompt response, the INV command begins to format the disk. This procedure may access every track on the disk depending on the options and the condition of the disk. A typical moving-head disk (DS10 for example) requires two minutes to complete the format function. The actual time varies with disk speed, access time, and capacity.

11. Enter the CV command to copy the master disk to the blank disk. For more information about the CV command, refer to the *DX10 Operations Guide (Volume II)*. The prompts are as follows:

```

COPY VOLUME
  SOURCE DISK UNIT:
  SOURCE VOLUME NAME:
  DESTINATION DISK UNIT:
  DESTINATION VOLUME NAME:
  LISTING DEVICE:
  VERIFY?: YES
  CONVERT SEQUENTIAL FILES?: NO
  CONVERT REL-REC FILES?: NO
  MORE COPIES?: NO

```

#### SOURCE DISK UNIT

The name of the source disk unit in the form DSxx, where xx is a disk unit number from 01 to 99. Enter DS01.

#### SOURCE VOLUME NAME

The volume name of the disk to be copied. Enter REL36.

#### DESTINATION DISK UNIT

The name of the destination disk unit in the form DSxx, where xx is a disk unit number from 01 to 99. Enter DS02.

#### DESTINATION VOLUME NAME

The volume name of the disk to which CV copies the disk identified by the source volume name. Enter the name that you entered in response to the VOLUME NAME prompt in the INV command.

#### LISTING DEVICE

Enter the device name of the terminal (STxx) that is to be the listing device. If a listing is not required, enter DUMMY.

#### VERIFY?

Although it is not recommended, a copy can be performed without verification if you enter NO in response to this prompt. The initial value is YES.

#### CONVERT SEQUENTIAL FILES?

If the two disks involved in the copy process do not have the same default physical record lengths, physical record length conversion allows more efficient use of the destination disk. You can request physical record length conversion for sequential files by responding YES. The initial value is NO.

#### **CONVERT REL-REC FILES?**

If the two disks involved in the copy process do not have the same default physical record lengths, physical record length conversion allows more efficient use of the destination disk. Also, if the two disks have the same sector sizes, responding YES to this prompt allows CV to perform full compression on unbounded program files. You can request physical record length conversion for relative record files by responding YES. The initial value is NO.

#### **MORE COPIES?**

Enter YES if you wish to make a series of copies. Your terminal then displays the full set of prompts for the next copy to be made. You can make as many as nine copies in a single execution of CV. The initial value is NO.

The new DX10 system disk has been built. You can now load the system from the new disk. If the standard ROM bootstrap loader is available, loading involves no more than pressing the HALT switch and then the LOAD switch on the front panel or, for a Business System 300, turning the power off and on one time. However, before loading the new system and beginning operations (or beginning to generate a custom system), make sure to copy your operating system and store the copy in a safe place. That way, if anything happens to your system disk, you do not have to build another. Refer to Volume II for information on backing up your system disk.

## **2.6 OBJECT KITS SUPPLIED ON MULTIPLE FLEXIBLE DISKETTE MEDIA**

This procedure explains how to use the Disk Build utility to transfer an object kit (typically, operating system software) from a backup copy on several flexible diskettes to a single fixed disk (the target disk). The Disk Build utility is designed for systems based on a WD500 disk drive. The WD500 has a single flexible diskette drive which cannot contain all of the DX10 software at one time and a disk drive whose disk cannot be removed. Therefore, the operating system must reside on the fixed disk, but must be copied to the fixed disk from a series of removable diskettes.

Disk Build can be used with any system that has both a flexible diskette drive and a rigid disk drive. This procedure assumes that the system includes a WD500 in the standard configuration (see Table 3-1). You may need to change your system to match the configuration in Table 3-1. You must also have a set of special flexible diskettes consisting of an initial build diskette (which contains the Disk Build utility) and three or more backup diskettes (which contain the new operating system and system files).

You use Disk Build for any of the following:

- Building the initial system
- Rebuilding a destroyed system
- Replacing an operating system with a new one

**NOTE**

Rebuilding a destroyed system (including your application files and programs) with the Disk Build utility requires that you have previously backed up your operating system software on flexible diskettes using the Backup Directory (BD) or Backup Directory to Device (BDD) command. If you want to accept the default values during Disk Build, you must name the backup file "DX.SYSTEM" when you issue the BD command.

**2.6.1 Disk Build Procedure with a WD500 in Standard Configuration**

The following procedure initiates and executes Disk Build. To terminate Disk Build prematurely, enter a dollar sign (\$) whenever a prompt offers an opportunity to respond. To begin the procedure, first mount the build diskette into the flexible diskette drive; do not write protect the diskette. Ensure that the Winchester disk(s) is in a ready state and write protected. To load the Disk Build utility from the diskette to system memory, perform an IPL from the flexible diskette, as follows:

1. If you are using a Business System 300, turn off power to the computer. Then turn power on again and continue with step 7 of this list.

If you are using a Business System 600 or 800, press the HALT switch and then the ALTERNATE LOAD switch on the front panel. Then continue with step 7 of this list.

If you are using equipment with a programmer panel (most 990 equipment), continue with step 2.

2. Press CLR and enter > 0084 on the programmer panel data switches.
3. Press ENTER MA on the programmer panel.
4. Press MDD and enter > 0200 on the programmer panel data switches.
5. Press MDE on the programmer panel.
6. Press LOAD. The loader code begins loading the Disk Build software.
7. Remove write protection from the Winchester disk.

After the system loads the Disk Build software, it immediately begins executing and displays the following message on one or more terminals:

```
***DISK BUILD UTILITY***
DO YOU WANT TO CHANGE ANY DEFAULT VALUES? (Y/N):
```

The appropriate response to this question is N except in three cases:

- When the target disk is not located at the TILINE address default value, > F800, unit 0



- When you wish to set special values for the following system disk parameters: volume name, physical record size, number of VCATALOG (the volume directory) entries, or hardware interleaving factor
- When you wish to specify a source file other than the default file (DX.SYSTEM) on the backup disks

In any of these three cases, enter a Y in response to the message and refer to the paragraph 2.6.2 for further explanation.

Whether you enter Y or N, only the terminal you are using remains active. All other terminal screens that displayed the introductory message become clear. A digital clock display appears in the bottom right corner of your screen and monitors the execution time of the Disk Build utility.

Disk Build now inspects the target disk and evaluates its condition. Depending on the condition of the disk, one of the following messages appears:

**DISK REQUIRES SURFACE ANALYSIS**

or

**DISK REQUIRES INITIALIZATION**

or

**VOLUME NAME: xxxxxxxx  
THE SPECIFIED DISK COULD CONTAIN SOME  
INFORMATION ABOUT YOUR BUSINESS.  
IF N IS ENTERED, THE DISK WILL BE  
ERASED. IF Y IS ENTERED, ONLY SYSTEM  
INFORMATION WILL BE ERASED. IF THERE IS  
INFORMATION THAT MUST BE SAVED, ENTER  
Y. IF THE INFORMATION CAN BE  
REPLACED, ENTER N.**

**SHOULD THE USER INFORMATION ON THE SYSTEM  
DISK BE SAVED?(Y/N)**

The message **DISK REQUIRES SURFACE ANALYSIS** appears if Disk Build does not find a bad track map on the target disk. As Disk Build begins the work of analyzing and formatting the disk, the following message appears:

**BEGIN STEP 1.**

Disk Build automatically issues an IDS command, which analyzes the target disk surface for physical flaws and lists the location of any flaw on a bad track map. As Disk Build performs the IDS command, a graph is displayed to show the percentage of surface analysis completed. When IDS completes, a list of bad tracks is displayed. Then, Disk Build performs an INV command, which creates a volume directory, assigns a volume name, and installs the volume on the target disk. In this case, Disk Build uses default values for the INV parameters.

The message **DISK REQUIRES INITIALIZATION** appears on the screen if Disk Build determines that an **IDS** has been performed on the target disk but that **VCATALOG** does not exist. The following message appears:

**BEGIN STEP 1.**

Disk Build automatically issues an **INV** command on the target disk using default values.

The third message appears on the screen if Disk Build finds **VCATALOG** on the target disk. This message means that software or data might already exist on the target disk. If you have previously created directories or installed programs, you may wish to preserve them. If so, enter **Y**. Disk Build then issues the **Install Volume (IV)** command. If you do not wish to save the previously existing software, enter **N**. Disk Build will then issue an **INV** command.

Under any circumstances Disk Build deletes the following system files:

- **.\$PROGA**
- **.\$IMAGES**
- **.\$LOADER**
- **.\$TCALIB**
- **.\$OVLYA**
- **.\$SDS\$**

Any data in these files is destroyed.

If Disk Build issues an **IDS** command, a graph is displayed to show the percentage of surface analysis completed. When **IDS** completes, a list of bad tracks is displayed.

Once the new volume is installed, Disk Build copies the remainder of the Disk Build operating system software from the build diskette to the target disk using the **CD** command. During this process, the following message appears on the terminal:

**BEGIN STEP 2.**

During the **CD** process, Disk Build creates a listing file called **.\$LISTCD** on the target disk. At this time, the system files are recreated. The following message appears on the terminal:

**REMOVE INITIAL BUILD MEDIA.  
THEN TYPE Y TO CONTINUE:**

Remove the build diskette from the diskette drive.

When you type Y, Disk Build automatically loads the system from the newly created files on the target disk. The following message appears on the screen:

SYSTEM LOAD INITIATED — WAIT ONE MINUTE.  
IF NO MESSAGES APPEAR, START OVER AT STEP 1.

- If no messages appear on the screen after one minute, the load process was not successful. The data lights on the front panel should display a crash code. For an explanation of crash codes, refer to *DX10 Error Reporting and Recovery Manual (Volume VI)*. After correcting the cause of the crash, repeat this procedure from the IPL.
- If the load is successful, the following message appears on the terminal:

MOUNT BACKUP DATA VOLUME 1.  
THEN TYPE Y TO CONTINUE:

Insert the first backup diskette, labeled Backup Disk Volume 1.

When the diskette is ready, enter Y. A response other than Y (or \$, which aborts Disk Build) causes the message MOUNT BACKUP DATA VOLUME 1 to reappear.

Disk Build issues a Restore Directory (RD) command to copy the first part of the operating system software from the source diskette to the target disk. The following message appears on the screen:

BEGIN STEP 3.

During this step, Disk Build deletes any files that could cause a conflict with the new system. Specifically, Disk Build deletes .\$\$PROGA, .\$\$SDS\$, and .\$\$IMAGES. Disk Build issues a Modify File Protection (MFP) command to allow replacement of .\$\$LOADER, .\$\$TCALIB, and .\$\$OVLYA.

Disk Build requests additional volumes as needed; the following message appears on the screen:

MOUNT VOLUME < x > ; TYPE \$ TO QUIT; Y TO CONTINUE:

In this message, < x > is the number of the next volume to install.

Remove the current diskette and mount volume < x > as requested.

When Disk Build has copied the entire DX10 operating system to the target disk, the following message appears on the screen:

BUILD PROCESS COMPLETED.  
REMOVE BACKUP MEDIA.  
THEN TYPE Y TO CONTINUE:

Remove the diskette. When you type Y, the following message appears on the screen:

SYSTEM LOAD INITIATED — WAIT 1 MINUTE.  
THEN LOG ON.

After four seconds, the system initiates an IPL and the screen becomes blank. When the IPL finishes, the DX10 operating system is loaded, and your computer is functional. To access SCI, press the Attention key, release it, and hold down the Shift key while you press the exclamation point (!) key. (Refer to Appendix A for an explanation of keycap names.) The following message appears on the bottom line of the screen:

**WARNING! SYSTEM IS NOT INITIALIZED:**

If this warning message does not appear, observe the data lights on the front panel for a crash code.

At this point you have installed the DX10 operating system. It is necessary to issue an IS command to the system. The prompts are as follows:

[ ] IS

INITIALIZE DATE AND TIME

YEAR:  
MONTH:  
DAY:  
HOUR:  
MINUTE:

Respond to the prompts as follows:

**YEAR**

Enter a 2- or 4-digit integer that represents the year. For example, you can enter 1982 or 82.

**MONTH**

Enter a 1- or 2-digit integer that represents the month. For example, you can enter 1 or 01 to indicate January.

**DAY**

Enter a 1- or 2-digit integer that represents the current day.

**HOUR**

Enter a 1- or 2-digit integer that represents the current hour (according to the 24-hour clock). For example, 2 PM would be 14.

**MINUTE**

Enter a 1- or 2-digit integer that represents the current minute.

After you initialize the system, the following message and prompts appear. You should initialize the system log so that it only uses files for log messages. The following example demonstrates this:

```
INITIALIZE SYSTEM LOG
      ATTENTION DEVICE:  DUMY
      LOGGING DEVICE:    DUMY
SYSTEM LOG PROCESSING TASK ID: 05E
      ANALYSIS OUTPUT PRINTER: LP01
      USER LOG PROCESSOR ID:  0
```

The warmstart procedure is complete. The system briefly displays the following message:

```
WARMSTART PROCEDURE COMPLETE:
```

When initialization is complete, you can customize your system, if necessary (see Section 3).

#### NOTE

The following file names are reserved pathnames. Do not use these pathnames for your own files because the Disk Build deletes them:

- .B\$PROGA
- .B\$LOADER
- .B\$PASS
- .B\$CONTRL
- .B\$LISTRD
- .B\$LISTCD
- .B\$OVLYA

#### 2.6.2 Variations to Disk Build Procedure when Changing Default Values

When the Disk Build utility begins execution, the message DO YOU WANT TO CHANGE DEFAULT VALUES? (Y/N) appears on the terminal screen. Enter Y in any of the following cases:

- The system disk (target disk) is not located at TILINE address > F800, unit 0.
- You wish to change one or more volume characteristics.
- You wish to specify your own backup file (other than the default backup file, DX.SYSTEM). If your response is N, this paragraph does not apply to your Disk Build procedure.

During the execution of Disk Build, you have three opportunities to change default values:

- When Disk Build is being loaded, you can specify the location of the system (target) disk.
- Before Disk Build copies itself from the initial build diskette to the target disk, you can change the default volume characteristics of the target disk.
- Before Disk Build restores the first backup diskette to the target disk, you can name your backup file.

**2.6.2.1 Specifying the Target Disk.** Immediately after you respond Y to the previously displayed question, DO YOU WANT TO CHANGE DEFAULT VALUES?, a message appears on the screen. The content of this message varies with each system, depending on what drives are configured and online with the particular system. The following message is a typical example:

```

SYSTEM WILL BE BUILT ON ONE OF THE FOLLOWING DISKS:
      ID      ADDRESS      UNIT      INTERRUPT      TYPE
      1      F800         01         13      WD500 — 5M
      2      F800         00         13      WD500 — 5M
      3      F820         00          9      FD1000 — 1M
INPUT ID NUMBER OF DESIRED DISK UNIT:

```

The table of disk drives listed in the message represents all of the disks that are online in the computer system. Choose the disk you want to be the system disk (target disk) for your computer, and enter the corresponding ID number from the left-hand column of the table.

**2.6.2.2 Changing Volume Characteristics of the Target Disk.** Next, Disk Build inspects the target disk and evaluates its condition. Depending on the condition of the disk, one of the following messages appears:

DISK REQUIRES SURFACE ANALYSIS

or

DISK REQUIRES INITIALIZATION

or

VOLUME NAME: xxxxxxxx  
 THE SPECIFIED DISK COULD CONTAIN SOME  
 INFORMATION ABOUT YOUR BUSINESS.  
 IF N IS ENTERED, THE DISK WILL BE  
 ERASED. IF Y IS ENTERED, ONLY SYSTEM  
 INFORMATION WILL BE ERASED. IF THERE IS  
 INFORMATION THAT MUST BE SAVED, ENTER  
 Y. IF THE INFORMATION CAN BE  
 REPLACED, ENTER N.

SHOULD THE USER INFORMATION ON THE SYSTEM  
 DISK BE SAVED?(Y/N)

The prompts that follow the first two messages (DISK REQUIRES SURFACE ANALYSIS and DISK REQUIRES INITIALIZATION) allow you to change default values. The other message is the same as if you were accepting defaults; treat it as described previously.

If the message DISK REQUIRES SURFACE ANALYSIS appears, Disk Build has determined that it must issue an IDS and an INV command on the target disk. The following prompts appear:

VOLUME NAME: (SYSTEM)  
NUMBER OF VCATALOG ENTRIES:  
PHYSICAL RECORD SIZE: (xxx)  
HARDWARE INTERLEAVING FACTOR:

You can change these four target disk parameters that are set during the INV command.

#### VOLUME NAME

If the default volume name SYSTEM is acceptable, press the Return key on your terminal. If you want to designate another name, enter that name at this time. The volume name can be up to eight characters long.

#### PHYSICAL RECORD SIZE

The default value of the physical record size depends on the model of the target disk. To designate a particular physical record size, enter that value in decimal notation. Refer to Volume III for more information on default physical record size.

#### NUMBER OF VCATALOG ENTRIES

To designate your own value for this prompt, enter that value now. Although no default value appears for this prompt, Disk Build supplies a default value if you press the Return key without entering a value. The default value varies depending on the model of the target disk.

#### HARDWARE INTERLEAVING FACTOR

To designate your own value for this prompt, enter that value now. Although no initial value appears, Disk Build supplies a default value if you press the Return key without entering a value.

After you answer this set of prompts, the following prompt appears:

RESTORE BAD TRACK LIST? \_\_

If you want to restore the bad track list if available, answer Y. If you answer N or receive an error after answering Y, Disk Build performs the IDS command.

When you answer Y, the following prompts appear:

ENTER BAD TRACKS IN THE FORMAT:

HEAD, CYLINDER;  
OR  
HEAD, CYLINDER; HEAD, CYLINDER; ETC.

TO END LIST, ENTER AN EMPTY LINE

Enter the list of known bad tracks in the requested format using decimal numbers for the values (for example, 2, 235; 0, 15;). All entries, including the last, must end with a semicolon. Improper positioning or absence of the punctuation marks causes an error to be returned.

When you use the INV command for a DS25 disk, divide cylinder addresses in half before entering them in response to the BAD TRACK FORMAT message. Because the DS25 uses every other cylinder on the disk pack, all tracks with odd cylinder addresses are ignored. You must enter the addresses (divided by two) of even number tracks only. For example, to enter HEAD number 4, CYLINDER number 422, you would respond to the BAD TRACK FORMAT message as follows:

```
HEAD 4              CYLINDER 211
```

If the message DISK REQUIRES INITIALIZATION appears, Disk Build has determined that the target disk has been analyzed but not initialized. A series of prompts follow. These prompts also appear if you answer N to the question SHOULD THE USER INFORMATION ON THE SYSTEM DISK BE SAVED?(Y/N) in the third message and you previously answered Y to change the default values. The first prompt is as follows:

```
PERFORM INV?
```

If you want to reformat the disk, answer Y. (This is the equivalent of issuing the INV command with the forced clearing of disk specified.) Otherwise, answer N.

If you answer Y, the following prompt appears:

```
PERFORM IDS?
```

If you want to perform surface analysis of the disk regardless if it has been done before, answer Y. Otherwise, answer N.

The following prompts appear next:

```

                                VOLUME NAME: (SYSTEM)
      NUMBER OF VCATALOG ENTRIES:
                                PHYSICAL RECORD SIZE: (xxx)
      HARDWARE INTERLEAVING FACTOR:
```

You can change these four target disk parameters that are set during the INV command. These prompts and the following prompt are discussed in the preceding explanation for the DISK REQUIRES SURFACE ANALYSIS message.

```
RESTORE BAD TRACK LIST? __
```

At this point, Disk Build continues to run in the same way as if you had chosen to accept default values. Disk Build performs either an IDS or an INV command (step 1) or an IV command. Then, Disk Build copies the first diskette to the target disk (step 2) and performs an IPL. If the IPL is successful, a message appears instructing you to replace the initial build diskette with the first backup diskette and enter a Y to initiate an RD command.



**2.6.2.3 Specifying the Backup File.** When Disk Build restores the directory from the backup diskette to the target disk, the default pathname of the target directory is DX.SYSTEM. If you have chosen to change default values, the following message appears on the screen:

ENTER THE PATHNAME OF SEQUENTIAL BACKUP FILE: DX.SYSTEM

Enter the pathname of the backup file for the RD command.

This concludes the differences between the default-accepting mode and the default-changing mode. Disk Build will display messages requesting the backup disks. Disk Build continues from this point as described in the standard DSKBLD procedure described in paragraph 2.6.1.

### **2.6.3 Variations to Disk Build Procedure for FD1000**

You should configure your equipment so it conforms to the DS04 device as given in Table 3-1. If you are using equipment with a programmer panel (most 990 equipment), enter > 0082 on the data switches in step 2 instead of > 0084. In step 4, enter > F810 on the data switches instead of > 0200. Then continue with step 6. If you are using equipment with a CDM (most 990A13-based equipment), press ALT LOAD in step 2.

## **2.7 OBJECT KITS SUPPLIED ON TAPE MEDIA**

The following procedure describes how to generate a DX10 operating system on a disk cartridge from the two tape reels or cartridges that make up the tape medium software kit. The procedure assumes the use of the standard ROM loader, a base system configuration (see Table 3-1 or 3-2 in Section 3), and either a 911 VDT, a 940 EVT, or a Business System VDT.

### **CAUTION**

**The program for building the disk from tape assumes that the devices listed in Table 2-2 are assigned the corresponding interrupt. Otherwise, the system goes into an infinite loop when it tries to process an interrupt from that device.**

Table 2-2. Minimum Configuration for TAPBLD

Device	Address	Interrupt	Baud Rate
Disk	TILINE > F800	13 (unit 0)	
Magnetic tape	TILINE > F880	9 (unit 0)	
911 VDT	CRU > 0100	10	
Business System terminal	CRU > 1700	8	9600
or		or	
931 VDT (channel 0)*	TILINE > F980	11	9600
<b>Note:</b>			
* You can configure a Business System terminal or a 931 VDT at either CRU address > 1700, interrupt 8, or TILINE address > F980, interrupt 11. The terminal should use 9600 baud.			

To begin the Disk Build from magnetic tape, perform the following steps:

1. Mount the software kit magnetic tape labeled BUILD TAPE (TI part number 939151-1301) on tape unit 0 with write protection enabled and make tape unit 0 ready for operation.
2. Place a disk cartridge on the system disk drive (first controller) unit 0, and make disk unit 0 ready for operation with the write protection enabled.
3. If you are using a Business System 300, turn off power to the computer. Then, turn power back on and continue after step 9 of this list.

If you are using a Business System 600 or 800, press the HALT switch and then the ALTERNATE LOAD switch on the front panel. Then continue after step 9 of this list.

If you are using equipment with a programmer panel (most 990 equipment), continue with step 4. If you are using equipment with a CDM (most 990A13-based equipment), press the HALT switch and then the ALT LOAD switch, and continue after step 9 of this list.

4. Press the CLR switch, and enter > 0082 in the programmer panel data switches.
5. Press the ENTER MA switch on the programmer panel.
6. Press the CLR switch, and enter > F880 in the programmer panel data switches.
7. Press the MDE and MAI switches on the programmer panel (in succession).
8. Press the CLR switch, and enter > 8000 in the programmer panel data switches.
9. Press the MDE switch and then the LOAD switch on the front panel.

The tape unit then reads the TAPBLD program from the tape to memory. This program contains several steps performed automatically. Appropriate messages appear on the screen as each step executes. Once the program is loaded and begins execution, the following message is displayed on one or more of the terminals:

```
***DISK BUILD UTILITY***  
DO YOU WANT TO CHANGE ANY DEFAULT VALUES? (Y/N):
```

At this point, remove write protection from the disk unit.

The appropriate response to this question is N except when you wish to set special values for the following system disk parameters: volume name, physical record size, number of VCATALOG (the volume directory) entries, or hardware interleaving factor.

If you want to set special values for these items, enter a Y in response to the message and refer to paragraph 2.7.1 for further explanation.

Whether you enter Y or N, only the terminal you are using remains active. All other terminal screens that displayed the introductory message become clear. A digital clock display appears in the bottom right corner of your screen and monitors the execution time of the Tape Build utility.

Tape Build now inspects the target disk and evaluates its condition. A running account of the data tracks as they are accessed appears in the top righthand corner of the screen. Depending on the condition of the disk, one of the following messages appears:

DISK REQUIRES SURFACE ANALYSIS

or

DISK REQUIRES INITIALIZATION

The message DISK REQUIRES SURFACE ANALYSIS appears if Tape Build does not find a bad track map on the target disk. As Tape Build begins the work of analyzing and formatting the disk, the following message appears:

BEGIN STEP 1.

Tape Build automatically issues an IDS command, which analyzes the target disk surface for physical flaws and lists the location of any flaw on a bad track map. As Tape Build performs the IDS command, a graph is displayed to show the percentage of surface analysis completed. When IDS completes, a list of bad tracks is displayed. Then, Tape Build performs an INV command, which creates a volume directory, assigns a volume name, and installs the volume on the target disk. In this case, Tape Build uses default values for the INV parameters.

The message DISK REQUIRES INITIALIZATION appears on the screen if Tape Build determines that an IDS has been performed on the target disk but that VCATALOG does not exist. The following message appears:

BEGIN STEP 1.

Tape Build automatically issues an INV command on the target disk using default values.

Once the new volume is logically installed, Tape Build copies the remainder of the Tape Build operating system software from the tape to the target disk. During this process, the following message appears on the terminal:

BEGIN STEP 2.

At this time, the system files are recreated. As directories and files are created, their names appear on the screen, as follows:

```
**VCATALOG**
**S$DIAG **
**S$ROLLA **
**B$LOADER**
**B$PROC **
**B$OVLYA **
**S$IMAGES**
**B$PROGA **
**B$PASS **
```

S\$DIAG is not created on an FD1000 floppy disk.

Tape Build automatically loads the system from the newly created files on the target disk. The following message appears on the terminal:

```
REMOVE INITIAL BUILD MEDIA.
TYPE Y TO CONTINUE:
```

Unload and remove the tape. When you type Y, the following message appears on the terminal:

```
SYSTEM LOAD INITIATED — WAIT ONE MINUTE
IF NO MESSAGES APPEAR, START OVER AT STEP 1
```

If no further messages appear on the screen after one minute, the load process was not successful. The data lights on the computer chassis should display a crash code. For an explanation of crash codes, refer to Volume VI. After correcting the cause of the crash, repeat this procedure from step 1 in the preceding list.

If the load is successful, the following message appears on the terminal:

```
MOUNT BACKUP DATA VOLUME 1
THEN TYPE Y TO CONTINUE
```

Insert the backup tape, labeled Backup Tape Volume 1.

When the tape is ready, enter Y. A response other than Y (or \$, which aborts Tape Build) causes the message MOUNT BACKUP DATA VOLUME 1 to reappear.

Tape Build issues an RD command to copy the first part of the operating system software from the source tape to the target disk. The following message appears on the screen:

BEGIN STEP 3.

During this step, Tape Build deletes any files that could cause a conflict with the new system. Specifically, Tape Build deletes .S\$PROGA, .S\$SDS\$, and .S\$IMAGES. Tape Build issues an MFP command to allow replacement of .S\$LOADER, .S\$TCALIB, and .S\$OVLYA.

Tape Build requests additional volumes as needed; the following message appears on the screen:

**MOUNT VOLUME <x> ; TYPE \$ TO QUIT; Y TO CONTINUE:**

In this message, x is the number of the next volume to install.

When Tape Build has copied the entire DX10 operating system to the target disk, the following message appears on the screen:

**BUILD PROCESS COMPLETED.  
REMOVE BACKUP MEDIA.  
THEN TYPE Y TO CONTINUE:**

Unload and remove the tape. When you type Y, the following message appears on the screen:

**SYSTEM LOAD INITIATED — WAIT 1 MINUTE.  
THEN LOG ON.**

After four seconds, the system executes an IPL and the screen becomes blank. When the IPL is complete, the screen remains blank. The DX10 operating system is now loaded, and your computer is functional. To access SCI, press the Attention key, release it, and hold down the Shift key while you press the exclamation point (!) key. (Refer to Appendix A for an explanation of keycap names.) The following message appears on the bottom line of the screen:

**WARNING! SYSTEM IS NOT INITIALIZED:**

If this warning message does not appear, observe the data lights on the front panel for a crash code and refer to Volume VI.

Otherwise, issue an IS command to the system. The prompts are as follows:

[ ] IS

**INITIALIZE DATE AND TIME  
YEAR:  
MONTH:  
DAY:  
HOUR:  
MINUTE:**

Respond to the prompts as follows:

**YEAR**

Enter a 2- or 4-digit integer that represents the year. For example, you can enter 1982 or 82.

**MONTH**

Enter a 1- or 2-digit integer that represents the month. For example, you can enter 1 or 01 to indicate January.

**DAY**

Enter a 1- or 2-digit integer that represents the current day.

**HOUR**

Enter a 1- or 2-digit integer that represents the current hour (according to the 24-hour clock). For example, 2 PM would be 14.

**MINUTE**

Enter a 1- or 2-digit integer that represents the current minute.

After you initialize the system, the following message and prompts appear. Respond to the first two prompts by entering the device name ME, as follows:

```
INITIALIZE SYSTEM LOG
ATTENTION DEVICE:  ME
LOGGING DEVICE:   ME
```

Accept the initial values for the next three prompts that appear by pressing the Return key.

```
SYSTEM LOG PROCESSING TASK ID: 05E
ANALYSIS OUTPUT PRINTER:  LP01
USER LOG PROCESSOR TASK ID:  0
```

The warmstart procedure is complete. The system displays the following message:

**LOG STARTED**

**WARMSTART PROCEDURE COMPLETE:**

When initialization is complete, you can customize your system, if necessary (see Section 3).

**NOTE**

The following file names are reserved pathnames. Do not use these pathnames for your own files because the Tape Build deletes them:

- .B\$PROGA
- .B\$LOADER
- .B\$PASS
- .B\$CONTRL
- .B\$LISTRD
- .B\$OVLYA
- .B\$LISTCD

**2.7.1 Variations to Tape Build Procedure when Changing Default Values**

When the Tape Build utility begins execution, the message DO YOU WANT TO CHANGE DEFAULT VALUES? (Y/N) appears on the terminal screen. Enter Y if you want to change one or more volume characteristics. If your response is N, this paragraph does not apply to your Tape Build procedure.

After you respond Y to the question, a message appears on the screen. The content of this message varies with each system, depending on which drives are configured with the system. The following message is a typical example:

```
SYSTEM WILL BE BUILT ON ONE OF THE FOLLOWING DISKS:
      ID      ADDRESS      UNIT      INTERRUPT      TYPE
      1      F800         01         13      WD800 — 43MB
      2      F800         02         13      WD500A — 20MB
      3      F800         03         13      WD500A — 20MB
      4      F800         04         13      UNKNOWN DISK
INPUT ID NUMBER OF DESIRED DISK UNIT:
```

The table of disk drives listed in the message represents disks that may be specified to build the initial system. Choose the disk that is online to your system, and enter the corresponding ID number from the left-hand column of the table.

Before Tape Build copies itself from the initial build tape to the target disk, you can change the default volume characteristics of the system disk. Tape Build inspects the target disk and evaluates its condition. Depending on the condition of the disk, one of the following messages appears:

**DISK REQUIRES SURFACE ANALYSIS**

or

**DISK REQUIRES INITIALIZATION**

The prompts following the two messages, **DISK REQUIRES SURFACE ANALYSIS** and **DISK REQUIRES INITIALIZATION**, allow you to change default values.

If the message **DISK REQUIRES SURFACE ANALYSIS** appears, Tape Build has determined that it must issue an **IDS** and **INV** command on the target disk. The following prompts appear:

**VOLUME NAME: (SYSTEM)**  
**NUMBER OF VCATALOG ENTRIES:**  
**DEFAULT PHYSICAL RECORD SIZE: (xxx)**  
**INTERLEAVING FACTOR:**



You can change these four target disk parameters that are set during the INV command.

#### VOLUME NAME

If the default volume name SYSTEM is acceptable, press the Return key on your terminal. If you want to designate another name, enter that name at this time. The volume name can be up to eight characters long. Note that you cannot backspace the cursor while responding to this prompt. Be sure to enter the VOLUME NAME correctly the first time.

#### DEFAULT PHYSICAL RECORD SIZE

The default value of the physical record size depends on the model of the target disk. To designate a particular physical record size, enter that value in decimal notation. Refer to Volume III for more information on default physical record size.

#### NUMBER OF VCATALOG ENTRIES

To designate your own value for this prompt, enter that value now. Although no default value appears for this prompt, Tape Build supplies a default value if you press the Return key without entering a value. The default value varies depending on the model of the system disk.

#### INTERLEAVING FACTOR

To designate your own value for this prompt, enter that value now. Although no initial value appears, Tape Build supplies a default value if you press the Return key without entering a value.

After you answer this set of prompts, the following prompt appears:

RESTORE BAD TRACK LIST? \_\_

If you want to restore the bad track list if available, answer Y. If you answer N or receive an error after answering Y, Disk Build performs the IDS command.

When you answer Y, the following prompts appear:

ENTER HEAD AND CYLINDER ADDRESS OF KNOWN BAD TRACKS.  
 ENTER THE ADDRESSES ONE PER LINE.  
 END THE LIST BY ENTERING RETURN ONLY FOR THE NEXT HEAD.  
 HEAD #

Enter the addresses of any known bad tracks on this volume. Enter the decimal number for the head on the line prompting HEAD #. Then press the Return key. Do not use any special characters such as commas or semicolons. Tape Build then issues the prompt CYLINDER #. Enter the decimal value of the cylinder number immediately after this prompt. Then press the Return key. Do not use any special characters such as commas or semicolons. Tape Build then prompts you for the next head and cylinder number in the same manner. When the list of head and cylinder pairs is complete, press only the Return key in response to the HEAD # prompt to terminate your input.

When you use the INV command for a DS25 disk, divide cylinder addresses in half before entering them in response to the CYLINDER # prompt. Because the DS25 uses every other cylinder on the disk pack, all tracks with odd cylinder addresses are ignored. You must only enter the addresses (cylinder divided by two) of tracks with even cylinder numbers.

## 2.8 DUPLICATION OF A BUILT SYSTEM

You can create a copy of a system disk capable of performing an IPL with one of several methods. If your system has more than one disk drive, you can produce a copy directly by using the Copy Volume (CV) command. If the disks are the same type, you can use the CVD or DCOPY command. If you have only one disk and it is removable, and you have a magnetic tape, you can use DCOPY. If your only disks are not removable, the best way is to use the procedures in paragraph 2.10 to create a backup that can be used with Disk Build; then use BDD to back up your data files.

You can create a system disk by using the following steps 1 through 5. This method is useful if your output disk is too small to use CV, or if you wish to produce a system disk that has fewer functions than a complete DX10 system. If you are trying to produce a smaller version of a DX10 system, you can delete tasks from S\$IMAGES and S\$PROGA, and you can delete command procedures from S\$PROC. Volume II contains information in the SCI command index table on what parts of S\$PROGA are used to service the command procedures. Make a copy of the program files; delete the desired tasks, procedures, and overlays from the copy; and copy twice with CD to compress the program files.

The system files necessary to have an operational DX10 system are S\$PROC, S\$OVLYA, S\$LOADER, S\$PROGA, and S\$IMAGES. You need to have the procedures necessary for system initialization and application execution in S\$PROC.

1. Place a new disk volume in a secondary disk unit.
2. Initialize the volume using the INV command. Answer YES to the USED AS SYSTEM DISK prompt to install the bootstrap loader on the disk.
3. Issue the CD command to copy the desired files to the new disk. (These files are the system files mentioned previously and whatever other files and directories are necessary for the system's intended use.) If you use the RPRL option, be sure to recopy .S\$SYSGEN.GENDAT without RPRL. Use the Verify Copy (VC) command to verify the copy. Accept the default value to include the system files.
4. Enter the CSF command to create the .S\$PRINT, .S\$ROLLA, and .S\$CRASH system files.
5. Use the MVI command to specify to the system loader the names of the primary system image, the program file, the overlay file, and the loader file.

Refer to Volume II for detailed descriptions of the CSF and MVI commands.

## **2.9 BACKUP AND RESTORE**

Volume II gives specific information on how to back up your system disk. It explains backup procedures and differentiates among the following six commands:

- Disk Copy/Restore (DCOPY)
- Copy Directory (CD)
- Backup Directory (BD)
- Copy and Verify Disk (CVD)
- Copy Volume (CV)
- Backup Directory to Device (BDD)

Briefly, DCOPY is the fastest but offers no disk compression. DCOPY can copy from tape to disk and vice versa, and from disk to disk. CVD and CV can only copy from disk to disk. The CVD and CV commands offer file compression and the capability of avoiding bad tracks. CVD and DCOPY require that the input disk and output disk be the same size and type. DCOPY requires that the destination disk and source disk have the same number of tracks, sectors per track, and bytes per sector even if there is an intervening tape copy. CV can copy between disks of different sizes and types.

CD is a more general copy that copies disks of different types. It offers disk compression, selection of which files to copy, and conversion of physical record length of files. BD and BDD copy the disk onto one sequential file, magnetic tape, or disk (or DSDD diskette). They can copy onto several tape or disk volumes if necessary. If you are backing up a disk onto several volumes, you must use BD or BDD. See Volume II for backup and restore instructions.

## **2.10 BUILDING A CUSTOM DISKETTE OR TAPE TO USE WITH DISK BUILD**

You can produce a backup on tape or diskette for use with Disk Build that contains a custom system generation and your custom software. This can shorten the time it takes you to build a new system or recover from problems with your system disk if you have a system without any removable disks. This paragraph gives the details on what you need to include in the custom backup.

Your custom backup must contain the following system files:

- **S\$SDS\$**
- **S\$PROC**
- **S\$OVLYA**
- **S\$LOADER**
- **S\$PROGA**
- **S\$IMAGES**
- **S\$MVI**

The file **S\$SDS\$** contains the assembler, the Link Editor, and programs from installation of language packages such as the COBOL compiler. If you do not need any of these functions on the system you are building, you may omit this file.

**S\$PROC** need only contain those command procedures necessary for operation of the system you are building and its applications. You can delete the command procedures for functions you do not need.

**S\$OVLYA** and **S\$LOADER** must be present.

**S\$PROGA** need only contain those command processors necessary for operation of your applications and the operating system. Volume II contains a table that shows what parts of **S\$PROGA** are necessary for each of the standard DX10 command procedures. You can delete those command processors for the functions you do not need. However, it is wise to retain some functions needed for diagnosis of problems, such as **ANALZ**. Make a copy of **S\$PROGA**; delete the tasks, procedures, and overlays not needed; and copy it to the disk to be backed up with CD. The **BD** command that creates the backup finishes the process of compressing the program file.

The **S\$IMAGES** file must contain the system image designated for use by the contents of the **S\$MVI** file. The Modify Volume Information (MVI) processor creates **S\$MVI** when it is executed on a disk that is initialized as a system disk, has all the system files on it, and is installed in the system (IV). The file can also be created by the Text Editor. See Volume II for more information on the **MVI** command.

The Disk Build software takes a certain amount of space on the disk being built. Be sure the disk or directory you back up is small enough so the disk being built does not fill up during the restore directory process.

#### **NOTE**

The disk from which you copy to produce the backup should not contain any of the files that start with **B\$**. The presence of any such file in the backup may cause the **RD** process of the Disk Build to return an error.

The Disk Build program attempts to run MVI even if errors occur during the RD process. However, if errors occur, it does not attempt to perform an IPL for the system it just built. At that point, you should not perform an IPL. You can use the log-on sequence to run a skeleton SCI. Use a .SHOW .B\$LISTRD SCI primitive to inspect the listing from RD. The error messages in that file allow you to determine what the problem is.

If the system builds successfully but an IPL cannot be performed, the most likely cause is that the S\$MVI file is either missing or specifying the wrong system image.

## 2.11 SYSTEM FILES

DX10 maintains specific system files related to DX10 operations in each disk volume as a part of system overhead. System files allow the operating system to function.

Ordinarily, you do not want to back up these files because they are already on the system disk and they are not often changed or easily built. Therefore, select the NOSYSFILE option on the CD and BD commands to exclude these files. However, if you are building a system disk or generating an alternative system disk to use with a specific application, you want to be sure they are included in the copy procedure. Accept the default.

Because DX10 requires these files, do not change the characteristics of any of them and do not alter the contents of the first six listed except as patched by released patch batch streams. DX10 system files include the following:

System program file	.S\$PROGA
System image file	.S\$IMAGES
Overlay file	.S\$OVLYA
Roll file	.S\$ROLLA
System crash file	.S\$CRASH
System loader	.S\$LOADER
Diagnostic file	.S\$DIAG
SCI procedure directory	.S\$PROC
Background terminal communications area	.S\$BGTC
Foreground terminal communications area	.S\$FGTC
Background terminal local file	.S\$BTLFXX
Foreground terminal local file	.S\$FTLFXX
Terminal library	.S\$TCALIB
Volume directory	.VCATALOG
LP\$X directory	.S\$PRINT
System log file 1	.S\$\$\$SLG1
System log file 2	.S\$\$\$SLG2

where:

S\$ designates system files.

XX is the terminal identification number.

You can initialize system log files during the DX10 loading and start-up procedure by entering the Initialize System Log (ISL) command. The ISL command initializes the two log files that store device errors, input/output (I/O) errors, task errors, and messages generated by user programs. The command also permits selection of tasks to process the log file entries.

Do not name user files with file names starting with S\$. This nomenclature is reserved for system files cataloged directly in the volume directory .VCATALOG. Following this convention aids you in avoiding any conflict regarding file pathnames.

Under DX10, an operational system disk must include a program file, overlay file, roll file, loader file, and a system image file. These files are built from a software kit or are provided in an operational DX10 system disk provided with the system. You can maintain primary and secondary versions of the program file, image file, and overlay file by using the MVI command to modify disk volume information (refer to paragraph 2.13). If you maintain more than one system program file, image file, loader file, and overlay file under different names on a system disk, you can switch to a specific version suited to your applications.

- System program file — The system program file (S\$PROGA) contains system programs and optionally may contain user programs.
- System image name — The name of the system image in the S\$IMAGES file that contains the memory resident parts of DX10.
- System overlay file — The system overlay file (S\$OVLYA) is a relative record file that contains disk-resident overlays of DX10.
- Roll file — The roll file (S\$ROLLA) is used by DX10 to roll out disk-resident tasks. Roll files are created with the CSF command.
- System crash file — The system crash file (S\$CRASH) is used by DX10 to save the total memory image following a system crash. The system crash file must be as large as the whole memory. The default size is 64K words. The CSF command creates the system crash file; it uses the sector size specified in the command. The system crash file may be deleted; however, then it is not possible to save system crash information.
- System loader file — The system loader file (S\$LOADER) is an image file containing code that loads the DX10 operating system from the system image file. The file name must be eight characters in length.
- Diagnostic file — The diagnostic file (S\$DIAG) is created by the INV command. It is a read-only file containing pseudorandom data patterns used for system verification by diagnostic programs. This file occupies exactly one disk cylinder and is located as close as possible to the disk spindle. (If the closest cylinder has a bad track, the next closest is used, and so on. S\$DIAG always occupies an error-free cylinder.)
- SCI procedure directory — The SCI procedure directory (S\$PROC) contains SCI procedures and has a maximum of 457 entries.
- Output spool files — The directory S\$PRINT contains the output spool files for the pseudo-devices LP\$1 through LP\$9, which hold output for printer devices LP01 through LP09. The CSF command creates the directory.

- System log files 1 and 2 — The system log files (.S\$SLG1 and .S\$SLG2) are queues of system log messages that provide information about device errors, abnormal task termination, user messages, system log processor messages, memory errors, errors in memory cache, and statistics. Refer to Volume VI for more information on the system log.
- Writable control storage (WCS) file — The WCS file contains predefined XOP instructions. Refer to the *990/99000 Assembly Language Reference Manual* for a detailed description of the WCS file. The WCS file is for use on 990/12 computers only.

SCI generates and maintains the following files as required by DX10. SCI-related files are automatically created, maintained, and deleted as required. You should not alter or delete these files.

- Background terminal communications area file — The background terminal communications area file (\$\$BGTC) is used by SCI to control background program processing.
- Foreground terminal communications area file — The foreground terminal communications area file (\$\$FGTC) is used by SCI to control foreground program processing.
- Background terminal local file — The background terminal local file (\$\$BTLFXX) is used by SCI as an output file for background processes at the terminal level.
- Foreground terminal local file — The foreground terminal local file (\$\$FTLFXX) is used by SCI as an output file for foreground processes at the terminal level.
- Terminal library — The terminal library (\$\$TCALIB) contains data or user IDs defined to the system.
- VCATALOG — VCATALOG is the master directory for a volume.

## **2.12 MODIFYING DISK VOLUME INFORMATION**

Unless you already have a running version of DX10 and need to use the Modify Volume Information (MVI) command to recognize the newly built system as the primary system image, you need not use the MVI command before going on to Section 3, system generation.

You can change disk volume information by using the MVI command. Under MVI, you can list and/or change specific files and all files and volume information within those files. To terminate MVI, enter Q. For a detailed description of the MVI command, see Volume II.

## **2.13 MODIFYING INITIAL STATE SPECIFICATIONS**

For security reasons, you will probably want to control the log-on specifications of your system. Whether you modify these specifications before or after system generation is a matter of personal choice.

The DX10 user interface requires that certain characteristics relative to each terminal in the system be defined for all terminals. SCI allows specific terminals to operate with predetermined characteristics as follows:

- Terminal status (device ON/OFF)
- Terminal operational mode (TTY/VDT)
- Terminal operational mode default (TTY/VDT)
- Terminal log-on specification (YES/NO)
- Terminal privilege level (0 through 7)

You can modify these characteristics by using the commands described in the following paragraphs.

### **2.13.1 Initial States**

The initial state of DX10 terminals after IPL is the following:

- All terminals are on.
- No terminal requires log-on.
- TTY terminals are in TTY mode.
- VDT terminals are in VDT mode.
- All terminals are assigned the highest privilege level.

### **2.13.2 Modification Strategy**

Depending on the requirements of each DX10 installation, the system programmer may decide to assign users and specific terminals on a system a varying level of access privilege and unique user ID assignments. You can also specify the mode of operation (TTY or VDT) for each terminal. Through the Modify Terminal Status (MTS) command, you can initialize specific assignments for terminal use and access. You can redefine terminal specifications each time you do an IPL. We recommend that you put all VDTs in VDT mode and all 733/743 type terminals in TTY mode as the default mode.

DX10 also provides you the capability of identifying a particular system user. Through the following commands, the system programmer can assign user ID, passcode, and privilege code specification:

AUI — Assign User ID  
MUI — Modify User ID  
DUI — Delete User ID

See Volume II for descriptions and examples of these commands.



### **2.13.3 Modifying the IS Command**

The IS command is an SCI command that automatically calls several commands and performs several functions related to initializing DX10.

IS is a general purpose initialization command designed to be modified by each DX10 installation. We advise the person coordinating system use to employ the IS command in generating a specific configuration with DX10.

The command stream initialized by the IS command includes the following SCI commands:

- Initialize Date and Time (IDT)
- Initialize System Log (ISL)
- Modify Terminal Status (MTS)
- Modify LUNO Protection (MLP)

For descriptions and examples of these commands, see Volume II.

The following procedure describes how to facilitate modification of the IS command for custom terminal configurations.

1. Issue the Copy/Concatenate (CC) command to create a copy of the file `.$$PROC.IS`. Enter `.$$PROC.IS` in response to the INPUT ACCESS NAME prompt and `.$$PROC.ISCOPY` (the pathname of the copy) as the OUTPUT ACCESS NAME.
2. Text edit the procedure by issuing the Initiate Text Editor (XE) command with the pathname of `.$$PROC.ISCOPY`, given as the FILE ACCESS NAME. (Refer to the *DX10 Operating System Text Editor Manual (Volume IV)*.)
3. Using the Text Editor, add an MTS command to the procedure for each terminal, thereby defining terminal characteristics at the entry of the IS command. You can also add appropriate IV commands and Assign Global LUNO (AGL) commands.
4. Terminate the text edit session, specifying `.$$PROC.ISCOPY` as the OUTPUT ACCESS NAME.
5. Test the modified IS command by issuing the ISCOPY command. If the command works satisfactorily, issue a CC command, specifying `.$$PROC.ISCOPY` as the INPUT ACCESS NAME and `.$$PROC.IS` as the OUTPUT ACCESS NAME. With this action, the modified command procedure is installed as the standard IS procedure.

# System Generation

---

## 3.1 SYSTEM GENERATION

The DX10 operating system's purpose is to manage the complex and varied operations of a computer system. To accomplish this, DX10 must obtain information about the computer's physical and logical makeup (its parameters). Furthermore, DX10 must be directed to process these parameters into a form that is more suited to efficient management of the computer system. These processes of obtaining and processing the computer's parameters constitute system generation.

Generating a system involves the following stages:

- Preexecution considerations
- Execution of the Execute System Generation Utility (XGEN) command, during which you provide DX10 the necessary parameters of the computer system
- Execution of the Assemble and Link Generated System (ALGS) command
- Execution of the Patch Generated System (PGS) command
- Execution of the Test Generated System (TGS) command
- Execution of the Install Generated System (IGS) command

### NOTES

Hexadecimal numbers are preceded by a left angle bracket (>) throughout this manual. Numbers without the left angle bracket are assumed to be decimal. Follow this convention when entering numbers to DX10 through the keyboard.

Also, the names of keys are generic key names throughout this manual. In some cases, the names on the keycaps of the terminals match the generic key names, but in many cases they do not. Appendix A contains a table of key equivalents to identify the specific keys on the terminal you are using. Drawings that show the layout of the keyboard of each type of terminal are also included.

## 3.2 PREEXECUTION CONSIDERATIONS

Before issuing the XGEN command, consider two factors:

- Devices to be added in the future
- Ready knowledge of the computer system parameters that you must provide during the execution of the System Generation utility

### NOTE

Make sure you read the object installation documents for any communications product that you plan to install on the system.

If you anticipate adding a new device in the near future, you may want to include the device during system generation to avoid regenerating the entire system just to include that one device. Use the Modify Device State (MDS) command to place the device offline until you physically install it. If you should use this technique, the system software may require that the device's controller board be installed in the chassis. In addition, some devices cannot share interrupts. Refer to paragraph 3.2.3 on interrupts for a discussion of this requirement.

Before you begin system generation, fill out the configuration chart for your system so you can respond readily to the prompts in the device definition portion of the System Generation utility. The following paragraph describes basic system configurations.

### 3.2.1 Base System Configurations

TI supports a base system configuration with multiple disk drives and at least one interactive device, such as a 931 video display terminal (VDT). The base system configuration, stored in .S\$IMAGES, requires that the supported devices be physically configured so that it conforms to the minimum configuration shown in the following tables and figures.

Table 3-1 shows the base system configuration supported by the DX10 system shipped on hard disk or on flexible diskette build media. Table 3-2 shows the base system configuration supported by the DX10 system shipped on magnetic tape build media. See Section 2 for instructions on building a system from the media shipped to you from TI.

Figures 3-1, 3-2, and 3-3 show the suggested base hardware configuration for each of the indicated chassis. TI suggests that your computer hardware be configured to match that in the figures, with your other devices added to it. This enables you to build a DX10 system in the future with a minimum of configuration changes. Figure 3-1 shows the configuration for the 990A13 13-slot chassis. The Business System 600 and 800 use this chassis. The footnotes indicate the various combinations that can appear in a system configured by TI. The system shipped from the TI factory is capable of performing the system build, whichever media you have. You need to examine the configuration chart for your system to determine what devices and parameters to use in executing system generation. Figure 3-2 shows the configuration for the 990 13-slot chassis. Figure 3-3 shows the configuration for the 990 17-slot chassis. (Note that in these figures the right angle bracket (> ) precedes a hexadecimal value.)

**NOTE**

Systems are shipped with the configuration chart attached to the computer chassis. Update the chart whenever you add or remove devices from the system.

**Table 3-1. Base System Configuration for DX10 on Disks**

Device	TILINE or CRU Address	Interrupt	Devices Supported
Disk	> F800	13	3 Drives
	> F820	9	1 Drive
VDT	> 1700	8	931 VDT/940 EVT
	> 0100	10	911 VDT
	> F980	11	931 VDT/940 EVT on channel 0

**Table 3-2. Base System Configuration for DX10 on Magnetic Tape**

Device	TILINE or CRU Address	Interrupt	Devices Supported
Disk	> F800	13	4 Drives
Tape	> F880	9	1 Drive
VDT	> 1700	8	931 VDT/940 EVT
	> 0100	10	911 VDT
	> F980	11	931 VDT/940 EVT on channel 0

LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1	CPU				1	CPU			
2		>2E0		6	2		>2C0		15
3		>2A0		10	3		>280		8
4		>260		11	4		>240		12
5		>220		7	5		>200		3
6		>1E0		11	6		>1C0		11
7		>1A0		9	7		>180		13
8		>160		8	8		>140		9
9		>120		8	9	911 VDT	>100		10
10		>0E0		12	10		>0C0		11
11		>0A0		3	11		>080		7
12		>060		14	12		>040		4
13		>020		15	13		>000		6

SEE  
NOTE 1

SEE  
NOTES  
2 AND  
3

SEE  
NOTE 4

SEE  
NOTE 5

**NOTES:**

- (1) SOME STANDARD OPTIONS FOR SLOT 7 ARE AS FOLLOWS:
  - A FULL BOARD TILINE DISK CONTROLLER AT ADDRESS >F800, INTERRUPT 13 ON P2 SIDE.
  - A TILINE PERIPHERAL BUS INTERFACE (TPBI) BOARD FOR TAPE IN P1 AT TILINE ADDRESS >F880, INTERRUPT 9 AND FOR DISK IN P2 AT TILINE ADDRESS >F800, INTERRUPT 13 (FOR EXAMPLE, WD800).
  - A TPBI BOARD FOR DISK IN P1 AT TILINE ADDRESS >F820, INTERRUPT 9 AND FOR DISK IN P2 AT TILINE ADDRESS >F800, INTERRUPT 13 (FOR EXAMPLE, WD500).
- (2) SOME STANDARD OPTIONS FOR SLOT 8 ARE AS FOLLOWS:
  - A FULL BOARD FOR DISK AT TILINE ADDRESS >F820, INTERRUPT 9.
  - A FULL BOARD FOR TAPE AT TILINE ADDRESS >F880, INTERRUPT 9.
- (3) THE P2 SIDE IS USER PROGRAMMABLE TO BE INTERRUPT 9 OR 14.
- (4) AN OPTION FOR THE P1 SIDE OF SLOT 9 IS A 931 VDT OR 940 EVT AT CRU BASE ADDRESS >1700, INTERRUPT 8. THIS IS A C1402 CONTROLLER ON A SYSTEM WHERE THE CPU IS NOT A /10A.
- (5) A STANDARD OPTION FOR SLOT 10 IS A 931 VDT OR 940 EVT ON A C1403 OR C1404 AT TILINE ADDRESS >F980, INTERRUPT 11, CHANNEL 0.

2283055

**Figure 3-1. Base System Configuration for 990A13 13-Slot Chassis, Business System 600/800**

LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1	SMI				1	SMI			
2	AU				2	AU			
3	MEMORY		>FB00		3	MEMORY		>FB00	
4	MEMORY				4	MEMORY			
5		>220			5		>200		
6		>1E0			6		>1C0		
7	DISK	>1A0	>F800	13	7	DISK	>180	>F800	13
SEE NOTE 1	8	>160		9	8	DISK	>140	>F820	9
SEE NOTE 2	9	>120		8	9	911 VDT	>100		10
SEE NOTE 3	10	>0E0		12	10		>0C0		11
	11	>0A0		3	11		>080		7
	12	>060		14	12		>040		4
	13	>020		15	13		>000		6

## NOTES:

- (1) AN ALTERNATE CONFIGURATION THAT YOU CAN HAVE IS A FULL SLOT TAPE CONTROLLER AT TILINE ADDRESS >F880, INTERRUPT 9 ON P2 SIDE.
- (2) AN OPTION FOR THE P1 SIDE OF SLOT 9 IS A 931 VDT OR 940 EVT AT CRU BASE ADDRESS >1700, INTERRUPT 8. THIS IS A CI402 CONTROLLER ON A SYSTEM WHERE THE CPU IS NOT A /10A.
- (3) THE BASE CONFIGURATION CAN SUPPORT A 931 VDT OR 940 EVT ON CHANNEL 0 OF A CI403 OR CI404 MULTIPLEXER BOARD. THIS BOARD OCCUPIES A FULL SLOT WITH TILINE ADDRESS >F980, INTERRUPT 11.

2283056

Figure 3-2. Base System Configuration for 990 13-Slot Chassis

TOP OF CHASSIS (P1)					BOTTOM OF CHASSIS (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER-RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER-RUPT
1	SMI				1	SMI			
2	AU				2	AU			
3	MEMORY		>FB00		3	MEMORY		>FB00	
4	MEMORY				4	MEMORY			
5				11	5				11
6		>2E0		10	6		>2C0		10
7		>2A0		15	7		>280		15
8		>260		12	8		>240		12
9		>220		8	9		>200		8
10		>1E0		3	10		>1C0		3
11	DISK	>1A0	>F800	13	11	DISK	>180	>F800	13
12		>160		9	12	DISK	>140	>F820	9
13		>120		10	13	911 VDT	>100		10
14		>0E0		11	14		>0C0		11
15		>0A0		7	15		>080		7
16				14	16		>040		4
17		>020		6	17		>000		6

SEE NOTE 1

SEE NOTE 2

SEE NOTE 3

NOTES:

- (1) AN ALTERNATE CONFIGURATION THAT YOU CAN HAVE IS A FULL SLOT TAPE CONTROLLER AT TILINE ADDRESS >F880.
- (2) AN OPTION FOR THE P1 SIDE OF SLOT 9 IS A 931 VDT OR 940 EVT AT CRU BASE ADDRESS >1700, INTERRUPT 8. THIS IS A C1402 CONTROLLER ON A SYSTEM WHERE THE CPU IS NOT A /10A.
- (3) THE BASE CONFIGURATION CAN SUPPORT A 931 VDT OR 940 EVT ON CHANNEL 0 OF A C1403 OR C1404 MULTIPLEXER BOARD. THIS BOARD OCCUPIES A FULL SLOT WITH TILINE ADDRESS >F980, INTERRUPT 11.

2283057

Figure 3-3. Base System Configuration for 990 17-Slot Chassis

### 3.2.2 Determining CRU Addresses

The communications register unit (CRU) is a serial data bus used by slower devices, such as keyboard devices and line printers. When defining one of these devices, you must specify its CRU base address. A device's CRU base address is determined by the position of its controller board in the chassis.

Note that a controller board can be either full-size, filling an entire slot, or half-size, filling either the left or right half-slot. A half-size board inserted in a half-slot uses the CRU base address of that half-slot. For example, a half-size board in the left-hand half-slot of slot 14 in Figure 3-5 uses CRU base address  $> 0E0$ .

Most full-size CRU device controller boards actually implement two separate control circuits on the two halves of a single board. The left and right halves of the board use the left and right CRU base addresses, respectively. For example, a 911 VDT controller board can be inserted in slot 9 of Figure 3-4. The control circuit on the right side of the board controls a single 911 VDT and uses CRU base address  $> 100$ . The control circuit on the left side of the board controls another 911 VDT and uses CRU base address  $> 120$ .

Some full-sized controller boards use a single circuit to control several units of a particular kind of device. Such boards use only the right-hand (P2) CRU base address of that slot. For example, an FD800 serial flexible disk drive controller board in slot 11 controls one or more drives, but only uses CRU base address  $> 080$ .

**3.2.2.1 Arrangement of CRU Base Addresses in a Chassis.** The CRU base addresses in a chassis begin at  $> 000$ . Each following CRU base address is greater than the preceding CRU base address by  $> 020$ . Figure 3-4 shows how the CRU base addresses are arranged within the 13-slot chassis starting at the highest numbered slot, which is slot 13. CRU base address  $> 000$  is assigned to the right-hand (P2) half-slot of slot 13. The next CRU base address,  $> 020$ , is assigned to the left-hand (P1) half-slot of slot 13. The next CRU base address,  $> 040$ , is assigned to the P2 half-slot of the second highest numbered slot, which is slot 12. This pattern continues until the maximum CRU base address,  $> 2E0$ , is reached. CRU devices cannot be installed in slots without a CRU base address. These slots are reserved for other boards, such as central processing unit (CPU) and memory boards.

**3.2.2.2 CRU Base Addresses for a 17-Slot Chassis.** The arrangement of CRU base addresses in a 17-slot chassis is similar to that of the 13-slot chassis. Figure 3-5 shows the arrangement of CRU base addresses in a 17-slot chassis.



LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1					1				
2		>2E0			2		>2C0		
3		>2A0			3		>280		
4		>260			4		>240		
5		>220			5		>200		
6		>1E0			6		>1C0		
7		>1A0			7		>180		
8		>160			8		>140		
9		>120			9		>100		
10		>0E0			10		>0C0		
11		>0A0			11		>080		
12		>060			12		>040		
13		>020			13		>000		

2283058

Figure 3-4. CRU Base Addresses for 990 and 990A13 13-Slot Chassis

TOP OF CHASSIS (P1)					BOTTOM OF CHASSIS (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1					1				
2					2				
3					3				
4					4				
5					5				
6		>2E0			6		>2C0		
7		>2A0			7		>280		
8		>260			8		>240		
9		>220			9		>200		
10		>1E0			10		>1C0		
11		>1A0			11		>180		
12		>160			12		>140		
13		>120			13		>100		
14		>0E0			14		>0C0		
15		>0A0			15		>080		
16		>060			16		>040		
17		>020			17		>000		

2283059

Figure 3-5. CRU Base Addresses, 17-Slot Chassis

**3.2.2.3 CRU Addresses in Expansion Chassis.** The CRU base addresses in an expansion chassis are arranged in the same manner as in the main chassis except that the CRU base addresses are offset by a multiple of > 400, depending on the sequential position of the expansion chassis. Table 3-3 lists the values to be added to the CRU base addresses for each expansion chassis.

**Table 3-3. CRU Base Address Offsets for Expansion Chassis**

Expansion Chassis	Add to Base Address
1	> 0400
2	> 0800
3	> 0C00
4	> 1000
5	> 1400
6	> 1800
7	> 1C00

Figure 3-6 shows the CRU base addresses for a 13-slot chassis used as the first expansion chassis. Notice that slot 1 has not been assigned an address. This position is always occupied by a buffer board (the CRU expander board).

### 3.2.3 Interrupt Levels

During system generation, you must specify the hardware interrupt that each device generates. The location of the controller or interface board in the chassis determines the interrupt level. Figure 3-7 shows the standard interrupt assignments for a 990A13 13-slot chassis. Figure 3-8 shows the standard interrupt assignments for a 990 13-slot chassis. The interrupt levels for a 13-slot chassis are determined by the positions of jumper wires on the system interface board, which is attached to the rear of the 13-slot chassis. The jumper wires are inserted into a molded plastic connector plugged into the backboard above slot 1. Do not attempt to change the position of these wires or the connector. The 990A13 chassis uses an etched or programmable card that plugs into the system interface board above slot 1.

**3.2.3.1 Interrupts for Devices in Expansion Chassis.** If a device is configured in an expansion chassis, the device uses the same interrupt used by the expansion card in the main chassis. When a device in an expansion chassis is defined during system generation, you are prompted for an interrupt level. Respond with the level assigned to the expansion card for the expansion chassis. You are also prompted for the device's expansion position. Respond to this prompt with the interrupt level assigned to the device's position within the expansion chassis. Interrupt levels within a 13-slot expansion chassis are the same as those within a 13-slot main chassis. Similarly, the interrupt levels within a 17-slot expansion chassis are the same as those within a 17-slot main chassis.

LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1					1				
2		>6E0			2		>6C0		
3		>6A0			3		>680		
4		>660			4		>640		
5		>620			5		>600		
6		>5E0			6		>5C0		
7		>5A0			7		>580		
8		>560			8		>540		
9		>520			9		>500		
10		>4E0			10		>4C0		
11		>4A0			11		>480		
12		>460			12		>440		
13		>420			13		>400		

2283060

Figure 3-6. CRU Base Addresses, 13-Slot Expansion Chassis #1

LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1					1				
2				6	2				15
3				10	3				8
4				11	4				12
5				7	5				3
6				11	6				11
7				9	7				13
8				8	8				9*
9				8	9				10
10				12	10				11
11				3	11				7
12				14	12				4
13				15	13				6

NOTE:

(\* ) THE P2 SIDE IS USER PROGRAMMABLE TO BE INTERRUPT 9 OR 14 .

2283061

**Figure 3-7. 990A13 Interrupt Assignments, 13-Slot Chassis**

LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1					1				
2					2				
3					3				
4					4				
5					5				
6					6				
7				13	7				13
8				9	8				9
9				8	9				10
10				12	10				11
11				3	11				7
12				14	12				4
13				15	13				6

2283062

**Figure 3-8. Interrupt Assignments, 13-Slot Chassis**

**3.2.3.2 Devices Sharing Interrupts.** When configuring your system, observe the following guidelines:

- TILINE devices never share interrupts. However, the CI403 and CI404 communications controllers can share interrupts between each other.
- CRU devices can share interrupts. However, some combinations may not be designed to work, and others may work only with low-speed devices. You should not share interrupts for high-speed devices.
- A communication device performs best if it does not share an interrupt. This is strongly recommended.
- Interrupts cannot be shared between any device and a CRU expander card. It is best not to share the interrupts for CARD1 and CARD2, the two interrupts for total CRU expansion.

Frequently, several CRU devices are assigned the same interrupt during system generation or the same position (interrupt) in an expansion chassis. The device controllers may be absent from the chassis and generated with shared empty slots. However, some combinations with shared empty slots may not be designed to work. Refer to Section 4 to determine which interrupts are giving you problems. Some combinations may only work with low-speed devices. You should not share an empty slot for high-speed devices.

On 990 Models 20 and 30, the 17-slot chassis used as the main chassis in the 990/12 CPU has soldered connections that determine the interrupt levels. Figure 3-9 shows the standard interrupt level assignments for a 17-slot chassis.

#### **3.2.4 TILINE Addresses**

TILINE is a high-speed, bidirectional, 16-bit data bus. It serves as a path for communication between all high-speed elements in the system. The CPU, main memory, and high-speed peripheral devices are directly connected to the TILINE. TILINE peripheral devices include disk drives, magnetic tape transports, some communications ports, double-sided, double-density (DSDD) diskette drives, and asynchronous multiplexers.

You must supply the TILINE address when defining TILINE devices. This address, which the hardware converts to a memory address, is determined by the switch settings on the controller for the TILINE device. You should not alter these switches since the bits set by them differ from one type of controller to another. Figure 3-1, Figure 3-2, and Figure 3-3 show typical TILINE address assignments in 13- and 17-slot chassis. The configuration label attached to your computer's chassis records the TILINE addresses as they were set up when your computer was manufactured.

TOP OF CHASSIS (P1)					BOTTOM OF CHASSIS (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1					1				
2					2				
3					3				
4					4				
5				11	5				11
6				10	6				10
7				15	7				15
8				12	8				12
9				8	9				8
10				3	10				3
11				13	11				13
12				9	12				9
13				10	13				10
14				11	14				11
15				7	15				7
16				4	16				14
17				6	17				6

2283063

Figure 3-9. Interrupt Assignments, 17-Slot Chassis



Table 3-4 lists the conventional TILINE assignments. We recommend that your system follow this pattern.

If you put a CI403/CI404 controller in an expansion chassis, a TILINE coupler must be present, and all CI403/CI404 controllers must have their interrupts wired through the TILINE coupler. You generate them to have the interrupt level where the TILINE coupler is plugged into the main chassis.

**Table 3-4. Recommended TILINE Address Assignments**

TILINE Address	Device Type	Controller Number
	Memory Controller	
> FB00		1
> FB04		2
> FB08		3
	Cache Memory Controller	
> FB10		1
> FB14		2
> FB18		3
	Disk Controller	
> F800		1 (System Disk)
> F810		2
> F820		3
> F830		4
> F840		5
> F850		6
> F860		7
> F870		8
	Tape Controller	
> F880		1
> F890		2
	Communication Controller	
> F900		1
> F910		2
> F920		3
> F930		4
> F940		5
> F950		6
	CI403/404	
> F980		1
> F990		2
> F9A0 *		3

**Note:**

\* You can add additional controllers after this one. Add > 10 to this address to obtain the address of each subsequent controller.

### 3.3 GEN990 SYSTEM GENERATION UTILITY

As discussed in preceding paragraphs, DX10 must obtain the operating parameters of a computer system in order to control that system. When GEN990 executes, it displays prompts which ask you for these parameters. You aid GEN990 in the definition of the computer system by supplying the values of these parameters. Then, at your direction, GEN990 can build the data files that are used by the remainder of system generation.

GEN990 requires extensive interaction with you. The parameter values that you supply must reflect your computer's configuration accurately. Consequently, GEN990 is more vulnerable to error than the remaining parts of system generation, which require relatively little human input. The following paragraphs provide a description of GEN990. We recommend that you read the following description of GEN990 and gather the required parameter values before attempting to execute GEN990 (also called system generation).

The following paragraphs discuss these topics:

- Files created and used by GEN990
- GEN990 modes of operation (command and inquiry)
- GEN990 commands

#### 3.3.1 Files Created and Used by GEN990

The GEN990 process uses your responses to the parameter prompts and the contents of various data files to produce a new system. These files are stored in the directories called `.$SYSGEN` and `.PATCH`, which can be on any disk currently installed in the system. Identify the disk where they are located by responding to the first GEN990 prompt `DATA DISK/VOLUME`.

The GEN990 utility produces a directory called `.$SYSGEN.<output>`, where `<output>` is the name you call the system, entered in response to the third system generation prompt, `OUTPUT CONFIGURATION`. In addition to the directory, GEN990 builds the following files, which are cataloged under the new directory:

- `BATCHSTM` — A batch stream of System Command Interpreter (SCI) commands used by the `ALGS` command (see paragraph 3.5).
- `CONFIG` — The configuration file for the new system. The configuration file contains the parameter values given in response to GEN990 prompts. You can read and print this sequential file, but under no circumstances should you edit it.
- `D$SOURCE` — A file containing the assembly language source code for the data base module (`D$DATA`) of the new system.
- `LINKSTRM` — A file containing all of the Link Editor commands necessary to link all parts of the new system together.

### 3.3.2 GEN990 Modes of Operation

GEN990, the interactive System Generation utility invoked by the XGEN command, can operate in either the command or inquire mode. In the inquire mode, GEN990 prompts you for the parameter values required to define the system configuration. In the command mode, you can list, change, or delete portions of the configuration created in the inquire mode.

#### NOTE

In either the command or inquire mode, GEN990 understands commands and data that are entered in uppercase only. Commands and data entered in lowercase are misinterpreted.

**3.3.2.1 Inquire Mode.** GEN990 begins execution in the inquire mode. In the inquire mode, GEN990 prompts you for parameters of the system being generated. You supply the parameter values of the system being generated. You can enter the inquire mode from the command mode by responding to the COMMAND prompt with INQUIRE.

**3.3.2.2 Command Mode.** You can enter the command mode at any time during the execution of GEN990 by pressing the Command key on your terminal. When GEN990 is in the command mode, it prompts you to enter a command by displaying the following:

COMMAND?

Respond by entering one of the GEN990 commands listed in Table 3-5.

### 3.3.3 GEN990 Commands

Table 3-5 lists the legal commands that you can issue to GEN990 from the command mode. Table 3-6 lists the legal commands that you can issue from the inquire mode. The following paragraphs describe the commands and their uses.

**Table 3-5. Legal Commands in GEN990 Command Mode**

<b>Command</b>	<b>Result</b>
BUILD	This command enables you to initiate the build sequence to build the configuration, source, and Link Editor control files, and terminate GEN990. You can also do this in inquire mode.
CHANGE	This command enables you to change the value of a specified parameter. You can enter CHANGE only from the command mode.
DELETE	This command enables you to delete a device or supervisor call (SVC) from the system generation.
INQUIRE or HELP	These commands return GEN990 to inquire mode from the command mode. GEN990 prompts you to define the next parameter with NEXT.
LIST	This command causes GEN990 to display the parameter values of the current configuration.
STOP	This command terminates GEN990. You can also issue STOP in the inquire mode.

**Table 3-6. Legal Commands in GEN990 Inquire Mode**

<b>Command</b>	<b>Result</b>
BUILD	This command enables you to initiate the Build sequence to build the configuration, source, and Link Editor control files, and terminate GEN990.
DVC or GENERATE	These commands cause GEN990 to prompt you for a DEVICE TYPE.
STOP	This command terminates GEN990. You can also issue STOP in the command mode.
SVC	This command causes GEN990 to prompt you for the parameters for an SVC.
WHAT or ?	This command causes GEN990 to display an explanation of the parameter prompted.

**3.3.3.1 BUILD Command.** Refer to the entry for the STOP command.

**3.3.3.2 CHANGE Command.** Use the CHANGE command to modify a previously defined value. You can legally issue the CHANGE command only from the command mode, which you enter by pressing the Command key. When you enter CHANGE, GEN990 displays the following prompt on the terminal screen:

PARAMETER TO BE CHANGED?

If you want to change a system parameter (rather than a device parameter), enter the parameter keyword (the name of the parameter) that you wish to change, such as TIME SLICING ENABLED.

Instead of writing the entire parameter keyword, you can enter an abbreviation of the parameter keyword. DX10 interprets abbreviations using the Near Equality algorithm. The three rules for abbreviating keywords as implemented by the Near Equality algorithm are the following:

- The abbreviation and the keyword must begin with the same letter.
- All letters in the abbreviation must appear in the keyword in the same order of occurrence.
- The first letter following a blank in the keyword must match the first unmatched letter in the abbreviation.

For example, both TSE and TIME are legal abbreviations for TIME SLICING ENABLED.

Be sure that an abbreviation is unique to a keyword. For example, if you entered INTER as an abbreviation for the parameter keyword INTERTASK?, DX10 would interpret INTER as the keyword INTERRUPT DECODER?. Similarly, although the abbreviation INTERT appears to stand for INTERTASK?, DX10 interprets INTERT to be INTERRUPT DECODER?.

For a more detailed discussion of the Near Equality algorithm, refer to Volume III.

To change a parameter of a particular device, respond to the prompt PARAMETER TO BE CHANGED? with DVC. In the CHANGE command, GEN990 responds to DVC with the following prompt:

DEVICE NAME?

Respond to this prompt with the device name of the previously defined device, for example, DS02 or ST04. GEN990 then displays the prompt DEVICE TYPE. Enter the type of device that you are changing, such as DS or VDT.

If you press the Return key without entering a device type, then GEN990 deletes the device supplied to the prompt DEVICE NAME.

The following example demonstrates the way to change the access type of LP02, showing the prompts from GEN990 and appropriate responses. After you enter the command mode by pressing the Command key on your terminal, GEN990 prompts you for a command:

```
COMMAND? CHANGE
PARAMETER TO BE CHANGED? DVC
DEVICE NAME? LP02
DEVICE TYPE? LP
CRU ADDRESS? (> 60) 0080
ACCESS TYPE? (FILE) RECORD
9902 INTERFACE? (NO)
PRINT MODE? (SERIAL)
EXTENDED? (NO)
TIME OUT? (60)
INTERRUPT? (6)
```

The following example demonstrates the way to change an existing SVC, USVC2, to another SVC, USVC9. Note that the object code for USVC9 is already in the file .TESTER.USVC9. The example includes the prompts from GEN990 and appropriate responses. After you enter the command mode by pressing the Command key on your terminal, GEN990 prompts you for a command:

```
COMMAND? CHANGE
PARAMETER TO BE CHANGED? SVC
SVC ENTRY NAME? USVC2
SVC? USVC9
FILE? .TESTER.USVC9
BYTES? (1) 20
```

**3.3.3.3 DELETE Command.** Use the DELETE command to remove a device or SVC from the current configuration. When you issue the DELETE command, GEN990 prompts you for an ENTITY TO BE DELETED. Respond with either DVC, if you want to delete a device, or SVC if you want to delete an SVC.

The following example demonstrates the deletion of a line printer, LP03, showing the prompts from GEN990 and the appropriate responses.

After you enter the command mode by pressing the Command key, GEN990 prompts you for a command:

```
COMMAND? DELETE
ENTITY TO BE DELETED? DVC
DEVICE NAME? LP03
COMMAND?
```

When a device is deleted, GEN990 reorders the device names of similar devices to maintain an unbroken numerical sequence. The following chart demonstrates this:

Before deletion of LP03	After deletion of LP03
LP01	remains LP01
LP02	remains LP02
LP03	deleted
LP04	becomes LP03
LP05	becomes LP04

The following example demonstrates the deletion of the SVC USVC3, showing the prompts from GEN990 and the appropriate responses. After you enter the command mode by pressing the Command key, GEN990 prompts you for a command:

```
COMMAND? DELETE
ENTITY TO BE DELETED? SVC
SVC ENTRY NAME? USVC3
COMMAND?
```

**3.3.3.4 DVC Command.** Issue the DVC command from the inquire mode to define a device. When the DVC command is issued, GEN990 prompts you for a DEVICE TYPE. Respond with a device mnemonic, such as ST, LP, or DS. GEN990 then prompts you for appropriate parameter values. Numerous examples are provided in paragraph 3.4.5.

**3.3.3.5 INQUIRE and HELP Commands.** Issue either the INQUIRE command or the HELP command to return GEN990 to the inquire mode from the command mode. After receiving the INQUIRE command, GEN990 displays the following prompt:

```
NEXT?
```

**3.3.3.6 LIST Command.** The LIST command can only be issued from the command mode. The LIST command causes GEN990 to list the value of ALL parameters specified and their values. When the LIST command completes its display, GEN990 remains in the command mode and displays the COMMAND prompt.

#### NOTE

If you issue the LIST command at a VDT, the list of parameter values rolls up from the bottom of the screen until each row of the screen contains an entry. To advance the display, press the Return key. If you want a permanent hard-copy list of your configuration, you need to use the Print File (PF) command to print the CONFIG file.

**3.3.3.7 STOP and BUILD Commands.** Issue either the STOP command or the BUILD command to terminate GEN990. Before terminating, GEN990 displays the following prompt:

BUILD? (YES)

If you want GEN990 to build the system generation files based on the data you have supplied, accept the default value of YES by pressing the Return key. When you enter YES, GEN990 begins to process the configuration parameters that you supplied. This processing requires a certain amount of time, depending on the number and kind of devices included. Throughout this processing, GEN990 displays the following messages informing you of its status:

```
CONFIGURATION FILE SAVED
D$DATA SOURCE FILE IS NOW BEING BUILT
THE LINK EDIT COMMAND STREAM SOURCE FILE IS BEING BUILT
BATCH FILE FOR SYSGEN COMPLETION IS NOW BEING BUILT
SYSGEN COMPLETE
GEN990 TERMINATED
```

If you do not want to build the system files at this time, enter NO and press the Return key.

GEN990 then displays the following prompt:

DO YOU WANT TO SAVE THE CONFIGURATION FILE? (NO)

If you want to discard the parameters that you have entered and abort the system generation session, accept the default of NO by pressing the Return key. If you want to save these parameters in the CONFIG file, then enter the response YES and press the Return key. The following message appears:

```
CONFIGURATION FILE SAVED
```

After you respond to the above prompts, GEN990 displays the following message:

```
SYSGEN COMPLETE
```

After a few seconds, GEN990 replaces the above message with the following message:

```
***** GEN990 TERMINATED ***** :
```

Press the Command key to return to SCI.

**3.3.3.8 SVC Command.** Issue the SVC command from the inquire mode to include user-written SVC processors with the generated operating system. Refer to paragraph 3.4.6 for a description of an SVC and a list of parameters.

**3.3.3.9 GEN990 WHAT Command.** Issue the WHAT command (or a question mark (?)) to obtain an explanation of the most recently displayed prompt. If you issue the WHAT command in response to a parameter prompt by GEN990, GEN990 displays the explanation and reprompts the same parameter. For most parameters, two levels of explanation are available; that is, WHAT can be asked twice for the same parameter. The first issue of WHAT displays an abbreviated explanation and the second displays an expanded explanation.



### 3.4 EXECUTING GEN990

You initiate GEN990 after you perform an initial program load (IPL) and initialize the system. (Volume II describes a simplified IPL procedure. The *ROM Loader User's Guide* describes the IPL procedure in detail.) Initiate GEN990 by issuing the XGEN command at your terminal. The following prompts immediately appear at your terminal:

```
EXECUTE AUTO-SYSGEN
      DATA DISK/VOLUME:
      INPUT CONFIGURATION:
      OUTPUT CONFIGURATION:
```

Respond to the prompt DATA DISK/VOLUME with the device name or volume name of disk drive that stores the system generation directories \$\$\$SYSGEN and PATCH. The \$\$\$SYSGEN and PATCH directories contain the standard DX10 object modules, the data files used by GEN990 for prompting and building system generation files, and user-defined and released configuration files. To conserve space on your main system disk of a multiple disk system, you can copy the \$\$\$SYSGEN and PATCH directories to a secondary disk using the Copy Directory (CD) SCI command.

If this is the first generation of a particular configuration, respond to the prompt INPUT CONFIGURATION? by pressing the Return key. If you are regenerating an existing configuration, enter that configuration's unique name and press the Return key on your terminal.

If this is the first generation of a particular configuration, respond to the prompt OUTPUT CONFIGURATION? by entering a unique name. This name, identifying the configuration, must begin with an alphabetic character and contain no more than five characters. If you are regenerating an existing configuration, you can enter that configuration's name and the new configuration replaces the old one. You can preserve the existing configuration and create a new one by entering a new name.

When you respond to the above prompts, GEN990 displays the following heading:

```
GEN990-AUTO SYSTEM GENERATION DX10 X.X.X
```

GEN990 continues by prompting you for the parameter values in TTY mode (one line at a time). GEN990 often displays a parameter's default value enclosed in parentheses (for instance, 60 is the default value for LINE?). Table 3-7 lists and describes the XGEN prompts. The paragraphs following the table give more detailed explanations.

#### NOTE

Designate hexadecimal numbers with a leading zero (0) or a leading right-angle bracket (>), as in 0152 and > 152. DX10 considers numbers that are not so designated to be decimal numbers, such as 152.

Table 3-7. GEN990 Parameters

Prompt (and Default Value)	Value Description
LINE? (60)	Line frequency (50 or 60 Hz)
TIME SLICING ENABLED? (YES)	Do you want to time slice?
TIME SLICE VALUE? (1)	Number of system time units in one slice (one system time unit equals 50 milliseconds).
TASK SENTRY ENABLED? (YES)	Do you want to reduce automatically the priority of tasks that execute longer than a value defined during system generation without giving up the CPU?
TASK SENTRY VALUE? (60)	Number of system time units a task is allowed to execute without giving up the CPU before its priority is reduced (one system time unit equals 50 milliseconds).
TABLE?	Size of the system table area in words.
COMMON?	File which contains the system common image (optional).
INTERRUPT DECODER? (NONE)	File which contains the object code of a user-supplied interrupt decoder (optional).
FILE MANAGEMENT TASKS? (2)	Number of file managers to include in the system.
CLOCK? (5)	Interrupt level of the clock.
RESTART ID? (NONE)	Installed ID of the initial task to be executed (optional).
OVERLAYS? (2)	The number of system overlay areas (400 words per overlay area).
ONLINE DIAGNOSTIC SUPPORT? NO	Include optional disk performance diagnostics?
SYSLOG? (6)	The number of system log messages which may be queued.
BUFFER MANAGEMENT (BYTES)? (1024)	Size in bytes of the largest physical record.
I/O BUFFERS? (0)	Size in bytes of additional system area to be used as device I/O buffers.

Table 3-7. GEN990 Parameters (Continued)

Prompt (and Default Value)	Value Description
INTERTASK? (100)	Size in words of intertask communication area appended to system table.
ITC MESSAGES? (3)	Maximum number of intertask communication messages that a task can have in the intertask communication area.
KIF? (YES)	Do you want to include logic to support key indexed files (KIFs)?
COUNTRY CODE? (US)	In which country is this system being used?
POWER FAIL? (NO)	Do you want to include Model 911 VDT power recovery option?
AUTO MEDIA CHANGE RECOVERY ? (YES)	Do you want to protect disks from accidental volume changes?
SCI BACKGROUND? (2)	Maximum number of batch (background) SCIs which can be active at one time.
SCI FOREGROUND? (8)	Maximum number of interactive (foreground) SCIs allowed to be active at one time.
BREAKPOINT? (16)	Maximum number of breakpoints allowed under system.
PANEL DISPLAY (BAR CHART)? (YES)	Should the front panel data light-emitting diodes (LEDs) display disk and CPU activity (YES) or the current program counter value?
CARD 1?	Interrupt level of expansion chassis 1 – 4 (optional).
CARD 2?	Interrupt level of expansion chassis 5 – 7 (optional).
DEVICE TYPE?	Type of a device to be added to system (namely, LP, CRDR, ASR, KSR, DS, VDT, MT, DK, SD, COM, RTS).
CRU ADDRESS?	CRU address of device being added (hexadecimal, multiple of >20, range from >0000 to >1EE0).

Table 3-7. GEN990 Parameters (Continued)

Prompt (and Default Value)	Value Description
TILINE ADDRESS?	TILINE address of device being added (hexadecimal, multiple of > 20, range from > F800 to FC00).
ACCESS TYPE? or CASSETTE ACCESS TYPE	Either record- or file-oriented device.
TIME OUT? or CASSETTE TIME OUT?	Time interval allowed to elapse for I/O before error.
PRINT MODE? (SERIAL)	Either serial or parallel mode of printing.
EXTENDED? (YES)	Do you want the extended character set for this serial printer (applies only to TI Model 810 printer)?
INTERRUPT?	Interrupt level assigned to device (3 – 15).
EXPANSION CHASSIS?	Number of expansion chassis to which device is attached (1 – 7).
EXPANSION POSITION?	Interrupt level on expansion chassis (0 – 15).
CHARACTER QUEUE? (6)	Number of unrequested input characters buffered for device.
DRIVES? (1)	Number of drives on a multidrive device (such as a disk or magnetic tape).
DEFAULT RECORD SIZE? (768)	Physical record size, in bytes, given to disk files when the default is specified in the SVC block.
VDT TYPE?	Either 911 VDT, 913 VDT, 931 VDT, Business System terminal, or 940 EVT.
INTERFACE TYPE?	The type of interface the device is using. A peripheral is connected to the host through an interface board. Refer to Table 3-9.
CHANNEL NUMBER?	The channel number on the board to which the device is connected.
SPEED?	The speed at which the device is operating.
INTERRUPT CRU BIT?	The CRU bit (0 – 30) which indicates an interrupt.

Table 3-7. GEN990 Parameters (Continued)

Prompt (and Default Value)	Value Description
NAME?	The name of a special device, 1 – 4 characters.
KSB?	Starting label (1 – 6 characters) of the keyboard status block (KSB), if any, for special device.
DSR WORKSPACE?	Label of the device service routine (DSR) workspace for the special device.
INTERRUPT ENTRY?	Label of the interrupt entry point in the DSR.
PDT FILE?	Pathname of the file that contains the source code for the physical device table (PDT) for the device.
DSR FILE?	Pathname of the file which contains the DSR for the device.
OVERRIDE? (Y)	Do you want GEN990 to override PDT fields? (Y or N)
SVC?	Starting label of the SVC processor where the system is to transfer control.
FILE?	Pathname of the file which contains the object code for the SVC processor.
BYTES? (1)	Number of bytes in the SVC block.
NEXT?	Issue an inquire mode command.

#### 3.4.1 Defining the GEN990 Operating Prompts

GEN990 prompts you to define the operating parameters of your customized system. The operating parameters tell GEN990 where its input should be and where its output should go. To respond to the GEN990 prompts, read the following parameter descriptions.

#### 3.4.2 Defining the System Timing Parameters

The following parameters are used to define timing values which are used by the new operating system to perform multiprogramming in a real-time environment. All of the parameters have default values which can be used by users who are mainly interested in changing DX10 optional features or adding new devices.

**3.4.2.1 LINE Parameter.** The LINE parameter specifies the power line frequency to which the computer is connected. Line frequencies vary between different countries (for example, 60 Hz in the U.S.A. and 50 Hz in Europe). GEN990 will accept either 50 or 60, with 60 being the default value. For a Business System 300 or 300A, always specify 50.

**3.4.2.2 TIME SLICING ENABLED Parameter.** This parameter enables or disables time slicing. In a system with time slicing enabled, each task is allowed to execute for a interval of time defined during system generation. After a given task has executed for this time interval, DX10 checks to see if another task with the same priority is waiting to execute. If such a task is waiting, the first task is suspended, so allowing other tasks to execute. In a system with time slicing disabled, the highest priority task will execute until it completes or voluntarily suspends execution. The initial value is YES.

**3.4.2.3 TIME SLICE VALUE Parameter.** This parameter specifies the length of a time slice in system time units. (One system time unit equals 50 milliseconds.) In a system with time slicing enabled, each task is allowed to execute for a certain interval of time. After this time interval has expired, the task is suspended and other tasks are given a chance to execute. The length of this time interval is known as the time slice. The initial value is one system time unit.

**3.4.2.4 TASK SENTRY ENABLED Parameter.** This parameter enables or disables the Task Sentry. The Task Sentry, when enabled, keeps track of how long a task executes without voluntarily suspending execution (by calling an SVC that requires the task to be rescheduled after the SVC completes). When this time exceeds a value defined during system generation, the execution priority of the task is reassigned to the next lower priority level. The Task Sentry can be enabled concurrently with time slicing. However, if you do not choose to enable time slicing, you should enable the Task Sentry to prevent any single task from monopolizing the CPU. The initial value for this parameter is YES.

**3.4.2.5 TASK SENTRY VALUE Parameter.** This parameter specifies the length of time a task may execute without voluntarily giving up the CPU before its execution priority is reduced. The value is given in system time units. (One system time unit equals 50 milliseconds.) The default value is 60 system time units.

### **3.4.3 TABLE Parameter**

The TABLE parameter specifies the size of the system table area. The system table area is part of the operating system and is used to contain system log messages, intertask communication messages, buffered SVC blocks, and many system-built tables. There is no default value for this parameter. The response to the TABLE prompt must be an integer number of words of memory (for example, 4000). Table 3-8 is a guideline for estimating system table area. Using the values given in the table, estimate the size and complete the system generation. GEN990 will not accept a value smaller than 2048 words.

You can tune your system generation to have the maximum possible system table area by using the following procedure:

1. Perform a trial ALGS.
2. Find the length of the task (length of phase 0) from the system link map. Subtract that number from  $>F740$ .
3. Find the origin of the module name FILMG in phase 1 from the system link map (LINKMAP file). Subtract that number from  $>C000$ .
4. The system table area can be larger by the smaller of these two numbers. If both results are negative, the system root is too large and must be made smaller by the larger of the absolute values. If one result is negative, the system root must be made smaller by the absolute value of the one that is negative.
5. These calculations are in hexadecimal bytes. The TABLE parameter in system generation uses decimal words. You need to convert the hexadecimal number to decimal and divide by 2.
6. Repeat system generation and change the table area by the amount from step 5.
7. Repeat the ALGS and proceed with the rest of the steps to complete the system generation.

When the system has been used for a while with your application, the size can be fine tuned using the Show Memory Status (SMS) command. Enter the SMS command at a terminal with an active SCI. Subtract the number given as LARGEST AREA USED from the number given as SYSTEM TABLE AREA. The resulting figure is the amount of table area that has never been used since the last initial program load. If the amount is large (ten percent of the table size or more), the system may be regenerated with a correspondingly smaller TABLE parameter. Do not reduce the table area enough to remove all of the unused space. Allow a cushion of at least 500 words. Allow 1000 words if you choose to include KIFs.

Table 3-8. System Table Area Sizing Guideline

Features Included	Size of Table Area in Words
Base System (includes support for 1 disk drive, 1 VDT, 1 SCI foreground, 1 SCI background, 1 file management)	3200
Each additional foreground*	650
Each additional background*	800
Each additional disk drive	40
Each additional file management task (a system generation defined number; see paragraph 3.4.4.3)	190
Diskette	128
Each installed disk volume with an opened KIF	133
Each serial line printer	70
Each parallel printer	210

**Note:**

\* These figures are based on peak usage.

**3.4.4 Defining Optional Features**

The following prompts allow you to include or exclude many optional features in the new DX10 system. They all have default values that should be used if you are only interested in adding devices to the system.

**3.4.4.1 COMMON Parameter.** This parameter is the pathname of the file which contains user-supplied object for the system common area of memory. If no file is specified, no system common area will be created. System common is memory reserved within the system which is available to all tasks via the Get Common Data SVC (> 10), and thus may be used to exchange data between tasks.

The object file (usually produced by the assembler from an assembly source module) can also include code to define application dependent constants in system common. The source must be assembled before it can be used by GEN990.

**3.4.4.2 INTERRUPT DECODER Parameter.** This parameter is the pathname of the file which contains a user-supplied object for an interrupt decoder module. If a file name is entered, GEN990 includes your interrupt decoder instead of generating the standard interrupt decoders and tables. If you enter nothing, GEN990 creates an interrupt decoder which is suitable for most systems.



**3.4.4.3 FILE MANAGEMENT TASKS Parameter.** This parameter is the number of copies of the file management task which are to be included in the new system. File management tasks can process any file I/O request; therefore, more than one task can process requests at the same time. However, each copy of the file management task reduces the maximum possible system table area by about 370 bytes. A system will run with only one file management task. Two provides significantly better throughput than one even if only one controller is present. At the other extreme, one file management task per disk controller provides all necessary file management capabilities. More than three file management tasks, however, do not improve throughput significantly, regardless of the number of disk controllers present. The default value is two.

**3.4.4.4 CLOCK Parameter.** This parameter is the interrupt level to be assigned to the internal clock, and may be either 5 or 15, with 5 being the default value. If 15 is specified, the clock must be wired for the level 15 interrupt. This can be done by changing a jumper wire on the system interface board (see the *Model 990/10 or 990/12 Minicomputer Hardware Reference Manual* for details). Having a level 15 clock interrupt may affect time slicing, but it gives all other devices (particularly communication devices and card readers) a higher interrupt priority than the clock, reducing the possibility of timing errors on these devices.

**3.4.4.5 RESTART ID Parameter.** The number specified is the installed ID of a task which will automatically be the first task to execute each time an IPL occurs. Select an ID not reserved for DX10 tasks (see Appendix B) since the task must be installed on the system program file. If no task ID is specified, no task is automatically bid at IPL.

**3.4.4.6 OVERLAYS Parameter.** This parameter specifies the number of system overlay areas to be reserved in system memory. The system overlay area is used to contain overlays needed by DX10 tasks, such as the file management, bid task, and disk manager. Adding more system overlay areas to the system (the maximum is 14) decreases the number of disk accesses needed by the system to execute and improves system throughput. However, each additional overlay area reduces maximum possible system table area by 806 bytes. A system will function with only one system overlay area (a minimum of one is required). More than three overlay areas (five in a system that uses KIFs) offer diminishing improvement in throughput. Most systems run satisfactorily with two. The initial value is two overlay areas.

**3.4.4.7 ONLINE DIAGNOSTIC SUPPORT Parameter.** If selected, the Online Diagnostics function during normal system operation, collecting and providing information about the performance of disk drives. If selected, the extended disk and magnetic tape DSRs, needed to support the online diagnostic tests, are installed; if not selected, the regular DSRs are installed. The Online Disk Diagnostic provides online test capability and provides information that permits evaluation of the frequency and severity of errors experienced with disk drives. The magnetic tape DSR is only installed if tape devices (MTxx) are defined in the system.

**3.4.4.8 SYSLOG Parameter.** The number to be specified is the number of messages which may be queued for the system log. Any number greater than zero is valid. The initial value is six. If you choose too small a number, messages may arrive faster than they can be processed and, as a result, some messages may be lost. Although table area is used only while the message is queued, choosing too large a number risks table area overflow. This degrades performance and may inhibit the execution of some tasks; it may also cause a system crash.

**3.4.4.9 BUFFER MANAGEMENT Parameter.** This parameter specifies the size of the memory-resident file buffer in bytes. The memory-resident buffer is used by file management to perform file I/O for blocked files only (see Volume I for information on blocked files). It is made memory resident in order to allow tasks to use blocked file I/O in small memory configurations where the system may not be able to allocate free memory for a blocking buffer. You can make this buffer very small to maximize your task execution space, but this decision risks >DF errors. (Refer to Volume VI for a description of error codes.) Establishing a memory-resident buffer does not reduce system table area, but decreases the physical memory space available for task execution.

Note that only one block is stored in this buffer at one time regardless of how much space you allocate to the buffer. File management shares the memory-resident buffer between files as needed.

The size of this buffer should be as great as the largest physical record defined for any file to be used on the system. There is no point in making it larger. The initial value is 1024 bytes.

**3.4.4.10 I/O BUFFERS Parameter.** This parameter specifies the maximum amount of memory, in bytes, to be added to the system table area. It is used for buffering device I/O data blocks for CRU type devices only. The number need not account for the buffers associated with the physical device tables for standard devices (see Section 5 for a discussion of PDTs). It should include the total amount of buffer space defined for nonstandard CRU devices. You should also consider the expected number of Initiate I/O calls which will be made to the system. Having extra buffer space will allow a task to execute several Initiate I/O SVCs, which allows a device to service an I/O request. Instead of possibly having to wait for the calling task to be rolled in before the data can be unbuffered and then wait for the task to issue the next request, the buffer for the next queued I/O request can be used and the I/O request can be served immediately. The initial value is 0.

**3.4.4.11 INTERTASK Parameter.** This parameter specifies the maximum size of the intertask communications area (in words). This area is statically allocated out of user memory, and is used to contain messages created by the Put Data SVC. This area must be at least as large as the largest message your application sends to another program.

**3.4.4.12 ITC MESSAGES Parameter.** This parameter specifies the maximum number of intertask communication messages that can be in the intertask communications area for any given queue ID. An error is returned if the task attempts to exceed the maximum number. The initial value is 3. A greater value is necessary only if an application needs to store messages between the time they are sent and received.

**3.4.4.13 KIF Parameter.** You have the option of including or excluding the logic for KIFs in the system being generated. You must answer YES if you want to use KIFs, including KIFs with COBOL. Excluding the KIF logic saves 2.5K words of memory (which does not affect system table area), and reduces the maximum system table area load due to KIF I/O by about 210 words per file manager task. Legal responses to the KIF prompt are YES (include KIF logic) and NO (exclude KIF logic), with YES being the initial value.

**3.4.4.14 COUNTRY CODE Parameter.** This parameter specifies in which country this system is being used. The response may be either the full or abbreviated country name. Refer to Volume III for more information on international considerations. The initial value is US (United States).

**3.4.4.15 POWERFAIL Parameter.** If you include the Model 911 VDT powerfail recovery feature, the VDT screens are refreshed and rebuilt after a power failure occurs. You cannot use this feature unless the system has battery power support. If you respond YES to the POWERFAIL prompt, each 911 VDT in the system requires 2058 additional bytes of physical memory space for storing the screen image. The initial value is NO.

**3.4.4.16 AUTO MEDIA CHANGE RECOVERY Parameter.** With several types of disk drives, unloading a disk or powering down a drive while a volume is still logically installed causes a condition in which subsequent writes to the disk are prevented. If you accept automatic media change recovery, the disk DSR automatically reads the volume information from the disk to determine if the volume has been changed. If the volume that is currently loaded is the same as the volume that is logically installed, the write protection is turned off. If you do not accept automatic media change recovery, write protection remains in effect until you issue an Unload Volume (UV) command or a Check and Reset Volume (CRV) command to the drive. If media change occurs on the system disk, a crash results. The automatic media change recovery option adds 170 bytes to the memory-resident code, which reduces total space available for user programs, but does not reduce maximum possible system table area.

**3.4.4.17 SCI BACKGROUND and SCI FOREGROUND Parameters.** These two numbers specify the maximum number of foreground (interactive) and background (batch) activations of the SCI allowed at any one time. If the maximum number of foreground terminals is already logged-on, then any additional log-on attempt is cancelled and an error message appears at the terminal. If the background limit is exceeded, the request is queued and will be serviced when the number of background SCI activations is lower than the limit. By specifying these parameters, you can limit access to the operating system being generated, and thereby ease some of the demand for system resources. Any number greater than zero is valid. The recommended practice is to allow one foreground area for each terminal on the system and two background areas to be shared by the entire system. The initial value is 8 for FOREGROUND and 2 for BACKGROUND.

Background tasks, receiving the lowest priority, level three, only execute when all higher priority tasks are terminated or suspended. DX10 then divides available CPU time equally among level three tasks. The following examples demonstrate the consequences of different numbers of background areas.

In a system in which two background areas are specified, two simultaneously bid background tasks each require five minutes of execution time. Ten minutes are required for one of the tasks to finish. If four such tasks are simultaneously bid, ten minutes pass before two tasks are completed and ten more minutes pass before the other two are completed. On a system with 10 foreground and 10 background areas, if 10 terminals each simultaneously submit a 5 minute background task, 50 minutes pass at which time all 10 tasks complete. In the same system with only two background areas, only two tasks execute at one time and each one completes in 10 minutes. For this reason, the recommended value for BACKGROUND? is 2.

It is also true that background tasks requiring long execution times can lock out other background tasks. Suppose that 2 background tasks, each requiring 45 minutes, are submitted in a system allowing two background tasks. After 5 minutes, a third background task requiring 5 minutes is submitted. With 2 background areas available, one of the 45-minute tasks must complete (after 90 minutes) before the 5-minute task can even begin to execute. With 3 background areas available, the 5-minute task should complete within the next 15 minutes, delaying completion of the 45-minute tasks by about 5 minutes.

**3.4.4.18 BREAKPOINT Parameter.** This number specifies the number of breakpoints allowed under the system. Breakpoints are used by the system debugger for interrupting program execution (see Volume III for information on the debugger). The initial value is 16.

**3.4.4.19 PANEL DISPLAY(BAR CHART) Parameter.** The value of this parameter determines the data displayed on the data LEDs on the computer system's programmer panel. If you respond YES (the default value), the display is a bar chart representing the computer's performance. The eight leftmost LEDs represent the percent of system disk usage, such that if the four leftmost LEDs are lit, then 50 percent of the system disk's access potential is being used. Similarly, the eight rightmost LEDs represent the percent of available CPU cycles being used. If you respond NO, the current value of the program counter is displayed on the LEDs, and the Show Memory Map (SMM) command usage statistics for the system disk and the CPU reflect a value of 0. Note that if you respond YES, the chart display uses a small, but measurable, amount of CPU time.

**3.4.4.20 CARD 1 and CARD 2 (Defining Expansion Chassis).** GEN990 allows definition of expansion chassis by prompting the CARD parameters. These parameters specify the different interrupt levels in the main chassis to be assigned to CRU expansion cards 1 and 2. Card 1 supports expansion chassis 1, 2, 3, and 4. Card 2 supports expansion chassis 5, 6, and 7. Card 1 and card 2 can be implemented either on one full-size CRU expander board or on two separate boards. If two boards are used, only ports 5, 6, and 7 can be used on the second board. If port 8 is used, a system crash results. If no number is specified, GEN990 assumes that there is no expansion card. Refer to paragraph 3.2.3.1 for a discussion of interrupts within an expansion chassis. Refer to paragraph 3.2.2.3 for a discussion of CRU base addresses within an expansion chassis. Refer to paragraph 3.4.5.1 for a discussion of defining devices that are configured in an expansion chassis.

### 3.4.5 Defining Devices

To define all of the devices included in your system's physical configuration, enter the inquire mode by issuing the INQUIRE command. GEN990 displays the following prompt:

NEXT?

Respond to NEXT? with the following command:

DVC

GEN990 displays the following prompt:

DEVICE TYPE?

Reply to this prompt with the type of a specific device. The types of devices supported by DX10 are the following:

VDT or CRT	Model 911, 913, or 931 VDT; or 940 EVT
CRDR	Model 804 Card Reader
LP	Model 306, 588, 810, 850, 855, 2230, 2260 or 880 Line Printer; or 840 RO, LP300, LP600, or LQ45
ASR	Model 733 ASR
KSR	Model 703/707 KSR, 733/743 KSR, 820 KSR, or teleprinter device
DK	Model FD800 Single-Sided, Single-Density Flexible Diskette Drive
MT	Model 979A, MT1600, or MT3200 Magnetic Tape Drive, or WD800 Tape Cartridge Drive
DS	DS10, DS25, DS31, DS32, DS50, DS80, DS200, DS300, WD500, WD500A WD800, WD800A, WD900, CD1400/32, CD1400/96, or FD1000 Disk Unit
COM	Communications packages (refer to each object installation guide for each communications package)
SD	Special device (that is, nonstandard device)
RTS	Remote Terminal Subsystem Communications Line

For each device entered, GEN990 displays prompts for the parameters needed to define the device. The following paragraphs provide descriptions of these parameters.

For each device defined, GEN990 creates a four-character device name (a pair of alphabetic characters followed by a pair of numeric characters). The alphabetic characters indicate the device group such as DS for TILINE disk drives, DK for flexible diskette drives, and ST for VDTs, automatic send/receive (ASR) terminals, and keyboard send/receive (KSR) terminals. The two numeric characters are assigned in ascending order from 01 to 99. Each unique pair of alphabetic characters added to the system begins with numeric pair 01. For a full discussion of device names, refer to Volume I.

Table 3-9 describes the interface boards and their corresponding system generation names. You use these names when responding to the INTERFACE TYPE? prompt for various devices. These are not all the interface boards supported by TI; they are the ones you need to know about to perform the current system generation. Table 3-9 also describes which boards you can use on the Business System 300, 600, and 800.

Table 3-10 describes the relationship between these interface boards and some of the peripherals to which they may be connected. The board names used in the table are their system generation names.

Table 3-9. Description of Various Interface Boards

Interface Boards	Descriptions	System Generation Name	Used in:		
			BS300	BS600	BS800
CI401 (COMM I/F)	Represents old COMM I/F interface; CI401 is a new name.	CI401		Yes	Yes
TTY/EIA	Current TTY/EIA interface	TTY/EIA		Yes	Yes
9902	TMS9902 port on 990/10A and Business System 300.	9902	Yes	Yes	
CI402	Two-channel unbuffered CRU controller (two 9902 ports)	9902		Yes	Yes
CI421	Two-channel unbuffered CRU controller (one 9902 port and one 9903 port)	9902 9903	Yes		
CI422	Four-channel unbuffered CRU controller (four 9902 ports)	9902	Yes		
CI403	Four-channel buffered TILINE EIA controller	CI403		Yes	Yes
CI404	Four-channel buffered TILINE fiber optics controller	CI404		Yes	Yes

Table 3-10. Relationship Between Peripherals and Interface Boards

Peripherals	Attachable Interface Boards					
	CI401 (COMM I/F)	TTY/EIA	9902	9903	CI403	CI404
931 VDT	X		X	X	X	X
940 EVT	X		X	X	X	X
LP (Serial)		X	X		X	X
TPD	X	X	X			
Special Device	X	X	X	X	X	X

**3.4.5.1 Defining Devices in an Expansion Chassis.** TILINE devices in expansion chassis are physically linked to the main chassis by a pair of interconnected TILINE coupler boards, one in the main chassis and one in the expansion chassis. These TILINE couplers are not apparent to the software.

However, TILINE devices typically share an expansion chassis with CRU devices. CRU devices are physically linked to the main chassis by a pair of interconnected boards: a CRU expander board in the main chassis and a CRU buffer board in the expansion chassis. The CRU expander and the CRU buffer are apparent to the software. In fact, the INTERRUPT parameter for a CRU or TILINE device in an expansion chassis is the interrupt level assigned to the CRU expander in the main chassis.

Definition of devices in expansion chassis requires that you specify the two additional parameters:

- Expansion chassis
- Expansion position

When defining a device in an expansion chassis, GEN990 prompts you for EXPANSION CHASSIS. This parameter specifies the expansion chassis (1 through 7) to which the device is connected.

GEN990 then prompts the EXPANSION POSITION parameter. This parameter specifies the interrupt level (0 through 15) assigned to the slot on the expansion chassis to which the device is connected. Refer to paragraph 3.2.3.1 for more information on interrupts in expansion chassis.

**TILINE Devices in Expansion Chassis.** When you define a TILINE device in an expansion chassis, GEN990 recognizes the interrupt as the same one specified in the CARD 1 or CARD 2 parameter and knows to prompt for additional parameters.

The following example demonstrates the definition of a disk drive whose controller is installed in slot 7 of the second expansion chassis of a system. The second expansion chassis is also connected to Card 1 of a CRU expander in slot 11 of the main chassis.

```
DEVICE TYPE? DS
TILINE ADDRESS? (> F820)
DRIVES? (1) 1
DEFAULT RECORD SIZE? (864)
INTERRUPT? (13) 7
EXPANSION CHASSIS? 2
EXPANSION POSITION? 13
```

Your responses to the prompts are the same as if the controller board were installed in the main chassis with the following exceptions:

- The correct INTERRUPT parameter is the interrupt level assigned to the CRU expander board in the main chassis.
- The EXPANSION CHASSIS parameter is 2 in this example, to indicate that the disk controller board is installed in expansion chassis 2.
- The EXPANSION POSITION parameter in this example is 13, which is the interrupt level of the disk controller board within the expansion chassis.

**CRU Devices in an Expansion Chassis.** When you define a CRU device in an expansion chassis, be sure to assign the correct CRU base address. Refer to paragraph 3.2.2.3 for a discussion of CRU base addresses in expansion chassis. GEN990 recognizes this CRU base address to be within an expansion chassis and prompts you for the same additional prompts described in the preceding paragraph. The following example demonstrates the definition of a 911 VDT controlled by the right half of a controller board which is installed in slot 9 of expansion chassis 1. The CRU expander board is installed in slot 11 of the main chassis:

```

DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100) > 500
INTERRUPT? (10) 7
EXPANSION CHASSIS? 1
EXPANSION POSITION? 10

```

Your responses to the prompts are the same as if the controller board were installed in the main chassis with the following exceptions:

- The CRU ADDRESS parameter is modified to reflect in which expansion chassis the controller board is installed.
- The correct INTERRUPT parameter is the interrupt assigned to the CRU Expander board in the main chassis.
- The EXPANSION CHASSIS parameter is 1 in this example, to indicate that the 911 VDT controller board is installed in expansion chassis 1.
- The EXPANSION POSITION parameter in this example is 10, the interrupt level of the 911 VDT controller board within the expansion chassis.



**3.4.5.2 Defining Disk and Magnetic Tape Drives.** Supply the following GEN990 parameters to add a disk drive to a new DX10:

- DEVICE TYPE
- TILINE ADDRESS
- DRIVES
- DEFAULT RECORD SIZE
- INTERRUPT

The parameters required to add a tape transport are the same, except the DEFAULT RECORD SIZE is neither required nor prompted.

Respond to the DEVICE TYPE prompt with DS to define a disk and MT to define a magnetic tape drive.

The TILINE ADDRESS parameter is prompted by GEN990 when a TILINE device (in this case, a disk drive or magnetic tape unit) has been defined in the system. Respond to this parameter with the TILINE ADDRESS of the device just named. See paragraph 3.2.4 for a discussion of the TILINE.

The DRIVE parameter only applies to disk, diskette, and magnetic tape units, which may have more than one drive per controller. Enter the number of drives connected to the controller being defined, either 1, 2, 3, or 4. The initial value is 1.

GEN990 requests a DEFAULT RECORD SIZE parameter for each disk controller defined. DX10 uses the specified number of bytes to define the physical record size of a disk file when none is specified in the SVC block or in the directory under which the file is created at file creation time. The DX10 routine for volume initialization uses a multiple of three sectors for its unit of disk allocation.

The INTERRUPT parameter specifies the interrupt level (3 through 15) to be assigned to the device being described. The interrupt for each slot of your chassis is listed on the configuration label that is attached to the chassis.

The following examples show how to define disks in a new system:

```
DEVICE TYPE? DS
TILINE ADDRESS? (> F800)
DRIVES: (1) 3
DEFAULT RECORD SIZE: (768)
INTERRUPT: (13)
DEVICE TYPE? DS
TILINE ADDRESS? (> F800) > F820
DRIVES: (1) 2
DEFAULT RECORD SIZE: (768)
INTERRUPT: (13) 12
```

The following example shows the definition of a tape transport:

```

DEVICE TYPE? MT
TILINE ADDRESS? (F880)
DRIVES: (1)
INTERRUPT: (9)

```

**3.4.5.3 Defining Communications Devices.** DX10 optionally supports several types of communications packages. The parameters necessary to define communications devices are:

- DEVICE TYPE
- CRU OR TILINE ADDRESS
- NUMBER OF CHANNELS
- PROTOCOL
- BUFFER SIZE
- INTERRUPT

#### NOTE

Before you execute GEN990, you should read the object installation documents for each communications package that you want to install on your system.

The DEVICE TYPE parameter must be specified as COM or COMM. The CRU or TILINE parameter is the address of the slot in the main or expansion chassis in which the communication card is connected. If the CRU address is entered as a TILINE address, more than one communications device can share that address. If the CRU address is a non-TILINE address, that CRU address can be assigned to only one communications device. Calculation of a CRU address is explained in paragraph 3.2.2.

The NUMBER OF CHANNELS parameter should be entered as 1 for a device using a Communication Interface board and can be 1 to 4 for a multiple channel device. GEN990 does not verify that the value entered is legal since other devices with unknown characteristics can be added in the future.

The PROTOCOL prompt is displayed as many times as the value you entered for the NUMBER OF CHANNELS parameter. Valid responses to this prompt are 3780, 2780, or 3270. The 3780 and 2780 packages are normally used in a hardware configuration, which includes a console device and at least one other I/O device (for example, a line printer). The 3270 package is generally used in conjunction with VDTs and/or a line printer. The default is NONE. If the default is taken for all NUMBER OF CHANNELS prompts, GEN990 ignores the current COMM device definition.

If the response to the PROTOCOL prompt is 3780 or 2780, the BUFFER SIZE prompt is displayed. No BUFFER SIZE prompt is displayed if the PROTOCOL response is 3270. The BUFFER SIZE parameter specifies the size of the internal CRU buffer to be used by the communications device. This buffer is used to receive data from the communications interface card. The default for the 3780 and 2780 required buffer size is 528 bytes.

The INTERRUPT parameter specifies the hardware interrupt assigned to the slot in which the communications interface card is connected. This parameter is described in paragraph 3.2.3.

The following examples show how both a 3780 and a 3270 package are defined:

```
DEVICE TYPE? COMM
  CRU OR TILINE ADDRESS? (> 140)
  NUMBER OF CHANNELS (1)?
  CHANNEL NUMBER 00 PROTOCOL? (NONE) 3780
  INTERRUPT? (8)
  BUFFER SIZE (528)?

DEVICE TYPE? COMM
  CRU OR TILINE ADDRESS? (> 140) > 160
  NUMBER OF CHANNELS (1)?
  CHANNEL NUMBER 00 PROTOCOL? (NONE) 3270
  INTERRUPT? (8)
```

**3.4.5.4 Defining Display Terminals.** The parameters needed to define a 911, 913, or 931 VDT, or 940 EVT are the following:

- DEVICE TYPE
- ACCESS TYPE
- TIME OUT
- CHARACTER QUEUE
- VDT TYPE
- CRU ADDRESS
- INTERRUPT

In response to the DEVICE TYPE prompt, enter VDT or CRT.

The ACCESS TYPE should be specified as either RECORD or FILE. A device that is record oriented can be opened for I/O by many tasks concurrently. A file-oriented device can only be used by one task at a time. The initial value for VDTs is RECORD. Refer to Volume I for a discussion of file orientation.

The **TIME OUT** parameter specifies the number of seconds DX10 allows for an I/O operation to occur before declaring an error condition. Interactive devices are generally assigned no time-out value because the time required for an input operation is operator dependent. The initial value is 0 (no time-out).

The **CHARACTER QUEUE** parameter specifies the number of input characters to be buffered. The initial value is 6. The character queue must be an even number of characters.

The **VDT TYPE** parameter allows you to specify whether the device is a 911, 913, or 931 VDT, or 940 EVT. Enter either 911, 913, 931, or 940 in response to this prompt.

The **CRU** parameter specifies the CRU address of the chassis slot in which the VDT controller is connected, as described in paragraph 3.2.2. The initial value is > 100.

The **INTERRUPT** parameter specifies the interrupt level assigned to the chassis slot in which the VDT controller board is located. The initial value is 10.

Additional parameters needed to define a 931 VDT or 940 EVT are as follows:

- ASSOCIATED PRINTER
- TIME OUT
- SWITCHED LINE
- SPEED
- INTERFACE TYPE
- TILINE ADDRESS
- CHANNEL NUMBER

In response to the **ASSOCIATED PRINTER** prompt, enter YES if a printer is attached to the auxiliary port on the 931 or 940. An attached printer always has a value of YES in response to the **EXTENDED?** prompt (refer to paragraph 3.4.5.7). The initial value is NO.

The **TIME OUT** parameter specifies the number of seconds allowed by DX10 for an I/O operation to the printer. If more than the specified amount of time elapses, an I/O error is assumed. A response of zero signifies no time-out count (that is, the printer has as long as necessary to complete the operation). The initial value is 60 seconds.

The SWITCHED LINE prompt is used to specify the connection procedure for the terminal. Respond NO for such configurations as local direct connect cable, leased telephone line with modems, and switched telephone line with modems. A NO response to the SWITCHED LINE prompt causes the operating system to assert Data Terminal Ready (DTR) and Request to Send (RTS) signals unconditionally during IPL. This allows initiation of connection from either the terminal or the CPU. The response to this prompt is usually NO. A YES response to the SWITCHED LINE prompt causes the operating system to await a Ring Indicator (RI) signal from the modem to the CPU before asserting DTR and RTS signals to complete the connection process. Respond YES only if all the following conditions are met:

- The terminal and CPU are connected via a switched telephone line and modems.
- The modems require a RI signal prior to asserting DTR and RTS signals.
- The call is placed from the terminal to the CPU.

The Modify VDT or Printer Characteristics (MVPC) command can be used to modify a response to the SWITCHED LINE prompt without performing another system generation. The initial value is NO.

The SPEED prompt requests the baud rate at which the terminal is operating. Legal baud rates are 300, 600, 1200, 2400, 4800, 9600, and 19200 baud. Make sure your hardware matches one of the listed baud rates. The initial value is 1200.

The INTERFACE TYPE prompt specifies which interface board the terminal is using. The valid responses are CI401 (previously COMM I/F), CI403, CI404, 9902, and 9903. Refer to Table 3-9 for more details.

The TILINE prompt appears if the response to the INTERFACE TYPE prompt is either CI403 or CI404. Otherwise, the CRU ADDRESS prompt appears. The TILINE prompt specifies the TILINE address of the board to which the VDT is connected, as described in paragraph 3.2.4. The initial value is > F980.

The CHANNEL NUMBER prompt appears if the response to the INTERFACE TYPE prompt is either CI403 or CI404. This prompt specifies the channel number on the board to which the VDT is connected. The initial value is 0.

The following example shows the definition of all four VDT types:

```
DEVICE TYPE? CRT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100)
INTERRUPT? (10)
```

```
DEVICE TYPE? CRT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 913
CRU ADDRESS? (> 100) > 00
INTERRUPT? (10) 6
```

```
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 940
ASSOCIATED PRINTER? (NO)
SWITCHED LINE? (NO)
SPEED? (1200) 9600
INTERFACE TYPE? CI403
TILINE ADDRESS? (> F980)
CHANNEL NUMBER? (0)
INTERRUPT? (10) 7
```

```
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 931
ASSOCIATED PRINTER? (NO)
SWITCHED LINE? (NO)
SPEED? (1200) 9600
INTERFACE TYPE? 9902
CRU ADDRESS? (> 1700)
INTERRUPT? (10) 8
```

**3.4.5.5 Defining ASRs.** To define a 733 ASR, you must supply the following parameters:

- DEVICE TYPE
- CRU ADDRESS
- ACCESS TYPE
- TIME OUT
- CHARACTER QUEUE
- CASSETTE ACCESS TYPE
- CASSETTE TIME OUT
- INTERRUPT

To define an ASR, respond to the DEVICE TYPE prompt by entering ASR.

The ACCESS TYPE or CASSETTE ACCESS TYPE parameter specifies whether a device is record oriented or file oriented. Enter either RECORD or FILE. Record-oriented devices may be accessed by many tasks at the same time, but file-oriented devices may only be opened by one task at a time. Disks, magnetic tape, and diskettes are automatically record oriented. The default value is the standard access type for the device being described. Card readers, line printers, and cassettes are usually file oriented, while ASRs, KSRs, and VDTs are typically record oriented. Refer to Volume I for a discussion of device orientation.

The TIME OUT or CASSETTE TIME OUT parameter specifies the number of seconds which is to be used by DX10 as the time-out count for the device. When an I/O operation is initiated on a device, DX10 allows the device to wait for the time period specified here for the operation to complete, after which it assumes that an error has occurred. If no time-out is desired, enter 0. The default value is different for different devices: 30 seconds for the line printers and card readers, 3 seconds for cassettes, and no time-out for interactive devices.

The CHARACTER QUEUE parameter specifies the number of characters from a keyboard device that should be buffered for input. If more characters are entered in the queue, they are ignored. The initial value is 6.

The INTERRUPT parameter specifies the interrupt level (3 through 15) to be assigned to the device being defined, as described in paragraph 3.2.3. The initial value is 6.

The following example shows how to define an ASR. GEN990 assigns device names (such as ST01, ST02) to terminal devices in the order in which the devices are defined.

```
DEVICE TYPE? ASR
CRU ADDRESS? (> 00)
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
CASSETTE ACCESS TYPE? (FILE)
CASSETTE TIME OUT? (3)
INTERRUPT? (6)
```

When an ASR is defined, the cassette units are named left to right; in the previous example, CS01 is the left cassette drive and CS02 is the right drive. If you define more ASRs, GEN990 continues to number the cassettes sequentially.

**3.4.5.6 Defining KSRs (or Other Teleprinter Devices (TPDs)).** The types of KSRs supported by DX10 are 703, 707, 733, 743, 745, 763, 765, 781, 783, 785, 787, and 820.

#### NOTE

You can define a 743 or 745 terminal that is interfaced to a 990 using the TTY/EIA interface module and unswitched lines as a KSR using either the KSR or the TPD DSR. You can define an 820 terminal that is interfaced to a 990 using a TTY/EIA interface module and unswitched lines as a KSR using either the K820 or TPD DSR. You can define a 733 terminal as either an ASR or a KSR using the KSR DSR. It is recommended that 743, 745, or 820 terminals be defined using the TPD DSR. This allows you to access various utility programs not otherwise available. Future utility software developed to operate with hard-copy terminals may not operate for terminals that are not defined as TPD devices.

The parameters needed to define a KSR device are the following:

- DEVICE TYPE
- DSR TYPE(TPD/KSR/K820)
- CRU ADDRESS
- ACCESS TYPE
- TIME OUT
- CHARACTER QUEUE
- INTERRUPT



In response to the DEVICE TYPE prompt, enter KSR.

In response to the DSR TYPE(TPD/KSR/K820) prompt, enter either TPD, KSR, or K820, according to the nature of the device you are defining.

The CRU ADDRESS parameter specifies the CRU address of the chassis slot in which the interface is connected, as described in paragraph 3.2.2. The initial value is > 00.

The ACCESS TYPE should be specified as either RECORD or FILE. A device that is record oriented can be opened for I/O by many tasks concurrently. A file-oriented device can only be used by one task at a time. The initial value for TPD is RECORD. Refer to Volume I for a discussion of file orientation.

The TIME OUT parameter specifies the number of seconds that DX10 allows for an I/O operation to occur before declaring an error condition. Interactive devices are generally assigned no time-out value, since the time required for an input operation is operator dependent. The initial value is 0 (no time-out).

The CHARACTER QUEUE parameter specifies the number of input characters to be buffered. If the Bubble Memory Terminal Support software (BMTS-990) is used at this port, see the *BMTS-990 User's Guide* for additional information regarding this parameter. The initial value is 6.

The INTERRUPT parameter specifies the interrupt level assigned to the chassis slot in which the VDT controller board is connected, as described in paragraph 3.2.3. The initial value is 6.

Additional parameters needed to define a TPD are the following:

- TERMINAL TYPE
- INTERFACE TYPE
- SWITCHED LINE
- BAUD RATE
- ACU PRESENT
- ACU CRU\*
- FULL DUPLEX MODEM
- ECHO

\* Not always prompted.

The **TERMINAL** parameter allows you to specify which type of terminal you are defining. The allowed responses are 703, 707, 743, 745, 763, 765, 781, 783, 785, 787, and 820.

The **ECHO** prompt asks whether keystrokes should be echoed (sent back to the terminal) by the 990. The default is YES. Echo must be selected for character editing to work properly at the terminal. If echo is selected, the terminal must be configured without local copy. If a half-duplex modem is used, echo should not be selected.

**FULL DUPLEX MODEM** asks if the modem is full duplex or half duplex. If the modem is a Bell 103J or a Bell 212A or a Vadic 3400, answer YES. If no modem is used, enter YES. If the modem is a Bell 202S, enter NO.

Respond to the **SWITCHED LINE** prompt to specify the type of telecommunication used with the TPD. Remote terminals accessed using modems on the public telephone network are on switched (dial-up) lines. Directly connected terminals and terminals on leased remote lines are not switched. For the latter case, accept the initial value of NO. If the terminal uses dial-up lines, respond Y (YES).

The **INTERFACE TYPE** prompt specifies which interface board the terminal is using. The valid responses are CI401 (previously COMM I/F), TTY/EIA, and 9902. Refer to Table 3-9 for more details.

Acceptable **BAUD RATES** are 110, 300, 600, 1200, 2400, 4800, and 9600. The initial value is 300 baud. Make sure your hardware matches one of the listed baud rates.

If an automatic call unit is used with the terminal, enter YES in response to the **ACU PRESENT?** prompt; otherwise, accept the initial value of NO. If you respond YES, GEN990 displays another prompt, **ACU CRU?** Respond with the CRU base address of the ACU. The initial value is 0.

The following example shows the definition of three devices:

```
DEVICE TYPE? KSR
DSR TYPE(TPD/KSR/K820)? KSR
CRU ADDRESS? (> 00)
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
INTERRUPT? (6)
```

```
DEVICE TYPE? KSR
DSR TYPE(TPD/KSR/K820)? K820
CRU ADDRESS? (> 00) > 40
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
INTERRUPT? (6)
```

DEVICE TYPE? KSR  
DSR TYPE(TPD/KSR/K820)? TPD  
TERMINAL TYPE? (743)  
INTERFACE TYPE? CI401  
SWITCHED LINE? (NO)  
BAUD RATE? (300) 1200  
ACU PRESENT? (NO)  
FULL DUPLEX MODEM? (YES)  
ECHO (YES)?  
CRU ADDRESS? (> 00) > 60  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
INTERRUPT? (6)

**3.4.5.7 Defining Line Printers.** The types of line printers supported by DX10 are: Model 306, 588, 810, 820 RO, 840 RO, 850, 855, 2230, 2260, LP300, LP600, and LQ45. The prompts needed to define a line printer are the following:

- DEVICE TYPE
- ACCESS TYPE
- PRINT MODE
- EXTENDED
- SPEED\*
- TIME OUT
- INTERFACE TYPE
- CRU ADDRESS or TILINE ADDRESS
- CHANNEL NUMBER
- INTERRUPT

\* Not always prompted.

The DEVICE TYPE prompt requires you to enter LP to define a printer.

Specify the ACCESS TYPE as either RECORD (R) or FILE (F) oriented. Refer to Volume I for a discussion of file orientation. The initial value is FILE.

The PRINT MODE should be specified as either PARALLEL or SERIAL.

If a printer is specified as being serial, GEN990 prompts the EXTENDED? parameter. The 810 serial printer may have an extended character set. If the device being added has an extended character set, you should answer YES; if not, respond with NO. If you want to print lowercase characters, answer YES. The initial value is YES.

GEN990 also prompts the SPEED parameter if a printer is specified as being serial. The SPEED prompt requests the baud rate at which the printer operates. Legal baud rates are 110, 200, 300, 600, 1200, 2400, 4800, and 9600. Make sure your hardware matches one of the listed baud rates. The initial value is 4800.

The TIME OUT parameter specifies the number of seconds allowed by DX10 for an I/O operation to the printer. If more than the specified amount of time elapses, an I/O error is assumed. A response of zero signifies no time-out count (that is, the printer has as long as necessary to complete the operation). The initial value is 60 seconds.

The INTERFACE TYPE prompt appears if the response to the PRINT MODE prompt is SERIAL. This parameter specifies which interface board the printer is using. The valid responses are CI403, CI404, TTY/EIA, and 9902. Refer to Table 3-9 for more details.

The CRU parameter specifies the CRU address of the chassis slot in which the printer interface card is connected, as described in paragraph 3.2.2. The initial value is > 60.

The TILINE prompt appears if the response to the INTERFACE TYPE prompt is either CI403 or CI404. Otherwise, the CRU prompt appears. The TILINE parameter specifies the TILINE address of the interface board to which the printer is connected, as described in paragraph 3.2.4. The initial value is > F980.

The CHANNEL NUMBER prompt appears if the response to the INTERFACE TYPE prompt is either CI403 or CI404. This parameter specifies the channel number on the interface board to which the printer is connected. The initial value is 0.

The INTERRUPT parameter specifies the interrupt assigned to the chassis slot in which the printer interface card is connected, as described in paragraph 3.2.3. The initial value is 14.

The following example shows the definitions of three printers:

```

DEVICE TYPE? LP
ACCESS TYPE? (FILE)
PRINT MODE? (SERIAL)
EXTENDED? (YES)
SPEED? (4800)
TIME OUT? (60)
INTERFACE TYPE? TTY/EIA
CRU ADDRESS? (> 60)
INTERRUPT? (14) 10
EXPANSION CHASSIS? (1)
EXPANSION POSITION? 5

```

```

DEVICE TYPE? LP
ACCESS TYPE? (FILE)
PRINT MODE? (SERIAL) P
EXTENDED? (YES)
SPEED? (4800)
TIME OUT? (60)
CRU ADDRESS? (> 60) > 100
INTERRUPT? (14)

```

```

DEVICE TYPE? LP
ACCESS TYPE? (FILE)
PRINT MODE? (SERIAL)
EXTENDED? (YES)
SPEED? (4800)
TIME OUT? (60)
INTERFACE TYPE? CI403
TILINE ADDRESS? (> F980)
CHANNEL NUMBER? (0)
INTERRUPT? (14)

```

**3.4.5.8 Defining a Card Reader.** The parameters needed to define a card reader are the following:

- DEVICE TYPE
- CRU ADDRESS
- ACCESS TYPE
- TIME OUT
- INTERRUPT

Enter CR in response to the DEVICE prompt. The CRU parameter specifies the CRU address assigned to the card reader. CRU address calculation is described further in paragraph 3.2.2. The initial value is > 40.

## NOTE

Connecting a card reader to an expansion chassis is not recommended. A card reader generates many interrupts that must be serviced immediately by DX10; therefore, connect the card reader to the main chassis and assign it the lowest interrupt level possible. No other device can be assigned the same interrupt level as a card reader.

The ACCESS TYPE parameter indicates whether the card reader is RECORD (R) or FILE (F) oriented. Refer to Volume I for a discussion of file orientation. The initial value is FILE.

The TIME OUT parameter specifies the number of seconds in the time-out count for the card reader. The initial value is 30.

The INTERRUPT parameter specifies the interrupt level assigned to the chassis slot in which the card reader interface card is connected. This number should be as low as possible. The initial value is 4.

The following example shows definition of a card reader:

```
DEVICE TYPE? CR
CRU ADDRESS? (> 40) > 80
ACCESS TYPE? (FILE)
TIME OUT? (30)
INTERRUPT? (4)
```

**3.4.5.9 Defining Diskettes.** The parameters required to define Model FD800 single-sided, single-density diskette drives to GEN990 are the following:

- DEVICE TYPE
- CRU ADDRESS
- DRIVES
- TIME OUT
- INTERRUPT

Respond to the DEVICE TYPE prompt by entering DK.

The CRU ADDRESS parameter specifies the CRU address of the chassis slot in which the diskette controller is connected. The default value is > 80.

The DRIVES parameter specifies the number of diskette drives (1, 2, 3, or 4) which are connected to the controller being defined. GEN990 names each drive DKXX, where XX is the drive number, in the same way as disk drives.

The TIME OUT parameter specifies the number of seconds allowed by DX10 for an I/O operation to a diskette drive to complete. The initial value is 30.

The INTERRUPT parameter specifies the hardware interrupt level assigned to the chassis slot in which the diskette controller is connected. Paragraph 3.2.3 provides a more detailed explanation.

The following example shows the definition of two diskette drives:

```
DEVICE TYPE? DK
CRU ADDRESS? (> 80)
DRIVES? (1) 3
TIME OUT? (30)
INTERRUPT? (7)
```

```
DEVICE TYPE? DK
CRU ADDRESS? (> 80) > 220
DRIVES? (1) 2
TIME OUT? (30)
INTERRUPT? (7)
```

**3.4.5.10 Defining Nonstandard Devices.** The following parameters are required to define a device not supported by the standard DX10 software modules:

- DEVICE TYPE
- INTERFACE TYPE
- SPECIAL DEVICE ADDRESS
- CHANNEL NUMBER
- DEVICE NAME
- INTERRUPT BIT
- KSB ADDRESS
- DSR WORKSPACE
- INTERRUPT ENTRY
- PDT FILE
- DSR FILE
- OVERRIDE
- INTERRUPT

In order to include a nonstandard device, you will have to write a DSR to handle it. See Section 5 for instructions. You should write the DSR before beginning system generation.

Respond to the prompt **DEVICE TYPE** with **SD** (Special Device).

The **INTERFACE TYPE** parameter specifies which interface board the device is using. The valid responses are **CI403**, **CI404**, **9902**, and **NONE**. Refer to Table 3-9 for more details. Enter **NONE** if you are not using a **CI403**, **CI404**, or **9902**. The initial value is **NONE**. This information is used to construct the interrupt decoder tables and does not provide any routines for handling the interface.

In response to the **SPECIAL DEVICE ADDRESS** parameter, enter the **TILINE** or **CRU** address assigned to the device. If you respond **NONE**, the **INTERRUPT** parameter is also set to **NONE** and the **OVERRIDE** parameter is set to **NO**. The initial value for this parameter is **> 100**.

The **CHANNEL NUMBER** prompt appears if the response to the **INTERFACE TYPE** prompt is **CI403** or **CI404**. This prompt specifies the channel number on the board to which the device is connected.

Assign a name to the device in response to the prompt **DEVICE NAME**. The name must be four characters long and begin with an alphabetic character. For example, if the device is a plotter, then you can assign the name **PL01**. If a device is a special graphics terminal, you can assign the name **GR01**. For each subsequent definition of this type of terminal, you can use the names **GR02**, **GR03**, and so on.

The **INTERRUPT BIT** prompt appears if the response to the **INTERFACE TYPE** prompt is **NONE**. When several devices share the same interrupt level, **GEN990** must know the **CRU** interrupt bit that indicates which device sent the interrupt in order to build a suitable interrupt decoder. Respond to this prompt with the appropriate **CRU** interrupt bit. The default is **15**. The valid values are in the range **0** to **31**.

The **KSB ADDRESS** parameter identifies the label of the starting address of the keyboard status block (**KSB**) for this device as specified in bytes 12 and 13 of the physical device table (**PDT**). If the device does not have a keyboard or if you do not intend to bid **SCI** with the device, then accept the initial value of **NONE** by pressing the Return key. The device must have a **KSB** to have an **STxx** (station) device name.

Respond to the **DSR WORKSPACE** prompt with the label of the address of the workspace to be used by the **DSR** when processing device interrupts. The conventional value for this parameter is either the first word of the **KSB** (if the device has a **KSB**) or the third word of the **PDT** (for nonkeyboard devices).

For the **INTERRUPT ENTRY** parameter, enter the label of the entry point in the **DSR** for processing interrupts for the device.

Respond to the **PDT FILE** prompt with the pathname of the file containing the **PDT** source module.

Respond to the **DSR FILE** prompt with the pathname of the file containing the **DSR** object module.



When you write the PDT for the special device, you may not know in which chassis slot the device will reside. The data in certain fields in the PDT and KSB (CHASSIS/TILINE ADDRESS and STATUS FLG/INTERRUPT – 1 in the PDT and CRU INPUT ADDRESS in the KSB) is dependent on the slot number. Respond YES to the OVERRIDE parameter if you want GEN990 to fill in these fields according to information supplied during system generation. Respond NO if the PDT already contains this data. If the special device has a keyboard, GEN990 overrides the station ID field in the KSB.

Enter the interrupt level of the slot in which the special device is installed to respond to the prompt INTERRUPT. If your response is NONE, the SPECIAL DEVICE ADDRESS PARAMETER is also set to NONE and the OVERRIDE parameter is set to NO. The initial value is 15.

Several examples of special device definition are included in Appendix F.

### 3.4.6 Defining SVCs

You can write your own SVC processors and include them in the DX10 software. It is necessary to write the SVC processor and install it in a file prior to defining its parameters during system generation.

GEN990 assigns identifying numbers to user-written SVCs, starting at > 80. These numbers are assigned in the sequence that the SVCs are defined. These numbers are not displayed in the output of the LIST command, but they are recorded in the configuration file, which may be inspected. If you delete an SVC, the identifying numbers are reordered to maintain an unbroken sequence. If you modify an SVC using the CHANGE command, the SVC retains its order in the number sequence, even if the content of the SVC has changed.

To define an SVC, issue the SVC command from the inquire mode. The following prompts appear:

- SVC
- FILE
- BYTES

The SVC parameter is the label of the initial entry point to a user-supplied SVC processor. You must define this label in the processor. User-defined SVCs are assigned SVC codes by GEN990, starting at > 80 and incrementing by one for each SVC defined.

The FILE parameter is the pathname of the file which contains the object module of the SVC processor being defined.

The BYTES parameter specifies the number of bytes which are used by the call block of the SVC just defined. The number must be at least one, which is the initial value.

After you respond to these prompts, GEN990 returns to the inquire mode.

### 3.4.7 Terminating GEN990

Terminate GEN990 by issuing either the STOP command or the BUILD command. Refer to paragraph 3.3.3.7 for a description of these commands' actions.

### 3.4.8 Optional Processor System Generation Parameters

During system generation, the parameters you specify in response to the GEN990 prompts should take into account the optional processors you want to include. The optional processors that can have an effect on the system generation procedure include, but are not limited to, the following:

- 915 RTS
- Sort/Merge
- 3780/2780
- 3270/ICS
- Bubble Memory Terminal Support (BMTS-990)

The following paragraphs explain the system generation considerations if you want to include these processors.

**3.4.8.1 DX10 Remote Terminal Subsystem (RTS).** If you want to include RTS in a DX10 system, you must first perform the RTS pre-system generation procedures as specified in the *Model 990 Computer DX10 Remote Terminal Subsystem (RTS) System Generation and Programmer's Reference Manual*. After you complete these procedures, you can perform the standard DX10 system generation procedure with each RTS communications line included as a special device (in addition to all other devices that must be included in your system).

Refer to the *Model 990 Computer DX10 Remote Terminal Subsystem (RTS) System Generation and Programmer's Reference Manual* for information concerning responses to the DX10 system generation prompts to include RTS communications.

**3.4.8.2 Sort/Merge.** When using Sort/Merge with COBOL, FORTRAN, BASIC, or assembly language, you must specify an intertask communication area. In response to the INTERTASK prompt, specify the number of words of the system table area that may be used for intertask communications.

Refer to the system generation requirements section of the *Sort/Merge User's Guide* for information on how to compute the value required for the intertask communication area.

**3.4.8.3 3780/2780 and 3270/ICS Emulators.** The parameters necessary to define these processors during the system generation procedure are described in paragraph 3.4.5.3 on defining communications devices.

For additional information concerning the system generation procedures for these processors, refer to the *DX10 3780/2780 Emulator Object Installation* and the *Model 990 Computer DX10 3270 Interactive Communication Software (ICS) Object Installation* manuals.

**3.4.8.4 Bubble Memory Terminal Support.** To use the BMTS-990, you must include a KSR using the teleprinter DSR during system generation. The parameters necessary to define teleprinter devices during system generation are listed in paragraph 3.4.5.6. For further information on BMTS-990, see the *Bubble Memory Terminal Support 990 User's Guide*.

### 3.5 ASSEMBLING AND LINKING THE SYSTEM

After you define the system configuration using GEN990, the data files must be processed into a machine-useable form. DX10 provides the Assemble and Link Generated System (ALGS) command to perform this processing automatically.

#### NOTE

If you are generating any communications products, make sure you have performed BCD as described in the communications installation documents prior to performing ALGS.

#### 3.5.1 Executing the ALGS Command

Issue the ALGS command. The following prompts appear on the screen:

```
ASSEMBLE AND LINK GENERATED SYSTEM
      DATA DISK:  DS01
      TARGET DISK: DS01
      SYSTEM NAME: TEST
D$DATA LISTING:
      BATCH LISTING:
```

Respond to the prompt DATA DISK with the device name or volume name of the disk or disk drive that contains the GEN990 output files. This is the same disk that you specify for the XGEN command to use as the data disk. The GEN990 output files are stored under a directory `.$SYSGEN.<system name>`, where `<system name>` is the name you assigned to the prompt OUTPUT CONFIGURATION for the XGEN command.

The output of the ALGS command is sent to a system file `S$IMAGES` on the target disk. In response to the prompt TARGET DISK, enter the device name or volume name of the target disk.

#### NOTE

Prior to executing the ALGS command, verify that a `.$IMAGES` program file exists on the target disk. If this program file does not exist, you must create it using the Create Program File (CFPRO) command. Refer to Volume II for a description of the CFPRO command. Copy the procedure DUMMY from the existing `.$IMAGES` program file to the one created using the Copy Program Image (CPI) command.

Respond to the prompt SYSTEM NAME with the name of the output configuration assigned to the prompt OUTPUT CONFIGURATION for the XGEN command.

The value of the prompt D\$DATA LISTING is the access name of a sequential file or output device to which the macro assembler output listing is written. The default access name follows this pattern: < data disk> .S\$\$SYSGEN.< system name> .D\$LIST.

The value of the prompt BATCH LISTING is the access name of a sequential file or output to which the ALGS output listing is written. The default value is a pathname with the form < data disk> .S\$\$SYSGEN.< system name> .ALGSLIST, such that < data disk> is the value that you supplied to the prompt DATA DISK and < system name> is the value you supplied to the prompt SYSTEM NAME.

### 3.5.2 Results of the ALGS Command

ALGS executes SCI in batch mode. The ALGS command stream executes a batch command stream (developed by GEN990 and stored in the file < data disk> .S\$\$SYSGEN.< system name> .BATCHSTM) and sends a listing to the access name specified for the prompt BATCH LISTING. The batch command stream executes the Macro Assembler, the Link Editor, and a synonym assignment program that creates a patch file for the new system.

The ALGS command requires from 15 to 30 minutes to execute, depending on the size and complexity of your computer system.

**3.5.2.1 Normal Termination.** When the ALGS process terminates, a message appears at the terminal. If processing completes normally, the message is:

```
*** ALGS NORMAL TERMINATION ***
```

Press the Return key. The following message appears:

```
*** ALGS NORMAL TERMINATION ***
```

In this case, press the Return key and proceed by patching the generated system as described in paragraph 3.6.

A successful ALGS execution creates the following files:

- A D\$DATA object module output from the Macro Assembler in .S\$\$SYSGEN.< system name> .D\$OBJECT.
- A task image which is the linked system, in file .S\$IIMAGES on the target disk.
- A patch file (in the file .S\$\$SYSGEN.< system name> .PATCHFIL) ready to be applied to the system image.
- A link map of the linked system in the file .S\$\$SYSGEN.< system name> .LINKMAP.
- A file containing any error messages output by the patch generator in the file .S\$\$SYSGEN.< system name> .ERROR
- An SCI batch listing, if you accept the defaults for the BATCH LISTING parameter, in the file .S\$\$SYSGEN.< system name> .ALGSLIST.

**3.5.2.2 Abnormal Termination.** If ALGS does not terminate normally, one of the following error messages appears on the terminal screen:

```
*** ALGS — D$DATA ASSEMBLY ERROR = <code> ***:  
*** ALGS — SYSTEM LINK EDIT ERROR = <code> ***:  
*** ALGS — SYNONYM GENERATOR FATAL ERROR = <code> ***:  
*** ALGS — SYNONYM GENERATOR NON-FATAL ERROR = <code> ***:
```

where:

<code> = a 4- or 5-digit condition code

The first three error messages indicate unsuccessful completion of ALGS. The fourth error message indicates that a non-fatal error occurred and ALGS has successfully completed. If the fourth error message appears, proceed by patching the generated system as described in paragraph 3.6.

If the first error message, D\$DATA ASSEMBLY ERROR, appears, the error is a macro assembler error. The second error message indicates a fatal Link Editor error. The third error message indicates that a fatal error occurred in the synonym generator section of ALGS.

You need to examine the batch listings created by the ALGS command to determine the cause of a fatal error. These two files contain the listings:

- <data disk>.\$SYSGEN.<system name>.ALGSLIST — The batch SCI listing
- <data disk>.\$SYSGEN.<system name>.\$DLIST — The assembly listing of the new system's D\$DATA module

The last two lines of the batch SCI listing (ALGSLIST) contain the error messages that appeared on the terminal screen. Starting from the last line and working forward, inspect the batch SCI listing for the last SCI command listed. Between this point in the listing and the end is an error message and a 4-digit error code. Refer to Section 4. The following error codes may accompany the third error message:

- 8001 — Module or synonym name too long
- 8002 — Same synonym for different overlays
- 8003 — Duplicate module
- C001 — Symbol too long
- C002 — Unexpected EOF on link map
- C003 — Too many modules
- C004 — Bad link map format

Such errors indicate that the patch generator could not create a proper patch file from its input. If the software supplied by Texas Instruments has not been modified, see Volume VI.

### 3.5.3 ALGS Batch Stream

The following example shows the batch stream executed by ALGS for a system called TEST on DS01.

```

*===== SYSGEN COMPLETION BATCH STREAM =====
*****
***** ASSEMBLE D$DATA *****
*****
BATCH                ! 04/26/82-16:19:16
XMA  SOURCE = @$$DSCD.S$$SYSGEN.TEST.D$SOURCE,
      OBJECT = @$$DSCD.S$$SYSGEN.TEST.D$OBJECT,
      LIST  = XMA$L,
      OPTION = (BUN,TUN,DUN)
.SYN XMA$L = ""
.IF @$$CC,GE,04000
  CM R=ME,M="*** ALGS - D$DATA ASSEMBLY ERROR = @$$CC ***"
  .STOP T="*** ALGS - D$DATA ASSEMBLY ERROR ***"
.ENDIF
*****
***** LINK THE SYSTEM *****
*****
XLE  CONTROL= @$$DSCD.S$$SYSGEN.TEST.LINKSTRM,
      LINK   = @$$DSC$.S$IMAGES,
      LIST  = @$$DSCD.S$$SYSGEN.TEST.LINKMAP
.IF @$$CC,GE,04000
  CM R=ME,M="*** ALGS - SYSTEM LINK EDIT ERROR = @$$CC ***"
  .STOP T="*** ALGS - SYSTEM LINK EDIT ERROR ***"
.ENDIF
*****
***** RUN SYNONYM ASSIGNMENT PROG *****
*****
      XPS  LINK   = @$$DSCD.S$$SYSGEN.TEST.LINKMAP,
      INPUT = @$$DSCP.PATCH.MEMRES,
      OUTPUT = @$$DSCD.S$$SYSGEN.TEST.PATCHFIL,
      ERROR  = @$$DSCD.S$$SYSGEN.TEST.ERROR
.SYN $$CC = "$XPS$E"
.IF @$$CC,GT,00000
  .SHOW @$$DSCD.S$$SYSGEN.TEST.ERROR
.ENDIF
.IF @$$CC,GE,08000
  CM R=ME,M="*** ALGS - SYNONYM GENERATOR FATAL ERROR = @$$CC ***"
  .STOP T="*** ALGS - SYNONYM GENERATOR FATAL ERROR ***"
.ENDIF
.IF @$$CC,GT,00000
  CM R=ME,M="*** ALGS - SYNONYM GENERATOR NON-FATAL ERROR = @$$CC *
  .STOP T="*** ALGS - SYNONYM GENERATOR NON-FATAL ERROR ***"
.ENDIF
CM R=ME,M="*** ALGS - NORMAL TERMINATION ***"
.STOP T="*** ALGS - NORMAL TERMINATION ***"

```

### 3.6 PATCHING THE SYSTEM

After completion of ALGS, the patch file generated by ALGS must be applied to the new system before you can successfully perform an IPL. Issue the Patch Generated System (PGS) command to direct DX10 to automatically apply the patches to the system.

#### PATCH GENERATED SYSTEM

DATA DISK:  
TARGET DISK:  
SYSTEM NAME:  
BATCH LISTING:

Respond to the DATA DISK, TARGET DISK, and SYSTEM NAME prompts of this command with the same values as for the preceding ALGS command. The default value for the prompt BATCH LISTING is the file < data disk> .\$\$SYSGEN.< system name> .PGSLIST.

The output of the PGS command is a usable DX10 image and the SCI batch listing. The PGS command activates a batch SCI, which uses the patch file created by ALGS as input. The PGS process operates in background as ALGS does. If PGS terminates successfully, a successful completion message appears. After termination of PGS, the new DX10 image is ready to be installed and/or tested as the executing operating system.

PGS will write error messages to the batch listing. Consult Volume VI for a description of any error messages. See the following paragraph for instructions on PGS error recovery.

See Volume II for a description of the command and an explanation of the prompt responses. Appendix J lists an example patch file.

#### 3.6.1 PGS Error Recovery

PGS generates one of four possible messages. If you receive the following message, the PGS command was successful:

SYSTEM IS BOOTABLE

If you receive the following message, the error count was nonzero and the patches were unsuccessful. You need to reissue the PGS command.

SYSTEM IS NOT BOOTABLE

The following message may also appear:

SYSTEM ROOT EXCEEDS > C000

The DX10 system tasks are loaded in the system at address > C000. A system in which the root is greater than > C000 in length will overlap the system tasks causing unknown results.

The other message that may appear is as follows:

LONGEST OVERLAY PATH EXCEEDS > F800

If the longest overlay path exceeds > F800, the system extends into the TILINE Peripheral Control Space (TPCS). Improper execution and possibly disk data destruction will occur. The system root length must be shortened. This can be accomplished by making the system table area smaller (refer to paragraph 3.4.3) or by removing some of the devices or additional capabilities (user SVCs). The latter choice in most cases is unacceptable, leaving reduction of the system table area the only means of making the system root smaller. Appendix I can be used as an aid in determining the new system memory requirements.

### 3.6.2 Executing the Patch Synonym Assignment Program

Occasionally, when installing the operating system from linked object, you may find it desirable to execute the patch synonym assignment program without assembling and linking the system again. Accomplish this using the Execute Patch Synonym (XPS) command:

EXECUTE PATCH SYNONYM PROCESSOR

```
LINK      = [acnm]
INPUT     = [acnm]
OUTPUT    = [acnm]
ERROR     = [acnm]
```

Table 3-11 lists the XPS prompts. The synonym \$\$DSC\$ must be assigned the value of the disk drive name on which the system image is installed (that is, DS01 for the system disk).

**Table 3-11. XPS Command Prompt Responses**

Prompt	Response Type	Required or Optional	Default Value	Example
LINK	acnm	R	NONE	.\$\$SYSGEN.TEST.LINKMAP
INPUT	acnm	R	NONE	\$.PATCH.MEMRES
OUTPUT	acnm	R	NONE	.\$\$SYSGEN.TEST.PATCHFIL
ERROR	acnm	R	NONE	.\$\$SYSGEN.TEST.ERROR



## NOTE

The message DIFFERENT SYNONYMS FOR SAME OVERLAYS is an informative warning, not a fatal error.

### 3.6.3 Batch Stream Error Counter Errors

Errors encountered by commands in batch streams are collected by using the Batch Stream Error Counter (EC) command. This command tests the value of the synonym \$\$CC, which is set by calls to S\$STOP. When the value of \$\$CC is not zero, an error accumulation synonym, \$E\$C, is incremented by one. The value of \$E\$C may subsequently be displayed using the Create Message (CM) command.

### 3.6.4 Clear Secret Synonyms

Synonym table overflow errors occur when the synonym table reaches its capacity. System-defined synonyms and user synonyms share the same table area. Delete the system-defined secret synonyms using the Clear Secret Synonyms (Q\$SYN) command. Issue a Show File (SF) command on .S\$PROC.Q\$SYN to inspect the synonyms cleared by the Q\$SYN command.

## 3.7 TESTING THE SYSTEM

We strongly advise that you test the new system before installing it as the primary system. To test the new system, issue the TGS command. The following prompts appear on the screen:

```
TARGET DISK:  
SYSTEM NAME:
```

Respond to the prompts with the same values established in the preceding PGS command.

The TGS command designates the system under test to be the secondary system. The next time that an IPL occurs, this secondary system is loaded as the operating system. On subsequent IPLs, the primary system is loaded. This offers you an opportunity to ensure that the generated system will load successfully and function as designed. The following example displays a typical output from PGS using the imaginary systems S\$SUP and ITGS:

```
                IGS/TGS  
                SYSTEM IPL STATUS  
PAE5  
  
PRIMARY SYSTEM = S$SUP                SECONDARY SYSTEM = ITGS  
                TEST SECONDARY SYSTEM
```

If the secondary system does not act as expected, you can perform an IPL and the primary system will be loaded. Then you can repeat the system generation procedure for the system. If the secondary system functions properly, install it as the primary system using the IGS command as described in the following paragraph.

### 3.8 INSTALLING THE GENERATED SYSTEM

Issue the IGS command to install the system entered for the SYSTEM NAME prompt as the primary system. The prompts and responses are the same as for TGS:

TARGET DISK:  
SYSTEM NAME:

The following example displays a typical output of the IGS command:

```
                IGS/TGS
                PAE5   SYSTEM IPL STATUS
PRIMARY SYSTEM = ITGS                SECONDARY SYSTEM = $$$SUP
                IPL ON PRIMARY SYSTEM
```

# **System Generation Troubleshooting**

---

## **4.1 INTRODUCTION**

This section provides useful advice for the person responsible for maintaining a system. In particular, this section concerns rectifying a faulty system generation, but the advice is useful in troubleshooting any crash, if for no other reason, then to determine that the fault does not lie in the system generation.

## **4.2 MAINTAINING A SYSTEM LOGBOOK**

Make it a practice to keep a logbook of what happens to the system. Record all changes made to the system and all unexpected behavior. The following list contains typical entries to a logbook:

- New system generations
- Installation of patches
- Installation of new hardware
- Repairs to hardware
- Which disk packs were in use when disk and/or controller related problems were noticed
- Which disk packs were in use when the system crashes
- What the crash codes were
- Installation of new application software
- Modifications to application software
- When application problems occurred and what the error codes and messages were
- The resolution to any problem

Changes to the system are primary clues to the initial direction of investigation. Coincidence of problems with other operations on the system also gives important clues. Changes in system behavior without a change in software are important clues to possible hardware problems, and the error codes are the system's primary method of communicating its problems to you.

### 4.3 TROUBLESHOOTING SYSTEM GENERATION PROBLEMS

Problems with system generation fall generally into three classes:

- Problems with mistyped interrupts and controller addresses, which cause hangs and crashes
- Problems relating to the various components of system table area, their sizes, and how they affect the operation of the system
- Problems with operations, usually with getting the Assemble and Link Generated System (ALGS) and Patch Generated System (PGS) commands to execute properly

The text below is arranged by external symptoms. The titles are meant to match the visible manifestation of the problem. Find the paragraph title that most closely matches the symptom of your problem. If you cannot isolate a problem by the procedures below in a reasonable amount of time, obtain software help before calling for hardware service.

Sometimes a problem with a system generation may be coincident with a hardware problem, or a hardware problem may develop that manifests itself at the next initial program load (IPL). In the DX10 documentation, the primary characteristic of hardware problems is called *essential randomness* (defined in Section 8 of *DX10 Error Reporting and Recovery Manual, Volume VI*). One purpose of the following text is to define *essential coincidence* as it differs from essential randomness and to describe the essential coincidences that indicate problems with system generation.

#### 4.3.1 Crashes and Hangs

Crashes and hangs are abrupt indications that something may be wrong with the system generation. The following characteristics describe a crash:

- The system becomes inoperative; your data terminal does not respond to your input.
- The fault light on the front panel is lit.
- A crash code appears on the front panel.

The system is also inoperative during a hang. A hang differs from a crash in two ways:

- The fault light is not lit.
- No crash code appears on the front panel. The bar chart display of CPU and disk activity shows 100% activity or simply does not change at all.

The problem must be coincident with a new system generation to be associated with that system generation. However, if a device is defined during system generation but not accessed during IPL, a system generation problem with that device may not show itself until that device is accessed, but it will always appear when that device is first accessed. If the problem appears randomly, consult the troubleshooting guide in Volume VI.

If the crash always happens during IPL and is  $> 1F$  or less, the cause of the crash is a system generation problem. DX10 Releases 3.3 and earlier cannot have a crash code less than  $> 20$ . In these releases, issue the Analyze DX10 Crash File (XANAL) command to execute the ANALZ utility and use the General Information (GI) command to obtain the contents of the status register at the time of failure. Add one to the value of the rightmost digit to calculate the interrupt level that is assigned to something defined incorrectly during system generation. (If the result is 2, the problem is not system generation related.) When working with DX10 Release 3.4 and later, subtract  $> 10$  from the crash code to calculate the interrupt level assigned to something incorrectly defined during system generation. In Release 3.4 and later, the crash code cannot be less than  $> 13$ .

If the crash always occurs when a particular device is accessed, but not at IPL, the cause of the crash is probably due to a fault in the definition of that device during system generation. If it is not a system generation definition problem, the cause could be a miswired interrupt jumper plug in the chassis or a broken controller board.

Hangs occur when a device is accessed and the interrupt/address pair do not match. A hang appears during IPL on devices that interrupt when initialized by the power-up sequence of the device service routine (DSR), and at first access on other devices. If a system generation problem, it will always appear when the same device controller is accessed. Like the crash, if it is not a system generation problem, it is a miswired interrupt jumper plug or a broken controller board.

A hang requires some analysis to determine the interrupt level involved. With the CPU in hung condition, press HALT, then ST under DISPLAY. This causes the value of the status register to appear on the 16 data lights on the front panel. Note the value of bit 8. If bit 8 is on, the hang is in a task and the problem is not related to interrupts or addresses mistyped during system generation. If bit 8 is off, note the contents of bits 12 through 15. Press RUN, then repeat the process. A hang caused by a faulty system generation will show one interrupt level consistently. If other interrupt levels are displayed, sample the status register several times to determine which interrupt level occurs most frequently. One level should stand out clearly. Add one to the value of bits 12 through 15 to obtain the interrupt level of the problem device.

If one level does not stand out, the problem may not be a faulty system generation. Before checking for other problems, check on other devices. A lower level (higher priority) device may be interrupting frequently enough to superimpose its normal behavior on the problem level. This will cause the other interrupt level to also show up; you may want to investigate both levels for a system generation problem. A level consistently showing up as 4 may mean that there is a hardware malfunction in the realtime clock. Consistent appearance of level 15 with bit 8 set high means the hang is in a task. If the level shows 1 or below, the system's memory may have been changed affecting the system's ability to handle error conditions. In the latter case, it is often not possible to force a crash dump. Call your dealer for help with this software problem.

If the crash or hang occurs randomly after the system is up and running, and the device indicated by the analysis above has been successfully accessed—or at least accessed and a different error occurred—the problem is probably not the system generation. The cause of the problem is more likely to be a malfunctioning controller board or an intermittent problem in the interrupt wiring.

Sometimes the problem is at an interrupt level where an expansion chassis is defined. In this case, consult the troubleshooting guide in Volume VI. Refer to paragraph titled Continuous Interrupts to determine the chassis position of the problem device. Again, suspect a system generation problem only if the problem always happens during IPL or when a particular device is accessed.

A hang can also occur when two devices are defined during system generation to have the same interrupt (in the main chassis) or the same position (in an expansion chassis) where one device is in use and no controller board is installed for the other device. The solution is to do one of the following:

- Remove all controller boards that share the interrupts or position.
- Perform another system generation, defining only those controllers present.
- Install the missing controller board in the affected chassis.

#### **4.4 OTHER SYSTEM GENERATION PROBLEMS**

Occasionally, the following problems arise when running applications and/or system utilities whose solutions require system generation changes.

- Supervisor call (SVC) error > 05
- SVC error > 0E (to be distinguished from an error > 000E that can happen with a defective SCI command procedure that calls a COBOL program)
- SVC error > 89

##### **4.4.1 SVC Error > 05**

SVC error > 05 has three causes:

- The user program has made an SVC in which the call block itself, or the associated data buffer, extends outside the range of memory the task has for its use. This condition is usually regarded as a bug.
- The user program has made an Assign LUNO call with a pathname too long. This condition can arise in programs that construct their own pathnames, such as the Copy Directory (CD) utility. In that case, the problem is usually that an output disk volume has been used whose name is long enough for marginal pathnames to “spill over” and cause the problem.
- Physical record and/or logical record too long when accessing key indexed files (KIFs). Treat this error as you would SVC error > 89 since the cause is the same.

##### **4.4.2 SVC Error > 0E**

This error has only one cause: the additional I/O buffer area defined for the system during system generation is not sufficient to support the I/O that the system is being asked to handle. Fix this problem by enlarging that parameter during system generation. GEN990 (the System Generation utility) itself may not allocate sufficient I/O buffer area. The parameter ADDITIONAL I/O BUFFERS is added to what GEN990 calculated, and so increasing this parameter should fix the problem.

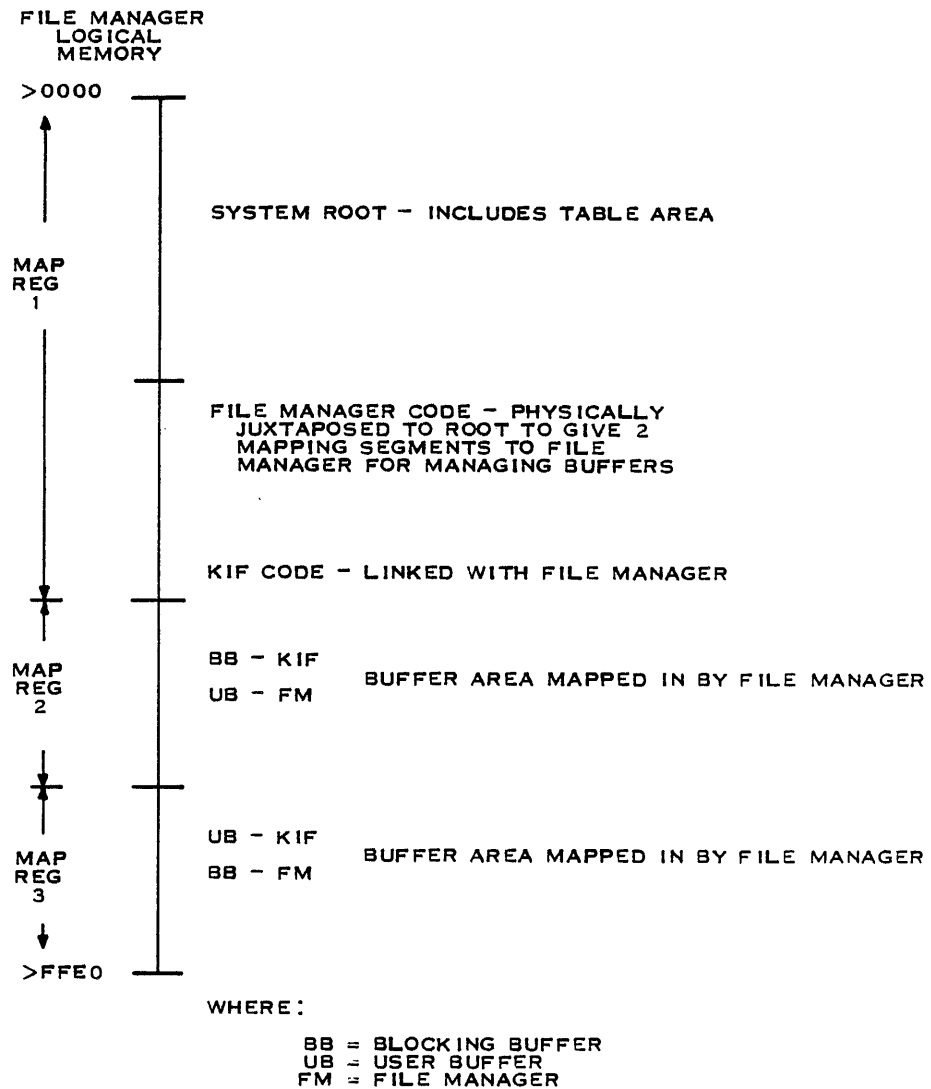
If the problem does not respond to this fix, or you have included communications packages in the system generation and the problem occurs when the communication package is inactive, something else is causing the > 0E error. Check the possible COBOL cause mentioned above and, if that is not the problem, obtain software help.

**4.4.3 SVC Error > 89**

This error is caused by having a combination of physical record length and logical record length too great for the file management processor to map into its address space. The following discussion explains Figure 4-1.

**NOTE**

Map reg 1 refers to the first of three segments, not to one of the map files of the CPU.



2283064

**Figure 4-1. Memory Used by File Management**

From Figure 4-1, it can be seen that the length of the system root directly affects the size of physical and logical records that the system can handle.

To calculate the amount of address space file management has to use for file blocking, inspect the system link map from .S\$\$SYSGEN and locate the phase with the module FILMG in it. Perform the following steps:

1. Add the length and origin of the phase together and round up to a beet boundary.
2. Add > 40 to the sum.
3. Subtract the result of step 2 from > FFE0. The difference is the blocking space.

The blocking space can be used for logical records and physical records of any size ratio so long as the sum of the two does not exceed blocking space. File management only maps in one blocking buffer and one logical record at any given time. Therefore, no “double blocking” occurs with small physical and logical records. In the event that blocking space is exceeded, you get a > 5 or > 89 error depending on which buffer is mapped into map segment 3. If the blocking buffer is mapped to map segment 3, an > 89 error occurs. If the user buffer is so mapped, a > 5 error occurs.

There are three basic ways to solve this problem:

- Reduce the physical record size of the file.
- Reduce the logical record size used by the program in its Read I/O SVC.
- Make the system root smaller (by reducing table area or eliminating devices and optional features).
- Make file management smaller by eliminating KIF.

#### **4.5 ALGS AND PGS PROBLEMS**

The most common operational problem with system generation is a Q\$\$SYN SCI command on the system that clears the synonyms used to pass information from the ALGS command or the PGS command to the batch stream that does the work. If those synonyms are being cleared, inspect the Q\$\$SYN command procedure, the BATCH command procedure, the Execute Batch (XB) command procedure, and the M\$00 command procedure for a .SYN or Assign Synonym (AS) command that clears the synonyms set by the ALGS or PGS commands. All of these are invoked during execution of the ALGS and PGS commands. Other problems are insufficient disk space to complete the system generation, the S\$\$SYSGEN directory is full, and the S\$IMAGES program file is full. The error messages returned by GEN990 and the Link Editor will indicate these problems.

If the error was in the macro assembler process, inspect the listing file given for the D\$DATA LISTING prompt and determine which statements failed. If a numeric error code appears, you can find an SVC error code in the two digits on the right. Check for missing files, full directory, full disk, and unassigned synonyms. These problems require you to provide the file, create space, or assign synonyms. If a macro expansion error occurs, determine if the macro at fault was supplied by TI or by the vendor of a non-TI software package. Contact the appropriate supplier or supplier’s representative for assistance.



If the error occurred during the Link Editor processing, inspect the file <data disk>.\$SYGEN.<system name>.LINKMAP to determine the errors. Check for the same things mentioned in the macro assembler process and also for Install Task and Install Overlay SVC errors. The message ADDRESS SPACE OVERFLOW means that the system is too large; in this case, the length of phase 0 may not be correct. Locate the longest phase 2 in the link map and add its origin to its length. Subtract >F800 from the result. Make the system table smaller by this number and perform another trial ALGS to see if further tuning is necessary.

#### 4.6 BUILDING A NEW SYSTEM DISK

Another common problem occurs when building a new disk. If you initialize the new disk using the Initialize New Volume (INV) command and accept the default value of NO for the USED AS SYSTEM DISK? prompt you cannot successfully use that disk as a system disk. If you specify such a disk as the TARGET DISK during system generation, errors occur that indicate that the directory .SYSGEN is missing, that the .S\$IMAGES program file is missing, or, if S\$IMAGES is present, something is wrong with it. Typically the latter appears as a Link Editor error. Create an S\$SYSGEN directory (it need contain no files); create S\$IMAGES and copy the DUMMY procedure from an operational S\$IMAGES. Other files must also be copied to make a running system. They currently are:

- S\$PROGA
- S\$LOADER
- S\$PROC
- S\$OVLYA
- S\$SDS\$ if the assembler, Link Editor, and other software installed there are needed



# How to Write a Device Service Routine

---

## 5.1 INTRODUCTION

DX10 supports a variety of peripheral devices by providing device service routines (DSRs) tailored to their requirements. If you wish to add a nonstandard device to your DX10 system and no standard DSR satisfies your needs, you must provide it with a DSR as well as the associated data structures and interrupt handling routines.

This section supplies the information you need for adding support for a nonstandard device to the DX10 operating system. It begins with a discussion of I/O request processing and the DSR entry points needed to handle the various types of interrupts. It then describes the data structures employed by the DSR and the coding conventions used for DX10 DSRs. The section then discusses the asynchronous DSR structure and the subroutines provided to simplify asynchronous DSR coding. It concludes with explanations of the common routines provided with DX10 to simplify DSR coding for both regular DSRs and asynchronous DSRs. Figure 5-22 contains two examples of DSRs.

### NOTE

There is no programmed limit on the number of special devices that can be defined to use a DSR. However, DX10 supports a maximum of 16 special DSRs. Each of the DSRs will be placed in its own map segment.

Before you begin, you should have a good understanding of the following topics:

- Hardware interface for the device.
- Interrupt handling. Refer to paragraph 5.2.5.
- Device data structures. Refer to paragraph 5.3.
- Computer hardware. Refer to the appropriate hardware manuals for your system.
- 990 assembly language. Refer to *990/10 and 990/12 Assembly Language Reference Manual*.
- DX10 system generation. Refer to Section 3.

## 5.2 DEVICE SERVICE ROUTINES (DSRs)

DSRs provide the interface between peripheral devices and tasks running on the 990 computer. A task initiates a device operation by performing an I/O supervisor call (SVC). The I/O supervisor transfers control to the DSR for the device, along with a pointer to information from the SVC. The DSR processes the SVC data, usually by sending an instruction to the device. Then it returns control to the I/O supervisor while waiting for the device to respond by generating a coded signal—an *interrupt*. When it receives an interrupt, the DX10 interrupt handler decodes the signal and transfers control back to the DSR. The DSR processes the interrupt and returns control to the I/O manager to wait for the next interrupt. After the DSR completes the SVC, it calls the ENDRCD routine (refer to paragraph 5.7.3), allowing the task to resume execution. Figure 5-1 illustrates this process.

When you write a DSR, you must provide routines to handle SVC calls and device interrupts. You must also supply routines for aborting device I/O and for initializing the data structures and/or the device during the initial program load (IPL). Optionally, you can include additional routines for internal use by the DSR. Figure 5-2 shows where to place the routines in the DSR and the following paragraphs explain their functions in detail. Subsequent discussions deal with the data structures used by the DSR, DSR coding conventions, and the common subroutines provided with DX10 to simplify DSR coding. The section concludes with two example DSRs.

### 5.2.1 I/O Call Routine

When the I/O supervisor receives an SVC for the device, it first makes a copy of the call block (buffers it) in the system table area. (Buffering is discussed in paragraphs 5.3.3 and 5.4.4.) Next it transfers control to the I/O Call routine in the DSR. This routine initiates the processing of the SVC, using the SVC data provided by the I/O supervisor. This usually involves decoding the SVC subopcode, taking the appropriate action, and returning control to the I/O supervisor. Typical subopcodes are Open, Close, Read, and Write. The characteristics of the device determine the actual processing required.

**5.2.1.1 Entry Point.** The I/O Call routine begins at the third word in the DSR. (The first and second words point to the Power Restored and I/O Abort routines.)

**5.2.1.2 Workspace.** The workspace used for the I/O Call routine is the physical device table (PDT) workspace for the device. This workspace contains the values shown in Figure 5-5 and described in paragraph 5.3.1. Using this information, you can access the SVC block for the call and address of the controller. You can use registers R5 through R11 for DSR working storage.

**5.2.1.3 Processing.** The operating system calls this routine with a BLWP instruction. That instruction always executes one instruction in the new context. (Consult the *990/10 and 990/12 Assembly Language Reference Manual* for a complete description of how BLWP works.) This instruction can be a LIM1 2 to mask all but level 1 interrupts preparatory to calling SETWPS, which sets the interrupt mask to the value defined in R2 of the PDT workspace. Prior to entry, the I/O supervisor sets the busy flag in the PDT (bit 1 of PDTDSF), which prevents entry to the DSR if another request for service occurs before the current one completes. The operating system queues such requests and then makes a new call to the DSR for all the queued requests.

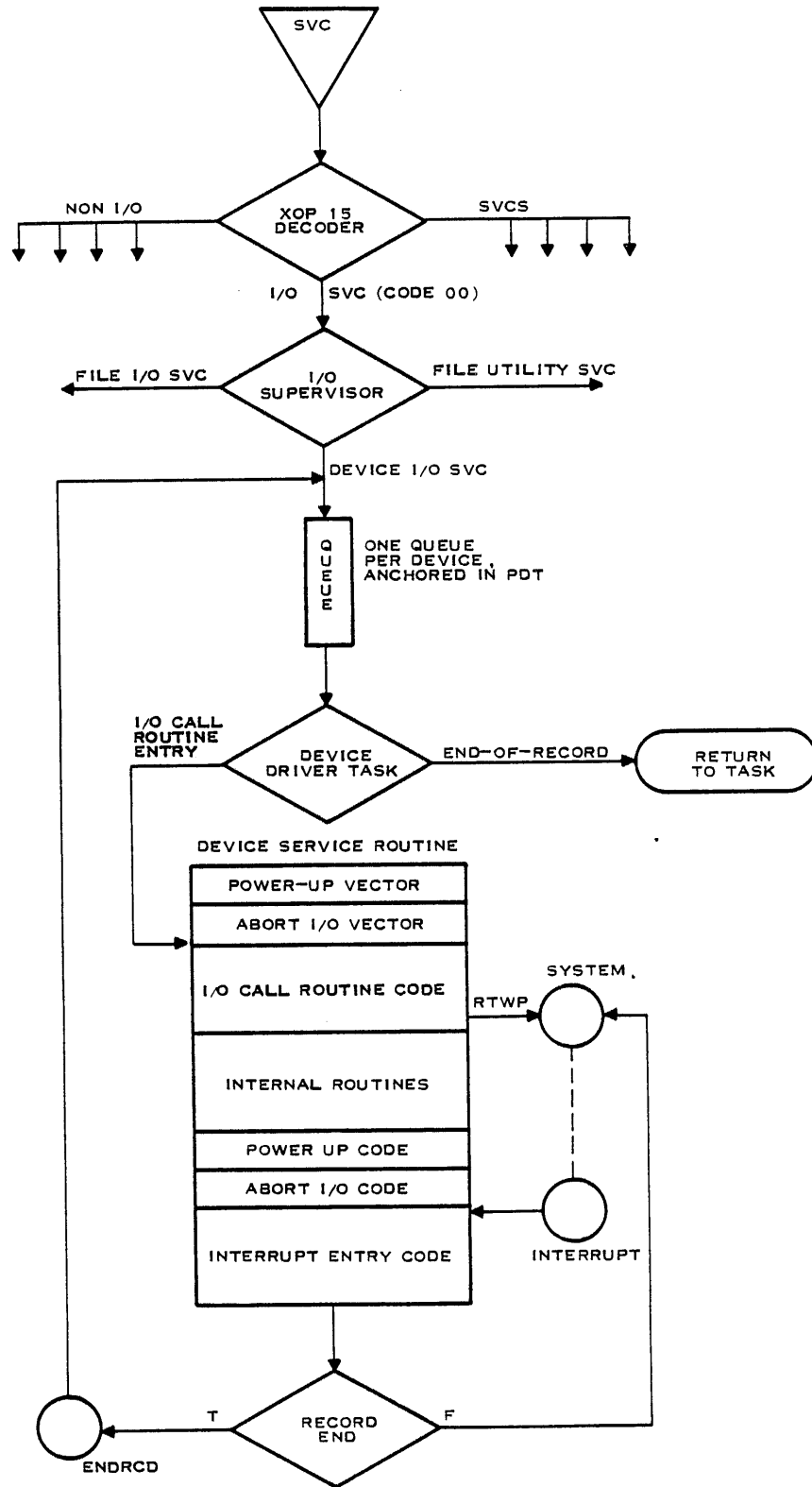
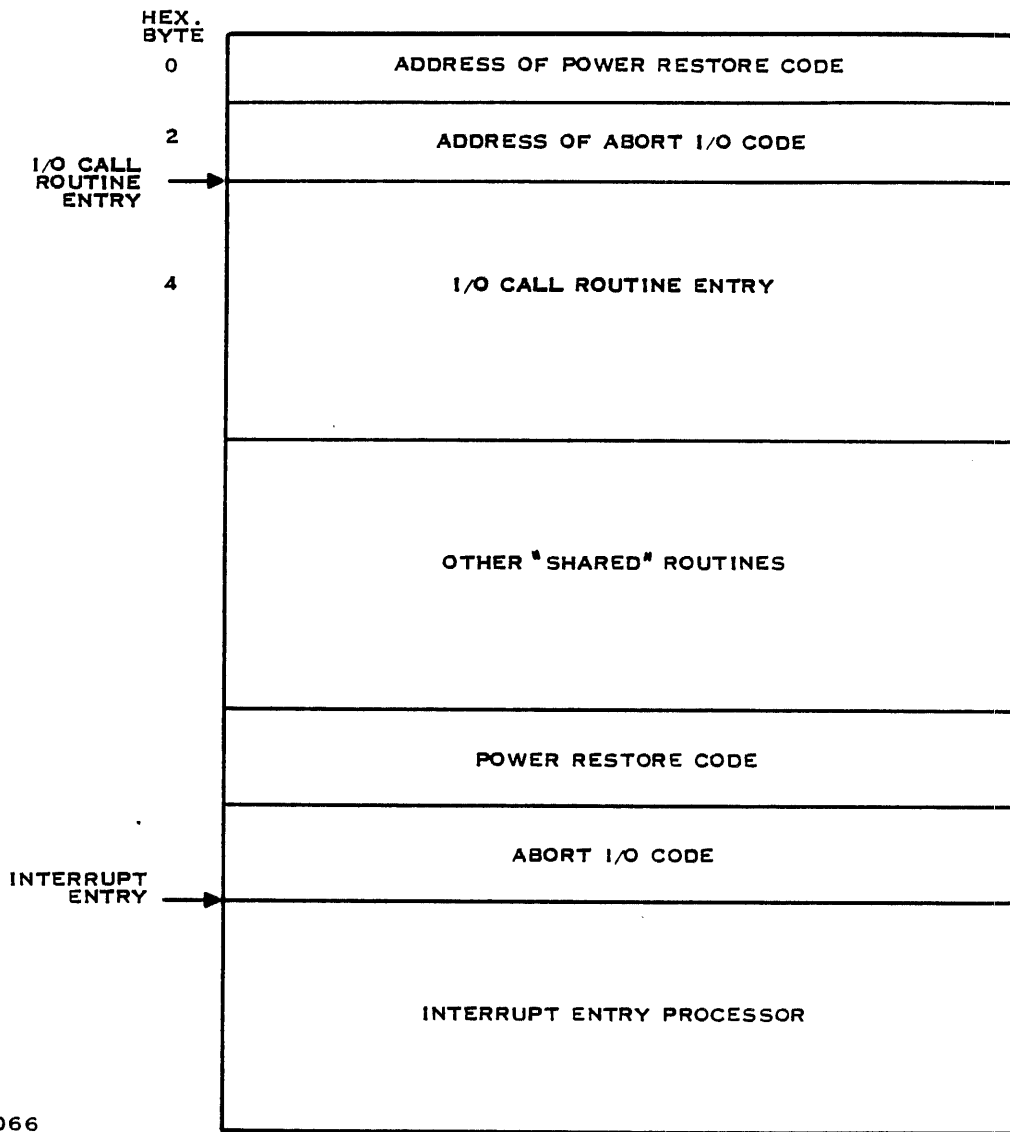


Figure 5-1. Flow of an I/O Call



2283066

Figure 5-2. DSR Structure

The value in R2 of the PDT workspace is defined during system generation by GEN990. You can choose to inhibit interrupts of varying levels, as noted below:

- You can use a mask value of 2, which inhibits all interrupts except power-down, power-up, and machine error (for example, illegal instruction). This is seldom necessary and is undesirable because all other interrupt activity of higher priority than necessary is also inhibited which could cause other devices to receive timing errors.
- A mask value of one less than the value defined for the device during system generation is used when the PDT workspace is also used for interrupt processing. This is the highest priority interrupt that must be masked to prevent the interrupt service routine (ISR) from being entered before initialization is complete.
- A mask value of > F allows all interrupts. This allows all other interrupts to be serviced, and may be useful because it will not cause timing errors to appear on any other device. If you elect to use this mask value, you allow the ISR to be entered at any time, including when the DSR may be processing an I/O SVC call. You must take precautions to ensure that the DSR and the ISR do not access their common data structures in a conflicting manner. This usually requires inhibiting interrupts to a level one less than the level the device is connected to during the time common data is being accessed. A DSR written to use this convention must use an interrupt workspace that is separate from the PDT workspace, usually the keyboard status block (KSB).
- A response of NONE to the INTERRUPT? prompt during system generation causes a mask value of > F to be generated. You should use it for a device that does not generate interrupts.

If the SVC contains a subopcode (sometimes called the I/O opcode), you can use the BRCALL or JMCALL routines to process it. BRCALL (paragraph 5.7.1) employs a branch table to transfer control to the appropriate routine within the DSR. JMCALL (paragraph 5.7.6) uses a jump table for the same purpose. When you use these routines, you must provide the branch table or jump table immediately after the call. The remainder of the I/O Call routine can consist of processors for the individual subopcodes and for error conditions. The I/O Call routine should terminate by executing an RTWP instruction.

### 5.2.2 Shared Routines

Following the I/O Call routine and subopcode processors, you might want to include one or more common routines. These shared routines can perform operations common to the subopcode processors or the other required routines.

### 5.2.3 Power Restored Routine

The Power Restored routine handles device initialization after the loading of DX10 or a power loss and recovery. In particular, it should call the I/O Abort routine whenever it finds an I/O request was pending at the time of a power failure. As shown in Figure 5-2, the Power Restore routine follows any shared routines included in the DSR.

**5.2.3.1 Entry Point.** The address of the Power Restored routine must be defined in the first word of the DSR. You do this by coding a DATA statement with the label of the first instruction in the Power Restored routine as the operand.

**5.2.3.2 Workspace.** When the routine is called, the workspace pointer (WP) register points to the PDT workspace for the device. You can use registers R5 through R11 in the PDT as scratch registers.

**5.2.3.3 Processing.** The Power Restored routine has two main responsibilities. First, it must initialize the device following a system load or power-up. How you do this depends on the characteristics of the device. Second, it must abort I/O when an I/O request was pending at the time of a power failure. You can use the common system routine, BZYCHK, to determine whether an I/O request was pending. If so, the Power Restored routine should call the DSR I/O Abort routine. If not, the routine should return control with an RTWP instruction.

#### **5.2.4 I/O Abort Routine**

When DX10 times out a device or executes an Abort I/O SVC, the I/O supervisor removes all physical record blocks (PRBs) from the queue for the device. Then it transfers control to the DSR at the entry point for the I/O Abort routine. This routine terminates the current I/O request and places the appropriate error code in the SVC block.

**5.2.4.1 Entry Point.** The address of the I/O Abort routine must be defined in the second word of the DSR, immediately after the address of the Power Restored entry point. You code it using a DATA statement with the label of the first instruction in the I/O Abort routine as the operand.

**5.2.4.2 Workspace.** When the routine is called, the WP register points to the PDT workspace for the device. You can use R5 through R11 as scratch registers.

**5.2.4.3 Processing.** The I/O Abort routine has the responsibility of terminating the processing of the request and posting an appropriate return code in the buffered I/O request block. It should also perform whatever initialization of the device is necessary to restore it to a known state.

If necessary, you can use the communications register unit (CRU) or TILINE address in R12 of the PDT workspace to inspect any status bits that may be available from the device. The error code returned to the calling task goes in PRBEC of the buffered I/O request block, which you address as @PRBEC(R1). The routine should report device errors to the system log by placing a nonzero error code in PDTERR of the PDT. It should also record the status of the device by setting bit 14, the Operation Failed flag, in PDTFLG. The routine should conclude with an RTWP instruction to return control to the I/O supervisor.



### 5.2.5 Interrupt Service Routine (ISR)

The 990 computer uses a vector table to handle up to 16 levels of interrupts. When the interrupt hardware detects an interrupt, it uses the table to transfer control to the interrupt handler or decoder (created by GEN990 during system generation) that processes the interrupt. Each transfer vector consists of two 16-bit words. The first contains the address of the workspace area for the interrupt, and the second contains the address of the entry point of the interrupt handler that services the interrupt. The computer hardware automatically uses map file 0 to access the interrupt vector.

Interrupt levels range from 0 through 15, with 0 being the highest priority and 15 the lowest. Bits 12 through 15 of the status register (ST) form an interrupt mask, whose value determines the lowest priority (highest numbered level) interrupt allowed. The 990 computer continuously compares the level of the highest pending interrupt request to the current value of the interrupt mask, honoring only interrupts with equal or lower levels than the mask. The processor hardware always finishes the current instruction before transferring control to the ISR.

When the CPU has a pending interrupt with an equal or lower level than the interrupt mask, it saves the current WP, program counter (PC), and ST registers. It changes to map file 0, subtracts one from the level of the interrupt, and places the result in the interrupt mask of the ST. Then, it multiplies the interrupt level by 4 to compute the memory address of the interrupt vector. It stores the WP, PC, and ST registers at the time of the interrupt in R13, R14, and R15 of the workspace pointed to by the WP component of the interrupt vector. Then, it loads the WP and PC registers with the values in the interrupt vector and executes one instruction regardless of any pending interrupt or the value of the interrupt mask. Then, it resumes normal execution. This process transfers control to the interrupt decoder built by GEN990. The interrupt decoder determines which interrupt workspace and entry address are associated with the device and enters the DSR using a BLWP instruction. Interrupt processing proceeds, and when it is complete, the ISR returns with an RTWP instruction. The interrupt decoder then enters the interrupt cleanup routine in DX10, known as trap return.

GEN990 produces both the interrupt vector table and the interrupt decoder, according to your responses to its device definition prompts. The interrupt decoder takes care of such things as shared interrupts and expansion chassis interrupts. You can provide your own vector table and interrupt decoder, if you prefer, but you will also have to replace the ones built by GEN990. To do this, you can tailor the source file built by GEN990 (.S\$\$SYSGEN.< sysname >.D\$\$SOURCE) to the requirements of your device. It is usually better to let GEN990 produce the interrupt table and then perform additional decoding in the ISR. Table 5-1 provides an example of an interrupt vector table.

**Table 5-1. Example Interrupt Vector Table**

Memory Address	Interrupt Vector	Vector Contents	Typical Assignment
0000	0	WP address for interrupt 0	Power On <sup>1</sup>
0002	0	PC address for interrupt 0	
0004	1	WP address for interrupt 1	Power Falling <sup>2</sup>
0006	1	PC address for interrupt 1	
0008	2	WP address for interrupt 2	Error <sup>2</sup>
000A	2	PC address for interrupt 2	
000C	3	WP address for interrupt 3	External Device
000E	3	PC address for interrupt 3	(Communications) <sup>3</sup>
0010	4	WP address for interrupt 4	External Device
0012	4	PC address for interrupt 4	(Card Reader) <sup>3</sup>
0014	5	WP address for interrupt 5	Line Frequency Clock (optional) <sup>3</sup>
0016	5	PC address for interrupt 5	
0018	6	WP address for interrupt 6	External Device
001A	6	PC address for interrupt 6	
001C	7	WP address for interrupt 7	External Device
001E	7	PC address for interrupt 7	
0020	8	WP address for interrupt 8	External Device
0022	8	PC address for interrupt 8	
0024	9	WP address for interrupt 9	External Device
0026	9	PC address for interrupt 9	(Mag Tape) <sup>3</sup>
0028	10	WP address for interrupt 10	External Device
002A	10	PC address for interrupt 10	(CRT) <sup>3</sup>
002C	11	WP address for interrupt 11	External Device
002E	11	PC address for interrupt 11	(CRT) <sup>3</sup>
0030	12	WP address for interrupt 12	External Device
0032	12	PC address for interrupt 12	
0034	13	WP address for interrupt 13	External Device
0036	13	PC address for interrupt 13	(Disk) <sup>3</sup>
0038	14	WP address for interrupt 14	External Device
003A	14	PC address for interrupt 14	
003C	15	WP address for interrupt 15	External Device
003E	15	PC address for interrupt 15	

**Notes:**

- <sup>1</sup> Level 0 is always the Power On/Up interrupt.
- <sup>2</sup> Predefined interrupts in the 990/10 and 990/12.
- <sup>3</sup> Typical interrupt levels for standard devices.

**5.2.5.1 Entry Point.** GEN990 issues the INTERRUPT ENTRY prompt to obtain the entry point of the ISR for the device being defined. It places the address of the ISR in the tables that the interrupt handler (decoder) uses to enter the ISR after it determines which DSR is to service the interrupt.

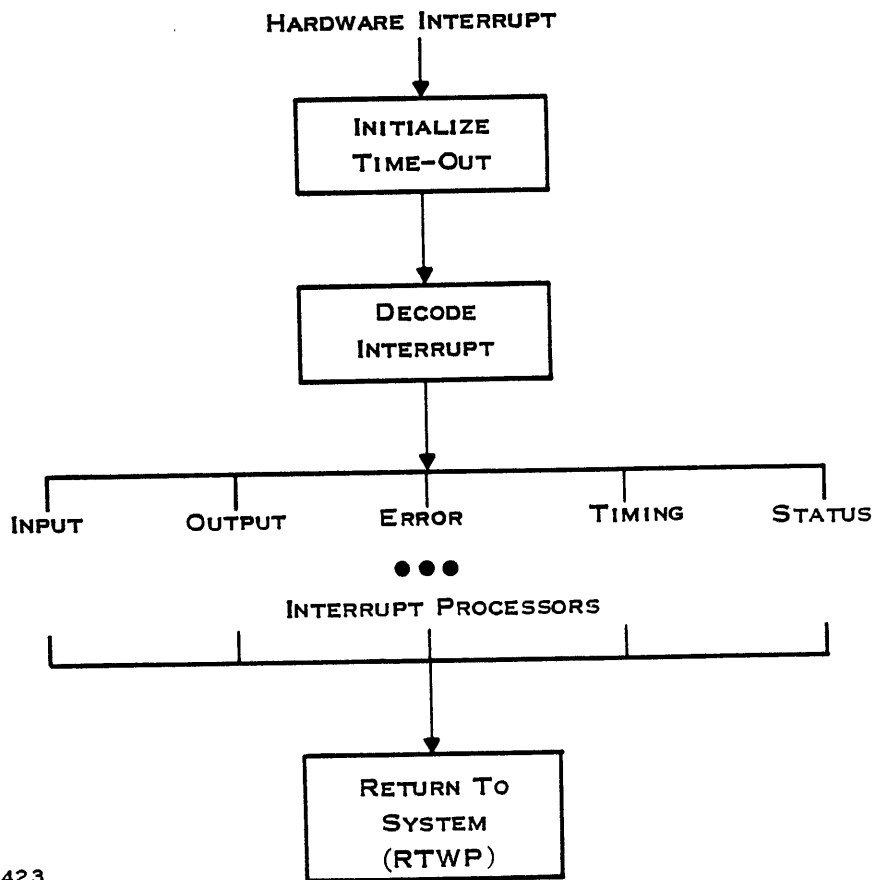
The routine beginning at the specified entry point must handle all interrupts for the device.

**5.2.5.2 Workspace.** GEN990 issues the DSR WORKSPACE prompt to obtain the location of the workspace for the ISR, usually PDT workspace in bytes 4 through 23 of the PDT. R5 through R11 are available for use by the DSR. You must not destroy the contents of the other PDT workspace registers.

For multiple device controllers, the ISR should use a separate workspace for decoding the interrupts instead of the PDT workspace. This workspace can be part of an extension to the PDT for the first device (which you define to suit the needs of the device). The workspace can also contain pointers to the PDT workspaces used by the DSR if needed.

For DSRs that service keyboard devices, the ISR should use the workspace in the KSB (paragraph 5.3.2) extension to the PDT. Some of the common routines (paragraph 5.7) used by keyboard DSRs use the KSB workspace.

**5.2.5.3 Processing.** Figure 5-3 illustrates the processing done by the ISR. You should design the ISR code to be reentrant, so that it can serve multiple devices of the same type. All data that the ISR changes must be in the PDT or its extension for the device being serviced. The ISR must service each interrupt very quickly.



2279423

Figure 5-3. ISR Processing

The ISR should begin by resetting the time-out counter for the device. To do this, move PDTTM1 to PDTTM2. You should do this even if you do not choose the time-out option during system generation.

Complex devices, such as interactive terminals or multiple-device controllers, require special treatment:

- The Keyboard ISR must be able to handle unsolicited characters as well as those generated in response to an I/O SVC. By using the routine PUTCBF, the system places characters into the KSB character queue. Event characters are handled the same way using the routine PUTEBF.
- If the characters read from the device interface do not fit the function table in Volume III, or the event/data character ranges defined for GETC, the ISR may need to transform those characters into character codes compatible with those functions. If so, it should perform the transformation before calling PUTCBF or PUTEBF. These character functions and character ranges are set by convention and you need to hold to the convention only if you need to use other parts of DX10 (such as SCI) that use that convention.
- For multiple-device controllers, the ISR should first use the workspace in the extension to the PDT to determine which device generated the interrupt. Then, use a BLWP instruction to enter the DSR with the workspace of the PDT for that device and process the interrupt. It should conclude with a RTWP instruction to return to the interrupt decoder. Figure 5-4 shows a specific configuration for a multiple disk drive controller.

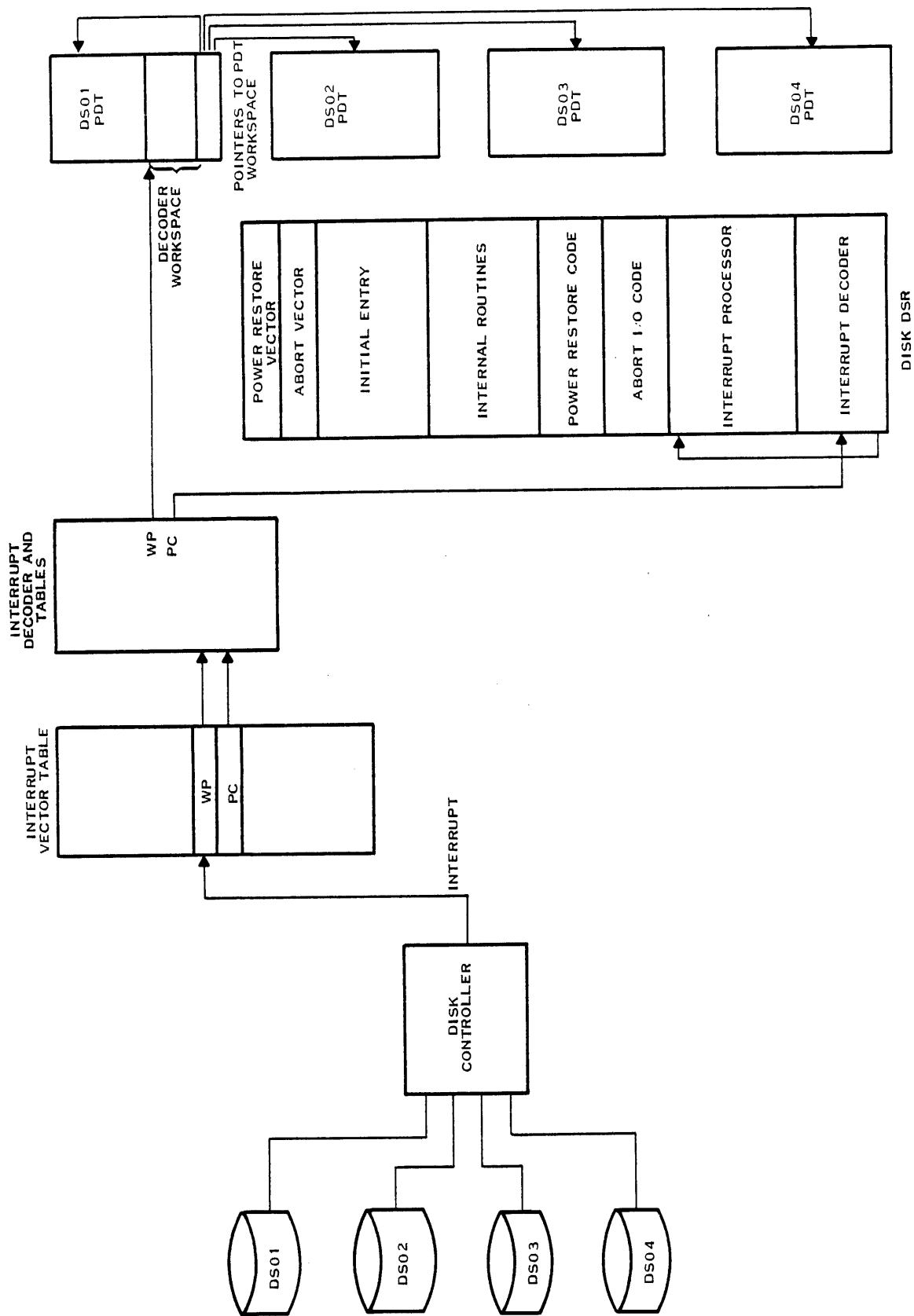


Figure 5-4. PDT and DSR Configuration for Multiple Disk Drive Controller

### 5.3 DATA STRUCTURES

The DSR uses several data structures maintained by DX10. These data structures can be found in `.$SYSGEN.SYSTEM.TABLES`. This paragraph describes the contents and use of these data structures:

- Physical device table (PDT) — Characteristics and status of a device, and some working storage.
- Keyboard status block (KSB) — Supplement to the PDT for keyboard devices.
- PDT workspace — I/O call information passed to the DSR. Do not depend on the PDT workspace registers to maintain status of a device from one I/O call to the next.
- Buffered I/O request block — Information from the I/O SVC block available to the DSR.
- Task status block (TSB) — All the information the system needs to know about tasks.
- Extension for terminals with keyboards (XTK) — Extension used by TI DSRs for terminals. It functions as an extension to the KSB.
- Asynchronous DSR local PDT extension — The local (system table area) PDT extension for the asynchronous DSR.
- Asynchronous DSR long-distance device extension — A second PDT extension area accessed via long-distance instructions by the asynchronous DSR.
- Any extension you may define to contain data related to the device.

This section does not provide details on the entire I/O SVC block, since it receives a thorough explanation in the Volume III.

#### 5.3.1 Physical Device Table (PDT)

The PDT represents a device to the DX10 operating system. It describes the current state and characteristics of the device, provides a workspace for the DSR, and holds the queue anchors for I/O requests for the device. Figure 5-5 shows the structure of the PDT. For a description of the PDT extensions for standard devices, refer to the *DX10 System Design Document*.

You must provide a PDT for each device you add to the system.

Some of the fields are reserved and receive their initial values from GEN990 or the I/O scheduler. You must provide the initial values for other fields. You must provide space for all the fields in the PDT, since the operating system accesses data by its position from the beginning of the PDT. Table 5-2 describes the PDT fields.

HEX BYTE				
>00		PDTLNK -- FORWARD LINK TO NEXT PDT		
>02		PDTMAP -- POINTER TO DSR MAP FILE		
>04	R0	PDTR0 -- DSR SCRATCH		
>06	R1	PDTPRB -- PRB ADDRESS		
>08	R2	PDTDSF -- DEVICE STATUS FLAGS		
>0A	R3	PDTDTF -- DEVICE TYPE FLAGS		
>0C	R4	PDTDIB -- DEVICE INFO BLOCK ADDRESS		
>0E	R5	DSR SCRATCH		
>10	R6			
>12	R7			
>14	R8			
>16	R9			
>18	R10			
>1A	R11	PDTR11		
>1C	R12	PDTCRU -- CRU OR TILINE BASE ADDRESS		
>1E	R13	PDTR13 -- SAVED WP REGISTER		
>20	R14	PDTR14 -- SAVED PC REGISTER		
>22	R15	PDTR15 -- SAVED ST REGISTER		
>24		PDT\$ -- PDT WORKSPACE ADDRESS		
>26		PDTDSR -- DSR ADDRESS		
>28		<table border="1" style="width:100%"> <tr> <td style="width:50%">PDTErr -- ERROR CODE</td> <td style="width:50%">PDTFLG -- FLAGS</td> </tr> </table>	PDTErr -- ERROR CODE	PDTFLG -- FLAGS
PDTErr -- ERROR CODE	PDTFLG -- FLAGS			
>2A		PDTNAM -- DEVICE NAME		
>2E		PDTSL1 -- SYSTEM LOG 1		
>30		PDTSL2 -- SYSTEM LOG 2		
>32		PDTBUF -- NOT USED		
>34		PDTBLN -- BUFFER LENGTH		
>36		PDTINT -- DSR REENTER-ME ADDRESS		
>38		PDTDVQ -- DEVICE QUEUE ANCHOR		
>42		PDTTM1 -- TIME OUT COUNT 1		
>44		PDTTM2 -- TIME OUT COUNT 2		
>46		PDTSRB -- SAVED PRB ADDRESS		
>48		PDTFQL -- PRIORITY DSR SCHEDULE QUEUE WORD		

2283068

Figure 5-5. PDT Structure

**Table 5-2. PDT Values**

Hex. Byte	Field Name	Description																		
> 00	PDTLNK	Address of the next PDT and the PDT expansion block for this PDT. All the PDTs are linked in a single list that is located in the D\$DATA module.																		
> 02	PDTMAP	Address of the DSR map file.																		
> 04	PDTR0	This word begins the PDT workspace to be used by the DSR I/O Call routine.																		
> 06	R1 PDTPRB	Address of I/O opcode byte in buffered I/O SVC block. This register is updated prior to DSR entry from the I/O subsystem.																		
> 08	R2 PDTDSF	Device status flags that are set by the system (bit 5 set by the DSR):																		
		<table border="1"> <thead> <tr> <th>Bit</th> <th>Meaning When Set</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Device is opened; that is, LUNOs are open to the device if it is file oriented.</td> </tr> <tr> <td>1</td> <td>DSR is busy (that is, it is processing an I/O request).</td> </tr> <tr> <td>2</td> <td>Kill I/O at this device is in progress.</td> </tr> <tr> <td>3</td> <td>Task doing I/O at this device is being killed.</td> </tr> <tr> <td>4</td> <td>Make this device available (unassigned) at the end of this I/O operation.</td> </tr> <tr> <td>5</td> <td>Signals the task scheduler to reenter the DSR (at the next time slice). A DSR can use this flag to wait for a device by setting the flag and then returning to the system.</td> </tr> <tr> <td>6</td> <td>End-of-record processing needs to be done for this PDT (data transfer is complete).</td> </tr> <tr> <td>7</td> <td>0 = ASCII. 1 = JISCII.</td> </tr> </tbody> </table>	Bit	Meaning When Set	0	Device is opened; that is, LUNOs are open to the device if it is file oriented.	1	DSR is busy (that is, it is processing an I/O request).	2	Kill I/O at this device is in progress.	3	Task doing I/O at this device is being killed.	4	Make this device available (unassigned) at the end of this I/O operation.	5	Signals the task scheduler to reenter the DSR (at the next time slice). A DSR can use this flag to wait for a device by setting the flag and then returning to the system.	6	End-of-record processing needs to be done for this PDT (data transfer is complete).	7	0 = ASCII. 1 = JISCII.
Bit	Meaning When Set																			
0	Device is opened; that is, LUNOs are open to the device if it is file oriented.																			
1	DSR is busy (that is, it is processing an I/O request).																			
2	Kill I/O at this device is in progress.																			
3	Task doing I/O at this device is being killed.																			
4	Make this device available (unassigned) at the end of this I/O operation.																			
5	Signals the task scheduler to reenter the DSR (at the next time slice). A DSR can use this flag to wait for a device by setting the flag and then returning to the system.																			
6	End-of-record processing needs to be done for this PDT (data transfer is complete).																			
7	0 = ASCII. 1 = JISCII.																			
	DSFASG																			
	DSFBSY																			
	DSFINT																			
	DSFKLL																			
	DSFCLS																			
	DSFREN																			
	DSFEOR																			
	DSFJIS																			



Table 5-2. PDT Values (Continued)

Hex. Byte	Field Name	Description																		
>09	—	Interrupt mask to be used by request entries to the DSR. This field is set during system generation or when the PDT is constructed.																		
>0A	R3 PDTDTF	Device type flags that are all set at system generation time, except for the system disk flag that is set by the system loader:																		
		<table border="1"> <thead> <tr> <th>Bit</th> <th>Meaning When Set</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>DTFFIL File-oriented device (if zero, the device is record oriented).</td> </tr> <tr> <td>1</td> <td>DTFTIL The DSR accesses the data buffer directly. The calling task is not rolled out. The buffer must be accessed with LDD and LDS instructions. This bit is also called the TILINE I/O flag.</td> </tr> <tr> <td>2</td> <td>DTFTIM The time-out logic should be enabled for this device. This flag can also be used by the DSR.</td> </tr> <tr> <td>3</td> <td>DTFPRI Device can only be used by privileged tasks.</td> </tr> <tr> <td>4</td> <td>DTFKSB This is a terminal (keyboard device) with a keyboard status block attached to the PDT.</td> </tr> <tr> <td>5</td> <td>DTFCOM This is a standard communications device.</td> </tr> <tr> <td>6</td> <td>DTFSYD This is the system disk.</td> </tr> <tr> <td>7</td> <td>DTFEXT A PDT extension exists. This bit is always set when bit 4 is set. It is set when bit 4 is not set to indicate that a non-KSB extension is present.</td> </tr> </tbody> </table>	Bit	Meaning When Set	0	DTFFIL File-oriented device (if zero, the device is record oriented).	1	DTFTIL The DSR accesses the data buffer directly. The calling task is not rolled out. The buffer must be accessed with LDD and LDS instructions. This bit is also called the TILINE I/O flag.	2	DTFTIM The time-out logic should be enabled for this device. This flag can also be used by the DSR.	3	DTFPRI Device can only be used by privileged tasks.	4	DTFKSB This is a terminal (keyboard device) with a keyboard status block attached to the PDT.	5	DTFCOM This is a standard communications device.	6	DTFSYD This is the system disk.	7	DTFEXT A PDT extension exists. This bit is always set when bit 4 is set. It is set when bit 4 is not set to indicate that a non-KSB extension is present.
Bit	Meaning When Set																			
0	DTFFIL File-oriented device (if zero, the device is record oriented).																			
1	DTFTIL The DSR accesses the data buffer directly. The calling task is not rolled out. The buffer must be accessed with LDD and LDS instructions. This bit is also called the TILINE I/O flag.																			
2	DTFTIM The time-out logic should be enabled for this device. This flag can also be used by the DSR.																			
3	DTFPRI Device can only be used by privileged tasks.																			
4	DTFKSB This is a terminal (keyboard device) with a keyboard status block attached to the PDT.																			
5	DTFCOM This is a standard communications device.																			
6	DTFSYD This is the system disk.																			
7	DTFEXT A PDT extension exists. This bit is always set when bit 4 is set. It is set when bit 4 is not set to indicate that a non-KSB extension is present.																			

Table 5-2. PDT Values (Continued)

Hex. Byte	Field Name	Description
		<p><b>Bit</b>                      <b>Meaning When Set</b></p> <p>8 – 11      Not used.</p> <p>12 – 15      Device type code, returned when the device is opened for I/O and returned for a Read Characteristics SVC. Device codes other than the ones in the following list might cause an error in some language packages (such as FORTRAN and COBOL). The number in parentheses after the device type name is the number of status bytes used when a log message is created.</p> <p>   0 — Dummy (0)     1 — Teleprinter (2)     2 — Line Printer (2)     3 — Cassette (2)     4 — Card Reader (2)     5 — Video Display Terminal (VDT) (4)     6 — Disk (16)     7 — Communications (4)     8 — Magnetic Tape (16)     E — AMPL Emulator (undefined)     F — AMPL Trace Module (undefined)</p>
> 0C	R4    PDTDIB	Pointer to the word after the PDT itself. This can be a disk PDT extension (DPD), tape PDT extension (TPD), line printer PDT extension (LPD), or your own extension, depending on the type of device. If a KSB is present, it must be the address of the KSB.
> 0E	PDTR5 through PDTR11	Scratch registers to be used by the DSR.
> 1C	PDTCRU	The CRU or TILINE address of the device.
> 1E	PDTR13 PDTR14 PDTR15	These three words contain the saved context (WP, PC, ST) to which the DSR returns control via a RTWP.
> 24	PDT\$	Pointer to the beginning of the PDT workspace (byte 4 of the PDT).
> 26	PDTDSR	A pointer to the first word of the DSR.

Table 5-2. PDT Values (Continued)

Hex. Byte	Field Name	Description
> 28	PDTERR	Error code returned by the DSR.
> 29	PDTFLG	Device flags as follows:
		<b>Bit</b> <b>Meaning When Set</b>
	DFGPRB	8      Use the buffered I/O request block in log message. If not set, PDTSL1 and PDTSL2 are used.
	DFGJAR	9      Receive mode for JISCI. Used by standard DSR to handle JISCI.
	DFGJAT	10     Transmit mode for JISCI. Used by standard DSR to handle JISCI.
	DFGSTA	11 – 12    Device state: online = 00, offline = 01, diagnostic = 10. Set by DX10.
	DFGOPF	14      Operation failed bit. Set by the DSR to cause the system to indicate a failure in the system log.
> 2A	PDTNAM	The 4-character device name. GEN990 renames devices said to have a KSB, STXX, where XX is the terminal ID.
> 2E	PDTSL1, PDTSL2	For CRU devices (indirect access), these words contain the controller image after an error. When you specify indirect access, the system log processor formats one or two words. For TILINE controllers (direct access), the first word contains the memory address of the controller image after an error. When you specify direct access, the system log processor goes to the address and formats the status bytes. The number of status bytes depends on the type of device (refer to the list under bits 12 – 15 of PDTDTF).
> 32	PDTBUF	Not used.
> 34	PDTBLN	Maximum length of a data buffer that can be transferred by the device in an I/O operation (for example, 80 for a special device). This is only necessary for indirect buffer access (DTFTIL set to 0). The size of the data buffer times the number of initiate I/O requests (maximum of 5) should be added to the response given to the I/O BUFFER prompt from GEN990.
> 36	PDTINT	The reenter-me address for the DSR.
> 38	PDTDVQ	The anchor for the queue of I/O requests for this device.

Table 5-2. PDT Values (Continued)

Hex. Byte	Field Name	Description
> 42	PDTTM1	The number of system time units in the time-out count for the device.
> 44	PDTTM2	The number of time units remaining in the time-out count before the system assumes that a device error has occurred. The DSR signals the operating system to perform time-out processing by setting the time-out enable flag in PDTDTF to one. When the DSR starts an I/O operation and after every interrupt, the DSR should move the time-out count in PDTTM1 to this word (PDTTM2). The scheduler then uses this word as the time-out counter. Each time a system time unit has elapsed, the scheduler decrements the time-out count in this word. If the counter goes to zero, the system assumes that a device error has occurred and enters the DSR at the Abort I/O entry as indicated for the ABORT processing.
> 46	PDTSRB	The address of the queued SVC block plus two. This is a copy of R1 as it was at the I/O Call routine entry to the DSR. See paragraphs 5.3.1 and 5.7.3.
> 48	PDTFQL	DSR priority schedule queue word.

Figure 5-6 shows a template for the PDT. You can probably tailor one of the PDTs in `.$SYSGEN.<sysname>.$SOURCE` to suit your device. After you write your PDT (in 990 assembly language), you must include it in the system. To do this, you give the pathname of its source file in response to the PDT FILE prompt issued by GEN990. Refer to paragraph 3.4.5.10.

```

*****
*   PHYSICAL DEVICE TABLE      (PDT)  05/20/80
*****
      DORG 0
PDTLNK BSS 2          FORWARD LINKAGE TO NEXT PDT
*   THERE IS AN EXPANSION BLOCK BEFORE THE NEXT PDT.
*   TO REFERENCE IT USE THE VALUE IN PDTLNK PLUS
*   ONE OF THE FOLLOWING OFFSETS.  WHEN PDTLNK IS
*   ZERO (DS01) USE THE VALUE IN PDTLST (GLOBAL
*   VARIABLE IN ROOT) .
PDTRED EQU -8        NUMBER OF READS
PDTWRT EQU -6        NUMBER OF WRITES
PDTOTH EQU -4        NUMBER OF OTHERS
PDTRTY EQU -2        NUMBER OF RETRIES
PDTLUN EQU -1        NUMBER OF LUNOS ASSIGNED
*
PDTMAP BSS 2          POINTER TO DSR MAP FILE
PDTR0 BSS 2          R0 - DSR SCRATCH
PDTPRB DATA $-$$    R1 - QUEUED PRB ADDRESS
PDTDSF BSS 2          R2 - DEVICE STATUS FLAGS
DSFASG EQU 0          ASSIGNED
DSFBSY EQU 1          BUSY
DSFINT EQU 2          KILL I/O IN PROGRESS
DSFKLL EQU 3          KILL TASK IN PROGRESS
DSFCLS EQU 4          CLOSE OUT
DSFREN EQU 5          RE-ENTER-ME
DSFEOR EQU 6          END-RECORD
DSFJIS EQU 7          JISCII FLAG (KATAKANA)
PDTDTF BSS 2          R3 - DEVICE TYPE FLAGS
DTFFIL EQU 0          FILE ORIENTED
DTFTIL EQU 1          TILINE DEVICE
DTFTIM EQU 2          ENABLE TIME-OUT
DTFPRI EQU 3          PRIVILEGED DEVICE
DTFKSB EQU 4          TERMINAL WITH A KSB
DTFCOM EQU 5          COMM DEVICE
DTFSYD EQU 6          SYSTEM DISC
DTFEXT EQU 7          EXPANSION BLOCK PRESENT
PDTDIB DATA PDTLNK+PDTSIZ R4 - DEVICE INFO BLOCK ADDRESS
PDTR5 BSS 2          R5 - DEVICE SERVICE ROUTINE SCRATCH
PDTR6 BSS 2          R6 - DSR SCRATCH
PDTR7 BSS 2          R7 - DSR SCRATCH
PDTR8 BSS 2          R8 - DSR SCRATCH
PDTR9 BSS 2          R9 - DSR SCRATCH
PDTR10 BSS 2         R10 - DSR SCRATCH
PDTR11 BSS 2         R11 - DSR SCRATCH
PDTCRU BSS 2         R12 - CRU OR TILINE ADDRESS
PDTR13 DATA $-$$    R13 - SAVED WP
PDTR14 DATA $-$$    R14 - SAVED PC
PDTR15 DATA $-$$    R15 - SAVED ST
PDT$ DATA PDTR0     PDTR0 ADDRESS
PDTDSR BSS 2         DSR ADDRESS
PDTErr BSS 1         SAVED ERROR CODE
PDTFLG BSS 1         DEVICE FLAGS

```

Figure 5-6. PDT Template (Sheet 1 of 2)

```

DFGPRB EQU 8      USE PRB IN LOG MESSAGE
DFGJAR EQU 9      JISCII RECIEVE MODE FLAG
DFGJAT EQU 10     JISCII TRANSMIT MODE FLAG
DFGSTA EQU 11     DEVICE STATE (TWO BITS)
*                ONLINE = 0 , OFFLINE = 1
*                DIAGNOSTIC = 2, UNDEFINED = 3
DFGOPF EQU 14     OPERATION FAILED
PDTNAM BSS 4      DEVICE NAME
PDTSL1 BSS 2      RESERVED FOR SYSTEM LOG
PDTSL2 BSS 2      RESERVED FOR SYSTEM LOG
PDTBUF BSS 2      CRU BUFFER ADDRESS
PDTBLN BSS 2      CRU BUFFER LENGTH
PDTINT BSS 2      DSR INTERRUPT ADDRESS
PDTDVQ BSS 10     DEVICE QUEUE ANCHOR
PDTT1 BSS 2       TIME OUT COUNT 1
PDTT2 BSS 2       TIME OUT COUNT 2
PDTSRB BSS 2      SAVED PRB ADDRESS
PDTFQL BSS 2      FAST REENTER ME QUEUE LINK
*
PDTSIZ EQU $
RORG
PAGE

```

Figure 5-6. PDT Template (Sheet 2 of 2)

### 5.3.2 Keyboard Status Block (KSB)

The KSB is appended to the PDT for keyboard devices. It provides additional data about the device and supplies a second workspace. The KSB workspace is used to service interrupts for keyboard devices. Figure 5-7 shows the structure of the KSB and Table 5-3 describes the information it contains.

You must provide a KSB for every new keyboard device that uses SCI. You might find it useful to provide KSBs even for devices that do not use SCI. Like the PDT, you must define all of the fields, even though the system provides some of the initial values.

HEX BYTE		
> 00	R0	KSBLDT -- STATION LDT ADDRESS
> 02	R1	KSBQOC -- QUEUE LENGTH
> 04	R2	KSBQIP -- QUEUE INPUT POINTER
> 06	R3	KSBQOP -- QUEUE OUTPUT POINTER
> 08	R4	KSBQEP -- QUEUE END POINTER
> 0A	R5	RESERVED
> 0C	R6	KSBFL -- KSB FLAG      KSBSN -- STATION NUMBER
> 0E	R7	KSBR7 -- SCRATCH
> 10	R8	KSBTSSB -- TSB ADDRESS/VALIDATION TABLE ADDRESS
> 12	R9	KSBR9 -- SCRATCH
> 14	R10	
> 16	R11	
> 18	R12	KSBCRU -- CRU BASE
> 1A	R13	KSBR13 -- SAVED WP
> 1C	R14	KSBR14 -- SAVED PC
> 1E	R15	KSBR15 -- SAVED ST
> 20		KSBLD0 -- PDT ADDRESS
> 22		KSBLD2 -- LUNO      KSBLD3 -- START I/O COUNT
> 24		KSBLD4 -- LDT FLAGS
> 26		KSBLD6 -- LDT LINK
> 28		KSBLD8 -- TSB ADDRESS
> 2A		KSBLCK -- LOCK COUNT
> 2C		

2283070

Figure 5-7. KSB Structure

Table 5-3. KSB Values

Hex. Byte	Field Name	Description																								
> 00	KSBLDT	Scratch.																								
> 02	KSBQOC	The number of characters currently in the input character queue. Initialize to zero.																								
> 04	KSBQIP	A pointer to the next byte of the character queue that is available to receive an input character (KSBBUF).																								
> 06	KSBQOP	A pointer to the oldest character in the input character queue, that is, the next character to be picked up by the DSR (KSBBUF).																								
> 08	KSBQEP	A pointer to the word after the character queue. That word contains the length of the queue (KSBBUF + KSBSIZ).																								
> 0A	RESERVED																									
> 0C	KSBFL	Flags as follows: <table border="1" data-bbox="479 1008 1388 1470"> <thead> <tr> <th>Field Name</th> <th>Bit</th> <th>Meaning When Set</th> </tr> </thead> <tbody> <tr> <td>KSBCHEM</td> <td>0</td> <td>Character mode (no mapping).</td> </tr> <tr> <td>KSBCEIE</td> <td>1</td> <td>Allow the command interpreter to be used on this terminal.</td> </tr> <tr> <td>KSBRCM</td> <td>2</td> <td>Keyboard is in record mode (always set).</td> </tr> <tr> <td>KSBCEIB</td> <td>3</td> <td>Bid task. Log-on request pending.</td> </tr> <tr> <td>KSBCEICP</td> <td>4</td> <td>The command interpreter is active.</td> </tr> <tr> <td>KSBCESET</td> <td>5</td> <td>Hold I/O.</td> </tr> <tr> <td>KSBCEKIO</td> <td>6</td> <td>Abort I/O.</td> </tr> </tbody> </table>	Field Name	Bit	Meaning When Set	KSBCHEM	0	Character mode (no mapping).	KSBCEIE	1	Allow the command interpreter to be used on this terminal.	KSBRCM	2	Keyboard is in record mode (always set).	KSBCEIB	3	Bid task. Log-on request pending.	KSBCEICP	4	The command interpreter is active.	KSBCESET	5	Hold I/O.	KSBCEKIO	6	Abort I/O.
Field Name	Bit	Meaning When Set																								
KSBCHEM	0	Character mode (no mapping).																								
KSBCEIE	1	Allow the command interpreter to be used on this terminal.																								
KSBRCM	2	Keyboard is in record mode (always set).																								
KSBCEIB	3	Bid task. Log-on request pending.																								
KSBCEICP	4	The command interpreter is active.																								
KSBCESET	5	Hold I/O.																								
KSBCEKIO	6	Abort I/O.																								
> 0D	KSBSN	Station (terminal) ID.																								
> 0E	KSB7	Scratch register for use by the DSR.																								
> 10	KSBTSB	The address of the TSB of the task currently using the terminal if the terminal is in character mode. If a validation table is being used, this field contains the validation table address.																								
> 12	KSB9, KSB10, KSB11	Scratch registers for use by the DSR.																								



Table 5-3. KSB Values (Continued)

Hex. Byte	Field Name	Description
> 18	KSBCRU	The CRU or TILINE address of the terminal.
> 1A	KSBR13	The saved context (WP, PC, ST) to which the DSR keyboard interrupt handling routine returns control via an RTWP instruction.
> 20	KSBLD0	These 10 bytes form a logical device table (see the <i>DX10 Design Document</i> ) that serves as an anchor for the terminal local LDT list. Flag bit 0 in byte > 24 is set to mark this LDT as an anchor. This LDT assigns terminal local LUNO 0 to the terminal itself.
> 2A	KSBLCK	The lock out count, which is a count of the number of Read with Event Characters SVCs issued for this terminal.
> 2C	KSBBUF	The input character buffer. Its size is defined by the response to the CHARACTER QUEUE prompt from GEN990. This buffer must hold an even number of bytes (at least two). Typical buffers have six to ten bytes. This queue is used to store unsolicited characters from the terminal.
> 2E	KSBSIZ	The length of the input character buffer. Must be an even number.

Figure 5-8 shows a template for the KSB. You might find it convenient to copy the source from one of the KSBs in `.$SYSGEN.<sysname>.$SOURCE` and tailor it to your device. You must append your KSB source to the PDT source, and place the address of the start of the KSB in R4 of the PDT. The device extension must follow the KSB.

```

*****
*   KEYBOARD STATUS BLOCK           (KSB)
*****
      DORG 0
KSBLDT DATA KSBLD0           R0 - STATION LDT ADDRESS
KSBQOC BSS 2                  R1 - QUEUE OUTPUT COUNT
KSBQIP BSS 2                  R2 - QUEUE INPUT POINTER
KSBQOP BSS 2                  R3 - QUEUE OUTPUT POINTER
KSBQEP BSS 2                  R4 - QUEUE END POINTER
KSBEFB DATA 0               R5 - EVENT CHARACTER BUFFER
KSBFL BSS 1                  R6 - KSB FLAGS
KSBCHM EQU 0                  CHARACTER MODE
KSBCHM EQU 1                  COMMAND INTERP ENABLE
KSBRCM EQU 2                  RECORD MODE
KSBICB EQU 3                  COMMAND INTERP BID
KSBICP EQU 4                  COMMAND INTERP ACTIVE
KSBSET EQU 5                  COMMAND I/O HOLD
KSBKIO EQU 6                  COMMAND I/O ABORT
KSBBRK EQU 7                  DEACTIVATE BREAK KEY
KSB SN BSS 1                  - STATION NUMBER
KSB R7 BSS 2                  R7 - SCRATCH
KSB TSB DATA $-$            R8 - TSB ADDRESS
KSBVTA EQU KSBTSB           VALIDATION TABLE ADDRESS
KSB R9 BSS 2                  R9 - SCRATCH
KSB R10 BSS 2                R10 - SCRATCH
KSB R11 BSS 2                R11 - SCRATCH
KSB CRU BSS 2                R12 - CRU BASE
KSB R13 DATA $-$           R13 - SAVED WP
KSB R14 DATA $-$           R14 - SAVED PC
KSB R15 DATA $-$           R15 - SAVED ST
KSB LD0 DATA 0             PDT ADDRESS (PDTLNK)
KSB LD2 BYTE 0              LUNO
KSB LD3 BYTE 0              INIT I/O COUNT
KSB LD4 BSS 2               LDT FLAGS
KSB LD6 BSS 2               LDT LINK
KSB LD8 DATA $-$           TSB ADDRESS
KSB LCK DATA 0             LOCK COUNT
KSB SIZ EQU $-KSBLDT
      RORG
      PAGE

```

Figure 5-8. KSB Template

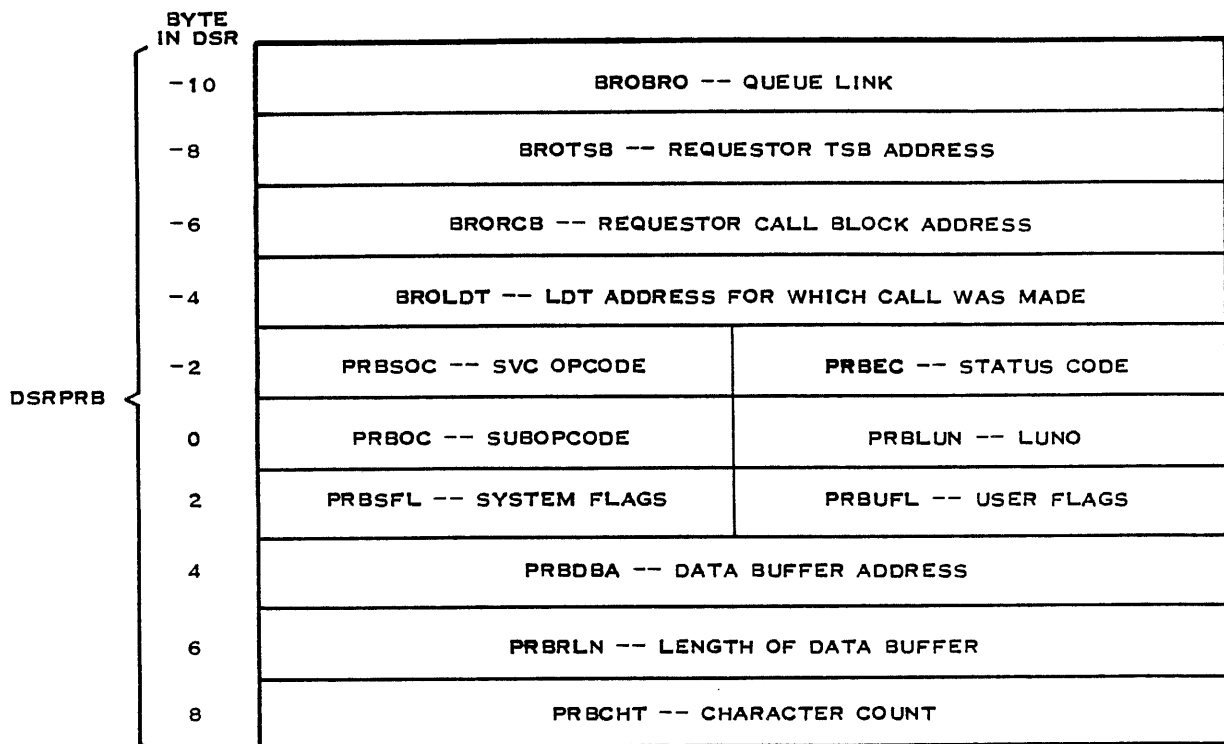
### 5.3.3 Buffered I/O Request Block

The buffered I/O request block is the portion of the SVC block that contains information for the DSR. Figure 5-9 shows the structure and contents of the buffered I/O request block. DSRPRB is a template defined in .SS\$SYSGEN.SYSTEM.TABLES. The DSR uses this template to reference elements of this data structure. Note that R1 contains the address of the I/O subopcode byte of the buffered I/O request block. Therefore, use offsets in your DSR as given in Figure 5-9, which are those defined by the DSRPRB template.

The buffered I/O request block is copied to the system table area by DXIOS. The DSR must not use LDS or LDD to access parts of the buffered I/O request block. When the DSR completes, DXIOS copies the buffered I/O request block back to the user task.

### 5.3.4 Task Status Block (TSB)

Tasks are represented within DX10 by a TSB. The bidding routines TMBID0 and TM\$BID build TSBs in the system table area. The termination task, TM\$DGN, releases a TSB when a task terminates, unless the task is a queue server. TSBs of inactive queue servers are not released unless more system table area is needed. Figure 5-10 shows the format of a TSB, and Table 5-4 describes the information it contains.



2284689

Figure 5-9. Buffered I/O Request Block

How to Write a Device Service Routine

HEX. BYTE		
>00	TSBQL -- QUEUEING LINK	
>02	TSBWP -- ACTIVE WP	
>04	TSBPC -- ACTIVE PC	
>06	TSBST -- ACTIVE ST	
>08	TSBPRI -- PRIORITY	TSBSTA -- TASK STATE
>0A	TSBFLG -- TASK FLAGS	
>0C	TSBEAC -- TRANSFER VECTOR ADDRESS	
>0E	TSBID -- INSTALLED ID	TSBRID -- RUN ID
>10	TSBSMF -- SAVED MAP FILE ADDRESS	
>12	TSBLNK -- FIXED TSB LINK	
>14	TSBKSS -- KSS ADDRESS	
>16	TSBFL2 -- TASK FLAGS (WD2)	
>18	TSBAR1 -- SID PARAMETER (1)	
>1A	TSBAR2 -- SID PARAMETER (2)	
>1C	TSBALT -- ALTERNATE TSB ADDRESS	
>1E	TSBCHR -- 913,911 CHARACTER	TSBIOC -- TILINE I/O COUNT
>20	TSBPR1 -- PSB ADDRESS (PROCEDURE 1)	
>22	TSBPR2 -- PSB ADDRESS (PROCEDURE 2)	
>24	TSBFCB -- PROGRAM FILE FCB ADDRESS	
>26	TSBERG -- DIAGNOSTIC ERROR CODE	
>28	TSBWPO -- DIAGNOSTIC WP	
>2A	TSBPCD -- DIAGNOSTIC PC	
>2C	TSBSTD -- DIAGNOSTIC ST	
>2E	TSBTD1 -- TIME DELAY COUNTER TSBTD2	
>32	TSBML1 -- MAP LIMIT 1	
>34	TSBMB1 -- MAP BIAS 1	
>36	TSBML2 -- MAP LIMIT 2	
>38	TSBMB2 -- MAP BIAS 2	
>3A	TSBML3 -- MAP LIMIT 3	
>3C	TSBMB3 -- MAP BIAS 3	
>3E	TSBPRF -- FIXED PRIORITY	TSBMRG -- TASK REGISTER NUMBER
>40	TSBPAR -- PARENT TSB ADDRESS	
>42	TSBSON -- OLDEST SON TSB ADDRESS	
>44	TSBRI1 -- OLDER SIBLING TSB ADDRESS	
>46	TSBRI2 -- YOUNGER SIBLING TSB ADDRESS	
>48	TSBBLN -- BEET LENGTH OF PROGRAM	
>4A	TSBTON -- OVERLAY NUMBER	
>4C	TSBOAD -- ADDRESS OF OVERLAY AREA DESTINATION	
>4E	TSBTO -- TIME TASK SUSPENDED	
>50	TSBT1 -- NUMBER OF TIME SLICES REMAINING	
>52	TSBSCR -- SCRATCH FOR GETMEM	
>54	TSBRLL -- LINK TO NEXT ROLLED TASK	
>56	TSBRN -- ROLL FILE STARTING PHYSICAL RECORD NUMBER	
>5A	TSBRRL -- NUMBER OF ROLL FILE RECORDS	
>5C	TSBLDF -- LOCAL LDT LIST FLAGS	
>5E	TSBLDA -- LOCAL LDT LIST ADDRESS	
>60	TSBEOR -- EOR COUNT	TSBIIP -- I/O COUNT
>62	TSBER -- QUEUE ANCHOR ADDRESS	
>64	TSBTSC -- TASK SENTRY COUNT	
>66		

2284695

Figure 5-10. TSB Structure

Table 5-4. TSB Values

Hex. Byte	Field Name	Description																																		
> 00	TSBQL	Link to the next TSB on the queue, when this TSB is queued.																																		
> 02	TSBWP, TSBPC, TSBST	The saved context (WP, PC, and ST values) for the task. When the task is scheduled to execute, these saved values are used to begin execution.																																		
> 08	TSBPRI	Task priority (0, 1, 2, 3, or > 81, > 82, > 83, ..., > FF, where > 81 is real-time priority 1 and > FF is real-time priority 127).																																		
> 09	TSBSTA	Task state as shown in Table 5-5.																																		
> 0A	TSBFLG	First word of task flags. The flags are as follows: <table border="1" data-bbox="755 835 1388 1354"> <thead> <tr> <th>Bit</th> <th>Meaning When Set</th> </tr> </thead> <tbody> <tr><td>0</td><td>System task (hardware and software privilege)</td></tr> <tr><td>1</td><td>Privileged task (software)</td></tr> <tr><td>2</td><td>Memory-resident task</td></tr> <tr><td>3</td><td>Take end action on error</td></tr> <tr><td>4</td><td>Roll out candidate</td></tr> <tr><td>5</td><td>Rolled out</td></tr> <tr><td>6</td><td>Abort/terminate task</td></tr> <tr><td>7</td><td>Activate call outstanding</td></tr> <tr><td>8</td><td>Reactivate bidding task at termination</td></tr> <tr><td>9</td><td>Serially reusable task</td></tr> <tr><td>10</td><td>Task quieting in progress</td></tr> <tr><td>11</td><td>Initial bid</td></tr> <tr><td>12</td><td>Leave task alone; do not abort</td></tr> <tr><td>13</td><td>Task is under control of alternate TSB</td></tr> <tr><td>14</td><td>SCI flag for scanning TSB chain</td></tr> <tr><td>15</td><td>Task is replicated image</td></tr> </tbody> </table>	Bit	Meaning When Set	0	System task (hardware and software privilege)	1	Privileged task (software)	2	Memory-resident task	3	Take end action on error	4	Roll out candidate	5	Rolled out	6	Abort/terminate task	7	Activate call outstanding	8	Reactivate bidding task at termination	9	Serially reusable task	10	Task quieting in progress	11	Initial bid	12	Leave task alone; do not abort	13	Task is under control of alternate TSB	14	SCI flag for scanning TSB chain	15	Task is replicated image
Bit	Meaning When Set																																			
0	System task (hardware and software privilege)																																			
1	Privileged task (software)																																			
2	Memory-resident task																																			
3	Take end action on error																																			
4	Roll out candidate																																			
5	Rolled out																																			
6	Abort/terminate task																																			
7	Activate call outstanding																																			
8	Reactivate bidding task at termination																																			
9	Serially reusable task																																			
10	Task quieting in progress																																			
11	Initial bid																																			
12	Leave task alone; do not abort																																			
13	Task is under control of alternate TSB																																			
14	SCI flag for scanning TSB chain																																			
15	Task is replicated image																																			
> 0C	TSBEAC	Transfer vector address.																																		
> 0E	TSBIID	Installed task ID.																																		
> 0F	TSBRID	Runtime ID assigned by system.																																		
> 10	TSBSMF	Address in the TSB of the saved map file register values (bytes > 32 – > 3D).																																		
> 12	TSBLNK	Link to the next TSB in the fixed list of TSBs. All TSBs in the system table area are linked onto this list when they are created. The list can be searched to find a task with a given runtime ID by the routine named TMTSCH (for example, to kill the task).																																		
> 14	TSBKSB	Address of the KSB of the terminal associated with this task (that is, the task was bid from the terminal).																																		

**Table 5-4. TSB Values (Continued)**

Hex. Byte	Field Name	Description																														
> 16	TSBFL2	Task flags as follows: <table border="1" style="margin-left: 40px;"> <thead> <tr> <th>Bit</th> <th>Meaning When Set</th> </tr> </thead> <tbody> <tr><td>0</td><td>Task to be suspended next time it executes</td></tr> <tr><td>1</td><td>Task is being controlled</td></tr> <tr><td>2</td><td>SVC traps to be taken when specified</td></tr> <tr><td>3</td><td>SVC switch: when 0, SVC traps are taken</td></tr> <tr><td>4</td><td>Execution stopped by scheduler</td></tr> <tr><td>5</td><td>Execution stopped by trapped SVC</td></tr> <tr><td>6</td><td>Execution stopped by XOP 15,15 (breakpoint)</td></tr> <tr><td>7</td><td>Dynamic priority management</td></tr> <tr><td>8</td><td>Roll in progress</td></tr> <tr><td>9</td><td>Task activated</td></tr> <tr><td>10</td><td>Initiate followed by execute I/O</td></tr> <tr><td>11</td><td>Extend time slice</td></tr> <tr><td>12</td><td>End action available for task</td></tr> <tr><td>13 – 15</td><td>Not used</td></tr> </tbody> </table>	Bit	Meaning When Set	0	Task to be suspended next time it executes	1	Task is being controlled	2	SVC traps to be taken when specified	3	SVC switch: when 0, SVC traps are taken	4	Execution stopped by scheduler	5	Execution stopped by trapped SVC	6	Execution stopped by XOP 15,15 (breakpoint)	7	Dynamic priority management	8	Roll in progress	9	Task activated	10	Initiate followed by execute I/O	11	Extend time slice	12	End action available for task	13 – 15	Not used
Bit	Meaning When Set																															
0	Task to be suspended next time it executes																															
1	Task is being controlled																															
2	SVC traps to be taken when specified																															
3	SVC switch: when 0, SVC traps are taken																															
4	Execution stopped by scheduler																															
5	Execution stopped by trapped SVC																															
6	Execution stopped by XOP 15,15 (breakpoint)																															
7	Dynamic priority management																															
8	Roll in progress																															
9	Task activated																															
10	Initiate followed by execute I/O																															
11	Extend time slice																															
12	End action available for task																															
13 – 15	Not used																															
> 18	TSBAR1	The two parameters that can be passed to the task by the Bid SVC and accessed by the task using the Get Bid Parameters SVC.																														
> 1C	TSBALT	TSB address of the alternate task. The alternate task is executed in place of this task.																														
> 1E	TSBCHR	913/911 character.																														
> 1F	TSBIOC	Number of outstanding TILINE I/O operations.																														
> 20	TSBPR1	Address of the procedure status block for attached procedure 1 (0 if none).																														
> 22	TSBPR2	PSB address for procedure 2 (0 if none).																														
> 24	TSBFCB	Address of the FCB that represents the program file on which this task is installed.																														
> 26	TSBERC	Error code that describes the error that caused the task to terminate (used by the termination task).																														
> 28	TSBWPD, TSBPCD, TSBSTD	The context (WP, PC, and ST registers) of the task when an error forced the task to terminate or take end action (used by the termination task). A Get End Action Status SVC returns these values.																														
> 2E	TSBTD1	Number of system time units remaining before this task will be reactivated from its time-delayed state (32 bits).																														

Table 5-4. TSB Values (Continued)

Hex. Byte	Field Name	Description
> 32	TSBML1	The map register values to be used when this task executes.
> 3E	TSBPRF	Map flags. Fixed priority of task.
> 3F	TSBMRG	The offset into the saved map file that marks the limit register that maps the task segment (that is, 0, 4, or 8).
> 40	TSBPAR	TSB family tree pointers.
> 42	TSBSON	TSB family tree pointers.
> 44	TSBBR1	TSB family tree pointers.
> 46	TSBBR2	TSB family tree pointers.
> 48	TSBBLN	Length of the entire program (task and procedures) in beets (32-byte blocks).
> 4A	TSBTON	The number of the system overlay in which this task was last executing (used for system tasks only).
> 4C	TSBOAD	The address of the overlay area in which the above overlay was loaded (the overlay MUST be reloaded in the same place).
> 4E	TSBT0	Number of time slices this task has been suspended.
> 50	TSBT1	Number of time slices still allotted to this task as the minimum number of time slices it must receive before it can be forcibly rolled-out by an equal priority task.
> 52	TSBSCR	Scratch used by the Get Memory SVC processor and the system overlay loader.
> 54	TSBRLL	Link to the TSB or PSB that represents the next rolled segment. The TSB or PSB of each rolled task or procedure is linked onto a list of rolled segments. The list is kept in order by increasing the roll file record number; that is, segments written at the beginning of the roll file appear at the beginning of the list. This linked list serves as a directory into the roll file, so that the various rolled segments can be retrieved for roll-in. Further roll information is kept in TSBs or PSBs.

Table 5-4. TSB Values (Continued)

Hex. Byte	Field Name	Description
> 56	TSBRRN	Number of the physical record in the roll file that begins the rolled image of the task segment. At initial bid time, this is the program file record number.
> 5A	TSBRRL	Number of roll file records occupied by the rolled task image. During initial bid, this is the length of the task in bytes.
> 5C	TSBLDF	Task local LDT list flags: bit 0 is the LDT anchor.
> 5E	TSBLDA	Pointer to the first task local LDT, or the station local LDT list anchor (if no task local LDTs exist).
> 60	TSBEOR	Number of I/O end-of-records that need to be processed for this task. If this field is nonzero, the device driver routine (DDT) receives the next time slice that would otherwise have been awarded to this task.
> 61	TSBIIP	The number of I/O operations outstanding for this task.
> 62	TSBSER	The address of the anchor for the queue served by this task (used only for queue servers).
> 64	TSBTSC	The task sentry count.
> 66	*	

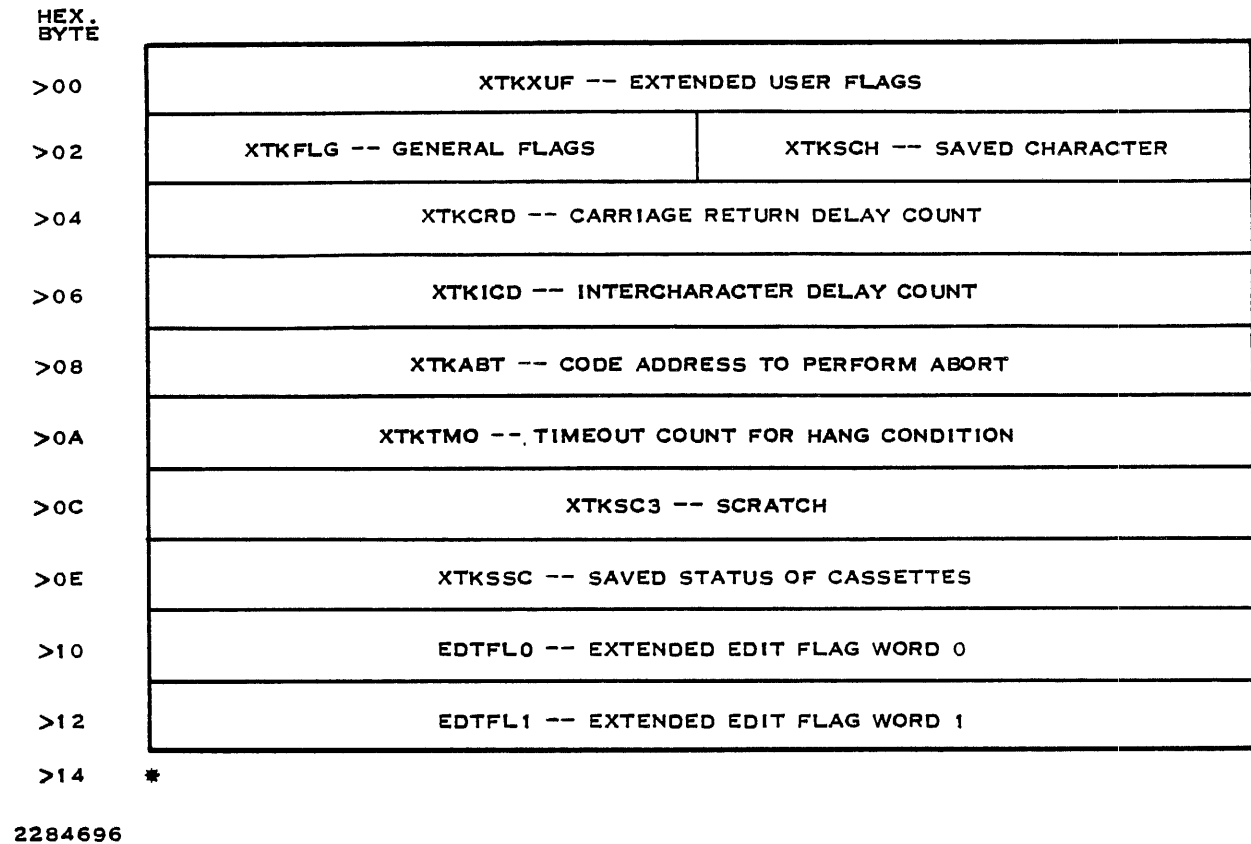


Table 5-5. Task State Codes

Task State	Significance
00	Active task, priority level 0
01	Active task, priority level 1
02	Active task, priority level 2
03	Active task, priority level 3
04	Terminated task
05	Task in time delay
06	Suspended task
07	Currently executing task
08	Reserved
09	Task awaiting completion of I/O
0A	Task awaiting assignment of device for I/O
0B	Task awaiting disk file utility services
0C	Reserved
0D	Task awaiting file management services
0E	Task awaiting overlay loader services
0F	Task awaiting initial load
10	Reserved
10	Task awaiting disk management services
12	Task awaiting tape management services
13	Waiting on system overlay loader services
14	Waiting on task driven SVC processor
15	Task waiting on GETMEM request
16	Not used
17	Suspended for co-routine activation
18	Task waiting on termination task services
19	Task awaiting completion of any I/O
1A	Waiting on MM\$FND door
1B	Task eligible for rollout when requested I/O is complete
1C	Task activated while roll in progress
1D	Suspended for initiate I/O threshold
1E	Suspended for locked directory
1F	Suspended for task management directory buffer
24	Task suspended for queue input
FF	Dummy task state

**5.3.5 Extension for Terminals with Keyboards (XTK)**

The XTK is an extension used by TI DSRs for terminals. It functions as an extension to the KSB. Figure 5-11 shows the structure of the XTK, and Table 5-6 describes the values it contains. Figure 5-12 shows the template for the XTK.



**Figure 5-11. XTK Structure**

Table 5-6. XTK Values

Hex. Byte	Field Name	Description														
>00	XTKXUF	Extended user flags from BRB														
>02	XTKFLG	General flags: <table border="1" data-bbox="747 609 1396 1008"> <thead> <tr> <th>Bit</th> <th>Meaning When Set</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>KSFHNG — Hang-up condition on 745</td> </tr> <tr> <td>1</td> <td>KSFTMS — Time-out switch for 745</td> </tr> <tr> <td>2</td> <td>KSFSCI — SCI active during hang up</td> </tr> <tr> <td>3</td> <td>KSFDCD — Data carrier drop detected</td> </tr> <tr> <td>4</td> <td>KSFSIO — Shift in/shift out JISCII</td> </tr> <tr> <td>5</td> <td>KSFDIF — Direct character input requested</td> </tr> </tbody> </table>	Bit	Meaning When Set	0	KSFHNG — Hang-up condition on 745	1	KSFTMS — Time-out switch for 745	2	KSFSCI — SCI active during hang up	3	KSFDCD — Data carrier drop detected	4	KSFSIO — Shift in/shift out JISCII	5	KSFDIF — Direct character input requested
Bit	Meaning When Set															
0	KSFHNG — Hang-up condition on 745															
1	KSFTMS — Time-out switch for 745															
2	KSFSCI — SCI active during hang up															
3	KSFDCD — Data carrier drop detected															
4	KSFSIO — Shift in/shift out JISCII															
5	KSFDIF — Direct character input requested															
>03	XTKSCH	Saved character for JISCII terminals														
>04	XTKCRD	Carriage return delay count														
>06	XTKICD	Intercharacter delay count														
>08	XTKABT	Code address to perform abort														
>0A	XTKTM0	Time-out count for hang condition														
>0C	XTKSC3	Scratch														
>0E	XTKSSC	Saved status of cassettes														
>10	EDTFL0	Edit flag word 1 <table border="1" data-bbox="747 1470 1396 1858"> <thead> <tr> <th>Bit</th> <th>Meaning When Set</th> </tr> </thead> <tbody> <tr> <td>0 – 7</td> <td>Reserved</td> </tr> <tr> <td>8</td> <td>MDTCHK — Post data modified on read</td> </tr> <tr> <td>9</td> <td>EXVAL — Extended character validation</td> </tr> <tr> <td>10</td> <td>NULFLG — Null character</td> </tr> <tr> <td>11</td> <td>CNBFLG — Covert null to blank</td> </tr> <tr> <td>12 – 15</td> <td>Reserved</td> </tr> </tbody> </table>	Bit	Meaning When Set	0 – 7	Reserved	8	MDTCHK — Post data modified on read	9	EXVAL — Extended character validation	10	NULFLG — Null character	11	CNBFLG — Covert null to blank	12 – 15	Reserved
Bit	Meaning When Set															
0 – 7	Reserved															
8	MDTCHK — Post data modified on read															
9	EXVAL — Extended character validation															
10	NULFLG — Null character															
11	CNBFLG — Covert null to blank															
12 – 15	Reserved															

**Table 5-6. XTK Values (Continued)**

Hex. Byte	Field Name	Description												
> 12	EDTFL1	Edit flag word 2												
		<table border="0"> <thead> <tr> <th data-bbox="724 569 760 594">Bit</th> <th data-bbox="1000 569 1219 594">Meaning When Set</th> </tr> </thead> <tbody> <tr> <td data-bbox="708 625 760 651">0 – 1</td> <td data-bbox="837 625 943 651">Reserved</td> </tr> <tr> <td data-bbox="708 682 721 707">2</td> <td data-bbox="837 682 1300 707">LEFARO — Terminate read on left arrow</td> </tr> <tr> <td data-bbox="708 739 776 764">3 – 11</td> <td data-bbox="837 739 943 764">Reserved</td> </tr> <tr> <td data-bbox="708 795 737 821">12</td> <td data-bbox="837 795 1325 821">RITSARO — Terminate read on right arrow</td> </tr> <tr> <td data-bbox="708 852 792 877">13 – 15</td> <td data-bbox="837 852 943 877">Reserved</td> </tr> </tbody> </table>	Bit	Meaning When Set	0 – 1	Reserved	2	LEFARO — Terminate read on left arrow	3 – 11	Reserved	12	RITSARO — Terminate read on right arrow	13 – 15	Reserved
Bit	Meaning When Set													
0 – 1	Reserved													
2	LEFARO — Terminate read on left arrow													
3 – 11	Reserved													
12	RITSARO — Terminate read on right arrow													
13 – 15	Reserved													

```

UNL
*****
*
*   EXTENSION FOR A TERMINAL      (XTK)           02/12/82
*   WITH A KEYBOARD
*
*****
      DORG XTKBGN
XTKXUF WORD 0           EXTENDED USER FLAGS FROM BRB
XTKFLG FLAGS 8         XTK GENERAL FLAGS
      FLAG KSFHNG       HANG UP CONDITION ON 745
      FLAG KSFTMS       TIME OUT SWITCH FOR 745
      FLAG KSFSCI       SCI ACTIVE DURING HANG UP
      FLAG KSPDCD       DATA CARRIER DROP DETECTED
      FLAG KSFSIO       SHIFT IN/SHIFT OUT JISCII
      FLAG KSPDIF       DIRECT CHAR INPUT REQUESTED
XTKSCH BYTE 0         SAVED CHAR FOR JISCII TERMINAL
XTKFIL EQU XTKFLG     FILL CHARACTER
XTKEVT EQU XTKFLG+1   EVENT CHARACTER
XTKCRD WORD 0         CARRIAGE RETURN DELAY COUNT
XTKPOS EQU XTKCRD     WITHIN FIELD CURSOR POSITION
XTKICD WORD 0         INTER-CHARACTER DELAY COUNT
XTKDEF EQU XTKICD     START OF FIELD CURSOR POSITION
XTKVTA EQU KSBVTA     VALIDATION TABLE ADDRESS
XTKABT WORD 0         CODE ADDRESS TO PERFORM ABORT
XTKSC1 EQU XTKABT     SCRATCH # 1
XTKTMO WORD 0         TIMEOUT COUNT FOR HANG CONDITON
XTKSC2 EQU XTKTMO     SCRATCH # 2
XTKSC3 WORD 0         SCRATCH # 3
XTKSSC WORD 0         SAVED STATUS OF CASSETTES
XTKJIN EQU XTKSSC     ASCII/JISCII INTENSITY MASK
*
*-----
*   EDIT FLAG WORD 0 AT END OF XTK                *CM*
*-----
EDTFLO WORD 0         EXTENDED EDIT FLAG WORD 0    *CM*
*
*   NOTE: BITS 0-7 ARE USED !
MDTCHK EQU 8         POST DATA MODIFIED ON READ   *CM*
EXVAL EQU 9         EXTENDED CHAR VAL              *CM*
NULFLG EQU 10       NULL CHARACTER                 *CM*
CNBFLG EQU 11       CONVERT NULL TO BLANK          *CM*
*
*   NOTE: BIT 12 IS USED !
*   NOTE: BITS 13-15 RESERVED
*-----
*   EDIT FLAG WORD 1 AT END OF XTK                *CM*
*-----
EDTFL1 WORD 0         EXTENDED EDIT FLAG WORD 1    *CM*
*
*   NOTE: BITS 0-1 ARE USED !
LEFARO EQU 2         1=TERMINATE RD ON LEFT ARROW  *CM*
*
*   NOTE: BITS 3-11 ARE USED !
RITARO EQU 12       1=TERMINATE RD ON RIGHT ARROW *CM*
*
*   NOTE: BITS 13-15 ARE USED !
*
XTKSIZ EQU $
      RORG
      PAGE
      LIST

```

Figure 5-12. XTK Template

### 5.3.6 Asynchronous DSR Local PDT Extension

The asynchronous DSR structure requires a PDT extension. Figure 5-13 shows the structure of this extension, and Figure 5-14 shows the template for this extension. The pathname for this template is .S\$\$SYSGEN.SYSTEM.TABLES.DSALLLEX.

This extension starts immediately after the KSB. The first two words of this extension are used to access a second DSR data structure (PDT extension) outside the local address space of the DSR. Paragraph 5.5.5 discusses the procedure for initializing these two words. The next five words, PDXFLG through PDXCP3, are reserved for HSR use. The remaining local PDT extension words are for TSR/ISR use. The size of this portion of the local PDT extension varies for different asynchronous devices.

HEX BYTE	
>0	PDXSMB -- LONG DISTANCE EXTENSION MAP BIAS
>2	PDXSMP -- LONG DISTANCE EXTENSION MAP POINTER
>4	PDXFLG -- HSR PARAMETER BYTE 0
>5	PDXCHN -- HSR PARAMETER BYTE 1
>6	PDXCP1 --- HSR PARAMETER BYTES 2 AND 3
>8	PDXCP2 -- HSR PARAMETER BYTES 4 AND 5
>A	PDXCP3 -- HSR PARAMETER BYTES 6 AND 7
>C	PDXCP4 -- TSR/ISR PARAMETER BYTES 0 AND 1
>E	PDXCP5 -- TSR/ISR PARAMETER BYTES 2 AND 3
>10	PDXCP6 -- TSR/ISR PARAMETER BYTES 4 AND 5
>12	PDXCP7 -- TSR/ISR PARAMETER BYTES 6 AND 7

2284697

**Figure 5-13. Asynchronous DSR Local PDT Extension Structure**

```

*****
*
*   LOCAL ASYNC EXTENSION TO PDT
*
*****
      DORG KSBSIZ
PDXSMB BSS 2          LONG DIST EXT MAP BIAS
PDXSMP BSS 2          LONG DIST EXT MAP POINTER
PDXFLG BSS 1          HSR MEMORY AREA
PDXCHN BSS 1          "
PDXFCT BSS 2          "
PDXCP1 BSS 2          "
PDXCP2 BSS 2          "
PDXCP3 BSS 2          "
PDXCP4 BSS 2          TSR MEMORY AREA
      "
      "
      "
      RORG

```

**Figure 5-14. Asynchronous DSR Local PDT Extension Template**

### 5.3.7 Asynchronous DSR Long-Distance Device Extension

The asynchronous DSRs use a long-distance extension for part of the PDT extension area. This memory must be accessed using long-distance instructions. The long-distance extension is divided into several areas. Figure 5-15 shows the structure of this extension, and Figure 5-16 shows the template for this extension. The pathname for this template is `.$SYSGEN.SYSTEM.TABLES.DSALLREX`.

The first 32 bytes beginning with `HSRBGN` are reserved for HSR module use. The next 112 bytes provide memory for a software transmit FIFO maintained by the HSR for non-buffered controllers. The only buffered asynchronous controllers are the `CI403` and the `CI404`. The remainder of the long-distance extension is for `TSR/ISR` use. Its size varies with the functions that the `TSR` and `ISR` modules perform. The example template defines areas for an implementation that keeps a memory copy of the screen image for `VDT` support. Forty-eight bytes are provided for `TSR/ISR` use. The `TSR` uses the memory starting at `SIBUFF` to maintain a memory image of the `VDT` screen.

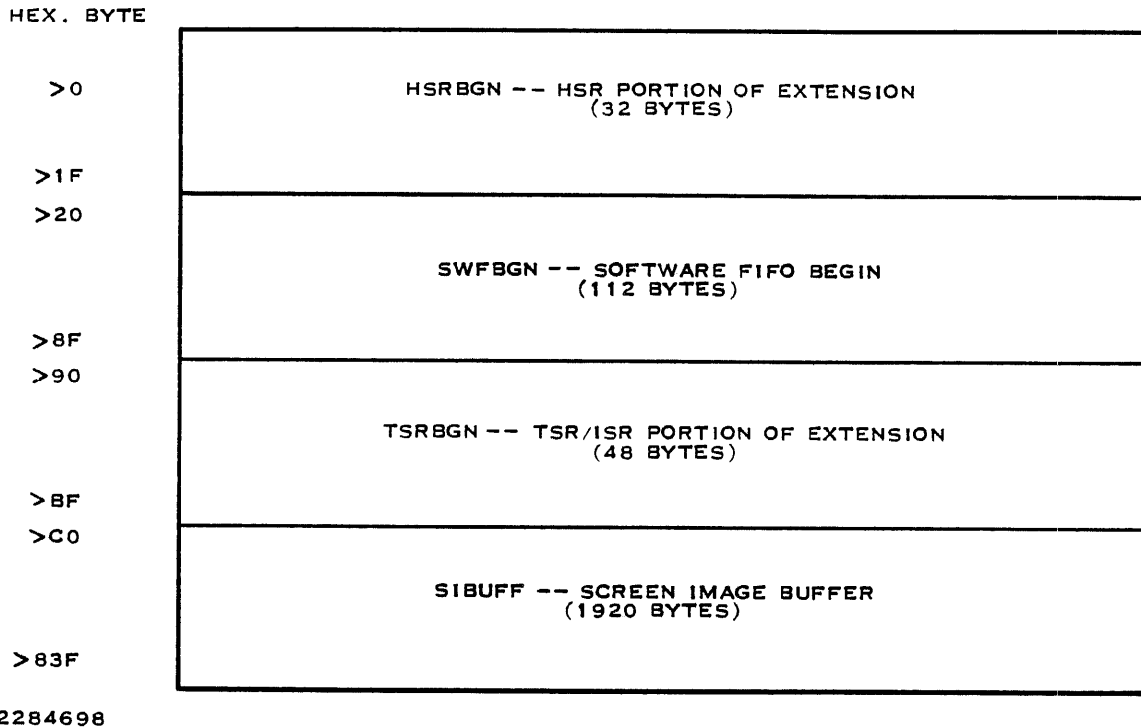


Figure 5-15. Asynchronous DSR Long-Distance Device Extension Structure

```

*****
***  ASYNC LONG DISTANCE EXTENSION  ***
*****
      DORG 0
HSRBGN EQU $           HSR PORTION OF DEVICE EXTENSION
      BSS >20          HSR DEPENDENT BLOCK
HSREND EQU $
*
SWFBGN EQU HSREND     SOFTWARE XMIT FIFO
      BSS >70
SWFEND EQU $
*
TSRBGN EQU SWFEND     TSR PORTION OF DEVICE EXTENSION
      BSS >30
TSREND EQU $
*
SIBUFF EQU TSREND     SCREEN IMAGE BUFFER
      BSS >780        1920 BYTE SCREEN IMAGE BUFFER
SIEND  EQU $
RORG 0
    
```

Figure 5-16. Asynchronous DSR Long-Distance Device Extension Template



## 5.4 DSR CONVENTIONS AND TECHNIQUES

This paragraph describes some of the methods used in writing the standard DX10 DSRs. When you write your own DSR, you can use these conventions and techniques to simplify your task and to make your DSR easier to maintain. The example DSRs provided at the end of this section demonstrate many of these methods.

### 5.4.1 Workspace for Keyboard ISRs

When you write an ISR for a keyboard device, you need to buffer the input keystrokes. Common subroutines provided with the operating system can handle keyboard input only if you provide a KSB. Since these routines use the workspace in the KSB, your DSR interrupt decoder and processor should also employ this workspace.

### 5.4.2 Decoding Interrupts

It is often necessary for a DSR to return to the system to wait for the next interrupt, and provide the address of a routine in the DSR which is coded to handle that interrupt. Since GEN990 allows only one interrupt entry point for an ISR, DX10 DSRs commonly use R6 in the interrupt processing workspace as an interrupt entry vector. When the DSR needs to wait for an interrupt, the DSR loads the address of an interrupt processor into R6 and executes an RTWP instruction. When the interrupt occurs, the ISR resets the time-out and branches to the address in R6. If you use a KSB, be sure to only transfer control from the ISR to the other DSR routines (that use the PDT workspace) by using a BLWP or the reenter-me mechanism. Use KSB R6 only for ISR entries and use PDT R6 only for entries to DSR routines that use the PDT workspace.

### 5.4.3 Reporting Errors to the System Log

Most DSRs report device errors to the system log by placing an appropriate error code in PDERR of the PDT and setting the Operation Failed flag in PDTFLG. You can use the device controller address in R12 of the PDT workspace to obtain controller status information. Most CRU controllers have one word of status information.

If the device buffer access method selected by the DTFTIL bit of PDTDTF is not direct (that is, the value of the DTFTIL bit is 0), you should store status information in the system log words (PDTSL1 and PDTSL2) to report the error to the system log. If the buffer access method is direct (the value of the DTFTIL bit is 1), your PDT must have an area set aside for log data and PDTSL1 must contain the address of that area. Refer to the following paragraph for a discussion of buffer access methods.

#### 5.4.4 Accessing the Data Buffer

You can access the data buffer associated with the I/O SVC in two ways. You select the method when you code the PDT by setting the value of bit 1 of PDTDTF (DTFTIL) as follows:

- 0 = Indirect access. DX10 DSRs use this method for CRU devices such as data terminals and line printers. It provides the flexibility for DX10 to roll out the calling task, and is suitable for devices (such as data terminals) that may have a relatively long response time under some conditions. The operating system makes a copy of the data buffer in system table area, and the DSR references that copy.
- 1 = Direct access. DX10 DSRs use this method for TILINE devices such as disks and magnetic tapes. The calling task cannot be rolled while I/O is in progress and is suitable for devices that have a relatively short response time under all conditions. The I/O scheduler does not make a copy of the data buffer, so this method is also suitable for devices with very long buffers.

The use of direct and indirect access need not be limited to TILINE and CRU type of devices. DX10 does not really care what the underlying device type is. You need to decide which access mode suits the needs of your device. Note that if you select direct access for a device that can have a long response time, the system can appear to lock up until the I/O completes. This symptom is caused by the following condition. Task A has an I/O request pending on a device for which direct access is specified, and the operating system does not have enough memory to roll in task B for execution. It flags task A so that it is rolled out when the I/O completes instead of being placed in execution. Task B is not loaded until then. You can relieve this condition by making the task that makes such an access a memory-resident task.

**5.4.4.1 Indirect Access.** For buffers accessed indirectly, DX10 creates a copy of the data buffer in the system table area. It modifies the buffer address in the system's copy of the buffered I/O request block (pointed to by R1 in the PDT) to point to the system's copy of the buffer. The DSR should access this copy of the buffer. When the DSR calls ENDRCD (see paragraph 5.7.3), the system copies the buffered I/O request block and the buffer back to the task that issued the SVC. The system also takes care of all task rollin/rollout processing.

**5.4.4.2 Direct Access.** For buffers accessed directly, DX10 makes a copy of only the buffered I/O request block. The buffer address in the buffered I/O request block is relative to the address space of the calling task. The DSR has its own map file, loaded into map file 0 of the CPU. That map file brings together the system root, the I/O common routines, and the DSR all into one address space. The system root is the part of the operating system linked in phase 0. (Refer to the *DX10 Design Document* for a discussion of system memory mapping.) The calling task executes in CPU map file 1. The DSR must load CPU map file 2 using the LDD and LDS instructions with the same map file used to execute the task, and must use the buffer address in the next instruction. For additional details, refer to the *990/10 and 990/12 Assembly Language Reference Manual*.

You can access the map file for the task by accessing its TSB. The address of the TSB is located in the buffered request overhead block, which immediately precedes the buffered I/O request block. The TSB has a pointer to the map file for the task at TSBSMF. The following code places the first word of the data buffer in R4:

```

MOV  @PRBDBA(R1),R5      R5 < = ADDRESS OF TASK BUFFER
MOV  @BROTSB(R1),R7      R7 < = ADDRESS OF TSB
MOV  @TSBSMF(R7),R7      R7 < = ADDRESS OF MAP FILE
LDS  *R7
MOV  *R5,R4              R4 < = FIRST WORD OF BUFFER

```

#### 5.4.5 Reenter-Me Processing

Some DSRs employ reenter-me processing to produce time delays or to allow an interrupt to bid a task. (For more information on bidding tasks from the DSR, refer to paragraph 5.4.7.) The reenter-me flag is DSFREN in PDTDSF in the PDT. You can use it in conjunction with the reenter-me address in PDTINT. When you set the reenter-me flag to one, the task scheduler sets the interrupt mask to the value in DSFINT in PDTDSF, resets the reenter-me flag to zero, and transfers control to the reenter-me address specified in PDTINT of the PDT.

In DX10, VDT DSRs use this flag to transfer control from the keyboard ISR (which runs with interrupts masked in the KSB workspace) to the I/O Call routine (which runs with interrupts unmasked in the PDT workspace). Whether your DSR uses this flag depends on the needs of your device. Keyboard devices such as the 911 VDT need to handle characters typed when a Read I/O is not active (unsolicited input). When the user enters data without a read operation pending, DSR911 queues the characters in the KSB, placing them in a Read I/O SVC data buffer only after it receives the SVC. Since DSR911 runs in the PDT workspace with all interrupts enabled, it needs to keep interrupt processing separate from call block processing to avoid interference between the DSR and ISR. Such interference is still possible between the PUTCBF and GETC routines, so the DSR inhibits interrupts for the duration of PUTCBF and GETC calls. When a character is received, and a Read I/O SVC is active, the ISR uses reenter-me because it cannot directly call the DSR. A DSR that runs with interrupts inhibited to a level that prevents concurrent execution of the ISR does not need to use the reenter-me flag. It can call the DSR (using BLWP) directly from the ISR. This allows a device to have a KSB without the extra coding necessary for reenter-me processing. The KSB and reenter-me flag provide the flexibility needed to handle these situations.

#### 5.4.6 DSR Priority Schedule

This procedure allows you to activate the DSR faster than the reenter-me processing. The operating system enters the DSR before any task executes. All operating system functions complete before the DSR is activated. PDTs are placed on a queue for processing. The routines that manage the queue make sure a PDT is only on the queue once. Byte 8 must contain a Branch instruction to the routine to process this entry.

To place an entry on the queue, put the starting address of the PDT in R0 of whichever workspace is in use. Use a BLWP instruction to branch to the routine TM\$FST. The code to place the current PDT address on the DSR priority schedule queue is as follows:

```

MOV R4,R0
AI R0,PDTLNK
BLWP @TM$FST

```

This code assumes the use of the template DSRPDT, which is offset by R4.

DX10 Release 3.6 changed the DX10 scheduler to implement DSR priority scheduling. The scheduler now returns to any map file 0 code before checking for queued PDTs or for a task. Therefore, if the DSR priority schedule routine in the DSR is interrupted, control immediately returns to the DSR.

When the PDT is selected from the queue, the interrupt mask level is set to the value contained in PDTDTF in the PDT. This value should be > F to allow interrupts to occur in a normal manner.

#### 5.4.7 Bidding Tasks from a DSR

You can execute a task as a result of an interrupt by bidding it from within a DSR. The bid takes place in the ISR and reenter-me processor. In the ISR, you set the flags for bidding the task. In the reenter-me routine, you call the subroutine TMBID0, which bids the task. (You must bid the task within the reenter-me routine to ensure that no executing task has extended the time slice.)

The task must be installed on the system program file. If you want fast execution, install it memory resident with an appropriate real-time priority.

**5.4.7.1 ISR Procedure for Bidding a Task.** The ISR must do the following preprocessing in response to an interrupt to bid a task:

1. Reset the interrupt from the device that caused the ISR to be entered.
2. Check your bid task in progress flag. If a bid task is in progress, ignore this request to bid a task.
3. Set your bid task in progress flag.
4. Provide a way to handle multiple interrupts received before the first bid is complete. Typically, you ignore subsequent bid requests until the one in progress is complete by checking the flag set in step 2.
5. Set the reenter-me flag in the PDT (bit DSFREN in word PDTDSF of the PDT).
6. Increment the value of BIDTSK by one. The scheduler uses this word to check if any bid requests are pending.
7. Check the global flag TMESLC. If zero, then set the global flag TM\$DFR to -- 1. TMESLC indicates whether a system task has inhibited scheduling and TM\$DFR forces the scheduler to execute as soon as scheduling is enabled. The following code performs this test:

```
                MOV @TMESLC,R0
                JNE CONT
                SETO @TM$DFR
CONT    < next instruction>
```

8. Exit using an RTWP instruction.

**5.4.7.2 Reenter-Me Procedure for Bidding a Task.** The reenter-me routine must do the following processing to bid a task:

1. Initialize the workspace registers as follows:

R1	Installed ID of the task in the first (upper) byte; second (lower) byte set to zero.
R2	Bid parameter #1.
R3	Bid parameter #2.
R4	Station ID for the task in the first (upper) byte (> FF if no station); second (lower) byte set to zero.
R10	Pointer to a block of memory that the DX10 routines that are used for bidding a task can use for a stack. Provide at least 15 words.
R11	Used by subroutine linkage.

2. Use the BL instruction to call the common routine TMBID0. TMBID0 places the task on the active queue. Reference the symbol TMBID0 at the beginning of the DSR.
3. Reset your bid task in progress flag.
4. Decrement the value BIDTSK.
5. Check the error returned by TMBID0 and take the appropriate action. You may ignore the bid request, bid an alternate task, or write a message to the terminal. TMBID0 returns the following information:

R0	Error code:
	0 — No error.
	1 — Invalid station number specified.
	2 — No runtime task ID available.
	3 — No system table area available.
	4 — Invalid program file LUNO.
R1	Run time ID in first byte; bit 15 indicates whether the request has been queued:
	0 — Request not queued; task was memory resident.
	1 — Request queued; task was disk resident and must be loaded.
R2	TSB address of bid task.

## 5.5 ASYNCHRONOUS DSR STRUCTURE

This paragraph describes the DSR structure for asynchronous device support. Throughout this paragraph, the term DSR refers to an asynchronous DSR. Table 5-7 shows the device and controller combinations that the asynchronous DSRs provided by TI support.

**Table 5-7. Asynchronous Device Support**

Controllers	Devices					
	931	940	Business System Terminal	810	840	85X
CI401	Y	Y	Y			
CI402	Y	Y	Y	Y		Y
CI421	Y		Y	Y <sup>1</sup>		Y <sup>1</sup>
CI422	Y		Y	Y		Y
/10A <sup>2</sup>	Y	Y	Y	Y		Y
CI403	Y	Y	Y	Y		Y
CI404	Y	Y <sup>3</sup>	Y <sup>3</sup>	Y <sup>3</sup>		Y <sup>3</sup>
931 <sup>4</sup>				Y		Y
940 <sup>4</sup>				Y	Y	Y
Business System Terminal <sup>4</sup>				Y	Y	Y

**Notes:**

<sup>1</sup> On the CI421 controller, printers are supported only on the 9902 port.

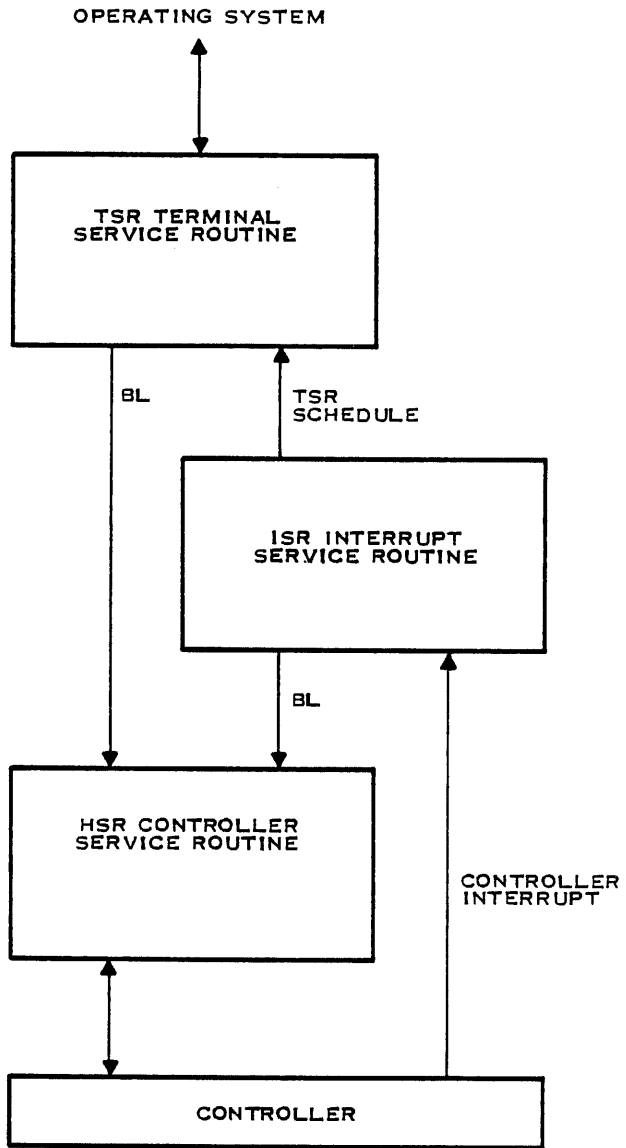
<sup>2</sup> /10A refers to the TMS9902 communications port on the 990/10A processor printed circuit board.

<sup>3</sup> These devices connect to the CI404 via the fiber optics to EIA RS-232C converter module.

<sup>4</sup> In the controller column, 931, 940, and Business System terminal refer to the auxiliary (AUX1) port of the VDT.

The asynchronous DSR design separates controller and device support into different software modules. Figure 5-17 displays a block diagram of the DSR structure. The DSR consists of three basic modules. The hardware controller service routine (HSR) module provides the controller support and is supplied by TI. The terminal service routine (TSR) module provides device support. The interrupt service routine (ISR) module has interrupt and high priority processing responsibility. You can include the TSR and ISR in the same source file if you want. The following list describes the basic functions of the DSR components:

- **TSR**
  - All DSR entry points except interrupt entry (I/O Call, Power Restored, Abort, DSR priority schedule, and reenter-me)
  - Request and completion reporting interface to DX10
  - Runs in PDT workspace
  - Provides software interface to device
  - Contains device-dependent logic
- **ISR**
  - Contains interrupt entry of the DSR
  - Interface to HSR for interrupt processing
  - High priority receive character processing
  - Runs in DSR interrupt workspace (not the PDT workspace)
- **HSR**
  - Generic (subroutine) software interface to the controller hardware
  - Contains all controller dependent logic
  - Contains all direct access to controller
  - Presents a buffered controller interface to other DSR modules



2284699

Figure 5-17. Asynchronous DSR Structure



The asynchronous DSR design can support several device and controller combinations. HSR object modules are provided with the operating system for the controllers listed in Table 5-8. Table 5-8 also documents the final node in the pathname for each HSR. The directory that contains the HSR object modules is <vol>.\$SYSGEN, where <vol> is the volume name of the disk you specify as the data disk.

**Table 5-8. HSR Object Modules**

Name	Controller Type
DS403HSR	CI403/CI404
DS923HSR	TMS 9902 and 9903 controllers *
DS401HSR	CI401 (previously COMM I/F)
<b>Note:</b>	
* Includes CI402, CI422, CI421, the 990/10A 9902 port, and the 9902 interface associated with the internal terminal on the Business System 300 computer.	

These HSR modules are available for users implementing DSRs for special devices connected to these controllers. These modules are used for DSRs following the asynchronous DSR design. This design must be followed for user-written DSRs if both of the following conditions are true:

- The special device is connected to a CI403 or CI404 controller.
- A standard TI DSR supports any of the communication channels of the CI403 or CI404.

When both of these conditions are not true, you have a choice of DSR designs for asynchronous device support. You can implement either the asynchronous DSR design or a design of your choice. Remember that any user design must obey all DX10 constraints.

### 5.5.1 Asynchronous DSR Design Overview

Figure 5-18 displays a detailed DSR-flow diagram. This figure displays data flow paths as well as the DSR logic flow. Refer to Figure 5-18 during the following discussion of the DSR logic and data flow. The TSR module contains all DSR entry points except the interrupt entry. It accepts requests from and reports completions to the I/O supervisor of DX10. The primary function of the TSR is to provide a software interface to the peripheral device. The actual functions vary considerably based on the type of device.

The TSR performs initial processing for all requests. The TSR calls the HSR for the output of data. The HSR stores output data in a transmit first-in, first-out (FIFO) buffer until the data can be transmitted on the communications line.

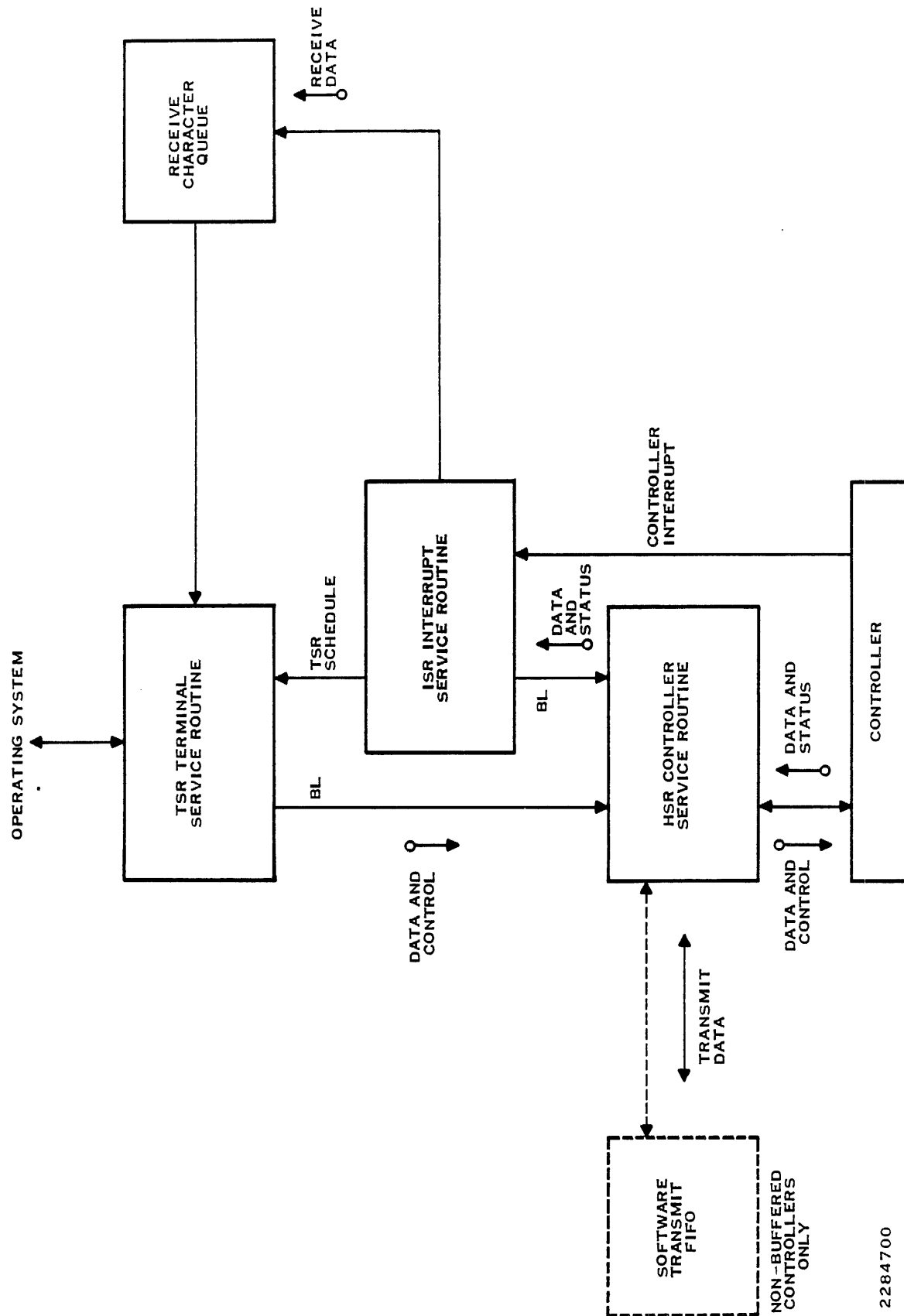


Figure 5-18. Asynchronous DSR Logic Flow

## NOTE

Buffered controllers such as the CI403 contain a hardware transmit FIFO. For non-buffered controllers such as the CI422, the HSR maintains a software transmit FIFO.

The HSR cannot accept data when the transmit FIFO fills with data waiting to be transmitted. In this event, the TSR requests notification from the ISR when the HSR can accept more data. The user writing TSRs and ISRs defines a mechanism for the TSR to make this request to the ISR. This can be a flag set by the TSR and monitored by the ISR. The HSR notifies the ISR when it can accept more transmit data, and the ISR schedules the TSR using the DSR priority schedule or reenter-me mechanism. The TSR can then resume transferring output data to the HSR. Figure 5-18 shows the logic paths followed in this process. Under normal conditions, the TSR reports completion of the output request before the HSR has actually transmitted all the data on the communications line.

Read requests, in most cases, require the cooperation of the TSR and ISR modules. This discussion assumes a receive character queue is utilized to capture unsolicited input from the peripheral device. This is the KSB queue for SCI keyboard terminals. The receive character queue is a mechanism for passing data from the ISR to the TSR as indicated in Figure 5-18. The TSR attempts to satisfy the read by moving received data from the receive character queue to the user's read data buffer. When the TSR satisfies the read, it reports completion to the user task via the operating system I/O support routines. When the receive queue does not contain enough characters to satisfy the read, the TSR must wait. To do this, the TSR requests notification from the ISR when the receive character queue contains more data. The TSR requests notification using some mechanism defined by the writer of the TSR and ISR. This mechanism can be a flag set by the TSR and monitored by the ISR. The TSR then releases control. When the ISR stores data in the receive character queue, it schedules the TSR for execution. Other I/O service requests are processed by the TSR with the aid of the ISR if required.

The ISR module contains some functions that you can consider device support and some that you can consider controller support. The ISR module contains the interrupt entry to the DSR and uses an interrupt workspace different than the PDT workspace. The ISR module runs with controller interrupts masked; it calls the HSR to decode the controller interrupt.

For the most part, ISR processing is independent of request processing of the DSR. Receive data is stored in the receive character queue even when no read request is active at the DSR. Error recovery action must be taken when the receive character queue becomes full. The ISR processes events requiring immediate attention. It also schedules the TSR module to start or resume processing.

The HSR provides access to the controller hardware. It provides a generic interface to the controller. This allows other DSR modules to be written independent of the asynchronous controller type. HSR functions include controller and communications channel initialization, transmission of data, timer services, monitoring of modem signals, and controller interrupt decoding. The HSR does not support the concept of a read request. The HSR decodes the controller interrupt and reports the cause for the interrupt to the ISR. If the cause of the interrupt was a received data character, the HSR also passes the data character back to the ISR. The HSR does not store the receive data.

### **5.5.2 Terminal Service Routine (TSR)**

The TSR module is the interface to the I/O subsystem of DX10. It must implement all the following DX10 interface functions similar to conventional DSRs:

- **DX10 I/O subsystem entry points**
  - Power Restored
  - Abort/Time-Out
  - Reenter-me
  - SVC request
  - DSR priority schedule
- **Data structures**
  - PDT
  - Buffered I/O request block
  - KSB/interrupt workspace
- **I/O subsystem routines**
  - BZYCHK
  - ENDRCD
  - GETC
  - PUTCBF, PUTEBF
  - JMCALL, BRCALL, and BRSTAT
  - KEYFUN

The following two mechanisms allow scheduling the TSR from an ISR:

- Reenter-me
- DSR priority schedule

You can use these mechanisms in any DSR whether the DSR uses the asynchronous design or not.

Both of these mechanisms enter the DSR in the PDT workspace. The DSR is reentered when the next system clock interval expires if the reenter-me mechanism is invoked. The reenter-me mechanism is described in paragraph 5.4.5. Another mechanism for scheduling DSR non-interrupt processing (such as a TSR) from DSR interrupt processing (such as an ISR) is the DSR priority schedule mechanism. This mechanism reenters the DSR after all interrupt processing for the system is complete, but before the DX10 task scheduler or any task executes. This is a more direct reentry path to the DSR. It is intended for only the highest priority (non-interrupt) processing. If you use this mechanism arbitrarily, it can interfere with high priority processing of other DSRs.

Table 5-9 describes the requirements for DSR (TSR) entry points when the DSR priority schedule mechanism is used. The first two entry point addresses (Power Up and Abort I/O) are defined by two DATA statements. These two entry point addresses reside at relative address 0 and 2 of the DSR. At DSR relative address 4, there must be a branch (B) instruction with its operand referencing the SVC entry point of the TSR. The DSR priority schedule entry point is at relative address 8.

**Table 5-9. DSR/TSR Entry Points**

Address	Code	Meaning
0000	DATA POWERU	DSR Power Up entry point address
0002	DATA ABORT	DSR Abort I/O entry point address
0004	B @SVCENT	Entry point to I/O Call routine
0008	B @PRISCH	Branch to scheduled routine

Refer to the description of the DSR priority schedule mechanism in paragraph 5.4.6 for further details. You can define other interface mechanisms between the TSR and ISR within the constraints of the operating system. Paragraph 5.6 describes the interface to the HSR.

### 5.5.3 Interrupt Service Routine (ISR)

The ISR contains the interrupt entry to the DSR and executes in the interrupt workspace of the DSR.

#### NOTE

Each channel of an interface supported by the asynchronous DSR structure must have an interrupt workspace different than the PDT workspace. The CI402, CI421, and CI422 are not considered multiple channel interfaces.

The ISR interfaces with both the TSR and the HSR as well as the operating system interrupt decoder. The design of the TSR/ISR interface is not dictated by the asynchronous DSR design. For the most part, you can specify it to fit your needs. The following list provides examples of ISR functions for standard keyboard devices:

- Bid application task
- Suspend output
- Abort output
- Abort application task (hard break)

Figure 5-19 shows the flow of control during interrupt processing. The PC component of the interrupt trap is an address within the operating system interrupt decoder. A controller interrupt trap gives control to the operating system interrupt decoder and starts its execution. The operating system decoder contains logic to determine which DSR is assigned to service the interrupting controller. The DSR is then entered at its interrupt entry point via a BLWP instruction. The workspace upon entry is the DSR interrupt workspace. This is the same workspace that the ISR uses.

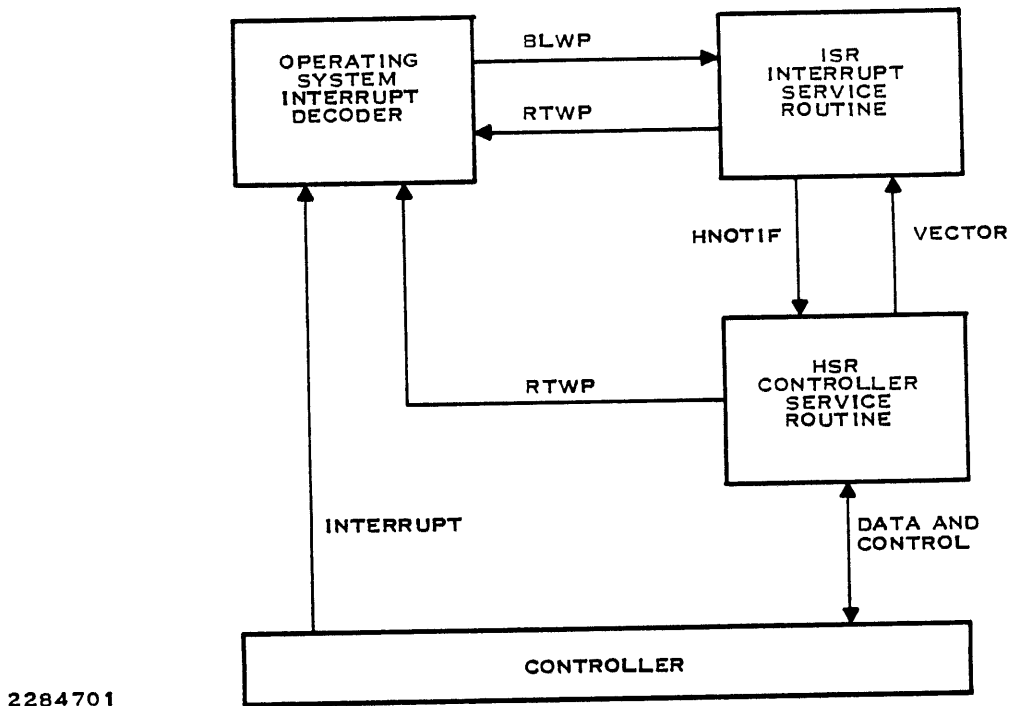
The ISR is responsible for controlling further interrupt decoding by the DSR. The ISR calls the HSR subroutine HNOTIF (refer to paragraph 5.6.9) to determine what type of controller interrupt occurred. The ISR provides return vectors for each interrupting condition of the controller. Figure 5-19 uses the word VECTOR to denote this process. The description of the HNOTIF subroutine documents the set of generic interrupt conditions. The HNOTIF subroutine takes the return vector associated with the current controller interrupt.

The ISR code for the specific type of interrupt takes the proper action to service the interrupt. When complete, the ISR returns to the operating system interrupt decoder via a RTWP instruction. The ISR can invoke the reenter-me or DSR priority schedule mechanism to cause the TSR to be executed. The operating system decoder takes the necessary action to restore normal system execution.

Figure 5-19 shows one other path for interrupt processing. The HSR exits/returns directly to the operating system interrupt decoder if the ISR calls HNOTIF when no controller interrupt is pending. This special return exists to support certain ISRs; most user ISRs never use this path.

#### **5.5.4 Hardware Controller Service Routine (HSR)**

The generic interface to the HSR consists of a set of subroutines with a branch and link (BL) call interface. A subroutine implements one or more generic functions for the specific controller in use. For example, the TSR makes a Set DTR subroutine call. The HSR for a CRU controller might implement this as a SBO DTR CRU instruction. However, the HSR for a TILINE controller may implement the same subroutine by using a SOC @DTR, @OUTSIG(R12) instruction to access the TILINE Peripheral Control Space (TPCS) of the controller. Identical requests from the TSR/ISR invoke identical functions for all controllers. Provision is made for controller hardware differences. A “not supported” return is provided for most HSR subroutines. This return is taken when the requested function is not supported by the controller hardware.



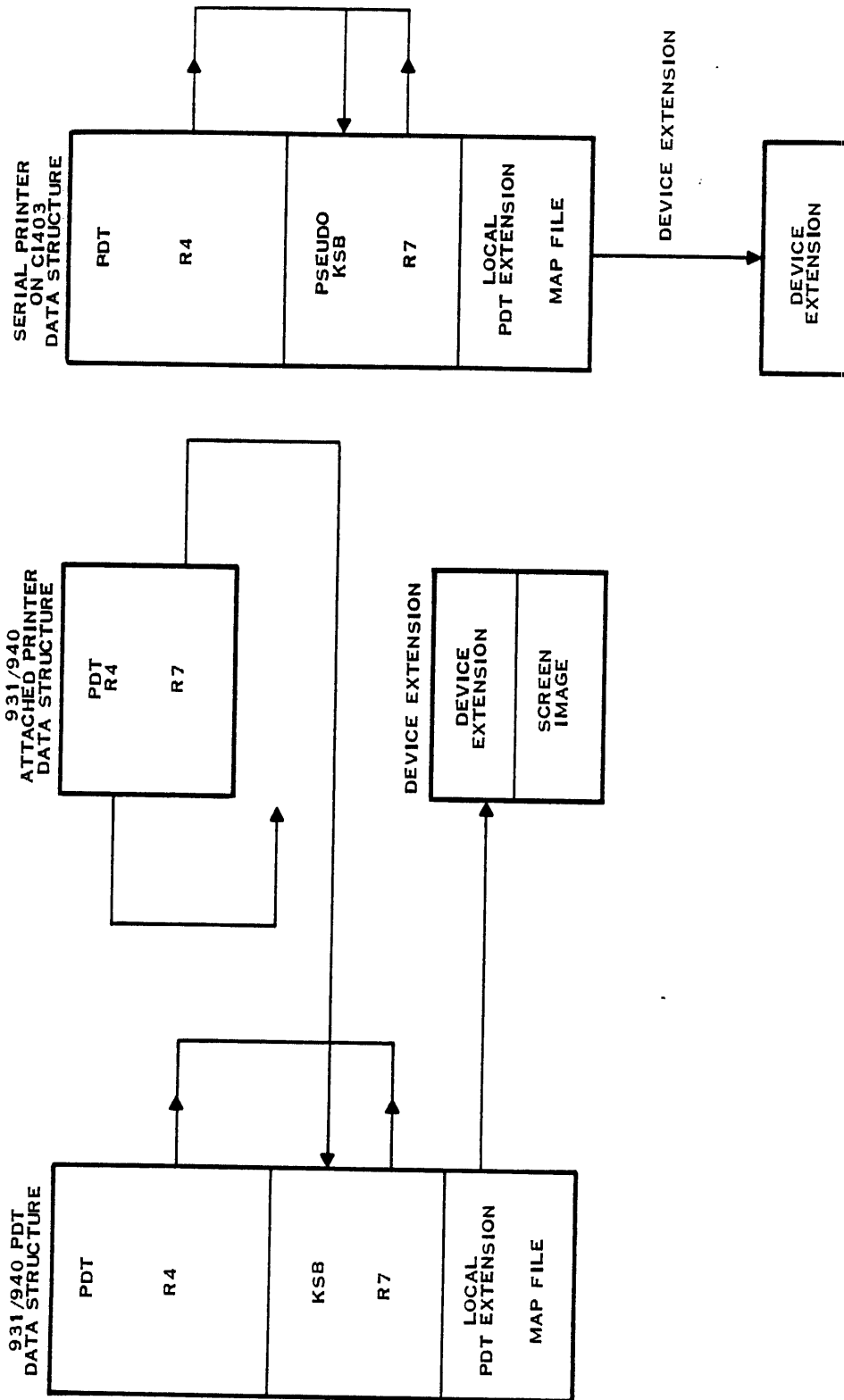
2284701

Figure 5-19. Interrupt Processing Flow

**5.5.5 Asynchronous Data Structure Allocation**

The PDT extension contains two segments. One segment is physically contiguous to the PDT. This segment contains data that requires the most frequent access. The other segment contains data for which an increased access time does not significantly affect overall performance. This second segment must be accessed with long-distance instructions. Other data structures included in the long-distance extension are VDT screen images for screen image DSRs. Refer to Figure 5-20.

All the data structures that are not accessed via long-distance instructions are allocated during system generation. These data structures are available when the operating system is loaded into memory from disk. The long-distance data structures must be allocated during the IPL of the operating system.



2284702

Figure 5-20. Asynchronous Data Structure Linkages



This discussion references labels defined by data structure templates for the PDT extension for asynchronous DSRs. The first two words of this extension, PDXSMB and PDXSMP, provide access to a second long-distance PDT extension. These words must be initialized properly for user-written DSR's that use the asynchronous DSR structure (that is, DSRs that use an HSR provided by Texas Instruments). R11 of the PDT workspace must contain the address of PDXSMB. The first word of the extension, PDXSMB, should always be initialized (in source code) to contain the value > FFFF. The second word, PDXSMP, must contain the size in bytes of the long-distance PDT extension. This size must be at least 144 bytes for non-buffered controllers and at least 32 bytes for buffered controllers (CI403/CI404). You have the option of specifying additional space for the user-written TSR/ISR. You can also lengthen the local PDT extension by reserving more memory in the PDT provided to the system generation program. The pathname of the user PDT is identified during system generation in response to the PDT FILE prompt.

## **5.6 HSR COMMON SUBROUTINES**

The information required to interface to the HSR module is as follows:

- Subroutine names
- Functions provided by each subroutine
- Subroutine calling conventions

The following paragraphs provide this information.

The following list describes the HSR subroutine classes. Each class contains several subroutines. These subroutines provide one or more HSR functions.

- Power-up initialization
- Write output signal or function
- Read input signal or function
- Enable/disable status change notification
- Output a character
- Write operational parameters
- Read operational parameters
- Request timer interval notification
- Controller interrupt decoding

All HSR subroutines are called via BL instructions. Thus, they use the caller's workspace during execution. Parameters required by the subroutines are passed to the HSR in workspace registers. Information is returned to the caller in one of two ways. Data is returned to the caller in workspace registers. Other status is returned via alternate subroutine returns. The caller specifies alternate return addresses as operands of assembler DATA directives immediately following the BL subroutine call. The following shows an example of HSR alternate return addresses:

BL	@HSRSUB	SUBROUTINE CALL
DATA	ALT1	FIRST ALTERNATE RETURN
DATA	ALT2	SECOND ALTERNATE RETURN
< next instruction >		NORMAL RETURN (CODE)

The caller execution resumes at one of the alternate return addresses or at the normal return address (the instruction following all alternate return DATA statements). The number of alternate returns varies for different HSR subroutines.

The HSR subroutines follow some general register conventions. The subroutines normally use R0 and R10 as working registers. These two registers are also used when parameters are passed to or from the HSR. Exceptions are noted in some of the HSR subroutine descriptions. In most cases, R7 is used as a pointer to the PDT. This pointer points to the end of the required PDT. R12 contains the TILINE or CRU base address of the controller or controller channel. Other register usage is documented with specific HSR subroutines.

### 5.6.1 Power-Up Initialization

This subroutine class allows the HSR to perform any initialization required before operation begins. For the CI403 and CI404, the HSR must ensure that the controller has successfully executed the self-test for each channel specified during system generation. For other controllers, the HSR may be required to build a software transmit FIFO in a long-distance memory buffer that the operating system obtains.

The three subroutines that perform power-up initialization functions are HRESET, HSWPWR, and HMRST.

The HRESET subroutine performs power-up initialization that must be performed for all I/O channels of the controller. This subroutine can be called once per channel for multiple channel controllers. However, there is only one master reset of the controller for each power-up occurrence.

The HSWPWR subroutine performs all channel-oriented initialization. The HSR data structures for the channel are initialized. Controller interrupts are enabled when the normal return is taken.

The HMRST subroutine performs the same initialization functions as the HRESET routine except that a master reset of the controller is unconditionally performed. This subroutine is provided so that diagnostic software can force a controller master reset for testing purposes.

The TSR normally makes two calls to the HSR for power-up initialization. The HRESET subroutine is called first, followed by the HSWPWR subroutine. These two subroutines are called for each channel of the controller. The calling conventions are identical for each of the HSR power-up subroutines. If HPOWER is considered a synonym for any of the three power-up subroutine names, then the calling conventions for all HSR power-up subroutines are as follows:

Calling convention:

```

BL    @HPOWER
DATA  XXXX          POWER UP FAILURE RETURN VECTOR
< next instruction> NORMAL RETURN (CODE)
    
```

### 5.6.2 Write Output Signal or Function

The subroutine names for setting output signals or functions to 1 (logic true) are of the form HSTxxx, where xxx specifies a signal or function. The subroutine names for resetting output signals or functions to 0 (logic false) are of the form HRTxxx, where xxx specifies a signal or function. The following list describes the output signals supported by HSRs:

Subroutine	Signal
HSTAL, HRTAL	AL — Analog Loopback
HSTDTR, HRTDTR	DTR — Data Terminal Ready
HSTRTS, HRTRTS	RTS — Request to Send
HSTSRS, HRTSRS	DSRS — Data Signal Rate Select
HSTSRT, HRTSRT	SRTS — Secondary Request to Send (or Reverse Channel Signal)

The following list describes the output functions supported by HSRs:

Subroutine	Function
HSTBIL, HRTBIL	BIL — Board Internal Loopback
HSTCR, HRTCR	CR — Channel Reset
HSTCTH, HRTCTH	CTH — Channel Transmitter Halt
HSTRS, HRTRS	RS — Receiver Squelch
HSTTB, HRTTB	TB — Set Transmit Break Condition
HSTUIL, HRTUIL	UIL — UART Internal Loopback

The following paragraphs describe each of the functions in detail.

**HSTBIL Subroutine.** The Set Board Internal Loopback subroutine HSTBIL sets a more general (controller or board) internal loopback mode than the universal asynchronous receiver/transmitter (UART) internal loopback mode. None of the current asynchronous controllers support this second level of loopback.

**HSTCR Subroutine.** The Set Channel Reset subroutine HSTCR performs a hardware reset of the channel. This does not disturb any other communication channels, nor does it enable the channel interrupts. The HRTCR subroutine enables interrupts and allows normal operation. The following example shows a typical sequence for TSR use of these subroutines:

1. TSR issues an HSTCR call to quiet HSR activity on the channel.
2. TSR issues an HSWPWR call to initialize the HSR data structures and status.
3. TSR initializes its channel data structures and status.
4. TSR issues an HRTCR call to begin normal operation.

**HSTCTH Subroutine.** The Set Channel Transmitter Halt subroutine HSTCTH temporarily suspends output of data to the communications line. Any data in the transmit FIFO is not transmitted. The HRTCTH subroutine resumes transmission of data in the transmit FIFO.

**HSTRS Subroutine.** The Set Receiver Squelch subroutine HSTRS enables half-duplex operation. The receiver squelch function disables reception of data during transmission. The HRTRS subroutine turns the receiver squelch off and allows full-duplex operation.

**HSTTB Subroutine.** The Set Transmit Break subroutine HSTTB initiates the transmission of a break sequence (spacing/logic 0). This continues until stopped with the HRTTB subroutine.

**HSTUIL Subroutine.** The Set UART Internal Loopback subroutine HSTUIL places the UART (communications chip) for the channel in loopback mode. In general, this causes the UART to return all transmitted data as received data on the same channel. Refer to UART documentation for more detailed information. The HRTUIL subroutine changes the UART from UART internal loopback to normal mode.

The calling conventions for the HSTxxx and HRTxxx subroutines are identical. The following calling convention uses the HSTxxx name as an example:

Calling convention:

BL @HSTxxx  
DATA VVVV  
< next instruction >

WHERE xxx IS DTR, RTS, ETC.  
SIGNAL NOT SUPPORTED RETURN VECTOR  
NORMAL EXIT (CODE)

### 5.6.3 Read Input Signal or Function

This subroutine class allows reading controller input signals and HSR function states. The HSR subroutine name is of the form HRDxxx, where xxx identifies a signal or function. The following list describes the input signals or function states that can be read:

Subroutine	Signal/Function
HRDBIL	BIL — Board Internal Loopback (Signal)
HRDCR	CR — Channel Reset (Function)
HRDCTH	CTH — Channel Transmission Halted (Function)
HRDCTS	CTS — Clear to Send (Signal)
HRDDCD	DCD — Data Carrier Detect (Signal)
HRDDSR	DSR — Data Set Ready (Signal)
HRDRI	RI — Ring Indicator (Signal)
HRDSCT	SCTS — Secondary Clear to Send (Signal)
HRSDCD	SDCD — Secondary Data Carrier Detect (or Speed Indication Signal)
HRDSSS	SSS — Split Speed Supported (Function)
HRDTB	TB — Transmit Break (Function)
HRDUIL	UIL — UART Internal Loopback (Signal)

#### Calling Convention:

BL @HRDxxx	WHERE xxx IS DSR, CTS, ETC.
DATA WWWW	SIGNAL NOT SUPPORTED RETURN VECTOR
DATA YYYY	CONTROLLER FAILURE RETURN VECTOR
DATA ZZZZ	SIGNAL FALSE RETURN VECTOR
< next instruction >	SIGNAL TRUE RETURN (CODE)

### 5.6.4 Enable/Disable Status Change Notification

This subroutine class provides a mechanism for the ISR (and therefore, indirectly the TSR) to receive status change notification from the HSR. There are subroutines to enable notification and to disable notification. Once enabled, most of the signals or functions that are supported remain enabled until explicitly disabled.

The Transmit Shift Register Empty (TSRE) function is an exception to this rule. If the HSR makes a TSRE status notification, it automatically disables further notification for this same condition. You must enable the TSRE function with another HDSTSR call to the HSR if you want subsequent notification. Refer to the HSR interrupt decoder description for more details about the method of notification.

Notification is made when the signal changes from 0 to 1 or from 1 to 0 for the first five signals. Some controllers notify only on the ring signal changing from 0 to 1. The TSRE notification is made only when the condition occurs.

The subroutine names for enabling status change notification are of the form HESxxx, where xxx specifies a signal or function. The subroutine names for disabling status change notification are of the form HDSxxx, where xxx specifies a signal or function. The following list describes the notification conditions that the HSR supports:

<b>Subroutine</b>	<b>Status Change Notification</b>
HESCTS, HDSCTS	CTS — Clear to Send
HESDCD, HSDSCD	DCD — Data Carrier Detect
HESDSR, HSDSR	DSR — Data Set Ready
HESRI, HDSRI	RI — Ring Indicator
HESST, HDSSCT	SCTS — Secondary Clear to Send
HESSDC, HDSSDC	SDCD — Secondary Data Carrier Detect
HESSTSR, HDSTSR	TSRE — Transmit Shift Register Empty

The calling conventions for the HESxxx and HDSxxx subroutines are identical. The following calling convention uses the HESxxx name as an example:

Calling Convention:

BL @HESxxx	WHERE xxx IS DSR, RI, ETC.
DATA VVVV	NOT SUPPORTED RETURN VECTOR
< next instruction >	NORMAL RETURN (CODE)

### 5.6.5 Output a Character

This HSR subroutine accepts characters to be output on the communications channel. The subroutine provides a character interface to the output channel (that is, only one character is passed to the HSR for each HOUTPx subroutine call). In all cases, the output data is stored in a transmit FIFO before transmission on the communications line. For buffered controllers, the FIFO is on the hardware controller. For non-buffered controllers, the FIFO is a software data structure that the HSR manages. An alternate (character not output) return from the HOUTPx routine is taken if the FIFO becomes full. The caller is responsible for saving the data character. The HSR notifies the ISR when the transmit FIFO is empty. This notification takes place as a “transmit interrupt” exit from the HSR interrupt decoder subroutine. This notification causes the output data to flow to the HSR again. Refer to the HSR interrupt decoder subroutine description for more details.

The output character subroutines are HOUTP4 and HOUTP7. The only difference in these two subroutines is that workspace register four (R4) contains the PDT pointer for the HOUTP4 subroutine and R7 contains the PDT pointer for the HOUTP7 subroutine. Refer to the PDTDIB description in Table 5-2.

Calling Convention:

BL	@HOUTPx	x = 4 IF R4 IS PDT POINTER
		x = 7 IF R7 IS PDT POINTER
DATA	XXXX	CHARACTER NOT OUTPUT — RETURN VECTOR
	< next instruction >	NORMAL (CHARACTER OUTPUT) RETURN

where:

R4 or R7 contains the pointer to the PDT.  
 R5 is the output character, left byte.

Volatile Registers:

R0 and R5  
 R5 preserved for character not output.

**5.6.6 Write Operational Parameters**

The following list describes the operational parameters that the HSRs support:

- Baud rate selection — The transmit baud rate and the receive baud rate are specified in the most significant byte (MSB) and least significant byte (LSB) of R0, respectively. The MSB and LSB must be identical for controllers not supporting split speed.
- Data format:
  - Parity selection: even, odd, mark, space, or none.
  - Character length selection: 5, 6, 7, or 8 bit data.
  - Stop bit selection: 1, 1.5, or 2 stop bits.

The write parameters subroutines are Set Channel Speed (Baud Rate) and Set Data Character Format. The calling conventions for these two subroutines are very similar. The only difference is the parameter information passed in R0.

**5.6.6.1 Set Channel Speed (Baud Rate).** This subroutine specifies the transmit and receive speeds for the channel. The transmit and receive speeds may differ only when split speeds are supported.

Calling Convention:

BL     @HSPSPD	SET CHANNEL SPEED
DATA  VVVV	PARAMETER NOT SUPPORTED RETURN VECTOR
< next instruction >	NORMAL RETURN (CODE)

where:

R0   MSB contains the transmit speed code;  
       LSB contains the receive speed code.  
       Refer to Table 5-10.

**Table 5-10. HSR Baud Rate Codes**

Speed Code	Baud Rate
00	50
01	75
02	110
03	134.5
04	150
05	200
06	300
07	600
08	1,200
09	1,800
0A	2,400
0B	3,600
0C	4,800
0D	7,200
0E	9,600
0F	14,400
10	19,200
11 – FF	Reserved



**5.6.6.2 Set Data Character Format.** This subroutine sets the character length, parity selection, and the number of stop bits for data characters.

Calling Convention:

BL    @HSPPSL DATA VVVV < next instruction >	SET DATA CHARACTER FORMAT PARAMETERS PARAMETER NOT SUPPORTED RETURN VECTOR NORMAL RETURN (CODE)
--	---

where:

R0    contains the parameter information in the following format:

Bit	Contents
0 – 1	Reserved
2 – 3	Parity selection: 00 = odd parity 01 = even parity 10 = mark parity 11 = space parity
4	Parity enable (if 1)
5 – 7	Reserved
8 – 9	Number of stop bits: 00 = 1 stop bit 01 = 1.5 stop bits 10 = reserved 11 = 2 stop bits
10 – 11	Data character length: 00 = 5 bit character 01 = 6 bit character 10 = 7 bit character 11 = 8 bit character
12 – 15	Reserved

**5.6.7 Read Operational Parameters and Information**

This class of subroutines allows the following operational parameter values to be read from the HSR:

Subroutine	Operational Parameter Value
HRPDAT HRPPSL	HSR module revision level Data format: Parity selection: even, odd, mark, space, or none Character length selection Stop bit selection
HRPSPD HRPTYP	Baud rate Controller type ID

The parameter information is returned in R0 of the caller's workspace. For the HRPSPD and HRPPSL subroutines, the format of the information in R0 is identical to the format in the HSPSPD and HSPPSL subroutines, respectively.

Calling Convention:

```
BL    @HRPxxx           WHERE xxx IS SPD OR PSL
< next instruction >   RETURN
```

The HRPDAT subroutine returns the current revision level of the HSR software module in R0. The revision level is a hexadecimal number starting at 0 for the initial level and incrementing by 1 for each revision. The HRPTYP subroutine returns a code right justified in R0 that identifies the controller type. Table 5-11 lists controller type codes.

**Table 5-11. Controller Type Codes**

Code	Controller
> 0001	CI401 (previously COMM I/F)
> 0006	Business System 300 internal 9902 port
> 0007	990/10A 9902 port
> 0008	CI402
> 0009	CI421 9902 port
> 000A	CI422
> 0023	CI403
> 0024	CI404
> 0030	CI421 9903 port

### 5.6.8 Request Time Interval Notification

This subroutine (HTIMER) requests notification after a specified time interval. You specify the time interval as some number of 250-millisecond periods. The HSR interrupt decoder performs the notification by taking the timer interrupt vector return to the ISR. Refer to the discussion of the HSR interrupt decoder for more details. You disable timer notification by specifying a zero as the number of 250-millisecond intervals (R0 = 0). A "not supported" exit is not provided for this subroutine.

Calling Convention:

```
BL    @HTIMER
```

where:

R0 specifies the number of 250-millisecond intervals.

### 5.6.9 Controller Interrupt Decoder

The ISR calls this subroutine (HNOTIF) to perform controller interrupt decoding. The subroutine executes in the DSR interrupt workspace and with interrupts masked to the interrupt level of the controller channel. The ISR provides several return vector addresses via DATA directives immediately following the call. A return vector is provided for each interrupt type possible from the controller. If the subroutine finds no controller interrupt pending, the return is to the operating system interrupt decoder rather than to the caller.

#### Calling Convention:

```

BL    @HNOTIF
DATA  XXXX    RECEIVE INTERRUPT VECTOR
DATA  YYYY    TRANSMIT INTERRUPT VECTOR
DATA  ZZZZ    SIGNAL OR FUNCTION CHANGE VECTOR
DATA  AAAA    TIMER INTERRUPT VECTOR
DATA  BBBB    ILLEGAL/INVALID INTERRUPT VECTOR

```

#### Receive Interrupt Return:

R10 Received character left byte; line status in right byte. Line status is as follows:

Bit	Meaning
0 – 2	Reserved
3	Break received
4	Framing error
5	Parity error
6	Overrun error
7	Reserved

#### Transmit Interrupt Return:

This return is taken when the transmit FIFO empties.

Signal or Function Change Return:

R10 Current signal or function states are returned in bits 0 – 3 and bits 8 – 10. Bits 4 – 7 and 12 – 15 are delta flags that indicate which signals or functions changed.

Bit	Contents
0	DCD
1	RI
2	DSR
3	CTS
4	Delta DCD
5	Delta RI
6	Delta DSR
7	Delta CTS
8	SCTS
9	SDCD
10	TSRE
11	Reserved
12	Delta SCTS
13	Delta SDCD
14	Delta TSRE
15	Reserved

## 5.7 COMMON ROUTINES

The following paragraphs describe the routines provided by DX10 for use in your DSR. Using these routines can reduce the code required for multiple routines that frequently perform the same operations. You can also use them to remove DX10-specific routines from your DSR.

### 5.7.1 BRCALL — Branch Table Call Routine

This routine decodes the I/O sub-opcode in PRBOC in the buffered I/O request block and transfers control to the appropriate routine. When you enter BRCALL, R1 must contain the address of the buffered I/O request block, which contains the I/O sub-opcode in its first byte, as defined in paragraph 5.3.3. If you use the alternate entry point BRCALT, R0 must contain the sub-opcode. The routine leaves the contents of R1 unchanged, but modifies R0 and R11.

**5.7.1.1 Parameters.** You must also supply a parameter list that contains the maximum number of sub-opcode processors, an error return address, and the address of each processor in order corresponding to the sub-opcode values. BRCALL decodes the sub-opcode and uses it as an index to the parameter list, where it obtains the address of the processor for the sub-opcode.

### 5.7.1.2 Calling Sequence.

```

BL      @BRCALL (or @BRCALT)
DATA   < highest legal sub-opcode (n)>
DATA   < error return address>
DATA   < sub-opcode 0 processor address>
DATA   < sub-opcode 1 processor address>
      .
      .
      .
DATA   < sub-opcode n processor address>

```

### 5.7.2 BZYCHK — Busy Check Routine

This routine determines whether the device was busy at the time of a power failure. You should call it in your Power Restored routine, which receives control when power is reapplied. BZYCHK runs in the PDT workspace. BZYCHK tests the busy flag (bit 1 of PDTDSF) in the PDT.

- If it finds the busy flag set to one, BZYCHK checks the ENDRCD flag. If it is set, BZYCHK returns to the first word following the call. If it is not set, BZYCHK sets the reenter-me flag. In the call block pointed to by R7, BZYCHK sets the status code of the call block (PRBEC) to > 04 to indicate that the device has lost a record due to the power failure. It also sets the error flag (bit 1 of PRBSFL). Then, it returns control to the DSR at the second word following the call. That word should contain a call to the end-of-record processor.
- If it finds the busy flag set to zero, BZYCHK returns control to the DSR at the word following the call. The instruction in this word should be a jump to the appropriate routine for handling the not-busy condition.

**5.7.2.1 Parameters.** BZYCHK does not have formal parameters other than the instructions for branching to the busy and not-busy routines. On return, the contents of R0 have been changed and not restored.

#### 5.7.2.2 Calling Sequence.

```

MOV    @PDTSRB – PDTSIZ(R4),R7  LOAD THE CALL ADDRESS INTO REGISTER R7
BL     @BZYCHK                  BRANCH AND LINK TO BZYCHK
JMP    NOTBSY                   JUMP TO NOT-BUSY ROUTINE
< busy routine>

```

### 5.7.3 ENDRCD — End-of-Record Routine

This routine notifies the operating system that the I/O operation has terminated by setting the end-of-record-needed flag (DSFEOR in PDTDSF) in the PDT. You should return control to the operating system with a RTWP instruction following the return from ENDRCD. The operating system then returns the I/O request to the user task. If the DSR has detected an error, it should set the error flag (bit 1 of PRBSFL) in the buffered I/O request block before calling ENDRCD.

The ENDRCD routine uses the following registers and does not restore them when it returns:

R0	R9
R1	R10
R8	R11

**5.7.3.1 Parameters.** ENDRCD has no parameters other than PDTSRB must point to the I/O subopcode byte of the buffered I/O request block for the SVC whose I/O is completed.

#### 5.7.3.2 Calling Sequence.

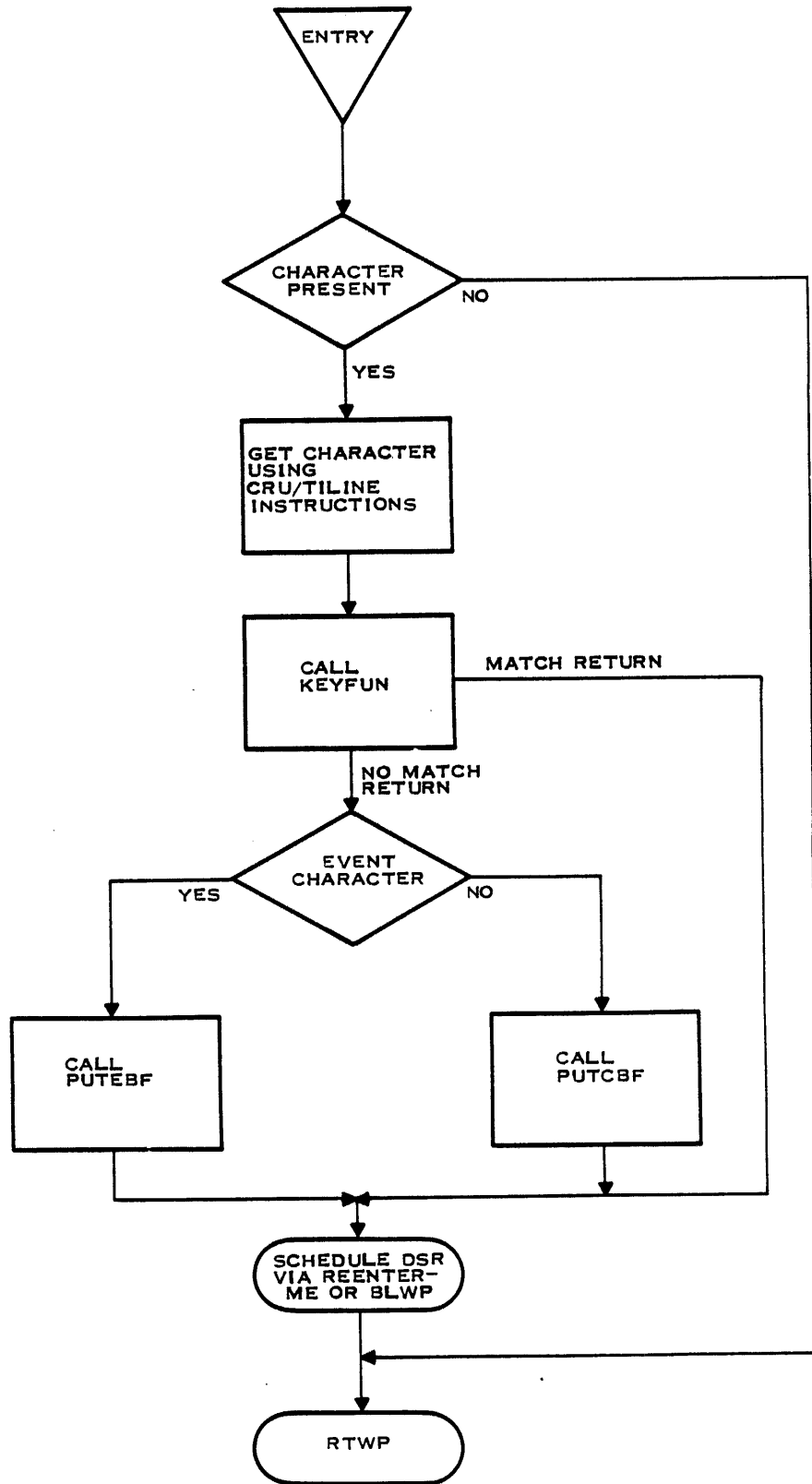
BL	@ENDRCD	BRANCH AND LINK TO ENDRCD
RTWP		RETURN CONTROL TO DX10

### 5.7.4 KEYFUN — Keyboard Function Routine

This routine processes four special keys found on keyboard devices. These keys allow the keyboard user to hold I/O, abort I/O, bid a task, or kill a task, depending on the sequence used.

HOLD	Stops normal character interpretation by the DSR. The next keystroke determines the subsequent action. If the user presses a key other than HARD BREAK, ABORT, or BID, I/O resumes from the point where it was interrupted. (The Attention key serves as the HOLD key.)
ABORT	Aborts the I/O being processed. (The Return key serves as the ABORT key.) The active SVC call terminates with an error code of 06.
BID	Bids SCI. (The exclamation point (!) key serves as the BID key.) The operating system bids task 2 in .S\$PROGA when this flag is set.
HARD BREAK	Terminates the task running at the terminal. (The Control/X key combination serves as the HARD BREAK key.) The DSR does not need to take any special action for this to occur. The operating system bids task 2 in .S\$PROGA when this flag is set.

KEYFUN uses the KSB workspace and expects the input character to be in the upper byte of R10. It compares the character to the HOLD, ABORT, BID, and HARD BREAK key values. When it detects a match, it sets the appropriate flag in R6 of the KSB and takes the special exit, which should skip normal DSR character buffering. Figure 5-21 depicts the use of KEYFUN in keyboard interrupt processing.



228 3074

Figure 5-21. Keyboard Interrupt Processing

**5.7.4.1 Parameters.** The four bytes following the KEYFUN call must contain the values for the HOLD, ABORT, BID, and HARD BREAK keys. The following word should contain the address of the special exit used to process those keys.

**5.7.4.2 Calling Sequence.**

MOV	@CHAR,R10	R10 <= CHARACTER
BL	@KEYFUN	BRANCH AND LINK TO KEYFUN
BYTE	< hold key>	'ATTENTION' KEY
BYTE	< abort key>	'RETURN' KEY
BYTE	< bid key>	'EXCLAMATION POINT' KEY
BYTE	< hard break key>	'CONTROL/X' KEY SEQUENCE
DATA	< special exit>	RETURN ON MATCH
	< next instruction>	RETURN ON NO MATCH

**5.7.5 GETC — Get Character Routine**

This routine retrieves a character from the KSB receive character queue for an interactive terminal. It uses the workspace in the PDT and assumes that R4 points to the KSB.

GETC checks if the LUNO was opened for event characters and the call uses an extended call block. If not, GETC discards event characters until it finds a data character. If the LUNO is opened for event characters and the call uses an extended call block, GETC processes the first character it removes. The character is processed according to Table 5-12.

**Table 5-12. Character Ranges Used by GETC**

Code	Character Type
> 00 – > 7F	Data
> 80 – > 86	Event
> 87 – > 95	Data
> 96 – > 9A	Event
> 9B	Ignored. Another character is removed.
> 9C – > 9F	Event
> A0 – > FF	Data

If the LUNO was opened for event characters but the call does not use an extended call block, the event character is placed back in the queue and the event character exit is taken. A Get Event Character SVC must be issued to remove the event character before more data characters can be read.

The character is placed in R9 left justified and control is returned to the routine indicated in the Table 5-12.



If GETC finds no characters in the input buffer, it places the empty buffer routine address in R6 and executes an RTWP instruction, returning control to the system. Because of this return, keyboard DSRs that use GETC must use a special convention for returning control to the system. The DSR must load R6 with the address of the processor for the next interrupt before each return to the system, and the DSR entry for processing interrupts should consist of a branch to the address in R6, as follows:

```
ISRENT MOV @PDTSRB(R4),R1  R1 <= SAVED PRB ADDRESS (PDTSRB)
          B      *R6          RESUME EXECUTION
```

**5.7.5.1 Parameters.** The call to GETC must be followed by two return points. The first provides a vector to the routine that handles an empty character buffer. The second provides a vector that handles event characters. The third return point is the instruction following the event character vector and is the routine for processing the data characters.

#### 5.7.5.2 Calling Sequence.

```
BL      @GETC
DATA    < empty buffer routine>
DATA    < event character routine>
        < first instruction of the data character routine is here>
```

#### 5.7.6 JMCALL — Jump Table Call Routine

This routine decodes the I/O sub-opcode in the buffered I/O request block and transfers control to the appropriate routine. When you enter JMCALL, R1 must contain the address of the buffered I/O request block, which contains the I/O sub-opcode in its first byte as described in paragraph 5.3.3. If you use the alternate entry point JMCALT, R0 upper byte must contain the sub-opcode. The routine leaves the contents of R1 unchanged, but modifies R0 and R11.

**5.7.6.1 Parameters.** You must also supply a parameter list that contains byte entries for the maximum number of sub-opcode processors, an error return address, and the address of each processor in order corresponding to the sub-opcode values. The entries for the error and sub-opcode processors give the offset in words (positive eight-bit value) from the label of the first byte in the list to the beginning of the processor. (Since the byte values are unsigned displacements, the displacement cannot exceed 255 words and cannot reference a processor that precedes the table.) For example, if the table begins at JTABLE and a processor begins at PROC1, you should use the following jump table entry for PROC1:

```
BYTE (PROC1 – JTABLE)/2
```

JMCALL decodes the sub-opcode and uses it as an index to the parameter list, where it obtains the offset for the processor for the sub-opcode. This index is zero based. If the sub-opcode is greater than the highest legal sub-opcode, the routine takes the error routine exit. R0 is destroyed.

### 5.7.6.2 Calling Sequence.

```
< table> BL    @JMCALL (or @JMCALE)
          BYTE < highest legal sub-opcode (n)>
          BYTE (< error return> - < table>)/2
          BYTE (< sub-opcode 0 processor> - < table>)/2
          BYTE (< sub-opcode 1 processor> - < table>)/2
          .
          .
          .
          .
          .
          BYTE (< sub-opcode n processor> - < table>)/2
```

### 5.7.7 PUTCBF — Put Character in Buffer Routine

This routine places a data character into the KSB input character queue.

**5.7.7.1 Parameters.** PUTCBF expects to find a data character in the upper byte of R10 of the KSB workspace and the address of the input character queue in R2 and R3. If it finds the queues full, PUTCBF returns control to the routine indicated in the word following the call. Otherwise, it returns control to the second word following the call.

#### 5.7.7.2 Calling Sequence.

```
MOV  @CHAR,R10           R10 < = DATA CHARACTER
BL   @PUTCBF             CALL PUTCBF
DATA < buffer full routine> RETURN IF BUFFER FULL
< next instruction>     NORMAL RETURN
```

### 5.7.8 PUTEBF — Put Event Character in Buffer Routine

This routine places an event character in the KSB character queue.

**5.7.8.1 Parameters.** PUTEBF sets bit 0 of R10 of the KSB workspace to a one to make it an event character. If it finds the event character queue full, PUTEBF returns control to the routine indicated in the word following the call. Otherwise, it returns control to the second instruction following the call.

#### 5.7.8.2 Calling Sequence.

```
MOV  @CHAR,R10           R10 < = EVENT CHARACTER
BL   @PUTEBF             CALL PUTEBUF
DATA < buffer full routine> RETURN IF BUFFER FULL
< next instruction>     NORMAL RETURN
```

**5.7.9 SETWPS — Set Interrupt Mask Routine**

This routine resets the interrupt mask so that the I/O Call routine can complete before the ISR must handle any interrupts. Your I/O Call routine should begin with a LIMI instruction to prevent an interrupt for the device during the execution of SETWPS.

**5.7.9.1 Parameters.** SETWPS expects R2 of the DSR workspace to contain the current interrupt mask. It does not alter the workspace pointers.

**5.7.9.2 Calling Sequence.**

< initial entry >	LIMI 0	MASK ALL INTERRUPTS
	BL @SETWPS	CALL SETWPS
	< next instruction >	RETURN POINT

**5.7.10 BRSTAT — Branch Table Call with Statistics Routine**

This routine performs the same function as BRCALL and JMCALL but also keeps a count for each type of I/O call. Each time BRSTAT is entered, an appropriate statistics counter is incremented in the PDT before branching to the specific opcode processor.

**5.7.10.1 Parameters.** BRSTAT requires two tables: one that lists opcode processor entry points and one that lists offsets to counters in the PDT. The opcode processor entry table must immediately follow the BL @BRSTAT call and have as its first word a count of the number of entry points in the table. R10 is either 0 for no statistics, or it is the address of the statistics table. The table contains a byte entry for each opcode handled by the DSR. If the byte value is 0, no statistics for that opcode are kept. Otherwise, the byte is used as an offset into the PDT from R4. (Refer to the asynchronous DSR example in Figure 5-22.)

**5.7.10.2 Calling Sequence.**

LI	R10,STATTB	
BL	@BRSTAT	
DATA	< highest legal sub-opcode (n) >	
DATA	< error return address >	
DATA	< sub-opcode 0 processor address >	
DATA	< sub-opcode 1 processor address >	
.	.	.
.	.	.
.	.	.
DATA	< sub-opcode n processor address >	
.	.	.
.	.	.
.	.	.
STATTB BYTE	> FC,> FD,> FC	

```
SDSMAC                                14:12:26 WEDNESDAY, JUN 28, 1978.
ACCESS NAMES TABLE                    PAGE 0001

SOURCE ACCESS NAME=                    SYSBLD.DEVDSR.SOURCE.LPDSR
OBJECT ACCESS NAME=                    SYSBLD.DEVDSR.OBJECT.LPDSR
LISTING ACCESS NAME=                   SYSBLD.DEVDSR.LIST.LPDSR
ERROR ACCESS NAME=
OPTIONS=                                XREF, TUNLST
MACRO LIBRARY PATHNAME=

LINE   KEY  NAME
0056   A   DSC.SYSTEM.TABLES (PDT)
        =>SYSBLD.SYSTEM.TABLES (PDT)
0057   B   DSC.SYSTEM.TABLES (LDT)
        =>SYSBLD.SYSTEM.TABLES (LDT)
0058   C   DSC.SYSTEM.TABLES (PRB)
        =>SYSBLD.SYSTEM.TABLES (PRB)
```

**Figure 5-22. Example DSRs (Sheet 1 of 60)**

LPDSR

14:12:26 WEDNESDAY, JUN 28, 1978

PAGE 0002

```

0001 *
0002 *(C) COPYRIGHT, TEXAS INSTRUMENTS INCORPORATED, 1977. ALL
0003 *RIGHTS RESERVED. PROPERTY OF TEXAS INSTRUMENTS INCORPOR-
0004 *ATED. RESTRICTED RIGHTS - USE, DUPLICATION OR DISCLOSURE
0005 *SUBJECT TO RESTRICTIONS SET FORTH IN TI'S PROGRAM LICENSE
0006 *AGREEMENT AND ASSOCIATED DOCUMENTATION.
0007     IDT 'LPDSR'
0008
0009 *O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O
0010 * TITLE:    LPDSR - LINE PRINTER DEVICE SERVICE ROUTINE
0011 * REVISION: 04/01/76 - ORIGINAL
0012 * REVISION: 06/01/76 - BLANK ADJUSTMENT
0013 * REVISION: 11/29/76 - DX10 REL. 3.0
0014 * COMPUTER: 990
0015 * ABSTRACT:
0016 *     FOR PDTEMP = 0                 ST$DM EQU 0
0017 *     THIS IS THE HANDLER FOR I/O TO VARIOUS MODELS
0018 *     OF PRINTERS INTERFACED VIA THE STANDARD TI
0019 *     PARALLEL INTERFACE, DATA MODULE.
0020 *     FOR PDTEMP = 2                 ST$EIA EQU 2
0021 *     THIS IS THE HANDLER FOR I/O TO VARIOUS MODELS
0022 *     OF THE CENTRONICS AND TI LINE PRINTERS, THAT
0023 *     USE AN RS-232-C INTERFACE.
0024 *
0025 *O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O
0026 *
0027 *     EXTERNAL DEFINITIONS
0028 *
0029 *     DEF LPHAN             DSR ADDRESS
0030 *     DEF LPINT            INTERRUPT ENTRY
0031 *     DEF LPSPUR           SPURIOUS INTERRUPT CLEAR
0032 *
0033 *     EXTERNAL REFERENCES
0034 *
0035 *     REF SETWPS           SAVE WPS ROUTINE
0036 *     REF ENDRCD           END-OF-RECORD ROUTINE
0037 *     REF BZYCHK           PDT BUSY CHECK ROUTINE
0038 *     REF JMCALL           IOCOMX - JUMP CALL PROCESSOR
0039 *     REF ASCCHK           ASCII CHARACTER CHECK
0040 *     REF BYTE40
0041 *
0042 *     CRU BIT EQUATES
0043 *
0044 *     0007 EIAPAR EQU 7       OUTPUT - DATA PARITY BIT
0045 *     0009 EIADTR EQU 9       OUTPUT - DATA TERMINAL READY
0046 *     000B EIAWRQ EQU 11      INPUT/OUTPUT - (CLEAR) WRITE REQ
0047 *     000D EIANSF EQU 13      INPUT - NEW STATUS
0048 *     000E EIADSR EQU 14      INPUT - DATA SET READY
0049 *
0050 *     0007 DMSTR EQU 7        OUTPUT - DATA OUTPUT STROBE
0051 *     0009 DMVFC EQU 9        OUTPUT - VERTICAL FORMS CONTROL
0052 *     000D DMDMD EQU 13       OUTPUT - DEMAND FOR A CHARACTER
0053 *     000E DMINT EQU 14       OUTPUT - INTERRUPT ENABLE BIT
0054 *     000F DMCIN EQU 15       OUTPUT - INTERRUPT ACKNOWLEDGE BIT

```

Figure 5-22. Example DSRs (Sheet 2 of 60)





```
LPDSR                                14:12:26 WEDNESDAY, JUN 28, 1978.
LPDSR - LINE PRINTER DSR                                PAGE 0005

0148      0064' BYTE02 EQU $
0149 0064 0206      LI R6,LPSPUR      SPURIOUS INTERRUPT VECTOR
      0066 00EC'
0150 0068 06A0      BL @ENDRCD        GO TO END-OF-RECORD ROUTINE
      006A 0000
0151 006C 0380      RTWP
```

**Figure 5-22. Example DSRs (Sheet 5 of 60)**





LPDSR  
LPDSR - LINE PRINTER DSR

14:12:26 WEDNESDAY, JUN 28, 1978.

PAGE 0007

00B6 0074  
0210 00B8 0380 RTWP

**Figure 5-22. Example DSRs (Sheet 7 of 60)**

14:12:26 WEDNESDAY, JUN 28, 1978.

PAGE 0008

LPDSR

LPDSR - LINE PRINTER DSR

```

0212           **O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O
0213           * ABSTRACT:
0214           *   LPINT  - THIS IS THE ENTRY POINT FOR DEVICE
0215           *           INTERRUPT.   THE TIME-OUT COUNTER IS RESET AND
0216           *           THE PROPER ROUTINE IS ENTERED VIA R6.
0217           *   PWRON  - THIS ROUTINE IS ENTERED WHEN THE SYSTEM
0218           *           INITIALLY STARTS AND WHEN THE SYSTEM RECOGNIZES
0219           *           A POWER RE-START.   THE INTERFACE IS PROPERLY
0220           *           INITIALIZED.
0221           *   LPSPUR  - HANDLE SPURIOUS INTERRUPTS.
0222           *   ABORT  - THIS ROUTINE HANDLES I/O ABORT.
0223           *
0224           **O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O=O
0225           *
0226 00BA 0754 LPINT ABS *R4           DM INTERFACED LP ?
0227 00BC 1602           JNE LPI$05        NO -
0228 00BE 1E0F           SNZ DMCIN          YES- CLEAR INPUT INTERRUPT
0229 00C0 1001           JMP LPI$10
0230 00C2 1E0D LPI$05 SBZ EIAN$F          CLEAR NEW STATUS INT
0231 00C4 C924 LPI$10 MOV @PDTIM1(R4),@PDTIM2(R4)
0232 00C6 FFFA
0232 00C8 FFFC
0232 00CA 0456           B *R6
0233           *
0234 00CC 0754 PWRON ABS *R4           DM INTERFACED LP ?
0235 00CE 1604           JNE PWR$05        NO -
0236 00D0 1E0F           SBZ DMCIN          YES- CLEAR INPUT INTERRUPTS
0237 00D2 1D0E           SBO DMINT          ENABLE INTERRUPTS
0238 00D4 1D09           SBO DMVFC          DISABLE VFC
0239 00D6 1003           JMP PWR$10
0240 00D8 1D09 PWR$05 SBO EIADTR          *
0241 00DA 3020 LDCR @INIT,0          * INITIALIZE INTERFACE
0242 00DC 0114
0242 00DE C1C1 PWR$10 MOV R1,R7          R7=PRB ADDRESS
0243 00E0 06A0           BL @BZYCHK         GO TO BUSY CHECK ROUTINE
0243 00E2 0000
0244 00E4 1006           JMP NOTBZY         NOT BUSY RETURN
0245 00E6 0206           LI R6,EXIT         BUSY RETURN
0245 00E8 005A
0246 00EA 0380           RTWP
0247           *
0248 00EC 0754 LPSPUR ABS *R4           DM INTERFACED LP ?
0249 00EE 1301           JEQ NOTBZY         YES-
0250 00F0 1E0B           SBZ EIAWRQ         NO - RESET INTERRUPT
0251 00F2 0206 NOTBZY LI R6,LPSPUR      AND IGNORE IT
0251 00F4 00EC
0252 00F6 0380           RTWP
0253           *
0254 00F8 C1C1 ABORT MOV R1,R7          R7=PRB ADDRESS
0255 00FA 00FA BYTE06 EQU $
0256 00FA 06A0           BL @BZYCHK         GO TO BUSY CHECK
0256 00FC 00E2
0257 00FE 10AD           JMP EXIT           NOT BUSY RETURN
0258 0100 D860 MOV @BYTE06,@PRBEC-2(R1) BUSY RETURN, ERROR CODE=6
0258 0102 00FA
0259 0104 FFFF
0259 0106 0242           ANDI 2,>FBFF       RESET REENTER-ME FLAG
0259 0108 FBFF
0260 010A 10A4           JMP EREXIT         EXIT
0261           *

```

Figure 5-22. Example DSRs (Sheet 8 of 60)

```
LPDSR                                14:12:26 WEDNESDAY, JUN 28, 1978.
LPDSR - LINE PRINTER DSR                                PAGE 0009

0262                                *
0263 010C 7F PAGE1 BYTE >7F          DATA TABLES FOR THE NON-OUTPUT
0264 010D 0C                BYTE >0C          CR
0265 010E 0D PAGE2 BYTE >0D          FF
0266 010F 0C                BYTE >0C
0267 0110 7F CR             BYTE >7F          NUL
0268 0111 0D                BYTE >0D          CR
0269 0112 0A CRLF          BYTE >0A
0270 0113 0D                BYTE >0D
0271 0114 4600 INIT        DATA >4600      INITIALIZE INTERFACE LDCR VALUE
0272                                END
NO ERRORS
```

Figure 5-22. Example DSRs (Sheet 9 of 60)

14:12:26 WEDNESDAY, JUN 28, 1978. PAGE 0010

LPDSR LABEL	VALUE	DEFN	REFERENCES
\$	0116		A0008 A0008 A0034 A0034 A0035 A0035 A0036 A0036 A0052 A0053 B0036 C0017 0118 0121 0124 0127 0128 0129 0130 0134 0135 0145 0148 0171 0175 0184 0255
ABORT	00F8	0254	0077
ASCCHK R		0039	
BASE	0014	0082	0083 0085 0086 0087 0088 0089 0090 0091 0092 0093 0094 0095 0096 0097 0098 0099 0100
BYTE02	0064	0148	0143
BYTE06	00FA	0255	0258
BYTE40 R	0056	0040	0144
BZYCHK R	00FC	0037	0243 0256
CLOSE	0026	0118	0086
CLSEOF	0038	0127	0087
CLSUNL	0038	0130	0089
CR	0110	0267	0122
CRLF	0112	0269	0119
DECCNT	0084	0184	0181 0190
DMCIN	000F	0054	0228 0236
DMDMD	000D	0052	0179
DMINT	000E	0053	0237
DMLCHR	00A8	0203	0192
DMSTR	0007	0050	0178 0205 0207
DMVFC	0009	0051	0238
DSFASG	0000	A0010	
DSFBSY	0001	A0011	
DSFCLS	0004	A0014	
DSFEOR	0006	A0016	
DSFINT	0002	A0012	
DSFKLL	0003	A0013	
DSFREN	0005	A0015	
DTFCOM	0005	A0023	
DTEFFIL	0000	A0018	
DTFKSB	0004	A0022	
DTFPRI	0003	A0021	
DTFSYD	0006	A0024	
DTFTIL	0001	A0019	
DTFTIM	0002	A0020	
EIADSR	000E	0048	0182
EIADTR	0009	0045	0240
EIANSF	000D	0047	0230
EIAPAR	0007	0044	0196 0198
FIATST	0080	0182	0177
EIAWRQ	000B	0046	0172 0174 0250
ENDRCD R	006A	0036	0150
EREXIT	0054	0144	0260
EXIT	005A	0145	0091 0092 0093 0100 0141 0186 0245 0257
ILLOP	004E	0143	0083 0094 0095
INIT	0114	0271	0241
JMCALL R	0012	0038	0081
LDCHR	0094	0191	
LDFAI0	000D	B0021	
LDFANC	0000	B0010	
LDFBSY	000A	B0018	
LDFCBA	0003	B0012	
LDFFIL	0009	B0017	
LDFIIO	000C	B0020	
LDFRCM	000B	B0019	
LDFSYS	0008	B0016	
LDFTAP	000F	B0023	
LDFUNB	000E	B0022	

Figure 5-22. Example DSRs (Sheet 10 of 60)

LPDSR LABEL	VALUE	DEFN	REFERENCES		
				14:12:26 WEDNESDAY, JUN 28, 1978.	
				PAGE 0011	
LDTBLK	0018	B0032			
LDTBN	0012	B0030			
LDTFCB	000C	B0028			
LDTFLG	0004	B0008			
LDTFLL	000A	B0027			
LDTFWT	0007	B0015			
LDTIOC	0003	B0007			
LDTLDT	0006	B0025			
LDTLRN	000E	B0029			
LDTLUN	0002	B0006			
LDTNU	001B	B0034			
LDTOCB	0016	B0031			
LDTORC	001A	B0033			
LDTPDT	0000	B0005			
LDTSIZ	001C	B0036			
LDTTSB	0008	B0026			
LPHAN	D 0000	0076	0029		
LPI\$05	00C2	0230	0227		
LPI\$10	00C4	0231	0229		
LPINT	D 00BA	0226	0030		
LPSPUR	D 00EC	0248	0031	0149	0251
NOTBZY	00F2	0251	0244	0249	
OPEN	002C	0121	0085		
OPNRWD	0032	0124	0088		
PAGE1	010C	0263	0125		
PAGE2	010E	0265	0131		
PDT\$	0024	A0037			
PDTBLN	0034	A0045			
PDTBUF	0032	A0044			
PDTCRU	001C	A0033			
PDTDIB	000C	A0025			
PDTDSF	0008	A0009			
PDTDSR	0026	A0038	A0056		
PDTDTF	000A	A0017			
PDTDVQ	0038	A0047			
PDTEMP	0048	A0053	A0025		
PDTERR	0028	A0039	0146		
PDTFLG	0029	A0040			
PDTIM1	FFFA	A0054	0231		
PDTIM2	FFFC	A0055	0231		
PDTINT	0036	A0046			
PDTLNK	0000	A0005			
PDTMAP	0002	A0006			
PDTNAM	002A	A0041			
PDTFRB	0006	A0008			
PDTR0	0004	A0007	A0037		
PDTR10	0018	A0031			
PDTR11	001A	A0032			
PDTR13	001E	A0034			
PDTR14	0020	A0035			
PDTR15	0022	A0036			
PDTR5	000E	A0026			
PDTR6	0010	A0027			
PDTR7	0012	A0028			
PDTR8	0014	A0029			
PDTR9	0016	A0030			
PDTSIZ	0048	A0052	A0054 A0055 A0056 A0057	0146	0147
PDTSL1	002E	A0042	0147		
PDTSL2	0030	A0043			
PDTSRA	FFDE	A0056			

Figure 5-22. Example DSRs (Sheet 11 of 60)

14:12:26 WEDNESDAY, JUN 28, 1978. PAGE 0012

LPDSR LABEL	VALUE	DEFN	REFERENCES
PDTSRB	0046	A0050	A0057
PDTTM1	0042	A0048	A0054
PDTTM2	0044	A0049	A0055
PRBCHT	000A	C0013	0137 0140
PRBDBA	0006	C0011	0136
PRBEC	0001	C0006	0143 0146 0258
PRBLUN	0003	C0008	
PRBOC	0002	C0007	
PRBRLN	0008	C0012	
PRBRN1	000C	C0014	
PRBRN2	000E	C0015	
PRBSFL	0004	C0009	0144
PRBSIZ	0010	C0017	
PRBSOC	0000	C0005	
PRBUFL	0005	C0010	
PWR\$05	00D8	0240	0235
PWR\$10	00DE	0242	0239
PWRON	00CC	0234	0076
R1	0001		0136 0137 0140 0143 0144 0146 0242 0254 0258
R4	0004		0146 0147 0176 0191 0226 0231 0231 0234 0248
R6	0006		0149 0199 0209 0232 0245 0251
R7	0007		0119 0122 0125 0131 0136 0187 0242 0254
R8	0008		0080 0137 0185
R9	0009		0187 0188 0189 0194 0203 0204
READST	0048	0140	0090
RWND	0038	0129	0099
SAVPRB	FFFE	A0057	
SETONE	00A0	0198	0195
SETWPS	R 000A	0035	0079
TSTDSR	0074	0175	0120 0123 0126 0132 0138 0208 0209
TSTWRQ	006E	0171	0199
WAITDS	00B4	0209	0180 0183
WAITWR	00A2	0199	0173 0197
WRITEA	003E	0135	0096
WRITED	003E	0134	0097
WRTEOF	0038	0128	0098

Figure 5-22. Example DSRs (Sheet 12 of 60)

```
SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983. PAGE 0001
ACCESS NAMES TABLE
SOURCE ACCESS NAME= D2.AMX.OSM.DX.DEVDSR.SOURCE.DSSPTSR
OBJECT ACCESS NAME= D2.AMX.OSM.DX.DEVDSR.OBJECT.DSSPTSR
LISTING ACCESS NAME= D2.AMX.OSM.DX.DEVDSR.LIST.DSSPTSR
ERROR ACCESS NAME= f{^ {
OPTIONS= XREF
MACRO LIBRARY PATHNAME=
LINE KEY NAME
0001 A DSC.CONDASM.OS
=>D2.AMX.OSM.DX.CONDASM.OS
0115 B DSC.SYSTEM.TABLES.DSRIRB
=>D2.AMX.OSM.DX.SYSTEM.TABLES.DSRIRB
0116 C DSC.SYSTEM.TABLES.DSRPDT
=>D2.AMX.OSM.DX.SYSTEM.TABLES.DSRPDT
0117 D DSC.SYSTEM.TABLES.KSB
=>D2.AMX.OSM.DX.SYSTEM.TABLES.KSB
0118 E DSC.SYSTEM.TABLES.AJMPMAC
=>D2.AMX.OSM.DX.SYSTEM.TABLES.AJMPMAC
0119 F DSC.SYSTEM.TABLES.DSALLLEX
=>D2.AMX.OSM.DX.SYSTEM.TABLES.DSALLLEX
0120 G DSC.SYSTEM.TABLES.DSALLREX
=>D2.AMX.OSM.DX.SYSTEM.TABLES.DSALLREX
```

Figure 5-22. Example DSRs (Sheet 13 of 60)



```

DSSPTSR      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.      PAGE 0002

0001          COPY   DSC.CONDASM.OS
0010 0000
0015          IDT   'DSSPTSR'
0019          *
0020          * (C) COPYRIGHT, TEXAS INSTRUMENTS INCORPORATED, 1983.
0021          * ALL RIGHTS RESERVED. PROPERTY OF TEXAS INSTRUMENTS
0022          * INCORPORATED. RESTRICTED RIGHTS -- USE, DUPLICATION, OR
0023          * DISCLOSURE SUBJECT TO RESTRICTIONS SET FORTH IN TI'S
0024          * PROGRAM LICENSE AGREEMENT AND ASSOCIATED DOCUMENTATION.
0025          *
0026          *
0027          *
0028          *****
0029          * TITLE:      ASYNC TERMINAL SUBSYSTEM SERIAL PRINTER TSR
0030          * REVISION:  01/26/83      ORIGINAL
0031          * COMPUTER:  990
0032          * ABSTRACT:
0033          * THIS IS THE TERMINAL SERVICE ROUTINE FOR THE
0034          * SERIAL PRINTER DSR. IT PROVIDES THE PROCESSING
0035          * OF I/O CALLS TO THE DEVICE AND INTERRUPTS
0036          * GENERATED BY THE DEVICE.
0037          *
0038          * THE DSR HAS ENTRY POINTS TO FOUR ROUTINES
0039          * WHICH PROCESS DIFFERENT STATES OF I/O:
0040          *
0041          * PWRON - POWER RESTORED ROUTINE
0042          * THIS ROUTINE IS ENTERED WHEN POWER IS
0043          * RESTORED OR WHEN DX10 IS LOADED. THE
0044          * DEVICE INTERFACE IS INITIALIZED HERE.
0045          * THE ENTRY POINT OF THIS ROUTINE IS DEFINED
0046          * IN THE FIRST WORD OF THE DSR.
0047          *
0048          * ABORT - I/O ABORT ROUTINE
0049          * THE DSR IS ENTERED HERE WHEN A DEVICE TIME-
0050          * OUT OCCURS OR A KILL I/O SUPERVISOR CALL
0051          * IS ISSUED.
0052          *
0053          * HANSLP - INITIAL I/O CALL HANDLER
0054          * THE THIRD WORD OF THE DSR IS THE INITIAL
0055          * ENTRY POINT OF THE DX10 I/O SUPERVISOR TO
0056          * HSR TEST. THE I/O CALLS OF USER TASKS ARE
0057          * PROCESSED IN THIS ROUTINE.
0058          *
0059          * REMSLP - REENTER-ME PROCESSING
0060          * REENTER-ME IS USED START AND CONTINUE
0061          * CHARACTER OUTPUT TO THE PRINTER.
0062          *
0063          *

```

Figure 5-22. Example DSRs (Sheet 14 of 60)

```
DSSPTSR      SDSMAC 3.6.0 83.111    13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0003
0065          *
0066          *   EXTERNAL DEFINITIONS
0067          *
0068          *
0069          *   DEF  A13HAN,A23HAN,A33HAN  GLOBAL - START OF DSR
0070          *   DEF  A13INT,A23INT,A33INT  GLOBAL - INTERRUPT ENTRY
0071          *   DEF  A13REM,A23REM,A33REM  GLOBAL - REENTER-ME ENTRY
0072          *
0073          *   COMMON EXTERNAL REFERENCES
0074          *
0075          *   REF  BRSTAT                OPCODE JUMP TABLE ROUTINE
0076          *   REF  ENDRCD                END OF RECORD ROUTINE
0077          *
0078          0001 IRB   EQU  R1
0079          0007 PDT   EQU  R7
0080          0000 TT    EQU  0                SET MACRO JUMP TABLE = TSR
```

Figure 5-22. Example DSRs (Sheet 15 of 60)

```

DSSPSTR      SDSMAC 3.6.0 83.111      13:16:24 FRIDAY, MAY 06, 1983.      PAGE 0004
DSSPSTR - DX10 - TSR SECTION - PDT WORKSPACE
0082      *
0109 0000
0113      REF BYTE02      UNDEFINED BIT IN PDERR(PDT)
0114      REF MASTAB      TABLE OF WORD DEFINITIONS
0115      COPY DSC.SYSTEM.TABLES.DSRIRB
*****
B0001      *
B0002      *
B0003      *      PHYSICAL RECORD BLOCK      (DSRIRB)      10/26/81
B0004      *
B0005      *      POINTED TO BY R1 IN DEVICE SERVICE ROUTINES.
B0006      *      REVISION
B0007      *      03/02/82 - ADD FLAG FOR ON-LINE DIAG
B0008      *      PROCESSING.
B0009      *      *****
B0010 FFFE      DORG -2
B0011 FFFE      00 IRBSOC BYTE 0      SUPERVISOR OPCODE
B0012 FFFF      00 IRBEC BYTE 0      ERROR CODE
B0013      *
B0014 0000      DORG 0
B0015 0000      00 IRBOC BYTE 0      SUB OPCODE
B0016 0001      00 IRBLUN BYTE 0      LUNO
B0017 0002      00 IRBSFL BYTE 0      SYSTEM FLAGS
B0018      0000 IRFBSY EQU 0      BUSY
B0019      0001 IRFERR EQU 1      ERROR
B0020      0002 IRFE0F EQU 2      END OF FILE
B0021      0003 IRFVNT EQU 3      EVENT CHAR
B0022      0007 IRBDIA EQU 7      DIAGNOSTIC EXTENSION TO PRB
B0023 0003      00 IRBUFL BYTE 0      USER FLAGS
B0024      0000 IRFINT EQU 0      INITIATE REQUEST
B0025      0001 IRFRPY EQU 1      REPLY REQUESTED
B0026      0001 IRFVAL EQU 1      READ WITH VALIDATION
B0027      0001 IRFKFG EQU 1      KEY SPECIFIES FLAG
B0028      0001 IRFBFI EQU 1      BUFFERED FORMAT INFO
B0029      0002 IRFRES EQU 2      RESERVED
B0030      0003 IRFOS EQU 3      HEAD OFFSET
B0031      0003 IRFACC EQU 3      ACCESS PRIVILEGES (2 BITS)
B0032      0004 IRFACL EQU 4      ACCESS PRIVILEGES (2ND BIT)
B0033      0018 IRFACM EQU >0018      ACCESS PRIV. BIT MASK
B0034      0004 IRFOSF EQU 4      HEAD OFFSET FORWARD
B0035      0005 IRFLOC EQU 5      LOCK/UNLOCK
B0036      0005 IRFMDS EQU 5      MASTER DO NOT SUSPEND
B0037      0005 IRFTIH EQU 5      TRANSFER INHIBIT
B0038      0005 IRFIMO EQU 5      IMMEDIATE OPEN FOR TPD DSR
B0039      0006 IRFOWN EQU 6      OWNERSHIP LEVEL
B0040      0006 IRFEXR EQU 6      EXTENDED USER FLAGS
B0041      0007 IRFBAD EQU 7      BLANK ADJ/SET EVENT MODE
B0042      0007 IRFWPM EQU 7      WORD PROCESSING MODE
B0043      0007 IRFRTY EQU 7      NO RETRIES
B0044 0004 0000 IRDBEA DATA 0      DATA BUFFER ADDRESS
B0045 0006 0000 IRBICC DATA 0      RECORD LENGTH
B0046 0008 0000 IRBOCC DATA 0      CHARACTER COUNT
B0047 000A 0000 IRBRN1 DATA 0      RECORD # PART 1
B0048      0008 OLDBFR EQU 8      ON-LINE DIAG I/O BUFFER
B0049      *      1 = AVAILABLE
B0050      *      0 = NO AVAILABLE
B0051 000C 0000 IRBRN2 DATA 0      RECORD # PART 2
B0052      *
B0053      000E PRBSIZ EQU $
B0054      *
B0055      *      DIRECT DISK EQUATES (TRACK/ADU)

```

Figure 5-22. Example DSRs (Sheet 16 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111      13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE
B0056        *
B0057        000A PRBTRK EQU IRBRN1      [ TRACK # ] [ ADU # ]
B0058        000C PRBSCT EQU IRBRN2      [(SCT/RCD)/SCT #][SCT OFFSET]
B0059        000C PRBBPS EQU IRBRN2      BYTES/SECTOR (VAR SECT LENGTH)
B0060        000C PRBLTA EQU IRBRN2      LOG TRACK ADDR (WRITE FORMAT)
B0061        *
B0062        *          CALL BLOCK EXTENSION
B0063        *
B0064        000A IRBRPY EQU IRBRN1      REPLY BLOCK ADDRESS
B0065        000A IRBVTA EQU IRBRN1      VALIDATION TABLE ADDRESS
B0066        000C IRBXFL EQU IRBRN2      EXTENDED USER FLAGS
B0067        0000 IRFCSF EQU 0          CURSOR START OF FIELD DEF
B0068        0001 IRFNTN EQU 1          INTENSITY
B0069        0002 IRFFKR EQU 2          BLINKING CURSOR (FLICKER)
B0070        0003 IRFGRA EQU 3          GRAPHICS DISPLAY (CHAR LT >20)
B0071        0004 IRFEBA EQU 4          8-BIT ASCII
B0072        0005 IRFTER EQU 5          ENABLE TASK EDIT CARR CONTROL
B0073        0006 IRFBP EQU 6          BEEP
B0074        0007 IRFRDB EQU 7          RIGHT DISPLAY EDGE BOUNDARY
B0075        0008 IRFCIF EQU 8          CURSOR IN-FIELD DEF
B0076        0009 IRFFC EQU 9          FILL CHAR DEFINED
B0077        000A IRFIF EQU 10         INITIALIZE FIELD
B0078        000B IRFRFF EQU 11        REMAIN IF FIELD IS FULL
B0079        000C IRFECO EQU 12        NO ECHO
B0080        000D IRFVRQ EQU 13        CHAR VALIDATION REQUIRED
B0081        000E IRFVER EQU 14        CHAR FIELD IS IN ERROR
B0082        000F IRFWBP EQU 15        WARNING BEEP
B0083 000E 00 IRBFCH BYTE 0          FILL CHARACTER
B0084 000F 00 IRBEVT BYTE 0          EVENT CHARACTER
B0085 0010 00 IRBCRO BYTE 0          CURSOR POSITION ROW
B0086 0011 00 IRBCCO BYTE 0          CURSOR POSITION COLUMN
B0087 0012 00 IRBFRO BYTE 0          FIELD DEFINITION ROW
B0088 0013 00 IRBFCO BYTE 0          FIELD DEFINITION COLUMN
B0089        *
B0090        0014 IRBSIZ EQU $
B0091 0000        RORG

```

Figure 5-22. Example DSRs (Sheet 17 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                PAGE 0006
0116      COPY   DSC.SYSTEM.TABLES.DSRPDT
C0001     *****
C0002     *
C0003     *   PHYSICAL DEVICE TABLE           (PDT)   06/18/80
C0004     *
C0005     *   POINTED TO BY R4 IN DEVICE SERVICE ROUTINES
C0006     *   PSIZ MUST BE THE SIZE OF THE PDT
C0007     *   RTS 3.1 - ADDED EXTENDED EDIT FLAGS FOR DSR911 4/25/81
C0008     *****
C0009     004A PSIZ EQU 74           PDT SIZE
C0010     0000 PDTSIZ EQU 0
C0011     FFB6 DORG PDTSIZ-PSIZ
C0012     FFB6 PDTLNK BSS 2           FORWARD LINKAGE TO NEXT PDT
C0013     *   THERE IS AN EXPANSION BLOCK BEFORE THE NEXT PDT.
C0014     *   TO REFERENCE IT USE THE VALUE IN PDTLNK PLUS
C0015     *   ONE OF THE FOLLOWING OFFSETS. WHEN PDTLNK IS
C0016     *   ZERO (DS01) USE THE VALUE IN PDTLST (GLOBAL
C0017     *   VARIABLE IN ROOT).
C0018     *
C0019     FFF8 PDTRC EQU -8           NUMBER OF READS
C0020     FFFA PDTWC EQU -6           NUMBER OF WRITES
C0021     FFFC PDTMC EQU -4           NUMBER OF OTHERS
C0022     FFFE PDTRTY EQU -2          NUMBER OF RETRIES
C0023     FFFF PDDLUN EQU -1          NUMBER OF LUNOS ASSIGNED
C0024     *
C0025     FFB8 PDTMAP BSS 2           POINTER TO DSR MAP FILE
C0026     FFBA PDTR0 BSS 2            R0 - DSR SCRATCH
C0027     FFBC 0000 PDTPRB DATA $-$  R1 - QUEUED PRB ADDRESS
C0028     FFBE PDTDSF BSS 2           R2 - DEVICE STATUS FLAGS
C0029     0000 DSFASG EQU 0           ASSIGNED
C0030     0001 DSFBYSY EQU 1          BUSY
C0031     0002 DSFINT EQU 2           KILL I/O IN PROGRESS
C0032     0003 DSFKLL EQU 3           KILL TASK IN PROGRESS
C0033     0004 DSFCLS EQU 4           CLOSE OUT
C0034     0005 DSFREN EQU 5           REENTER-ME
C0035     0006 DSFEOR EQU 6           END-RECORD
C0036     0007 DSFJIS EQU 7           JISCI FLAG (KATAKANA)
C0037     000A DSFWPM EQU 10          WORD PROCESSING MODE
C0038     FFC0 PDTDTF BSS 2           R3 - DEVICE TYPE FLAGS
C0039     0000 DTFFIL EQU 0           FILE ORIENTED
C0040     0001 DTFTIL EQU 1           TILINE DEVICE
C0041     0002 DTFTIM EQU 2           ENABLE TIME-OUT
C0042     0003 DTFPRI EQU 3           PRIVILEGED DEVICE
C0043     0004 DTFKSB EQU 4           TERMINAL WITH A KSB
C0044     0005 DTFCOM EQU 5           COMM DEVICE
C0045     0006 DTFSYD EQU 6           SYSTEM DISK
C0046     0007 DTFEXT EQU 7           EXPANSION BLOCK PRESENT
C0047     FFC2 0000 PDTDIB DATA PDTLNK+PSIZ R4 - DEVICE INFO BLOCK ADDRESS
C0048     FFC4 PDTR5 BSS 2            R5 - DSR SCRATCH
C0049     FFC6 PDTR6 BSS 2            R6 - DSR SCRATCH
C0050     FFC8 PDTR7 BSS 2            R7 - DSR SCRATCH
C0051     FFCA PDTR8 BSS 2            R8 - DSR SCRATCH
C0052     FFCC PDTR9 BSS 2            R9 - DSR SCRATCH
C0053     FFCE PDTR10 BSS 2           R10 - DSR SCRATCH
C0054     FFD0 PDTR11 BSS 2           R11 - DSR SCRATCH
C0055     FFD2 PDTCRU BSS 2           R12 - CRU OR TILINE ADDRESS
C0056     FFD4 0000 PDTR13 DATA $-$  R13 - SAVED WP
C0057     FFD6 0000 PDTR14 DATA $-$  R14 - SAVED PC
C0058     FFD8 0000 PDTR15 DATA $-$  R15 - SAVED ST
C0059     FFDA FFBA PDT$ DATA PDTR0 PDTR0 ADDRESS

```

Figure 5-22. Example DSRs (Sheet 18 of 60)

```

DSSPTRS   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTRS - DX10 - TSR SECTION - PDT WORKSPACE                               PAGE 0007
C0060 FFDC      PDTDSR BSS 2      DSR ADDRESS
C0061 FFDE      PDTERR BSS 1      SAVED ERROR CODE
C0062 FFDF      PDTFLG BSS 1      DEVICE FLAGS
C0063   0008    DFGPRB EQU 8      USE PRB IN LOG MESSAGE
C0064   0009    DFGJAR EQU 9      JISCII RECEIVE MODE FLAG
C0065   000A    DFGJAT EQU 10     JISCII TRANSMIT MODE FLAG
C0066   000B    DFGSTA EQU 11     DEVICE STATE (TWO BITS)
C0067          *                   ONLINE = 0 , OFFLINE = 1
C0068          *                   DIAGNOSTIC = 2, UNDEFINED = 3
C0069   000E    DFGOPF EQU 14     OPERATION FAILED
C0070   000F    DFGRTY EQU 15     NO RETRIES DESIRED
C0071 FFE0      PDTNAM BSS 4      DEVICE NAME
C0072 FFE4      PDTSL1 BSS 2      RESERVED FOR SYSTEM LOG
C0073 FFE6      PDTSL2 BSS 2      RESERVED FOR SYSTEM LOG
C0074 FFE8      PDTBUF BSS 2      CRU BUFFER ADDRESS
C0075 FFEA      PDTBLN BSS 2      CRU BUFFER LENGTH
C0076 FFEC      PDTINT BSS 2      DSR INTERRUPT ADDRESS
C0077 FFEE      PDTDVQ BSS 10     DEVICE QUEUE ANCHOR
C0078 FFF8      PDTTM1 BSS 2      TIME-OUT COUNT 1
C0079 FFFA      PDTTM2 BSS 2      TIME-OUT COUNT 2
C0080 FFFC      PDTSRB BSS 2
C0081 FFFE      PDTFQL BSS 2      FAST REENTER-ME QUEUE LINK
C0082          *
C0083 0000      RORG

```

Figure 5-22. Example DSRs (Sheet 19 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111 . 13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0008
0117      COPY   DSC.SYSTEM.TABLES.KSB
D0001     *****
D0002     *   KEYBOARD STATUS BLOCK      (KSB)                            02/09/77
D0003     *****
D0004 0000      DORG 0
D0005 0000 0020 KSBLDT DATA KSBLD0      R0 - STATION LDT ADDRESS
D0006 0002      KSBQOC BSS 2              R1 - QUEUE OUTPUT COUNT
D0007 0004      KSBQIP BSS 2              R2 - QUEUE INPUT POINTER
D0008 0006      KSBQOP BSS 2              R3 - QUEUE OUTPUT POINTER
D0009 0008      KSBQEP BSS 2              R4 - QUEUE END POINTER
D0010 000A 0000 KSBEBF DATA 0           R5 - EVENT CHARACTER BUFFER
D0011 000C      KSBFL BSS 1              R6 - KSB FLAGS
D0012      0000 KSBCHM EQU 0              CHARACTER MODE
D0013      0001 KSBCEI EQU 1              COMMAND INTERP ENABLE
D0014      0002 KSBRCM EQU 2              RECORD MODE
D0015      0003 KSBCEB EQU 3              COMMAND INTERP BID
D0016      0004 KSBICP EQU 4              COMMAND INTERP ACTIVE
D0017      0005 KSBSET EQU 5              COMMAND I/O HOLD
D0018      0006 KSBKIO EQU 6              COMMAND I/O ABORT
D0019      0007 KSBBRK EQU 7              DEACTIVATE BREAK KEY
D0020 000D      KBSN BSS 1                - STATION NUMBER
D0021 000E      KSB7 BSS 2                R7 - SCRATCH
D0022 0010 0000 KSBTSB DATA $-$        R8 - TSB ADDRESS
D0023      0010 KSBVTA EQU KSBTSB        VALIDATION TABLE ADDRESS
D0024 0012      KSB9 BSS 2                R9 - SCRATCH
D0025 0014      KSB10 BSS 2               R10 - SCRATCH
D0026 0016      KSB11 BSS 2               R11 - SCRATCH
D0027 0018      KSB12 BSS 2               R12 - CRU BASE
D0028 001A 0000 KSB13 DATA $-$        R13 - SAVED WP
D0029 001C 0000 KSB14 DATA $-$        R14 - SAVED PC
D0030 001E 0000 KSB15 DATA $-$        R15 - SAVED ST
D0031 0020 0000 KSBLD0 DATA 0         PDT ADDRESS (PDTLNK)
D0032 0022 00   KSBLD2 BYTE 0          LUNO
D0033 0023 00   KSBLD3 BYTE 0          INIT I/O COUNT
D0034 0024      KSBLD4 BSS 2             LDT FLAGS
D0035 0026      KSBLD6 BSS 2             LDT LINK
D0036 0028 0000 KSBLD8 DATA $-$        TSB ADDRESS
D0037 002A 0000 KSBLCX DATA 0          LOCK COUNT
D0038      002C KSBISZ EQU $-KSBLDT
D0039 0000      RORG

```

Figure 5-22. Example DSRs (Sheet 20 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                               PAGE 0009
  0118      COPY   DSC.SYSTEM.TABLES.AJMPMAC
E0002      *****
E0003      *
E0004      *           TITLE:  ASYNC MUX JUMP TABLE                       *
E0005      *
E0006      *           REVISION: 25 JAN 1983 ORIGINAL                       *
E0007      *           REVISION: 22 FEB 1983 ADD DIAGNOSTIC ENTRY POINTS   *
E0008      *           REVISION: 28 FEB 1983 USE COMMON "JMPHSR" LABEL    *
E0009      *           REVISION: 23 MAR 1983 DROP JUMP TABLES            *
E0010      *
E0011      *           COMPUTER: 990/10A 990/10 990/12                     *
E0012      *           ABSTRACT: THIS COPY MEMBER PRODUCES A TEMPLATE OR   *
E0013      *           TABLE FOR INTERFACE BETWEEN A TSR AND HSR. THE     *
E0014      *           TSR TABLE PRODUCES FIXED OFFSET INTO A TABLE WHOSE *
E0015      *           ADDRESS IS CONTAINED IN A REGISTER. THE HSR TABLE  *
E0016      *           PRODUCES A TABLE OF BRANCH INSTRUCTIONS FOR ACCESS  *
E0017      *           TO THE HSR SUBROUTINE MODULES.                       *
E0018      *
E0019      *           SET TT VARIABLE BEFORE TABLE COPY STATEMENT        *
E0020      *           TT = 0 TSR TEMPLATE                                  *
E0021      *           = 1 HSR SUBROUTINE ENTRIES                          *
E0022      *
E0023      *****
E0099      REF   HRESET   NORMAL POWER UP/RECOVERY SUBROUTINE
E0100      REF   HMRST    DIAGNOSTIC POWER UP/RECOVERY SUBR.
E0101      REF   HSWPWR   NORMAL SOFTWARE INITIALIZATION SUBR.
E0102      REF   HNOTIF   HSR INTERRUPT DECODER SUBROUTINE
E0103      REF   HOUTP4   HSR OUTPUT CHARACTER SUBROUTINE
E0104      REF   HOUTP7   HSR OUTPUT CHARACTER SUBROUTINE
E0105      REF   HTIMER   HSR SET TIMER DURATION SUBROUTINE
E0106      *
E0107      REF   HSTDTR   HSR DTR SIGNAL SET SUBROUTINE
E0108      REF   HRTDTR   HSR DTR SIGNAL RESET SUBROUTINE
E0109      REF   HSTRTS   HSR RTS SIGNAL SET SUBROUTINE
E0110      REF   HRTSTS   HSR RTS SIGNAL RESET SUBROUTINE
E0111      REF   HSTSPT   HSR SRTS SIGNAL SET SUBROUTINE
E0112      REF   HRTSRT   HSR SRTS SIGNAL RESET SUBROUTINE
E0113      REF   HSTSRS   HSR DSRS SIGNAL SET SUBROUTINE
E0114      REF   HRTSRS   HSR DSRS SIGNAL RESET SUBROUTINE
E0115      REF   HSTAL    HSR AL SIGNAL SET SUBROUTINE
E0116      REF   HRTAL    HSR AL SIGNAL RESET SUBROUTINE
E0117      REF   HSTRS    HSR RS FUNCT. SET SUBROUTINE
E0118      REF   HRTRS    HSR RS FUNCT. RESET SUBROUTINE
E0119      REF   HSTUIL   HSR UIL FUNCT. SET SUBROUTINE
E0120      REF   HRTUIL   HSR UIL FUNCT. RESET SUBROUTINE
E0121      REF   HSTCTH   HSR CTH FUNCT. SET SUBROUTINE
E0122      REF   HRTCTH   HSR CTH FUNCT. RESET SUBROUTINE
E0123      REF   HSTCR    HSR CR FUNCT. SET SUBROUTINE
E0124      REF   HRTCRC   HSR CR FUNCT. RESET SUBROUTINE
E0125      REF   HSTBIL   HSR BIL FUNCT. SET SUBROUTINE
E0126      REF   HRTBIL   HSR BIL FUNCT. RESET SUBROUTINE
E0127      REF   HSTTB    HSR TB FUNCT. SET SUBROUTINE
E0128      REF   HRTTB    HSR TB FUNCT. RESET SUBROUTINE
E0129      *
E0130      REF   HRDSDR   HSR DSR SIGNAL READ SUBROUTINE
E0131      REF   HRDCTS   HSR CTS SIGNAL READ SUBROUTINE
E0132      REF   HRDSCT   HSR SCTS SIGNAL READ SUBROUTINE
E0133      REF   HRDDCD   HSR DCD SIGNAL READ SUBROUTINE
E0134      REF   HRSDSDC  HSR SDCD SIGNAL READ SUBROUTINE
E0135      REF   HRDRI    HSR RI SIGNAL READ SUBROUTINE

```

Figure 5-22. Example DSRs (Sheet 21 of 60)



```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0010
E0136          REF  HRDUIL      HSR UIL  FUNCT.  READ  SUBROUTINE
E0137          REF  HRDCTH      HSR CTH  FUNCT.  READ  SUBROUTINE
E0138          REF  HRDCR      HSR CR   FUNCT.  READ  SUBROUTINE
E0139          REF  HRDBIL      HSR BIL  FUNCT.  READ  SUBROUTINE
E0140          REF  HRDTB      HSR TB   FUNCT.  READ  SUBROUTINE
E0141          REF  HRDSSS      HSR SSS  FUNCT.  READ  SUBROUTINE
E0142          *
E0143          REF  HESDSR      HSR DSR  SIGNAL  CHANGE NOTIFY ENABLE
E0144          REF  HSDSR      HSR DSR  SIGNAL  CHANGE NOTIFY DISABLE
E0145          REF  HESCTS      HSR CTS  SIGNAL  CHANGE NOTIFY ENABLE
E0146          REF  HDSCTS      HSR CTS  SIGNAL  CHANGE NOTIFY DISABLE
E0147          REF  HESSCT      HSR SCTS SIGNAL  CHANGE NOTIFY ENABLE
E0148          REF  HDSSCT      HSR SCTS SIGNAL  CHANGE NOTIFY DISABLE
E0149          REF  HESDCD      HSR DCD  SIGNAL  CHANGE NOTIFY ENABLE
E0150          REF  HSDSDC      HSR DCD  SIGNAL  CHANGE NOTIFY DISABLE
E0151          REF  HESSDC      HSR SDCD SIGNAL  CHANGE NOTIFY ENABLE
E0152          REF  HDSSDC      HSR SDCD SIGNAL  CHANGE NOTIFY DISABLE
E0153          REF  HESRI      HSR RI   SIGNAL  CHANGE NOTIFY ENABLE
E0154          REF  HDSRI      HSR RI   SIGNAL  CHANGE NOTIFY DISABLE
E0155          REF  HESTSR      HSR TSRE FUNCT.  CHANGE NOTIFY ENABLE
E0156          REF  HDSTSR      HSR TSRE FUNCT.  CHANGE NOTIFY DISABLE
E0157          *
E0158          REF  HSPSPD      HSR SET  BAUD  RATE  SUBROUTINE
E0159          REF  HSPPSL      HSR SET  PARITY, CHAR LENGTH, STOP BIT
E0160          *
E0161          REF  HRPSPD      HSR READ BAUD  RATE  SUBROUTINE
E0162          REF  HRPSSL      HSR READ PARITY, CHAR LENGTH, STOP BIT
E0163          *
E0164          REF  HRPDAT      HSR READ HSR  MODULE ID
E0165          REF  HRPTYP      HSR READ HARDWARE INTERFACE TYPE
E0166          REF  HWUART      HSR TILINE UART WRITE SUBROUTINE
E0167          REF  HRUART      HSR TILINE UART READ SUBROUTINE

```

Figure 5-22. Example DSRs (Sheet 22 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111    13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                PAGE 0011
0119          COPY      DSC.SYSTEM.TABLES.DSALLLEX
F0001        *****
F0002        *
F0003        *   LOCAL ASYNC EXTENSION TO PDT
F0004        *
F0005        *
F0006        *
F0007        *****
F0008 002C   DORG   KSBSIZ
F0009 002C   PDXSMB BSS 2           SAVE MAP BIAS
F0010 002E   PDXSMP BSS 2           SAVE MAP POINTER
F0011 0030   PDXFLG BSS 1           HSR FLAGS
F0012 0031   PDXCHN BSS 1           HSR CHANNEL NUMBER
F0013 0032   PDXFCT BSS 2           HARDWARE SOFTWARE FIFO COUNT
F0014 0034   PDXCP1 BSS 2           HSR SCRATCH COPY
F0015 0036   PDXCP2 BSS 2           HSR SCRATCH COPY
F0016 0038   PDXCP3 BSS 2           HSR SCRATCH COPY
F0017 003A   PDXCP4 BSS 2           TSR SCRATCH/SYSGEN PARAMETERS
F0018 003C   PDXCP5 BSS 2           TSR SCRATCH/SYSGEN PARAMETERS
F0019 003E   PDXCP6 BSS 2           TSR SCRATCH/SYSGEN PARAMETERS
F0020 0040   PDXCP7 BSS 2           TSR SCRATCH/SYSGEN PARAMETERS
F0021        *
F0022        *   ADDITIONAL WORDS FOR TSR
F0023        *
F0024 0042   PDXCP8 BSS 2           1   TSR SCRATCH/SYSGEN PARAMETERS
F0025 0044   PDXCP9 BSS 2           2   TSR SCRATCH/SYSGEN PARAMETERS
F0026 0046   PDXCPA BSS 2           3   TSR SCRATCH/SYSGEN PARAMETERS
F0027 0048   PDXCPB BSS 2           4   TSR SCRATCH/SYSGEN PARAMETERS
F0028 004A   PDXCPC BSS 2           5   TSR SCRATCH/SYSGEN PARAMETERS
F0029 004C   PDXCPD BSS 2           6   TSR SCRATCH/SYSGEN PARAMETERS
F0030 004E   PDXCPE BSS 2           7   TSR SCRATCH/SYSGEN PARAMETERS
F0031 0050   PDXCPF BSS 2           8   TSR SCRATCH/SYSGEN PARAMETERS
F0032 0000   RORG
0120          COPY      DSC.SYSTEM.TABLES.DSALLLEX
G0001        *****
G0002        *
G0003        *   REMOTE DEVICE EXTENSIONS
G0004        *
G0005        *
G0006        *
G0007        *****
G0008 0000   DORG   0
G0009 0000   HSRBGN EQU $           HSR PORTION OF DEVICE EXTENSIONS
G0010 0000   BSS   >20
G0011 0020   HSREND EQU $
G0012        *
G0013 0020   SWFBGN EQU HSREND      SOFTWARE FIFO IN DEVICE EXTENTIONS
G0014 0020   BSS   >70
G0015 0090   SWFEND EQU $
G0016        *
G0017 0090   TSRBGN EQU SWFEND      TSR PORTION OF DEVICE EXTENSIONS
G0018 0090   BSS   >30
G0019 00C0   TSREND EQU $
G0020        *
G0021 00C0   SIBUFF EQU TSREND      SCREEN IMAGE BUFFER
G0022 00C0   BSS   >780            1920 BYTE SCREEN IMAGE BUFFER
G0023 0840   SIEND  EQU $
G0024 0000   RORG   0
0130 0000

```

Figure 5-22. Example DSRs (Sheet 23 of 60)

```

DSSPTR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0012
0132      *
0133      DBGINS $MACRO
0134          DATA >1000          MACRO DEBUG INSTRUCTION(S)
0135          $END DBGINS
0136      *
0137      *      DATA >2C40      SYSTEM DEBUG ENABLE WORD
0138      *      DATA >1000      NO-OP FOR "NORMAL" OPERATION
0139      *      DATA >10FF      JUMP HERE FOR NO DEBUG SUPPORT
0140      *      XOP  15,15      DX10 SYSTEM DEBUGGER
0141      *
0142      *
0143      0001 DRSCH EQU  1          DSR SCHEDULE WORD  0 = USE REENTER-ME
0144      *                                          1 = USE DSR SCHEDULE
0145      *
0146      *
0147      *      LONG DISTANCE PDT EXTENSIONS
0148      *
0149 0090          DORG TSRBGN
0150 0090      RTEXT0 BSS  2          RECEIVE BREAK COUNT
0151 0092      RTEXT1 BSS  2          FRAMING ERROR COUNT
0152 0094      RTEXT2 BSS  2          PARITY ERROR COUNT
0153 0096      RTEXT3 BSS  2          RECEIVER OVERRUN COUNT
0154 0098      RTEXT4 BSS  2          "OTHER" ERROR COUNT
0155 009A      RTEXT5 BSS  2          ILLEGAL INTERRUPT COUNT
0156 0000          RORG

```

Figure 5-22. Example DSRs (Sheet 24 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111    13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0013
0158          *
0159          *
0160          *   PDT LOCAL EXTENSIONS (AFTER THE KSB OR PSEUDO-KSB)
0161          *
0162 002C          DORG KSBSIZ
0163          002C PDXMAP EQU PDXSMB
0164          003A VDTEX0 EQU PDXCP4          TSR/ISR FLAG WORD
0165          *
0166          *   EQU 0          RESERVED
0167          0001 TFOVIA EQU 1          WRITE IS ACTIVE
0168          *   EQU 2          RESERVED
0169          0003 TFOOIA EQU 3          OTHER IS ACTIVE
0170          *
0171          0004 TF0DSR EQU 4          XMT HOLD DUE TO DSR LOSS
0172          0005 TF0DC3 EQU 5          XMT HOLD DUE TO DC3 RECEPTION
0173          0006 TF0KFG EQU 6          FLAG WHEN MODE CHARACTER TO BE SENT
0174          0007 TF0EPF EQU 7          EXTENDED PRINT FLAG
0175          *
0176          *   EQU 8          RESERVED
0177          0009 TFORRW EQU 9          REENTER-ME FOR WRITE
0178          *   EQU 10         RESERVED
0179          *   EQU 11         RESERVED
0180          *
0181          000C TF0CFE EQU 12         CHANNEL FAILED SELF TEST
0182          000D TF0RES EQU 13         THIS IS A POWER RECOVERY NOT REBOOT
0183          *   EQU 14         RESERVED
0184          *   EQU 15         RESERVED
0185          *
0186          003C VDTEX1 EQU PDXCP5     TRANSMIT/RECEIVE BAUD RATE WORD
0187          003E VDTEX2 EQU PDXCP6     OPERATIONAL PARAMETERS WORD
0188          0040 VDTEX3 EQU PDXCP7     TOTAL ERROR COUNT
0189 0000          *   RORG
0190          *
0191          000B ERRBI EQU 11          BREAK INTERRUPT ERROR
0192          000C ERRFE EQU 12          FRAMING ERROR
0193          000D ERRPE EQU 13          PARITY ERROR
0194          000E ERROE EQU 14          OVERRUN ERROR
0195          *
0196          *
0197          1820 DEFPRM EQU >1820     DEFAULT OPERATIONAL PARAMETERS
0198          *   EVEN PARITY, 7 BIT CHARACTERS
0199          *   ONE STOP BIT
0200          *
0201          1F1F SPDMSK EQU >1F1F     BAUD RATE MASK
0202          0002 DSRSET EQU 2          DSR RETURNED SIGNAL STATE BIT
0203          *

```

Figure 5-22. Example DSRs (Sheet 25 of 60)

```

DSSPTRS   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTRS - DX10 - TSR SECTION - PDT WORKSPACE                PAGE 0014
0205      *****
0206      *
0207      *   ABSTRACT:
0208      *     HANSLP - THIS IS THE MAIN ENTRY POINT INTO THE DSR.
0209      *     THE FIRST TWO WORDS OF THIS ROUTINE ARE POINTERS TO
0210      *     THE POWER UP ENTRY ROUTINE AND THE ABORT I/O ROUTINE.
0211      *     THE THIRD WORD IS THE ENTRY POINT FOR INITIAL I/O
0212      *     CALLS AND IS USED BY THE I/O SUPERVISOR.
0213      *
0214      *****
0215 0000   A13HAN
0216 0000   A23HAN
0217 0000   A33HAN
0235 0000
0239 0000 02CA^   DATA PWR000           POWER RESTORE ENTRY ADDRESS
0240 0002 02BC^   DATA ABT000           ABORT I/O ENTRY ADDRESS
0244 0004 0460    B   @HAN000           TSR SVC ENTRY POINT
0245 0008 0460    B   @REM000           TSR DSR SCHEDULE ENTRY POINT
0245 000A 01F8^
0250      *
0251      *   INITIAL I/O CALL ENTRY POINT
0252      *
0253 000C   HAN000
0254 000C   DBGINS
*0001 000C 1000   DATA >1000           MACRO DEBUG INSTRUCTION(S)
0255 000E C2D9    MOV  *R9,R11           GET TSR FLAG WORD
0256 0010 22E0    COC  @TF0CFE*2+MASTAB,R11 IS CHANNEL ERROR SET?
0257 0012 0018    JNE  HAN010           NO - TAKE JUMP
0258 0016
0259 0016 04CB    CLR  R11           CLEAN TEST REGISTER
0260 0018 D2E1    MOV  B @IRBOC(IRB),R11       GET THE SUBOPCODE
0261 001A 0000
0261 001C 1306    JEQ  HAN002           IF SUBOPCODE 0 - TAKE JUMP
0262 001E
0263 001E 028B    CI   R11,>1500         OPCODE 15 ?
0264 0020 1500
0264 0022 130C    JEQ  HAN010           YES - TAKE JUMP
0265 0024
0266 0024 028B    CI   R11,>0300        OPCODE 03 ?
0267 0026 0300
0267 0028 1605    JNE  HAN005           NO - TAKE JUMP
0268 002A
0269 002A D2E1    HAN002 MOV  B @IRBUFL(IRB),R11       GET USER FLAG BYTE
0270 002C 0003
0270 002E 22E0    COC  @IRFIMO*2+MASTAB,R11 IS IMMEDIATE OPEN BIT SET?
0271 0030 000A
0271 0032 1304    JEQ  HAN010           YES - TAKE JUMP
0272 0034
0273 0034 020B    HAN005 LI   R11,>0707         LOAD DEVICE ERROR CODE
0274 0036 0707
0274 0038 0460    B   @ERR005           EXIT THROUGH ERROR ROUTINE
0275 003A 01BE^
0275 003C
0276 003C   HAN010
0277 003C 020A    LI   R10,HAN040       GET STATISTICS TABLE ADDRESS
0278 003E 0074^
0278 0040 06A0    BL   @BRSTAT          OPCODE DECODER
0278 0042 0000

```

Figure 5-22. Example DSRs (Sheet 26 of 60)

```

DSSPTRS   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTRS - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0015
0280      *
0281      *   THE STATISTICS TABLE MUST BE MODIFIED IF THE OPCODE
0282      *   TABLE IS MODIFIED.
0283      *
0284 0044 0015 HAN020 DATA HAN030      LAST OPCODE USED
0285 0046 01BA DATA ILL010      OPCODE OUT OF RANGE RETURN
0286      *
0287 0048 008A DATA OPN010      OPEN (>00)
0288 004A 008A DATA CLS010      CLOSE (>01)
0289 004C 00A2 DATA REW010      CLOSE/EOF (>02)
0290 004E 00A2 DATA OPN100      OPEN/REWIND (>03)
0291 0050 00A2 DATA REW100      CLOSE/UNLOAD (>04)
0292 0052 013C DATA STA010      READ STATUS (>05)
0293 0054 01E6 DATA IGN010      FORWARD SPACE (>06)
0294 0056 01E6 DATA IGN100      BACKWARD SPACE (>07)
0295 0058 01BA DATA ILL010      ILLEGAL OPCODE (>08)
0296 005A 01BA DATA ILL100      READ ASCII (>09)
0297 005C 01BA DATA ILL200      READ DIRECT (>0A)
0298 005E 00A8 DATA WRT010      WRITE ASCII (>0B)
0299 0060 00A8 DATA WRT100      WRITE DIRECT (>0C)
0300 0062 00A2 DATA REW200      WRITE/EOF (>0D)
0301 0064 00A2 DATA REW300      REWIND (>0E)
0302 0066 01E6 DATA IGN200      UNLOAD (>0F)
0303 0068 01BA DATA ILL300      UNUSED (>10)
0304 006A 01BA DATA ILL400      UNUSED (>11)
0305 006C 01BA DATA ILL500      UNUSED (>12)
0306 006E 01B2 DATA DMP010      DUMP STATS (>13)
0307 0070 01BA DATA ILL600      UNUSED (>14)
0308 0072 00C0 DATA OLD010      OPCODE 15 (>15)
0309 0015 HAN030 EQU ($-HAN020-6)/2
0310      *
0311 0074 FC HAN040 BYTE PDTMC      OPEN (>00)
0312 0075 FC HAN040 BYTE PDTMC      CLOSE (>01)
0313 0076 FC HAN040 BYTE PDTMC      CLOSE/EOF (>02)
0314 0077 FC HAN040 BYTE PDTMC      OPEN/REWIND (>03)
0315 0078 FC HAN040 BYTE PDTMC      CLOSE/UNLOAD (>04)
0316 0079 00 HAN040 BYTE 0          READ STATUS (>05)
0317 007A 00 HAN040 BYTE 0          FORWARD SPACE (>06)
0318 007B 00 HAN040 BYTE 0          BACKWARD SPACE (>07)
0319 007C 00 HAN040 BYTE 0          ILLEGAL OPCODE (>08)
0320 007D F8 HAN040 BYTE PDTRC     READ ASCII (>09)
0321 007E F8 HAN040 BYTE PDTRC     READ DIRECT (>0A)
0322 007F FA HAN040 BYTE PDTWC     WRITE ASCII (>0B)
0323 0080 FA HAN040 BYTE PDTWC     WRITE DIRECT (>0C)
0324 0081 FA HAN040 BYTE PDTWC     WRITE/EOF (>0D)
0325 0082 FA HAN040 BYTE PDTWC     REWIND (>0E)
0326 0083 00 HAN040 BYTE 0          UNLOAD (>0F)
0327 0084 00 HAN040 BYTE 0          UNUSED (>10)
0328 0085 00 HAN040 BYTE 0          UNUSED (>11)
0329 0086 00 HAN040 BYTE 0          UNUSED (>12)
0330 0087 00 HAN040 BYTE 0          DUMP STATS (>13)
0331 0088 00 HAN040 BYTE 0          UNUSED (>14)
0332 0089 00 HAN040 BYTE 0          OPCODE 15 (>15)
0333      *

```

Figure 5-22. Example DSRs (Sheet 27 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0016
0335 008A          OPN010
0336 008A          CLS010
0337 008A 0206     LI   R6,CRCR          LOAD ADDRESS OF CR/CR
      008C 04E0'
0338 008E
0339 008E
0340 008E 0208     OPN020 LI   R8,2          LOAD COUNT
      0090 0002
0341 0092 E660     SOC   @TF00IA*2+MASTAB,*R9 SET OTHER ACTIVE
      0094 0006
0342 0096 E660     OPN030 SOC   @TF0WIA*2+MASTAB,*R9 SET TRANSMIT ACTIVE FLAG
      0098 0002
0343 009A E660     SOC   @TF0RRW*2+MASTAB,*R9 SET REENTER DUE TO XMT
      009C 0012
0344 009E 0460     B     @REM000          GO PROCESS IT
      00A0 01F8'
0345 00A2
0346 00A2
0347 00A2          OPN100
0348 00A2          REW010
0349 00A2          REW100
0350 00A2          REW200
0351 00A2          REW300
0352 00A2 0206     LI   R6,CRFF          LOAD ADDRESS OF CR/FF
      00A4 04E2'
0353 00A6 10F3     JMP   OPN020
0354 00A8
0355 00A8
0356 00A8

```

Figure 5-22. Example DSRs (Sheet 28 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0017
0358      *****
0359      *
0360      *   WRT010 - WRITE DIRECT ROUTINE
0361      *   WRT100 - WRITE ASCII ROUTINE (SAME AS WRITE DIRECT)
0362      *   THIS ROUTINE OPERATES IN THE PDT WORKSPACE AND, IF A
0363      *   NORMAL WRITE, WILL JUST SET AN ACTIVE FLAG AND CALL
0364      *   REENTER-ME TO OUTPUT CHARACTERS. AFTER A REQUEST
0365      *   IS COMPLETE, A REENTER-ME CODE WILL RETURN THE IRB TO
0366      *   THE USER.
0367      *
0368      *****
0369 00A8   WRT010
0370 00A8   WRT100
0371 00A8 C221   MOV   @IRBOCC(IRB),R8   GET XMT CHARACTER COUNT
      00AA 0008
0372 00AC 1604   JNE   WRT150   IF ANY - TAKE JUMP
0373 00AE 04E1   CLR   @IRBICC(IRB)   CLEAR XMT CURRENT COUNT
      00B0 0006
0374 00B2 0460   B     @RETURN   IF NONE - EXIT
      00B4 01E6
0375 00B6
0376 00B6 C1A1   WRT150 MOV   @IRBDBA(IRB),R6   SET XMT BUFFER ADDRESS
      00B8 0004
0377 00BA 4660   SZC   @TF00IA*2+MASTAB,*R9 RESET OTHER IS ACTIVE
      00BC 0006
0378 00BE 10EB   JMP   OPN030
0379 00C0

```

Figure 5-22. Example DSRs (Sheet 29 of 60)



```

DSSPTRS      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTRS - DX10 - TSR SECTION - PDT WORKSPACE                PAGE 0018
0381          *****
0382          *
0383          *   OLD010 - ON LINE DIAGNOSTIC SUPPORT
0384          *   THIS ROUTINE ALLOWS OPCODE 15 SUPPORT FOR ONLINE
0385          *   DIAGNOSTICS.
0386          *
0387          *****
0388          OLD010
0389          00C0      MOVB @PDTFLG(PDT),R11   GET PDT FLAG WORD
0390          00C2      FFDF
0391          00C4      024B      ANDI R11,>1800      LEAVE ONLY DEVICE STATE
0392          00C6      1800
0393          00C8      028B      CI   R11,>1000      IN DIAGNOSTIC MODE?
0394          00CA      1000
0395          00CC      1303      JEQ  OLD020      YES - TAKE JUMP
0396          00CE      020B      LI   R11,>9C9C      LOAD WRONG DEVICE STATE
0397          00D0      9C9C
0398          00D2      1075      JMP  ERR005      EXIT
0399          00D4
0400          00D4      OLD020
0401          00D4      C2E1      MOV  @IRBDBA(IRB),R11  GET OPCODE 15 BUFFER ADDRESS
0402          00D6      0004
0403          00D8      C2DB      MOV  *R11,R11          GET OPCODE 15 SUBOPCODE
0404          00DA      024B      ANDI R11,>FF00      MASK OFF UNUSED BYTE
0405          00DC      FF00
0406          00DE      028B      CI   R11,>3100      BELOW VALID SUBOPCODES?
0407          00E0      3100
0408          00E2      1A6B      JL   ILLCDE          YES - TAKE JUMP
0409          00E4      130B      JEQ  OLD100          IF OPCODE >31 - TAKE JUMP
0410          00E6
0411          00E6      028B      CI   R11,>3500      ABOVE VALID SUBOPCODES?
0412          00E8      3500
0413          00EA      1B67      JH   ILLCDE          YES - TAKE JUMP
0414          00EC      137C      JEQ  RETURN          IF OPCODE >35 - TAKE JUMP
0415          00EE
0416          00EE      028B      CI   R11,>3400      ABOVE VALID SUBOPCODES?
0417          00F0      3400
0418          00F2      131E      JEQ  OLD400          IF OPCODE >34 - TAKE JUMP
0419          00F4
0420          00F4      028B      CI   R11,>3200      OPCODE >32?
0421          00F6      3200
0422          00F8      1308      JEQ  OLD200          YES - TAKE JUMP
0423          00FA      1016      JMP  OLD300          OPCODE >33 - TAKE JUMP
0424          00FC

```

Figure 5-22. Example DSRs (Sheet 30 of 60)

```

DSSPTRS   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTRS - DX10 - TSR SECTION - PDT WORKSPACE                               PAGE 0019
0416      *
0417      *      WRITE UART SUB-SUBOPCODE >31
0418      *
0419 00FC C021 OLD100 MOV @IRBCRO(IRB),R0   GET THE PASSED UART WORD
      00FE 0010
0420 0100 06A0      BL @HWUART             CALL LOAD SUBROUTINE
      0102 0000
0421 0104 01BA     DATA ILLCDE           NOT SUPPORTED RETURN
0422 0106 0460 OLD150 B @PWR090          RETURN THE IRB AND EXIT
      0108 03A6
0423      *
0424      *      READ UART SUB-SUBOPCODE >32
0425      *
0426 010A C021 OLD200 MOV @IRBCRO(IRB),R0   GET THE PASSED UART WORD
      010C 0010
0427 010E 06A0      BL @HRUART             CALL READ SUBROUTINE
      0110 0000
0428 0112 01BA     DATA ILLCDE           NOT SUPPORTED RETURN
0429 0114 011C     DATA OLD250          FUNCTION FAILED RETURN
0430 0116 C840      MOV R0,@IRBCRO(IRB)    STORE THE RETURNED VALUE
      0118 0010
0431 011A 10F5      JMP OLD150             NO - RETURN THE IRB
0432 011C
0433 011C 0721 OLD250 SETO @IRBCRO(IRB)     STORE THE RETURNED VALUE
      011E 0010
0434 0120 0300      LIM1 2                 MASK INTERRUPTS
      0122 0002
0435 0124 0460      B @PWR100             RETURN THE IRB WITH ERROR
      0126 03BC
0436      *
0437      *      SOFTWARE POWER UP SUB-SUBOPCODE >33
0438      *
0439 0128 0300 OLD300 LIM1 2                 MASK INTERRUPTS
      012A 0002
0440 012C 0460      B @PWR032             GO TO SOFTWARE POWER UP
      012E 0356
0441      *
0442      *      HARDWARE RESET SUB-SUBOPCODE >34
0443      *
0444 0130 0300 OLD400 LIM1 2                 MASK INTERRUPTS
      0132 0002
0445 0134 06A0      BL @HMRST             CALL RESET SUBROUTINE
      0136 0000
0446 0138 03BC     DATA PWR100          ERROR RETURN
0447 013A 10E5     JMP OLD150            RETURN THE IRB AND EXIT
0448 013C

```

Figure 5-22. Example DSRs (Sheet 31 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.           PAGE 0020
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE
0450 *****
0451 *
0452 *
0453 * STA010 - READ DEVICE STATUS INFORMATION
0454 * UP TO 38 BYTES OF STATUS INFORMATION IS RETURNED
0455 * IN THE SCB DATA BUFFER.
0456 *
0457 *****
0458      2300 CI403 EQU >2300          CI 403 EQUATE
0459      2400 CI404 EQU >2400          CI 404 EQUATE
0460      0026 STACNT EQU 38           MAX RETURN SIZE
0461 *
0462      013C STA010 EQU $
0463 013C 04CB CLR R11                CLEAR REGISTER
0464 013E D80B MOVB R11,@STA110      CLEAR PORT ID NUMBER
      0140 0191
0465 0142 06A0 BL @HRPTYP           CALL READ HARDWARE TYPE
      0144 0000
0466 0146 06C0 SWPB R0              MOVE IT TO LEFT BYTE
0467 0148 D800 MOVB R0,@STA130          LOAD IT
      014A 0196
0468 014C 0280 CI R0,CI403          IS IT A CI403?
      014E 2300
0469 0150 1303 JEQ STA020           YES - TAKE JUMP
0470 0152 0280 CI R0,CI404          IS IT A CI404?
      0154 2400
0471 0156 1603 JNE STA050           NO - TAKE JUMP
0472 0158
0473 0158 D827 STA020 MOVB @PDXCHN(PDT),@STA110 LOAD CHANNEL NUMBER
      015A 0031
      015C 0191
0474 015E
0475 015E C80C STA050 MOV R12,@STA120    LOAD CRU/TILINE ADDRESS
      0160 0192
0476 0162 D827 MOVB VDTEX1(PDT),@STA140  LOAD SPEED CODE
      0164 003C
      0166 01A4
0477 0168
0478 0168 C2E1 MOV @IRBICC(IRB),R11      GET TRANSFER COUNT
      016A 0006
0479 016C 028B CI R11,STACNT          REQUESTED COUNT GT STACNT?
      016E 0026
0480 0170 1202 JLE STA060           NO - TAKE JUMP
0481 0172 020B LI R11,STACNT        FORCE MAX ALLOWABLE COUNT
      0174 0026
0482 0176
0483 0176 C84B STA060 MOV R11,@IRBOCC(IRB)  LOAD RETURN COUNT
      0178 0008
0484 017A 1335 JEQ RETURN           IF NONE - RETURN
0485 017C C2A1 MOV @IRBDBA(IRB),R10          GET BUFFER ADDRESS
      017E 0004
0486 0180 0200 LI R0,STA100           GET INFO ADDRESS
      0182 018C
0487 0184 DEB0 STA070 MOVB *R0+,*R10+    MOVE THE BYTE
0488 0186 060B DEC R11              DEC BYTE COUNT, FINISHED?
0489 0188 16FD JNE STA070           NO - TAKE JUMP
0490 018A 102D JMP RETURN            EXIT

```

Figure 5-22. Example DSRs (Sheet 32 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0021
0492 018C
0493          *          READ DEVICE STATUS INFORMATION
0494 018C
0495          018C' STAl00 EQU $
0496 018C 00FF          DATA >00FF          00 - 01          RESERVED
0497 018E 0000          DATA >0000          02 - 03          RESERVED
0498 0190 10           BYTE >10           04           DSR TYPE-DSRSPC
0499 0191 00          STAl10 BYTE >00          05           PORT ID NUMBER
0500 0192 0000        STAl20 DATA >0000          06 - 07          CRU/TILINE ADDR.
0501 0194 FFFF          DATA >FFFF          08 - 09          RESERVED
0502 0196 00          STAl30 BYTE >00          10           HARDWARE TYPE
0503 0197 00           BYTE >00          11           RESERVED
0504 0198 0000          DATA >0000          12 - 13          RESERVED
0505 019A 0000          DATA >0000          14 - 15          RESERVED
0506 019C 0000          DATA >0000          16 - 17          RESERVED
0507 019E 0000          DATA >0000          18 - 19          RESERVED
0508 01A0 0000          DATA >0000          20 - 21          RESERVED
0509 01A2 0000          DATA >0000          22 - 23          RESERVED
0510 01A4 00          STAl40 BYTE >00          24           SPEED CODE
0511 01A5 00           BYTE >00          25           RESERVED
0512 01A6 0000          DATA >0000          26 - 27          RESERVED
0513 01A8 0000          DATA >0000          28 - 29          RESERVED
0514 01AA 0000          DATA >0000          30 - 31          RESERVED
0515 01AC 0000          DATA >0000          32 - 33          RESERVED
0516 01AE 0000          DATA >0000          34 - 35          RESERVED
0517 01B0 0000          DATA >0000          36 - 37          RESERVED
0518 01B2

```

Figure 5-22. Example DSRs (Sheet 33 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                PAGE 0022
0520      *****
0521      *
0522      * RETURN - EXIT DSR
0523      * RESTORE PRB ADDRESS; CALLS END RECORD, AND BRANCHES
0524      * TO IDLHSR
0525      *
0526      * IGNXXX - IGNORE REQUEST
0527      * I/O CALLS THAT ARE TO BE IGNORED ENTER HERE FROM
0528      * THE OPCODE DECODER. IT JUMPS TO RETURN
0529      *
0530      * DMP010 - DUMP STATISTICS BUFFERS
0531      * AN ERROR CODE IS SET TO CAUSE SYSLOG TO DUMP
0532      * THE STATISTICS BUFFERS.
0533      *
0534      * ILLXXX - ROUTINE TO PROCESS AN ILLEGAL I/O OPCODE.
0535      *
0536      *****
0537      *
0538 01B2 070B DMP010 SET0 R11
0539 01B4 D9CB      MOV0 R11,@PDTERR(PDT)      SET DUMP ERROR CODE
0539      01B6 FFDE
0540 01B8 1016      JMP RETURN
0541 01BA
0542 01BA          ILL010
0543 01BA          ILL100
0544 01BA          ILL200
0545 01BA          ILL300
0546 01BA          ILL400
0547 01BA          ILL500
0548 01BA          ILL600
0549 01BA          ILLCDE
0550 01BA 020B      LI R11,>0202                LOAD ERROR CODE
0550      01BC 0202
0551      01BE ERR005 EQU $
0559 01BE
0563 01BE F9E0      SOCB @DFGPRB+MASTAB-16,@PDTFLG(PDT) SET TO LOG PRB
0563      01C0 FFF8
0563      01C2 FFDF
0567      *

```

Figure 5-22. Example DSRs (Sheet 34 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0023
0569 01C4      ERROR
0570 01C4      ERR010
0571 01C4 C067      MOV  @PDTSRB(PDT),IRB      RESTORE PRB ADDRESS
      01C6 FFFC
0572 01C8 130E      JEQ  ERR030      IF NONE, TAKE JUMP
0573 01CA E860      SOC  @IRFERR*2+MASTAB,@IRBSFL(IRB)  SET IRB ERROR
      01CC 0002
      01CE 0002
0574 01D0 D9CB      MOVB R11,PDTErr(PDT)      LOG THE ERROR?
      01D2 FFDE
0575 01D4 1303      JEQ  ERR020      NO - TAKE ERR020 JUMP
0576 01D6 F9E0      SOCB @BYTE02,@PDTErr(PDT)  SET OPERATION FAILED FLAG
      01D8 0000
      01DA FFDE
0577 01DC
0578 01DC      ERR020
0579 01DC 06CB      SWPB R11      GET THE IRB ERROR CODE
0580 01DE D84B      MOVB R11,@IRBEC(IRB)  SET THE IRB ERROR CODE
      01E0 FFFF
0581 01E2 04E7      CLR  @PDTS11(PDT)      CLEAR THE LOG ERROR
      01E4 FFE4
0582 01E6      ERR030
0583 01E6
0584 01E6      IGN010
0585 01E6      IGN100
0586 01E6      IGN200
0587 01E6      RETURN
0588 01E6 C067      MOV  @PDTSRB(PDT),IRB      RESTORE PRB ADDRESS
      01E8 FFFC
0589 01EA 1304      JEQ  EXIT      IF NONE, TAKE JUMP
0590 01EC 06A0      BL   @ENDRCD      SCHEDULE EOR PROCESSOR
      01EE 0000
0591 01F0 06A0      BL   @CLN010      CLEAN UP FOR EXIT
      01F2 03F8
0592 01F4
0593 01F4 0380      EXIT  RTWP
0594 01F6      DBGINS
*0001 01F6 1000      DATA >1000      MACRO DEBUG INSTRUCTION(S)
0595 01F8

```

Figure 5-22. Example DSRs (Sheet 35 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111    13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0024
0597          *****
0598          *
0599          *  REMTSR - THIS IS THE ENTRY POINT USED WHEN THE
0600          *  REENTER-ME FLAG IS SET.
0601          *****
0602 01F8
0603 01F8      A13REM
0604 01F8      A23REM
0605 01F8      A33REM
0606 01F8      REM000
0607 01F8      DBGINS
*0001 01F8 1000      DATA >1000      MACRO DEBUG INSTRUCTION(S)
0608 01FA C2D9      MOV *R9,R11      GET FLAG WORD
0609 01FC 22E0      REM100 COC @TF0WIA*2+MASTAB,R11 IS TRANSMIT ACTIVE FLAG SET?
0610 01FE 0002
0610 0200 164C      JNE REM400      NO - EXIT
0611 0202
0612 0202 22E0      REM200 COC @TF0RRW*2+MASTAB,R11 REENTER DUE TO XMT?
0613 0204 0012
0613 0206 1649      JNE REM400      NO - TAKE JUMP
0614 0208 4660      SZC @TF0RRW*2+MASTAB,*R9 RESET REENTER DUE TO XMT
0615 020A 0012
0615 020C
0616 020C 0608      REM220 DEC R8      DECREMENT XMIT COUNT
0617 020E D176      MOV *R6+,R5      GET TRANSMIT CHARACTER
0618 0210 C2D9      MOV *R9,R11      GET FLAG WORD
0619 0212 22E0      COC @TF0EPF*2+MASTAB,R11 EXTENDED PRINT FLAG SET?
0620 0214 000E
0620 0216 1308      JEQ REM230      YES - TAKE JUMP
0621 0218
0622 0218 0285      CI R5,>6100      CHARACTER LESS THAN >61?
0623 021A 6100
0623 021C 1A05      JL REM230      YES - TAKE JUMP
0624 021E
0625 021E 0285      CI R5,>7B00      CHARACTER HIGHER THAN >7B?
0626 0220 7B00
0626 0222 1402      JHE REM230      YES - TAKE JUMP
0627 0224 0245      ANDI R5,#>2000   MASK OFF LOWER CASE BIT
0628 0226 DFFF
0628 0228

```

Figure 5-22. Example DSRs (Sheet 36 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111    13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0025
0630 0228 20A0  REM230 COC @DSFJIS*2+MASTAB,R2 JISCII FLAG SET?
      022A 000E
0631 022C 1619          JNE  REM270          NO - TAKE JUMP
0632 022E D2E7          MOVB @PDTF LG(PDT),R11  GET PDT FLAGS
      0230 FFDF
0633 0232 C145          MOV  R5,R5          MSB (JISCII) SET?
0634 0234 1109          JLT  REM260          YES - TAKE JUMP
0635 0236
0643 0236
0647 0236 22E0          COC @DFGJAR*2+MASTAB-16,R11 PRINTER IN ALPHA MODE?
      0238 0002
0651 023A 1612          JNE  REM270          YES - TAKE JUMP
0652 023C
0660 023C
0664 023C 59E0          SZCB @DFGJAR*2+MASTAB-16,@PDTF LG(PDT) NO-SET ALPHA
      023E 0002          MODE
      0240 FFDF
0668 0242 0205          LI   R5,>0F00          LOAD SHIFT IN CODE
      0244 0F00
0669 0246 1008          JMP  REM265
0670 0248
0671 0248 0248` REM260 EQU $
0679 0248
0683 0248 22E0          COC @DFGJAR*2+MASTAB-16,R11 PRINTER IN ALPHA MODE?
      024A 0002
0687 024C
0688 024C 1309          JEQ  REM270          NO - TAKE JUMP
0689 024E
0697 024E
0701 024E F9E0          SOCB @DFGJAR*2+MASTAB-16,@PDTF LG(PDT) SET KATA MODE
      0250 0002
      0252 FFDF
0705 0254
0706 0254 0205          LI   R5,>0E00          LOAD SHIFT OUT CODE
      0256 0E00
0707 0258 0588  REM265 INC  R8          INC COUNT TO BXMIT BYTE
0708 025A 0606          DEC  R6          DECREMENT BUFFER ADDRESS
0709 025C E660          SOC  @TF0KFG*2+MASTAB,*R9  SET "SHIFT" CHARACTER
      025E 000C          FLAG
0710 0260
0711 0260 06A0  REM270 BL   @HOFTP4          CALL OUTPUT SUBROUTINE
      0262 0000
0712 0264 028E`          DATA REM300          FIFO FULL RETURN
0713 0266

```

Figure 5-22. Example DSRs (Sheet 37 of 60)



```

DSSPTSR      SDSMAC 3.6.0 83.111    13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0026
0715 0266 4660      SZC  @TF0KFG*2+MASTAB,*R9    RESET "SHIFT" CHAR XMIT
      0268 000C
0716 026A C208      MOV  R8,R8                    XMIT COMPLETE?
0717 026C 16CF      JNE  REM220                    NO - CONTINUE OUTPUT
0718 026E
0719 026E C067      MOV  @PDTSRB(PDT),IRB      RESTORE PRB ADDRESS
      0270 FFFC
0720 0272 1309      JEQ  REM290                    IF NONE, TAKE JUMP
0721 0274
0722 0274 C2D9      MOV  *R9,R11                    GET FLAG WORD
0723 0276 22E0      COC  @TF0OIA*2+MASTAB,R11  IS OTHER ACTIVE?
      0278 0006
0724 027A 1303      JEQ  REM280                    YES - TAKE JUMP
0725 027C C861      MOV  @IRBOCC(IRB),@IRBICC(IRB)  UPDATE COUNT
      027E 0008
      0280 0006
0726 0282
0727 0282 06A0      REM280 BL  @ENDRCD                    SCHEDULE EOR PROCESSOR
      0284 01EE
0728 0286
0729 0286 06A0      REM290 BL  @CLN010                    CLEAN IT UP
      0288 03F8
0730 028A 0460      B    @REM000                    CONTINUE LOOKING FOR MORE
      028C 01F8
0731 028E
0732 028E
0733 028E C2D9      REM300 MOV  *R9,R11                    GET FLAG WORD
0734 0290 22E0      COC  @TF0KFG*2+MASTAB,R11  "SHIFT" CHARACTER XMIT SET?
      0292 000C
0735 0294 1304      JEQ  REM500                    YES - TAKE JUMP
0736 0296 0588      INC  R8                    INCREMENT COUNT TO TRANSMIT
0737 0298 0606      DEC  R6                    DECREMENT BUFFER ADDRESS
0738 029A 029A      REM350 EQU  $
0739 029A 0460      REM400 B    @EXIT
      029C 01F4
0740 029E
0741 029E
0742 *              MODE CHANGE CHARACTER DID NOT GO OUT, RESTORE FLAGS
0743 029E
0744 029E 4660      REM500 SZC  @TF0KFG*2+MASTAB,*R9    RESET "SHIFT" CHAR XMIT
      02A0 000C
0745 02A2 D2E7      MOV  @PDNFLG(PDT),R11        GET FLAG WORD
      02A4 FFFD
0746 02A6
0754 02A6
0758 02A6 22E0      COC  @DFGJAR*2+MASTAB-16,R11  KATA RCV SET?
      02A8 0002
0762 02AA
0763 02AA 1604      JNE  REM550                    NO - TAKE JUMP
0764 02AC
0772 02AC
0776 02AC 59E0      SZCB @DFGJAR*2+MASTAB-16,@PDNFLG(PDT)  SET ALPHA MODE
      02AE 0002
      02B0 FFFD
0780 02B2
0781 02B2 10F3      JMP  REM350
0782 02B4
0790 02B4
0794 02B4 F9E0      REM550 SOCB @DFGJAR*2+MASTAB-16,@PDNFLG(PDT)  SET KANA MODE
      02B6 0002

```

Figure 5-22. Example DSRs (Sheet 38 of 60)

```
DSSPTR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.  
DSSPTR - DX10 - TSR SECTION - PDT WORKSPACE  
          02B8 FFDF  
0798 02BA 10EF           JMP  REM350  
0799 02BC
```

PAGE 0027

**Figure 5-22. Example DSRs (Sheet 39 of 60)**

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.           PAGE 0028
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE
*****
0801
0802      *
0803      *   ABORT - I/O ABORT ROUTINE
0804      *   THE SYSTEM ENTERS HERE WHEN A DEVICE TIME OUT OCCURS
0805      *   OR A KILL I/O SVC IS RECEIVED.  CURRENT OPERATION IS
0806      *   TERMINATED AND APPROPRIATE ERROR CODE IS PASSED
0807      *   TO THE SVC BLOCK.  DSR IS EXITED THROUGH 'EXIT'.
0808      *
0809      *
0810      *
*****
0811
0812
0813 02BC  ABT000
0814      *   NEED TO RESET CERTAIN FLAGS HERE
0815      *   NEED TO RESET CERTAIN WORDS HERE
0816 02BC  DBGINS
*0001 02BC 1000  DATA >1000  MACRO DEBUG INSTRUCTION(S)
0817 02BE 06A0  BL   @CLN010  CLEAN UP ALL FLAGS AND POINTERS
      02C0 03F8
0818 02C2 020B  LI   R11,>0606  ABORT I/O ERROR
      02C4 0606
0819 02C6 0460  B    @ERR010
      02C8 01C4
0820      *
0821      *
0822 02CA  PWR000
0823 02CA  DBGINS
*0001 02CA 1000  DATA >1000  MACRO DEBUG INSTRUCTION(S)
0824 02CC C1C4  MOV  R4,PDT  INITIALIZE PDT FOR SUBSYSTEM
0825 02CE C067  MOV  @PDTSRB(PDT),IRB  IRB ACTIVE?
      02D0 FFFC
0826 02D2 1312  JEQ  PWR020  NO - TAKE PWR020 JUMP
0827 02D4
0828 02D4 020B  LI   R11,>0404  LOAD POWER RESTORED CODE
      02D6 0404
0829 02D8 E860  SOC  @IRFERR*2+MASTAB,@IRBSFL(IRB)  SET ERROR IN PRB
      02DA 0002
      02DC 0002
0830 02DE
0831 02DE D9CB  MOVB R11,PDTErr(PDT)  LOG THE ERROR?
      02E0 FFDE
0832 02E2 1303  JEQ  PWR010  NO - TAKE PWR010 JUMP
0833 02E4 F9E0  SOCB @BYTE02,@PDTErr(PDT)  SET OPERATION FAILED FLAG
      02E6 01D8
      02E8 FFDE
0834 02EA
0835 02EA  PWR010
0836 02EA 06CB  SWPB R11  GET THE IRB ERROR CODE
0837 02EC D84B  MOVB R11,IRBEC(IRB)  SET THE IRB ERROR CODE
      02EE FFFF
0838 02F0 04E7  CLR  @PDTS11(PDT)  CLEAR THE LOG ERROR
      02F2 FFE4
0839 02F4 06A0  BL   @ENDRCD  CALL END RECORD
      02F6 0284

```

Figure 5-22. Example DSRs (Sheet 40 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0029
0841 02F8
0842 02F8          PWR020
0843 02F8 C247      MOV  PDT,R9          MOVE PDT POINTER TO R9
0844 02FA 0229      AI   R9,VDTEX0        ADD IN OFFSET TO FLAG WORD
      02FC 003A
0845 02FE C2D9      MOV  *R9,R11         GET EXTENTION WORD 1
0846 0300 22E0      COC  @TF0RES*2+MASTAB,R11 IS THIS POWER RESTORE?
      0302 001A
0847 0304 1325      JEQ  PWR030         YES - TAKE PWR030 JUMP
0848 0306
0877 0306
0878 0306 C147      MOV  PDT,R5          GET PDT ADDRESS
0879 0308 0225      AI   R5,PDXMAP      SET LONG DISTANCE MAP FILE
      030A 002C
0880 030C
0888 030C
0892 030C C9C5      MOV  R5,@KSBEF(PDT)  SET UP KSB REG 5
      030E 000A
0896 0310
0897 0310 020B      LI   R11,HSRBGN     LOAD REMOTE START ADDRESS
      0312 0000
0898 0314 0795      PWR023 LDS *R5          SET LONG DISTANCE
0899 0316 04FB      CLR  *R11+          CLEAR THE WORD
0900 0318 028B      CI   R11,TSREND    AT END OF TSR AREA
      031A 00C0
0901 031C 1AFB      JL   PWR023         NO - TAKE JUMP
0902 031E
0903 031E C9C4      MOV  R4,@KSBR7(PDT) SET UP KSB REGISTER 7
      0320 000E
0904 0322 C9C9      MOV  R9,@KSBR9(PDT) SET UP PDT/KSB REG 9
      0324 0012
0905 0326 E660      SOC  @TF0RES*2+MASTAB,*R9 SET POWER RESTORE FLAG
      0328 001A
0906 032A C2E7      MOV  @VDTEX1(PDT),R11 GET THE BAUD RATE FROM PDT
      032C 003C
0907 032E 024B      ANDI R11,SPDMSK     MASK OUT UNUSED BITS
      0330 1F1F
0908 0332 C9CB      MOV  R11,@VDTEX1(PDT) MOVE BAUD RATE
      0334 003C
0909 0336
0910 0336 4660      SZC  @TF0EPF*2+MASTAB,*R9 CLEAR EXTENDED FLAG
      0338 000E
0911 033A 0767      ABS  @VDTEX2(PDT)   EXTENDED PRINT?
      033C 003E
0912 033E 1302      JEQ  PWR026         NO - TAKE JUMP
0913 0340 E660      SOC  @TF0EPF*2+MASTAB,*R9 SET EXTENDED FLAG
      0342 000E
0914 0344

```

Figure 5-22. Example DSRs (Sheet 41 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                                PAGE 0030
0916 0344
0917 0344 020B PWR026 LI   R11,DEFPRM          LOAD UP DEFAULT PARMS
      0346 1820
0918 0348 C9CB          MOV   R11,@VDTEX2(PDT)      MOVE PARITY SELECT ET AL
      034A 003E
0919 034C 04E7          CLR   @VDTEX3(PDT)          CLEAR TOTAL ERROR COUNT
      034E 0040
0920 0350
0921 0350
0922 0350 06A0 PWR030 BL   @HRESET              CALL POWER UP ROUTINE
      0352 0000
0923 0354 03BC^        DATA PWR100              POWER UP FAILURE
0924 0356
0925 0356 4660 PWR032 SZC @TF0DSR*2+MASTAB,*R9 CLEAR DSR HOLD FLAG
      0358 0008
0926 035A 4660          SZC @TF0DC3*2+MASTAB,*R9 CLEAR DC3 HOLD FLAG
      035C 000A
0927 035E
0928 035E 06A0          BL   @HSWPWR              CALL SOFTWARE POWER UP
      0360 0000
0929 0362 03BC^        DATA PWR100              POWER UP FAILURE
0930 0364
0931 0364 C027          MOV   @VDTEX1(PDT),R0      GET BAUD RATE
      0366 003C
0932 0368 06A0          BL   @HSPSPD              CALL LOAD BAUD SUBROUTINE
      036A 0000
0933 036C 03BC^        DATA PWR100              FUNCTION FAILED RETURN
0934 036E
0935 036E C027          MOV   @VDTEX2(PDT),R0      GET PARITY, CHAR LEN, BREAK
      0370 003E
0936 0372 06A0          BL   @HSPPSL              CALL LOAD OPERATIONAL PARM
      0374 0000
0937 0376 03BC^        DATA PWR100              FUNCTION FAILED RETURN
0938 0378
0939 0378 06A0          BL   @HSTDTR              CALL SET DTR SUBROUTINE
      037A 0000
0940 037C 03BC^        DATA PWR100              FUNCTION FAILED RETURN
0941 037E
0942 037E 06A0          BL   @HSTRTS              CALL SET RTS SUBROUTINE
      0380 0000
0943 0382 03BC^        DATA PWR100              FUNCTION FAILED RETURN
0944 0384
0945 0384 06A0          BL   @HSTSRT              CALL SET SRTS SUBROUTINE
      0386 0000
0946 0388 038A^        DATA PWR035              FUNCTION FAILED RETURN
0947 038A

```

Figure 5-22. Example DSRs (Sheet 42 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                PAGE 0031
0949 038A 06A0 PWR035 BL @HRDDSR          CALL READ DSR SIGNAL
      038C 0000
0950 038E 03BC      DATA PWR100          UNSUPPORTED FEATURE RETURN
0951 0390 03BC      DATA PWR100          FUNCTION FAILED RETURN
0952 0392 0396      DATA PWR040          SIGNAL FALSE RETURN
0953 0394 1005      JMP PWR050            TRUE RETURN
0954 0396
0955 0396
0956 0396 E660 PWR040 SOC @TF0DSR*2+MASTAB,*R9 SET DSR HOLD FLAG
      0398 0008
0957 039A 06A0      BL @HSTCTH           CALL SET CHANNEL XMIT HALT
      039C 0000
0958 039E 03BC      DATA PWR100          UNSUPPORTED FEATURE RETURN
0959 03A0
0960 03A0 06A0 PWR050 BL @HESDSR          CALL ENABLE DSR CHANGE NOTIF
      03A2 0000
0961 03A4 03BC      DATA PWR100          UNSUPPORTED FEATURE RETURN
0962 03A6
0963 03A6
0964 03A6
0965 03A6
0966 03A6 PWR090
0967 03A6 C067      MOV @PDTSRB(PDT),IRB  IRB ACTIVE?
      03A8 FFFC
0968 03AA 1304      JEQ PWR095          NO - TAKE JUMP
0969 03AC 04E7      CLR @PDSSL1(PDT)   CLEAR THE LOG ERROR
      03AE FFE4
0970 03B0 06A0      BL @ENDRCD          CALL END RECORD
      03B2 02F6
0971 03B4
0972 03B4 06A0 PWR095 BL @CLN010          CLEAN UP FLAGS AND VECTORS
      03B6 03F8
0973 03B8 0460      B @EXIT            RETURN
      03BA 01F4
0974 03BC

```

Figure 5-22. Example DSRs (Sheet 43 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                               PAGE 0032
 0976 03BC          PWR100
 0977 03BC          DBGINS
*0001 03BC 1000     DATA >1000      MACRO DEBUG INSTRUCTION(S)
 0978 03BE C067     MOV  @PDTSRB(PDT),IRB      IRB ACTIVE?
          03C0 FFFC
 0979 03C2 130E     JEQ  PWR150          NO - TAKE JUMP
 0980 03C4
 0981 03C4 020B     LI   R11,>0707          LOAD DEVICE FAIL CODE
          03C6 0707
 0982 03C8 E860     SOC  @IRFERR*2+MASTAB,@IRBSFL(IRB)  SET ERROR IN PRB
          03CA 0002
          03CC 0002
 0983 03CE
 0984 03CE D9CB     MOVB R11,PDTErr(PDT)      LOG THE ERROR?
          03D0 FFDE
 0985 03D2 1303     JEQ  PWR110          NO - TAKE PWR110 JUMP
 0986 03D4 F9E0     SOCB @BYTE02,@PDTErr(PDT)  SET OPERATION FAILED FLAG
          03D6 02E6
          03D8 FFDE
 0987 03DA
 0988 03DA          PWR110
 0989 03DA 06CB     SWPB R11              GET THE IRB ERROR CODE
 0990 03DC D84B     MOVB R11,IRBEC(IRB)    SET THE IRB ERROR CODE
          03DE FFFF
 0991 03E0
 0992 03E0          PWR150
1001 03E0
1005 03E0 59E0     SZCB @DFGSTA*2+MASTAB-16,@PDTEFLG(PDT)  SET OFFLINE
          03E2 0006
          03E4 FFDF
1006 03E6 F9E0     SOCB @DFGSTA+1*2+MASTAB-16,@PDTEFLG(PDT)  SET OFFLINE
          03E8 0008
          03EA FFDF
1010 03EC
1011 03EC E660     SOC  @TFOCFE*2+MASTAB,*R9 SET CHANNEL FAIL FLAG
          03EE 0018
1012 03F0 06A0     BL   @HSTCR           CALL SET CHANNEL RESET
          03F2 0000
1013 03F4 03A6     DATA PWR090          FUNCTION FAILED RETURN
1014 03F6 10D7     JMP  PWR090          RETURN
1015 03F8

```

Figure 5-22. Example DSRs (Sheet 44 of 60)

```

DSSPTSR   SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
DSSPTSR - DX10 - TSR SECTION - PDT WORKSPACE                               PAGE 0033
1017      *****
1018      *   TITLE: CLN010 - CLEAN UP SUBROUTINE                          *
1019      *                                                                 *
1020      *   ENTERED VIA:                                                  *
1021      *                                                                 *
1022      *       BL   @CLN010   FOR CLEAN UP ROUTINE                      *
1023      *                                                                 *
1024      *   CREATED 10/12/82 BY CECIL GREATHOUSE                        *
1025      *****
1026      *
1027 03F8 04C8 CLN010 CLR R8          CLEAR CHARACTER COUNT
1028 03FA 04C6      CLR R6          CLEAR BUFFER ADDRESS
1029 03FC C247      MOV PDT,R9      SET UP REG 9
1030 03FE 0229      AI R9,VDTEX0    RESTORE FLAG WORD POINTER
      0400 003A
1031 0402 4660      SZC @TF0WIA*2+MASTAB,*R9 RESET TRANSMIT ACTIVE FLAG
      0404 0002
1032 0406 4660      SZC @TF0RRW*2+MASTAB,*R9 RESET REENTER XMT FLAG
      0408 0012
1033 040A 4660      SZC @TF0OIA*2+MASTAB,*R9 RESET OTHER ACTIVE FLAG
      040C 0006
1034 040E 4660      SZC @TF0KFG*2+MASTAB,*R9 RESET "SHIFT" CHAR XMIT
      0410 000C
1035 0412 045B      RT RETURN
1036 0414
1037 0414
1038 0414

```

Figure 5-22. Example DSRs (Sheet 45 of 60)



```

DSSPTRS      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
ISRSLP - ISR SECTION - KSB WORKSPACE                                PAGE 0034
1041          *****
1042          *
1043          *   INTSLP - THIS IS THE ENTRY POINT FOR A DEVICE INTERRUPT.
1044          *   THE INTERRUPT ROUTINE IS IN THE INTERRUPT SERVICE
1045          *   ROUTINE SECTION OF THE TSR/ISR CODE.
1046          *
1047          *
1048          *****
1053 0414     A13INT
1054 0414     A23INT
1055 0414     A33INT
1056 0414     INT000
1057 0414     DBGINS
*0001 0414 1000     DATA >1000     MACRO DEBUG INSTRUCTION(S)
1058 0416 06A0     BL   @HNOTIF     CALL NOTIFICATION ROUTINE
          0418 0000
1059 041A 0424     DATA IRC010     RECEIVE INTERRUPT
1060 041C 04A8     DATA ITC010     TRANSMIT INTERRUPT
1061 041E 04BE     DATA ISF010     SIGNAL/FUNCTION INTERRUPT
1062 0420 0414     DATA INT000     TIMER INTERRUPT
1063 0422 04D4     DATA ISE010     ILLEGAL/INVALID INTERRUPT
1064 0424

```

Figure 5-22. Example DSRs (Sheet 46 of 60)

```

DSSPTRS      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
ISRSLP - ISR SECTION - KSB WORKSPACE                                PAGE 0035
1066          *****
1067          *
1068          *   IRC010 - READ INTERRUPT RECEIVED
1069          *   CHARACTERS ARE READ INTO THE USER BUFFER WHEN
1070          *   A RECEIVE IS ACTIVE. WHEN NO RECEIVE IS ACTIVE THE
1071          *   RECEIVE CHARACTER IS TOSSED. WHEN A RECEIVE IS
1072          *   COMPLETE, THE USER BUFFER IS FULL, REENTER-ME LOGIC
1073          *   IS USED TO ACTIVATE THE TSR CODE TO RETURN THE
1074          *   IRB.
1075          *
1076          *   REGISTERS USED:
1077          *
1078          *   R5 - ADDRESS OF MAP FILE TO LONG DISTANCE BUFFER
1079          *   R7 - ADDRESS OF PDT
1080          *   R8 - SCRATCH
1081          *   R10 - RETURNED CHARACTER BYTE / ERROR FLAGS
1082          *   R12 - CRU/TILINE ADDRESS
1083          *
1084          *****
1085 0424      IRC010
1086 0424 C00A      MOV R10,R0          SAVE CHARACTER
1087 0426 0240      ANDI R0,>00FF      ERROR RETURNED?
1088           0428 00FF
1088 042A 1328      JEQ IRC050          NO - TAKE JUMP
1089 042C 05A7      INC @VDTEX3(PDT)    INC TOTAL ERROR COUNT
1089           042E 0040
1090 0430
1091 0430 2020      COC @ERRBI*2+MASTAB,R0  BREAK INTERRUPT?
1091           0432 0016
1092 0434 1605      JNE IRC020          NO -TAKE JUMP
1093 0436 0795      LDS *R5            SET LONG DISTANCE MAP FILE
1094 0438 05A0      INC @RTEXT0        INC RECEIVED BREAK COUNT
1094           043A 0090
1095 043C 4020      SZC @ERRBI*2+MASTAB,R0  RESET BIT
1095           043E 0016
1096 0440
1097 0440 2020      IRC020 COC @ERRFE*2+MASTAB,R0  FRAMING ERROR?
1097           0442 0018
1098 0444 1605      JNE IRC030          NO -TAKE JUMP
1099 0446 0795      LDS *R5            SET LONG DISTANCE MAP FILE
1100 0448 05A0      INC @RTEXT1        INCREMENT FRAMING ERROR COUNT
1100           044A 0092
1101 044C 4020      SZC @ERRFE*2+MASTAB,R0  RESET BIT
1101           044E 0018
1102 0450
1103 0450 2020      IRC030 COC @ERRPE*2+MASTAB,R0  PARITY ERROR?
1103           0452 001A
1104 0454 1605      JNE IRC035          NO -TAKE JUMP
1105 0456 0795      LDS *R5            SET LONG DISTANCE MAP FILE
1106 0458 05A0      INC @RTEXT2        INCREMENT PARITY ERROR COUNT
1106           045A 0094
1107 045C 4020      SZC @ERRPE*2+MASTAB,R0  RESET BIT
1107           045E 001A
1108 0460

```

Figure 5-22. Example DSRs (Sheet 47 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
ISRSLP - ISR SECTION - KSB WORKSPACE                                PAGE 0036
1110 0460 2020  IRC035 COC @ERROE*2+MASTAB,R0 RECEIVER OVERRUN ERROR?
      0462 001C
1111 0464 1605          JNE IRC040          NO -TAKE JUMP
1112 0466 0795          LDS *R5          SET LONG DISTANCE MAP FILE
1113 0468 05A0          INC @RTEXT3          INCREMENT OVERRUN ERROR COUNT
      046A 0096
1114 046C 4020          SZC @ERROE*2+MASTAB,R0 RESET BIT
      046E 001C
1115 0470
1116 0470 C000  IRC040 MOV  R0,R0          ANY "OTHER" BIT SET?
1117 0472 13D0          JEQ INT000          NO - TAKE JUMP
1118 0474 0795          LDS *R5          SET LONG DISTANCE MAP FILE
1119 0476 05A0          INC @RTEXT4          INCREMENT OTHER ERROR COUNT
      0478 0098
1120 047A 10CC          JMP  INT000
1121 047C
1122 047C
1123 047C 980A  IRC050 CB   R10,@DC3          IS RECEIVED CHARACTER A HOLD?
      047E 04E5
1124 0480 1606          JNE IRC100          NO - TAKE JUMP
1125 0482
1126 0482 E660          SOC @TF0DC3*2+MASTAB,*R9 SET DC3 HOLD FLAG
      0484 000A
1127 0486 06A0  IRC070 BL   @HSTCTH          CALL SET CHANNEL XMIT HALT
      0488 039C
1128 048A 0414          DATA INT000          UNSUPPORTED FEATURE RETURN
1129 048C 10C3          JMP  INT000
1130 048E
1131 048E
1132 048E 980A  IRC100 CB   R10,@DC1          RELEASE XMIT HOLD FOR DC3?
      0490 04E4
1133 0492 16C0          JNE INT000          NO - TAKE JUMP
1134 0494
1135 0494 4660          SZC @TF0DC3*2+MASTAB,*R9 RESET DC3 HOLD FLAG
      0496 000A
1136 0498 C299          MOV  *R9,R10          GET MAJOR FLAG WORD
1137 049A 22A0          COC @TF0DSR*2+MASTAB,R10 HOLD FOR DSR STILL ACTIVE?
      049C 0008
1138 049E 13BA          JEQ  INT000          YES - TAKE JUMP
1139 04A0
1140 04A0 06A0  IRC150 BL   @HRTCTH          CALL RESET CHANEL XMIT HALT
      04A2 0000
1141 04A4 0414          DATA INT000          UNSUPPORTED FEATURE RETURN
1142 04A6 1000          JMP  ITC010          FORCE XMIT INTO OPERATION
1143 04A8

```

Figure 5-22. Example DSRs (Sheet 48 of 60)

```

DSSPTS  SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983.
ISRSLP - ISR SECTION - KSB WORKSPACE PAGE 0037
1145 *****
1146 *
1147 * ITC010 - TRANSMIT WAS FULL AND TRANSMIT FIFO IS EMPTY
1148 * THIS WILL PUT THE TRANSMIT SECTION OF THE TSR
1149 * BACK INTO OPERATION.
1150 *
1151 *
1152 *****
1153 04A8 ITC010
1154 04A8 C9E7 MOV @PDTT1(PDT),@PDTT2(PDT) RESET TIMEOUT COUNT
      04AA FFF8
      04AC FFFA
1155 04AE E660 SOC @TFORRW*2+MASTAB,*R9 SET REENTER XMT
      04B0 0012 FLAG
1156 04B2
1160 04B2 C007 MOV PDT,R0 LOAD R0 WITH PDT ADDRESS
1161 04B4
1170 04B4
1174 REF TMSFST
1175 04B4 0220 AI R0,PDTLNK POINT TO THE PDT START
      04B6 FFB6
1176 04B8 0420 BLWP @TMSFST CALL DSR SCHEDULE CODE
      04BA 0000
1180 04BC
1184 04BC
1192 04BC
1193 04BC 10AB JMP INT000 CHECK FOR MORE TO DO
1194 04BE
1195 04BE
1196 04BE

```

Figure 5-22. Example DSRs (Sheet 49 of 60)

```

DSSPTSR      SDSMAC 3.6.0 83.111    13:16:24 FRIDAY, MAY 06, 1983.
ISRSLP - ISR SECTION - KSB WORKSPACE                                PAGE 0038
1198          *****
1199          *
1200          *   ISF010 - A SIGNAL OR FUNCTION HAS CHANGED AND ITS
1201          *   STATUS REPORT WAS REQUESTED
1202          *
1203          *
1204          *****
1205 04BE      ISF010
1206 04BE 4660  SZC  @TF0DC3*2+MASTAB,*R9 RESET DC3 HOLD FLAG
      04C0 000A
1207 04C2 22A0  COC  @DSRSET*2+MASTAB,R10 IS DSR NOW SET?
      04C4 0004
1208 04C6 1303  JEQ  ISF050                YES - TAKE JUMP
1209 04C8 E660  SOC  @TF0DSR*2+MASTAB,*R9 SET DSR HOLD FLAG
      04CA 0008
1210 04CC 10DC  JMP  IRC070
1211 04CE
1212 04CE
1213 04CE 4660  ISF050 SZC  @TF0DSR*2+MASTAB,*R9 RESET DSR HOLD FLAG
      04D0 0008
1214 04D2 10E6  JMP  IRC150
1215 04D4
1216 04D4
    
```

Figure 5-22. Example DSRs (Sheet 50 of 60)

```
DSSPTR      SDSMAC 3.6.0 83.111   13:16:24 FRIDAY, MAY 06, 1983.
ISRSLP - ISR SECTION - KSB WORKSPACE                                PAGE 0039
1218      *****
1219      *
1220      *   ISE010 - SPURIOUS OR ILLEGAL INTERRUPT OCCURRED
1221      *
1222      *****
1225 04D4  ISE010
1226 04D4 05A7      INC  @VDTEX3(PDT)      INCREMENT TOTAL ERROR COUNT
      04D6 0040
1227 04D8 0795      LDS  *R5              SET LONG DISTANCE MAP FILE
1228 04DA 05A0      INC  @RTEXT5         INCREMENT ILLEGAL INTERRUPT
      04DC 009A
1229 04DE 109A      JMP  INT000          SCAN FOR MORE TO DO
1230 04E0
```

Figure 5-22. Example DSRs (Sheet 51 of 60)

```
DSSPTRS      SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983.
ISRSLP - ISR SECTION - KSB WORKSPACE                                PAGE 0040
1232          *
1233 04E0 0D0D  CRCR   DATA >0D0D
1234 04E2 0D0C  CRFF   DATA >0D0C
1235 04E4   11  DC1    BYTE >11
1236 04E5   13  DC3    BYTE >13
1237          *
1238 04E6          EVEN
1239 04E6          PATCH BSS >50
1240 0536
1241          END
NO ERRORS,          NO WARNINGS
```

Figure 5-22. Example DSRs (Sheet 52 of 60)

DSSPTSR SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983.											
LABEL	VALUE	DEFN	REFERENCES								PAGE 0041
\$	0536		B0053	B0090	C0027	C0027	C0056	C0056	C0057	C0057	
			C0058	D0022	D0022	D0028	D0028	D0029	D0029	D0030	
			D0036	D0036	D0038	G0009	G0011	G0015	G0019	G0023	
			0462	0495	0551	0671	0738			0309	
A13HAN	D 0000	0215	0069								
A13INT	D 0414	1053	0070								
A13REM	D 01F8	0603	0071								
A23HAN	D 0000	0216	0069								
A23INT	D 0414	1054	0070								
A23REM	D 01F8	0604	0071								
A33HAN	D 0000	0217	0069								
A33INT	D 0414	1055	0070								
A33REM	D 01F8	0605	0071								
ABT000	02BC	0813	0240								
BRSTAT	R 0042	0075	0278								
BYTE02	R 03D6	0113	0576	0833	0986						
C\$DNOS	0004	A0014									
C\$DPOS	0004	A0013	A0014	0003	0084	0219	0553	0637	0654	0673	
			0748	0766	0784	0850	0882	0994	1163	0691	
C\$DX10	0003	A0012	A0019	A0020	0012	0111	0237	0561	0645	0662	
			0699	0756	0774	0792	0890	1003	1172	0681	
C\$DX34	0005	A0015									
C\$DX5	0001	A0010									
C\$DX7	0002	A0011									
C\$NAM1	0044	A0021									
C\$NAM2	0058	A0022									
C\$NAM3	0031	A0023									
C\$NAM4	0030	A0024									
C\$OS	0003	A0019	0003	0012	0084	0111	0219	0237	0553	0561	
			0645	0654	0662	0673	0681	0691	0699	0748	
			0766	0774	0784	0792	0850	0882	0890	0994	
			1163	1172						1003	
C\$OSV	0003	A0020									
C\$REL	0035	A0026									
C\$VER	0033	A0025									
CI403	2300	0458	0468								
CI404	2400	0459	0470								
CLN010	03F8	1027	0591	0729	0817	0972					
CLS010	008A	0336	0288								
CRCR	04E0	1233	0337								
CRFF	04E2	1234	0352								
DBGINS	M 0133	0254	0594	0607	0816	0823	0977	1057			
DC1	04E4	1235	1132								
DC3	04E5	1236	1123								
DEFPRM	1820	0197	0917								
DFGJAR	0009	C0064	0647	0664	0683	0701	0758	0776	0794		
DFGJAT	000A	C0065									
DFGOPF	000E	C0069									
DFGPRB	0008	C0063	0563								
DFGRTY	000F	C0070									
DFGSTA	000B	C0066	1005	1006							
DMP010	01B2	0538	0306								
DSFASG	0000	C0029									
DSFBSY	0001	C0030									
DSFCLS	0004	C0033									
DSFEOR	0006	C0035									
DSFINT	0002	C0031									
DSFJIS	0007	C0036	0630								
DSFKLL	0003	C0032									
DSFREN	0005	C0034									

Figure 5-22. Example DSRs (Sheet 53 of 60)



DSSPTSR SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983. PAGE 0042

LABEL	VALUE	DEFN	REFERENCES
DSFWPM	000A	C0037	
DSRSCH	0001	0143	0242 1158 1186
DSRSET	0002	0202	1207
DTFCOM	0005	C0044	
DTFEXT	0007	C0046	
DTFFIL	0000	C0039	
DTFKSB	0004	C0043	
DTFPRI	0003	C0042	
DTFSYD	0006	C0045	
DTFTIL	0001	C0040	
DTFTIM	0002	C0041	
ENDRCD	R 03B2	0076	0590 0727 0839 0970
ERR005	01BE	0551	0274 0394
ERR010	01C4	0570	0819
ERR020	01DC	0578	0575
ERR030	01E6	0582	0572
ERRBI	000B	0191	1091 1095
ERRFE	000C	0192	1097 1101
ERROE	000E	0194	1110 1114
ERROR	01C4	0569	
ERRPE	000D	0193	1103 1107
EXIT	01F4	0593	0589 0739 0973
HAN000	000C	0253	0244
HAN002	002A	0269	0261
HAN005	0034	0273	0267
HAN010	003C	0276	0257 0264 0271
HAN020	0044	0284	0309
HAN030	0015	0309	0284
HAN040	0074	0311	0277
HDSCTS	R	E0146	
HSDCD	R	E0150	
HSDSR	R	E0144	
HDSRI	R	E0154	
HDSSCT	R	E0148	
HDSSDC	R	E0152	
HDSTSR	R	E0156	
HESCTS	R	E0145	
HESDCD	R	E0149	
HESDSR	R 03A2	E0143	0960
HESRI	R	E0153	
HESSCT	R	E0147	
HSSDC	R	E0151	
HSTSR	R	E0155	
HMRST	R 0136	E0100	0445
HNOTIF	R 0418	E0102	1058
HOUTP4	R 0262	E0103	0711
HOUTP7	R	E0104	
HRBIL	R	E0139	
HRDCR	R	E0138	
HRDCTH	R	E0137	
HRDCTS	R	E0131	
HRDDCD	R	E0133	
HRDDSR	R 038C	E0130	0949
HRDRI	R	E0135	
HRDSCT	R	E0132	
HRSDC	R	E0134	
HRSSS	R	E0141	
HRDTB	R	E0140	
HRDUIL	R	E0136	
HRESET	R 0352	E0099	0922

Figure 5-22. Example DSRs (Sheet 54 of 60)

DSSPTSR SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983.										
LABEL	VALUE	DEFN	REFERENCES							
HRPDAT	R	E0164								
HRPPSL	R	E0162								
HRPSPD	R	E0161								
HRPTYP	R	0144	E0165 0465							
HRTAL	R	E0116								
HRTBIL	R	E0126								
HRTCR	R	E0124								
HRTCTH	R	04A2	E0122 1140							
HRTDTR	R	E0108								
HRTRS	R	E0118								
HRTSTS	R	E0110								
HRTSRS	R	E0114								
HRTSRT	R	E0112								
HRTTB	R	E0128								
HRTUIL	R	E0120								
HRUART	R	0110	E0167 0427							
HSPPSL	R	0374	E0159 0936							
HSPSPD	R	036A	E0158 0932							
HSRBGN		0000	G0009 0897							
HSREND		0020	G0011 G0013							
HSTAL	R	E0115								
HSTBIL	R	E0125								
HSTCR	R	03F2	E0123 1012							
HSTCTH	R	0488	E0121 0957 1127							
HSTDTR	R	037A	E0107 0939							
HSTRS	R	E0117								
HSTRTS	R	0380	E0109 0942							
HSTSRS	R	E0113								
HSTSRT	R	0386	E0111 0945							
HSTTB	R	E0127								
HSTUIL	R	E0119								
HSWPWR	R	0360	E0101 0928							
HTIMER	R	E0105								
HWUART	R	0102	E0166 0420							
IGN010		01E6	0584 0293							
IGN100		01E6	0585 0294							
IGN200		01E6	0586 0302							
ILL010		01BA	0542 0285 0295							
ILL100		01BA	0543 0296							
ILL200		01BA	0544 0297							
ILL300		01BA	0545 0303							
ILL400		01BA	0546 0304							
ILL500		01BA	0547 0305							
ILL600		01BA	0548 0307							
ILLCDE		01BA	0549 0401 0405 0421 0428							
INT000		0414	1056 1062 1117 1120 1128 1129 1133 1138 1141 1193							
IRB		0001	0078 0260 0269 0371 0373 0376 0397 0419 0426 0430							
			0433 0478 0483 0485 0571 0573 0580 0588 0719							
			0725 0725 0825 0829 0837 0967 0978 0982 0990							
IRBCCO		0011	B0086							
IRBCRO		0010	B0085 0419 0426 0430 0433							
IRBDBA		0004	B0044 0376 0397 0485							
IRBDIA		0007	B0022							
IRBEC		FFFF	B0012 0580 0837 0990							
IRBEVT		000F	B0084							
IRBFCH		000E	B0083							
IRBFCO		0013	B0088							
IRBFRO		0012	B0087							
IRBICC		0006	B0045 0373 0478 0725							

Figure 5-22. Example DSRs (Sheet 55 of 60)

DSSPTSR SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983. PAGE 0044

LABEL	VALUE	DEFN	REFERENCES
IRBLUN	0001	B0016	
IRBOC	0000	B0015	0260
IRBOCC	0008	B0046	0371 0483 0725
IRBRN1	000A	B0047	B0057 B0064 B0065
IRBRN2	000C	B0051	B0058 B0059 B0060 B0066
IRBRPY	000A	B0064	
IRBSFL	0002	B0017	0573 0829 0982
IRBSIZ	0014	B0090	
IRBSOC	FFFE	B0011	
IRBUFL	0003	B0023	0269
IRBVTA	000A	B0065	
IRBXFL	000C	B0066	
IRC010	0424	1085	1059
IRC020	0440	1097	1092
IRC030	0450	1103	1098
IRC035	0460	1110	1104
IRC040	0470	1116	1111
IRC050	047C	1123	1088
IRC070	0486	1127	1210
IRC100	048E	1132	1124
IRC150	04A0	1140	1214
IRFAC1	0004	B0032	
IRFACC	0003	B0031	
IRFACM	0018	B0033	
IRFBAD	0007	B0041	
IRFBFI	0001	B0028	
IRFBP	0006	B0073	
IRFBSY	0000	B0018	
IRFCIF	0008	B0075	
IRFCSF	0000	B0067	
IRFEBA	0004	B0071	
IRFECO	000C	B0079	
IRFEOF	0002	B0020	
IRFERR	0001	B0019	0573 0829 0982
IRFEXR	0006	B0040	
IRFFC	0009	B0076	
IRFFKR	0002	B0069	
IRFGRA	0003	B0070	
IRFIF	000A	B0077	
IRFIMO	0005	B0038	0270
IRFINT	0000	B0024	
IRFKFG	0001	B0027	
IRFLOC	0005	B0035	
IRFMDS	0005	B0036	
IRFNTN	0001	B0068	
IRFOS	0003	B0030	
IRFOSF	0004	B0034	
IRFOWN	0006	B0039	
IRFRDB	0007	B0074	
IRFRES	0002	B0029	
IRFRFF	000B	B0078	
IRFRPY	0001	B0025	
IRFRTY	0007	B0043	
IRFTER	0005	B0072	
IRFTIH	0005	B0037	
IRFVAL	0001	B0026	
IRFVER	000E	B0081	
IRFVNT	0003	B0021	
IRFVRQ	000D	B0080	
IRFWBP	000F	B0082	

Figure 5-22. Example DSRs (Sheet 56 of 60)

How to Write a Device Service Routine

DSSPTSR		SDSMAC 3.6.0 83.111		13:16:24 FRIDAY, MAY 06, 1983.							
LABEL	VALUE	DEFN	REFERENCES								
IRFWPM	0007	B0042									
ISE010	04D4	1225	1063								
ISF010	04BE	1205	1061								
ISF050	04CE	1213	1208								
ITC010	04A8	1153	1060 1142								
KSBBRK	0007	D0019									
KSBCHM	0000	D0012									
KSBCIB	0003	D0015									
KSBCIE	0001	D0013									
KSB CRU	0018	D0027									
KSBE BF	000A	D0010	0892								
KSBFL	000C	D0011									
KSBICP	0004	D0016									
KSBKIO	0006	D0018									
KSBLCK	002A	D0037									
KSBLD0	0020	D0031	D0005								
KSBLD2	0022	D0032									
KSBLD3	0023	D0033									
KSBLD4	0024	D0034									
KSBLD6	0026	D0035									
KSBLD8	0028	D0036									
KSBLDT	0000	D0005	D0038								
KSBQEP	0008	D0009									
KSBQIP	0004	D0007									
KSBQOC	0002	D0006									
KSBQOP	0006	D0008									
KSBR10	0014	D0025									
KSBR11	0016	D0026									
KSBR13	001A	D0028									
KSBR14	001C	D0029									
KSBR15	001E	D0030									
KSBR7	000E	D0021	0903								
KSBR9	0012	D0024	0904								
KSBRCM	0002	D0014									
KSBSET	0005	D0017									
KSBSIZ	002C	D0038	F0008 0162								
KSBSN	000D	D0020									
KSBTSB	0010	D0022	D0023								
KSBVTA	0010	D0023									
MASTAB R	0001	0114	0256 0270 0341 0342 0343 0377 0563 0573 0609								
			0612 0614 0619 0630 0647 0664 0683 0701 0709								
			0715 0723 0734 0744 0758 0776 0794 0829 0846								
			0905 0910 0913 0925 0926 0956 0982 1005 1006								
			1011 1031 1032 1033 1034 1091 1095 1097 1101								
			1103 1107 1110 1114 1126 1135 1137 1155 1206								
			1207 1209 1213								
OLD010	00C0	0388	0308								
OLD020	00D4	0396	0392								
OLD100	00FC	0419	0402								
OLD150	0106	0422	0431 0447								
OLD200	010A	0426	0412								
OLD250	011C	0433	0429								
OLD300	0128	0439	0413								
OLD400	0130	0444	0409								
OLDBFR	0008	B0048									
OPN010	008A	0335	0287								
OPN020	008E	0340	0353								
OPN030	0096	0342	0378								
OPN100	00A2	0347	0290								
PATCH	04E6	1239									

Figure 5-22. Example DSRs (Sheet 57 of 60)

DSSPTS											
SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983.											
LABEL	VALUE	DEFN	REFERENCES								PAGE
PDT	0007	0079	0389	0473	0476	0539	0563	0571	0574	0576	0581
			0588	0632	0664	0701	0719	0745	0776	0794	0824
			0825	0831	0833	0838	0843	0878	0892	0903	0904
			0906	0908	0911	0918	0919	0931	0935	0967	0969
			0978	0984	0986	1005	1006	1029	1089	1154	1154
			1160	1226							
PDT\$	FFDA	C0059									
PDTBLN	FFEA	C0075									
PDTBUF	FFE8	C0074									
PDTCRU	FFD2	C0055									
PDTDIB	FFC2	C0047									
PDTDSF	FFBE	C0028									
PDTDSR	FFDC	C0060									
PDTDTF	FFC0	C0038									
PDTDVQ	FFEE	C0077									
PDTERR	FFDE	C0061	0539	0574	0576	0831	0833	0984	0986		
PDTFLG	FFDF	C0062	0389	0563	0632	0664	0701	0745	0776	0794	1005
			1006								
PDTFQL	FFFE	C0081									
PDTINT	FFEC	C0076									
PDTLNK	FFB6	C0012	C0047	1175							
PDTLUN	FFFF	C0023									
PDTMAP	FFB8	C0025									
PDTMC	FFFC	C0021	0311	0312	0313	0314	0315				
PDTNAM	FFE0	C0071									
PDTPRB	FFBC	C0027									
PDTR0	FFBA	C0026	C0059								
PDTR10	FFCE	C0053									
PDTR11	FFD0	C0054									
PDTR13	FFD4	C0056									
PDTR14	FFD6	C0057									
PDTR15	FFD8	C0058									
PDTR5	FFC4	C0048									
PDTR6	FFC6	C0049									
PDTR7	FFC8	C0050									
PDTR8	FFCA	C0051									
PDTR9	FFCC	C0052									
PDTRC	FFF8	C0019	0320	0321							
PDTRTY	FFFE	C0022									
PDTISZ	0000	C0010	C0011								
PDTSL1	FFE4	C0072	0581	0838	0969						
PDTSL2	FFE6	C0073									
PDTSRB	FFFC	C0080	0571	0588	0719	0825	0967	0978			
PDTTM1	FFF8	C0078	1154								
PDTTM2	FFFA	C0079	1154								
PDTWC	FFFA	C0020	0322	0324	0325						
PDXCHN	0031	F0012	0473								
PDXCP1	0034	F0014									
PDXCP2	0036	F0015									
PDXCP3	0038	F0016									
PDXCP4	003A	F0017	0164								
PDXCP5	003C	F0018	0186								
PDXCP6	003E	F0019	0187								
PDXCP7	0040	F0020	0188								
PDXCP8	0042	F0024									
PDXCP9	0044	F0025									
PDXCPA	0046	F0026									
PDXCPB	0048	F0027									
PDXCPC	004A	F0028									
PDXCPD	004C	F0029									

Figure 5-22. Example DSRs (Sheet 58 of 60)

DSSPTS R SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983. PAGE 0047

LABEL	VALUE	DEFN	REFERENCES										
PDXCPE	004E	F0030											
PDXCPF	0050	F0031											
PDXFCT	0032	F0013											
PDXFLG	0030	F0011											
PDXMAP	002C	0163	0879										
PDXSMB	002C	F0009	0163										
PDXSMP	002E	F0010											
PRBBPS	000C	B0059											
PRBLTA	000C	B0060											
PRBSCT	000C	B0058											
PRBSIZ	000E	B0053											
PRBTRK	000A	B0057											
PSIZ	004A	C0009	C0011	C0047									
PWR000	02CA	0822	0239										
PWR010	02EA	0835	0832										
PWR020	02F8	0842	0826										
PWR023	0314	0898	0901										
PWR026	0344	0917	0912										
PWR030	0350	0922	0847										
PWR032	0356	0925	0440										
PWR035	038A	0949	0946										
PWR040	0396	0956	0952										
PWR050	03A0	0960	0953										
PWR090	03A6	0966	0422	1013	1014								
PWR095	03B4	0972	0968										
PWR100	03BC	0976	0435	0446	0923	0929	0933	0937	0940	0943	0950		
			0951	0958	0961								
PWR110	03DA	0988	0985										
PWR150	03E0	0992	0979										
R0	0000		0419	0426	0430	0466	0467	0468	0470	0486	0487		
			0931	0935	1086	1087	1091	1095	1097	1101	1103		
			1107	1110	1114	1116	1116	1160	1175				
R1	0001		0078										
R10	000A		0277	0485	0487	1086	1123	1132	1136	1137	1207		
R11	000B		0255	0256	0259	0260	0263	0266	0269	0270	0273		
			0389	0390	0391	0393	0397	0398	0398	0399	0400		
			0404	0408	0411	0463	0464	0478	0479	0481	0483		
			0488	0538	0539	0550	0574	0579	0580	0608	0609		
			0612	0618	0619	0632	0647	0683	0722	0723	0733		
			0734	0745	0758	0818	0828	0831	0836	0837	0845		
			0846	0897	0899	0900	0906	0907	0908	0917	0918		
			0981	0984	0989	0990							
R12	000C		0475										
R2	0002		0630										
R4	0004		0824	0903									
R5	0005		0617	0622	0625	0627	0633	0633	0668	0706	0878		
			0879	0892	0898	1093	1099	1105	1112	1118	1227		
R6	0006		0337	0352	0376	0617	0708	0737	1028				
R7	0007		0079										
R8	0008		0340	0371	0616	0707	0716	0716	0736	1027			
R9	0009		0255	0341	0342	0343	0377	0608	0614	0618	0709		
			0715	0722	0733	0744	0843	0844	0845	0904	0905		
			0910	0913	0925	0926	0956	1011	1029	1030	1031		
			1032	1033	1034	1126	1135	1136	1155	1206	1209		
			1213										
REM000	01F8	0606	0245	0344	0730								
REM100	01FC	0609											
REM200	0202	0612											
REM220	020C	0616	0717										
REM230	0228	0630	0620	0623	0626								

Figure 5-22. Example DSRs (Sheet 59 of 60)

DSSPTSR SDSMAC 3.6.0 83.111 13:16:24 FRIDAY, MAY 06, 1983. PAGE 0048

LABEL	VALUE	DEFN	REFERENCES				
REM260	0248	0671	0634				
REM265	0258	0707	0669				
REM270	0260	0711	0631	0651	0688		
REM280	0282	0727	0724				
REM290	0286	0729	0720				
REM300	028E	0733	0712				
REM350	029A	0738	0781	0798			
REM400	029A	0739	0610	0613			
REM500	029E	0744	0735				
REM550	02B4	0794	0763				
RETURN	01E6	0587	0374	0406	0484	0490	0540
REW010	00A2	0348	0289				
REW100	00A2	0349	0291				
REW200	00A2	0350	0300				
REW300	00A2	0351	0301				
RTEXT0	0090	0150	1094				
RTEXT1	0092	0151	1100				
RTEXT2	0094	0152	1106				
RTEXT3	0096	0153	1113				
RTEXT4	0098	0154	1119				
RTEXT5	009A	0155	1228				
SIBUFF	00C0	G0021					
SIEND	0840	G0023					
SPDMSK	1F1F	0201	0907				
STA010	013C	0462	0292				
STA020	0158	0473	0469				
STA050	015E	0475	0471				
STA060	0176	0483	0480				
STA070	0184	0487	0489				
STA100	018C	0495	0486				
STA110	0191	0499	0464	0473			
STA120	0192	0500	0475				
STA130	0196	0502	0467				
STA140	01A4	0510	0476				
STACNT	0026	0460	0479	0481			
SWFBGN	0020	G0013					
SWFEND	0090	G0015	G0017				
TF0CFE	000C	0181	0256	1011			
TF0DC3	0005	0172	0926	1126	1135	1206	
TF0DSR	0004	0171	0925	0956	1137	1209	1213
TF0EPF	0007	0174	0619	0910	0913		
TF0KFG	0006	0173	0709	0715	0734	0744	1034
TF0OIA	0003	0169	0341	0377	0723	1033	
TF0RES	000D	0182	0846	0905			
TF0RRW	0009	0177	0343	0612	0614	1032	1155
TF0WIA	0001	0167	0342	0609	1031		
TM\$FST R	04BA	1174	1176				
TSRBGN	0090	G0017	0149				
TSREND	00C0	G0019	G0021	0900			
TT	0000	0080	E0025				
VDTEX0	003A	0164	0844	1030			
VDTEX1	003C	0186	0476	0906	0908	0931	
VDTEX2	003E	0187	0911	0918	0935		
VDTEX3	0040	0188	0919	1089	1226		
WRT010	00A8	0369	0298				
WRT100	00A8	0370	0299				
WRT150	00B6	0376	0372				

Figure 5-22. Example DSRs (Sheet 60 of 60)

# How to Write a Supervisor Call Processor

---

## 6.1 INTRODUCTION

DX10 provides about 70 supervisor calls (SVCs) that perform services and provide access to data structures. If your environment requires additional SVCs, you can write your own SVC processor and include it as part of your DX10 operating system during system generation.

## 6.2 HOW TO WRITE AN SVC PROCESSOR

To add an SVC to DX10, you must design the SVC call block, build several tables, write a processor for the SVC, and include relevant information during system generation.

### 6.2.1 SVC Call Block

Because of the close relationship between the SVC call block and the SVC processor, the writer of the processor should also design the call block. Except for the first three bytes, you must determine the size and content according to the SVC functions to be performed.

Byte	Contents
0	SVC opcode. The standard SVCs use opcodes ranging from 0 through >7F. Your SVCs can use any of the opcodes in the range >80 through >FF.
1	Completion code. Zero indicates a normal completion. Nonzero codes indicate errors or warnings.
2	Sub-opcode. When you use sub-opcodes, you should place them in byte 2. When an SVC performs only one operation, you can use byte 2 of the SVC for any purpose.

To allow for adaptations or extensions to an SVC and its processor, you should include a reserved word at the end of the defined call block. You should also make the block an even number of bytes beginning on a word boundary.



### 6.2.2 SVC Structure

The SVC processing routine must DEF its own entry point and REF the system return point XPRT1A. However, SVC routines do not need to define a workspace, since a common workspace within DX10 is used for most SVCs. When the SVC routine is entered, the workspace contains:

- The entire SVC call block, or the first ten bytes (whichever is smaller) buffered in R0 – R4
- The address of the calling task's map file in R5
- The address of the calling task's task status block in R8
- A DX10 internal stack pointer in R10
- The address of the SVC call block in R11, relative to the calling task's map file (in R5)
- Return information (saved workspace pointer (WP), program counter (PC), and status register (ST)) in R13 – R15

Figure 6-1 shows the contents of the SVC routine workspace. The information in R10 and R13 – R15 must not be destroyed. SVC routines cannot issue SVCs.

When the SVC routine finishes processing the call, it should return to the system return point, as shown:

B @XPRT1A

### 6.2.3 Example of User-Defined SVC

```

        IDT 'USRSVC'
* THIS IS AN EXAMPLE OF A USER DEFINED SVC PROCESSOR.
* THIS ROUTINE GIVES THE CALLING TASK THE SPECIFIED
* NEW PRIORITY.
*
*      CALLING SEQUENCE:          XOP @PRI,15
*
*
*
*
*
*      PRI      DATA >8000
*              BYTE NEWPRI
*
*
*      WHERE NEWPRI IS THE NEW PRIORITY
*****
        DEF  USRSVC      MAKE ENTRY POINT AVAILABLE
        REF  XPRT1A     MAKE RETURN POINT AVAILABLE
        TSBPRI EQU 8    OFFSET TO PRIORITY BYTE IN TSB
*****
USRSVC CLR  R12        R12=0
        LDD  *R5        USE CALLING TASK'S MEMORY
        MOVB R12,@1(R11) CLEAR ERROR BYTE IN CALL BLOCK
        LDS  *R5        GET THE NEW PRIORITY
        MOVB @2(R11),R9 INTO R9
        MOVB R9,@TSBPRI(R8) PUT IT IN THE CALLING TASK'S TSB
        B   @XPRT1A     RETURN TO CALLING TASK
        END

```

REGISTER	
0	SUPERVISOR CALL BLOCK , BYTES 0 AND 1
1	SUPERVISOR CALL BLOCK , BYTES 2 AND 3
2	SUPERVISOR CALL BLOCK , BYTES 4 AND 5
3	SUPERVISOR CALL BLOCK , BYTES 6 AND 7
4	SUPERVISOR CALL BLOCK , BYTES 8 AND 9
5	SAVED MAP FILE ADDRESS
6	
7	
8	TASK STATUS BLOCK ADDRESS
9	
10	INTERNAL STACK POINTER (DO NOT DISTURB)
11	EFFECTIVE ADDRESS OF SUPERVISOR CALL BLOCK
12	
13	SAVED WORKSPACE POINTER
14	SAVED PROGRAM COUNTER
15	SAVED STATUS REGISTER

2283075

Figure 6-1. SVC Routine Workspace

#### **6.2.4 System Generation Requirements**

To add an SVC to the system, you must specify the following during system generation:

- **SVC parameter** — The entry label of the SVC processing routine
- **FILE parameter** — The pathname of the file containing the object code of the SVC processing routine
- **BYTE parameter** — The number of bytes in the call block for the SVC being defined

GEN990 enters the SVC entry label and call block byte length in tables in the DX10 Data Base module. It also includes the file which contains the SVC routine object module in the Link Editor control stream so that the newly defined SVCs will be incorporated in the new operating system. Refer to paragraph 3.4.6 for a discussion of defining an SVC during system generation.

GEN990 assigns user-defined SVCs an SVC code, starting at >80 and incrementing by one for each new SVC defined. You can obtain a list of the SVCs and their codes during GEN990 by issuing a LIST command.

# Privileged Supervisor Calls

---

## 7.1 GENERAL

Only privileged or system tasks can issue privileged supervisor calls (SVCs). Tasks are designated as privileged or system when they are installed. These SVCs are privileged because misuse of them can destroy the system (for instance, by irrevocably and improperly modifying the system disk) or hoard the system (for example, by killing other tasks). This section describes the SVCs, what they do, and how to use them. For the examples of SVCs, assume that this instruction has been assembled:

```
DXOP SVC, 15          DEFINE SVC AS XOP 15.
```

Each call block should start on a word boundary. Error codes returned by SVC processors are described in the *DX10 Error Reporting and Recovery Manual (Volume VI)*.

### NOTE

You should not install a task on the S\$\$SDS\$ program file. If you use the S\$\$SDS\$ program file, your task IDs and names will be deleted with each new release of DX10. It is recommended that you install tasks in your own library. This recommendation also applies to installing real-time tasks, procedures, and overlays.

## 7.2 INSTALL TASK (CODE > 25)

The function of this SVC is to install an object module, supplied on a specified file or device, in a specified program file.

0	SVC CODE		ERROR CODE	
2	PFLUNO		TASK ID	
4	TASK NAME			
12	FLAGS	R	PRIORITY	
14	PROC 1 ID		PROC 2 ID	
16	OBJLUNO		FLAGS	
18	RESERVED (=0)			

2283076

Byte	Description
0	SVC code (= > 25).
1	Error code returned by DX10.
2	LUNO assignment of the program file in which the task is to be installed. If 0, the system program file is assumed. This LUNO must be closed, except for the special form used by the Link Editor and other system utilities (see paragraph 7.2.1).
3	Optional task ID. If 0, an available ID is assigned, and returned in this byte. Appendix B contains a list of IDs reserved for system tasks on the system program file. Do not assign any of these IDs to a task that you intend to install on the system program file.
4 – 11	Task name, up to eight ASCII characters left justified with trailing blanks. The task name must be unique within each program file. If this field is binary 0, the IDT from the object module name will be used and returned in this field.

Byte	Description
12	<p data-bbox="532 346 613 373">Flags:</p> <p data-bbox="532 409 829 436">Bit 0 — Privileged task.</p> <p data-bbox="532 464 1489 531">Bit 1 — System task. Is mapped into system memory in addition to being privileged.</p> <p data-bbox="532 558 1489 657">Bit 2 — Memory-resident task. Must be installed on the system program file. Such a task remains disk resident until the next initial program load (IPL).</p> <p data-bbox="532 684 1489 751">Bit 3 — Delete protected. The task cannot be deleted until the delete-protect bit is set to zero.</p> <p data-bbox="532 779 1489 846">Bit 4 — Replicative. Copies of this task may be created and may execute concurrently, sharing procedures.</p> <p data-bbox="532 873 1489 972">Bits 5 – 6 — Associated with procedure 1 and 2 respectively. If 0, the procedure is to be loaded from the same program file as the task. If 1, the procedure is to be loaded from the system program file.</p> <p data-bbox="532 999 1489 1066">Bit 7 — Has special usage to support the Link Editor. If 1, see paragraph 7.2.1. Normally 0.</p>
13	<p data-bbox="532 1102 1489 1381">Priority. May be 0–4 or realtime 1–127. Priorities 0–3 correspond to the task’s execution priority level and are fixed, while priority 4 indicates that the priority level is to be dynamically managed by the operating system. The highest priority level is 0, while 3 is the lowest. Operating systems tasks are assigned priority 0 and 1. Critical system tasks are assigned the priority of 0. Priority 0 should be considered reserved for operating system use exclusively. Real-time priorities 1–127 logically fall between priority 0 and 1 and are indicated by setting the most significant bit of the byte.</p> <p data-bbox="532 1409 1489 1684">If priority 4 is specified, the task priority is managed by the system when the task is bid. The task’s priority is dependent upon its execution mode. If the parent task (that is, the task which issues the Bid Task SVC) is also installed with a priority of 4, then the bid task is considered to be executing in a foreground mode (not the same as System Command Interpreter (SCI) foreground). If the parent task does not have a 4 priority, the bid task is considered to be executing in the background mode and is assigned a priority of 3, in order not to degrade the response of foreground tasks.</p>

Byte	Description
	A foreground task initially executes at priority 1. Thereafter, each time the task executes a minimum number of time slices for its priority level, the priority level is lowered by one. When the task suspends for terminal input, it is reassigned priority level 1. If it suspends for any other non-TILINE I/O, it is assigned priority level 2 (unless the task already has a priority of 1). TILINE I/O does not change the priority. Whenever the task does terminal input, its priority becomes 1.
14	Procedure 1 ID. A zero value indicates no procedure.
15	Procedure 2 ID. A zero value indicates no procedure.
16	LUNO assigned to the object file from which the task object code is to be read. This LUNO must not be open.
17	Flags:  Bit 0 — Overflow (990/12 only). If arithmetic overflow occurs with this bit set to one, control will be passed to the task's end action routine.  Bit 1 — Writable control storage (990/12 only). If the task uses a writable control storage area, this bit must be set to one.  Bit 2 — Execute protected (990/12 only). If the task is to be execute protected, this bit must be set to one.  Bits 3 – 7 — Unused. Must be set to zero.
18 – 19	Reserved. Initialize to zero.

Sample code to install a privileged task on the system program file:

```

BLOCK  EVEN
        BYTE    > 25      SVC CODE FOR INSTALL TASK
        BYTE    0         RESERVED FOR RETURNED ERROR CODE
        BYTE    0         PUT THIS TASK ON THE SYSTEM FILE
        BYTE    > 83      ASSIGN ID > 83 TO THIS TASK
        TEXT    'MYTASK'  TASK NAME
        BYTE    > 88      MAKE THIS TASK PRIVILEGED AND REPLICATABLE
        BYTE    2         PRIORITY 2
        DATA   0         NO PROCEDURES
        BYTE    > 4A      OBJECT FILE IS LUNO > 4A
        BYTE    0         RESERVED, INITIALIZE TO 0
        DATA   0         RESERVED, INITIALIZE TO 0
        .
        .
        .
INSTAL  SVC      @BLOCK
    
```

### 7.2.1 Special Form of Install

Flag bit 7 is used to indicate that this is a special install operation. The bit, when set, indicates that the image for the program segment has already been written to the program file. This form of the Install SVC is used by the Link Editor in image mode to update the program file directory entry for the program segment. For this operation the object LUNO is not referenced. The Link Editor uses this SVC in conjunction with SVC > 37 (used to get space on the program file). The program file should be opened for exclusive write access privileges before executing the Install SVC. You must specify the following additional information for the special Install SVC:

Byte	Description
18 – 19	Load address.
20 – 21	Length of task root and longest overlay path.
22 – 23	Length of task root.

### 7.2.2 Install Real-Time Task

To install a task as a real-time task, the SVC must be set up the same as for INSTALL TASK except for the priority byte, byte 13. The first bit of byte 13 must be set to one and the remaining seven bits must contain a value between one and 127 inclusive. The highest real-time priority is one and the lowest is 127. Refer to the NOTE in paragraph 7.1.

### 7.3 INSTALL PROCEDURE (CODE > 26)

This SVC installs an object module from a given file or device (LUNO) as a procedure in a specified program file. Refer to the NOTE in paragraph 7.1. The install procedure SVC block is similar to the install task block and has the following format:

0	SVC CODE	RETURNED ERROR CODE
2	PFLUNO	PROCEDURE ID
4	PROCEDURE NAME	
12	FLAGS	OBJLUNO
14	RESERVED (=0)	
16	RESERVED (=0)	
18	RESERVED (=0)	

2283077



Byte	Description
0	SVC code (> 26).
1	Error code returned by DX10.
2	LUNO of program file on which procedure is to be installed. If 0, the system program file is assumed. This LUNO must not be open except for the special install function (see paragraph 7.2.1).
3	Procedure ID. If 0, the system will assign an ID and return it in this byte.
4 – 11	Procedure name, up to eight ASCII characters with trailing blanks. The procedure name must be unique in each program file. If this field is binary 0, the IDT name from the object module is used.
12	<p>Flags, which, when set, mean the following:</p> <p>Bits 0 – 1 — Not used, must be set to 0.</p> <p>Bit 2 — Memory resident. The procedure must be installed on the system program file, and will not become memory resident until the system is rebooted.</p> <p>Bit 3 — Delete protected. The procedure cannot be deleted until this bit is reset.</p> <p>Bit 4 — Writable control storage (990/12 only). If the procedure uses a writable control storage, this bit must be set to one.</p> <p>Bit 5 — Execute protected (990/12 only). If the procedure is to be execute protected, this bit must be set to one.</p> <p>Bit 6 — Write protected (990/12 only). If the procedure is to be write protected, this bit must be set to one.</p> <p>Bit 7 — Normally 0. Has special system usage if 1 is to support the Link Editor (see paragraph 7.2.1).</p>
13	LUNO assigned to the file from which the object module is to be read. This LUNO must not be open.
14 – 15	Reserved. Initialize to zero.

The following is sample code to install a procedure:

```

BLOCK  EVEN
        BYTE  > 26      INSTALL PROCEDURE SVC CODE
        BYTE  0         ERROR CODE
        BYTE  > AA     PUT THE PROCEDURE ON THE FILE DESIGNATED
                        BY LUNO > AA.
*      BYTE  0         LET THE SYSTEM ASSIGN AN ID
        TEXT  'MYPROC'  PROCEDURE NAME
        BYTE  0         NO SPECIAL FLAGS
        BYTE  > BB     THE OBJECT MODULE IS ON FILE
                        DESIGNATED BY LUNO > BB
        DATA 0        RESERVED
        .
        .
        .
INSTAL  SVC      @BLOCK  INSTALL THE PROCEDURE

```

A special form of the Install Procedure SVC is described in the following paragraph.

If flag bit 7 is set, the following bytes of information must be added (see paragraph 7.2.1 for a complete description):

Byte	Description
14 – 15	Load address.
16 – 17	Length of procedure in bytes.
18 – 19	Reserved. Initialize to 0.

#### 7.4 INSTALL OVERLAY (CODE > 27)

This SVC installs an object module from a given file or device as an overlay in the specified program file. Refer to the NOTE in paragraph 7.1. The call block is similar to the install task call block, and has the following format:

0	SVC CODE	RETURNED ERROR CODE
2	PFLUNO	OVERLAY ID
4	OVERLAY NAME	
12	FLAGS	ASSOCIATED TASK ID
14	OBJLUNO	RESERVED (=0)
16	RESERVED (=0)	

2283078

Byte	Description
0	SVC code (> 27).
1	Error code returned by DX10.
2	LUNO of program file on which overlay is to be installed. If 0, the system program file is used. This LUNO must not be open except for the special install function.
3	Overlay ID. If 0, the system will assign an ID and return it in this byte.
4 – 11	Overlay name. The overlay name must be unique in the program file and can have up to eight ASCII characters with trailing blanks. If this field is binary 0, the IDT name from the object file is used and returned in this field.
12	<p>Flags:</p> <p>Bit 0 — Relocatable.</p> <p>Bits 1 – 2 — Not used, must be 0.</p> <p>Bit 3 — Delete protected. The overlay cannot be deleted until this bit is reset.</p> <p>Bits 4 – 6 — Not used, must be 0.</p> <p>Bit 7 — Normally 0. Same as for install task and procedure.</p>
13	Associated task ID. If specified, the overlay must have been installed on the same program file as the task, and is automatically deleted when the task is deleted.
14	LUNO assigned to the file from which the object module is to be read. This LUNO must not be open.
15 – 17	Reserved. Initialize to zero.

The following code is an example of the call block and call:

```

LAYLA  EVEN
        BYTE   > 27      INSTALL OVERLAY CODE
        BYTE   0         RESERVED FOR ERROR CODE
        BYTE   0         INSTALL THIS ON THE SYSTEM FILE
        BYTE   0         LET SYSTEM ASSIGN ID
        TEXT   'OLAY1'   OVERLAY NAME
        BYTE   > 80      RELOCATABLE
        BYTE   > 83      ASSOCIATE WITH TASK > 83
        BYTE   > 0A      GET OVERLAY FROM LUNO > 0A
        BYTE   0         RESERVED
        DATA  0         RESERVED
        .
        .
        .
CALL    SVC    @LAYLA   INSTALL THE OVERLAY
    
```

If flag bit 7 is set, the information added is the same as for install task or procedure (see paragraph 7.2.1), that is:

Byte	Description
16 – 17	Load address.
18 – 19	Length of overlay in bytes.
20 – 21	Reserved. Initialize to zero.

### 7.5 DELETE TASK (CODE > 28)

The Delete Task SVC deletes the task with the specified ID from the given program file. All associated overlays are automatically deleted. The call block is six bytes in length and has the following format:

BYTE		
0	SVC CODE	RETURNED ERROR CODE
2	PFLUNO	ID
4	FLAGS	RESERVED (=0)

2283079

Byte	Description
0	SVC code (> 28).
1	Error code returned by DX10.
2	LUNO assigned to the program file from which the task is to be deleted. Deletion of delete protected tasks is not allowed. This LUNO must not be open, unless flag bit 7 is set.
3	ID of task to be deleted.
4	Flags, which when set mean:  Bits 0 – 6 — Not used, must be zero.  Bit 7 — Program file is open.
5	Reserved. Initialize to zero.

The following sample code deletes > 83, which was installed on the program file assigned to LUNO > 4A.

```

DELETE  EVEN
        BYTE    > 28      SVC CODE
        BYTE    0         RESERVED FOR ERROR CODE
        BYTE    > 4A     THE TASK IS ON LUNO > 4A
        BYTE    > 83     THE TASK ID IS > 83
        DATA   0         RESERVED
        .
        .
        .
DELTASK SVC    @DELETE

```

## 7.6 DELETE PROCEDURE (CODE > 29)

This SVC deletes the specified procedure from the given program file. The format and contents of the call block are the same as for delete task (see paragraph 7.5), except that the ID in byte 3 is the procedure ID.

### 7.7 DELETE OVERLAY (CODE > 2A)

This SVC deletes the specified overlay from the given program file. The format and contents of the call block are the same as for delete task (see paragraph 7.5), except that the ID in byte 3 is the overlay ID.

### 7.8 KILL TASK (CODE > 33)

The Kill Task SVC is issued by one task in order to terminate another running task. The call block is eight bytes and has the following format:

BYTE		
0	SVC CODE	ERROR CODE
2	RUN ID	STATION ID
4	STATE	RESERVED (=0)
6	RESERVED (=0)	

2283080

Byte	Description
0	SVC code (> 33).
1	Error code returned by DX10.
2	Run-time ID of the task to be terminated. Note that the run-time ID of a task may be different from the installed ID, if the task is replicatable, because the system chooses an ID for task replicas.
3	Optional station ID. If this byte is: 0 — Any running task with the specified ID will be terminated. > 1 — > FE — The task must be associated with this station number in order to be terminated. > FF — The task to be terminated must not be associated with a station.
4	The state of the task at termination is returned in this byte.
5 – 7	Reserved. Initialize to zero.

The following is an example of code for killing a task:

```

KILL      EVEN
          BYTE   > 33      SVC CODE FOR KILL TASK
          BYTE   0         ERROR CODE
          BYTE   > 83      KILL TASK > 83
          BYTE   0         KILL ANY TASK WITH RUN ID > 83
          BYTE   0         RETURN TASK > 83 STATE HERE
          BYTE   0         RESERVED
          DATA  0         RESERVED
          .
          .
          .
KILTSK    SVC      @KILL

```

### 7.9 SUSPEND AWAITING QUEUE INPUT (CODE > 24)

This is a special SVC used by queue serving tasks. When a queue server has processed all the entries in its queue, it issues a code > 24 SVC to suspend itself until another entry is put on the queue.

For a queue serving task this SVC serves the same purpose as an End Program SVC for a normal task, except that the suspended queue server is not processed by the termination task; therefore, any task local LUNOs must be released before this SVC is issued. The queue server is not rolled when it suspends for queue input, but will be released from memory only when it is chosen for rollout by task management. The call block is one byte, which contains the SVC code:

```

          BYTE
          0   24

```

2283081

\* THIS IS SAMPLE CODE FOR A SUSPEND  
\* AWAITING QUEUE INPUT SVC

```

SAQI      BYTE   24      THIS IS THE CALL BLOCK
          .
          .
          .
OUT       SVC     @SAQI   THIS IS THE CALL

```

### 7.10 READ/WRITE TSB (CODE > 2C)

This SVC reads/writes one word of data from/to the specified task's TSB and should be used with extreme caution. Some fields of a TSB cannot be modified without serious impact on the operating system. No protection is provided to prevent the use of this SVC from causing a system crash. Note that the use of this SVC assumes a TSB format which may be changed as new system requirements are implemented.



The logical sequence for using the Read/Write TSB SVC is as follows:

1. Extend the time slice
2. Read TSB
3. Write TSB

The call block format is as follows:

BYTE	
0	SVC CODE                      ERROR CODE
2	FLAGS
4	RUN ID                              DISPLACEMENT
6	VALUE
8	RESERVED (=0)

2283082

Byte	Description
0	SVC code (> 2C).
1	Error code returned by DX10.
2-3	Flags:  Bits 0 - 14 — Not used. Initialize to 0.  Bit 15 — 1 for write. 0 for read.
4	Run-time ID of the task whose TSB is to be read/written.
5	Byte offset into the TSB of the word to be read/written (should be even).
6-7	Value of the word read/written.
8-9	Reserved. Initialize to 0.

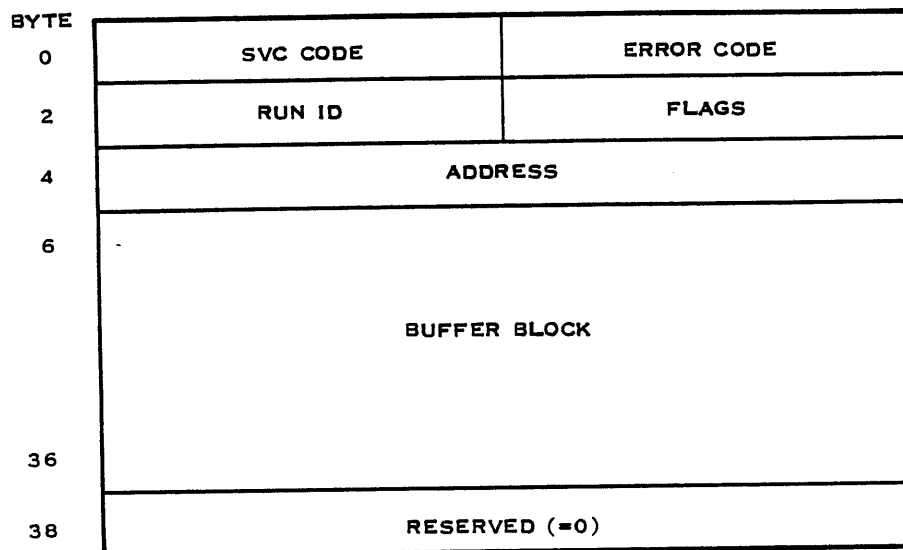
The following code is an example of a code > 2C SVC:

\* THIS ROUTINE WILL ALLOW TASK > 83 TO TAKE END ACTION AGAIN.

RWTTSB	EVEN	BYTE	> 2C	THE SVC CODE
		BYTE	0	RESERVED FOR ERROR CODE
SVCFLG	DATA	0	0	FLAGS 0 = READ, 1 = WRITE
		BYTE	83	TSB OF TASK > 83
		BYTE	10	THE DISPLACEMENT OF THE TASK FLAGS IN TSB
VALUE	DATA	0	0	RESERVED FOR VALUE RETURNED
		DATA	0	RESERVED
MEMRES	DATA	> 1000		END ACTION FLAG MASK
WRTFLG	DATA	1		WRITE TSB FLAG
EXTEND	BYTE	9,4		EXTEND TIME SLICE 4 SYSTEM TIME UNITS
	.			
	.			
	.			
	SVC	@EXTEND		EXTEND TIME SLICE
	SVC	@RWTTSB		READ THE CURRENT TSB WORD
	SOC	@MEMRES,	@VALUE	RESET END ACTION
	MOV	@WRTFLG,	@SVCFLG	SET WRITE FLAG IN BLOCK
	SVC	@RWTTSB		WRITE THE TSB WORD

### 7.11 READ/WRITE TASK (CODE > 2D)

This SVC is used to read/write up to 16 words of data from/to the specified address in the designated task. The task must be either in memory or in an unconditionally suspended state (this can be forced by halting the task). The call block is 40 bytes long and has the following format:



Byte	Description
0	SVC code (> 2D).
1	Error code returned by DX10.
2	Run-time ID of task to be read/written. A zero value indicates that the DX10 system root is to be read/written.
3	Flags: Bit 8 — 0 for read, 1 for write Bit 9 — 0 = Task ID, 1 = System Overlay Segment ID. Bits 10-11 — Reserved. Initialize to zero. Bits 12-15 — Number of words (a zero value specifies 16 words).
4 – 5	Relative address in the specified task of the first word of the data block.
6 – 37	16-word data block.
38 – 39	Reserved. Initialize to zero.

The following code is an example of a Read Task SVC:

```

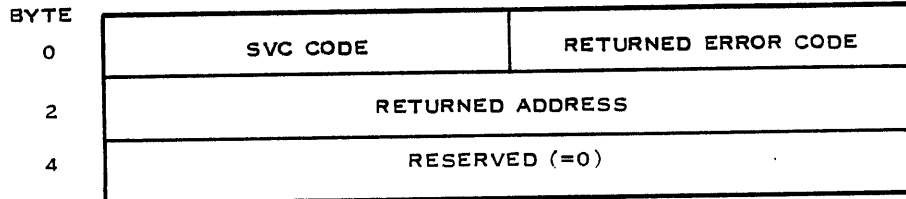
* THIS ROUTINE READS THE WORKSPACE OF TASK > 83,
* ASSUMING THAT THE WORKSPACE ADDRESS IS 6,
* AND THAT THE TASK IS MEMORY RESIDENT.

RDTSK    EVEN
          BYTE    > 2D          SVC CODE
          BYTE    0            RETURNED ERROR CODE
          BYTE    > 83        READ/WRITE FROM TASK > 83
          BYTE    0            READ 16 WORDS
          DATA   6            ADDRESS OF WORKSPACE

DATA     BSS     32            16-WORD BUFFER
          DATA   0            RESERVED
          .
          .
          .
          SVC     @RDTSK      READ THE WORKSPACE
    
```

### 7.12 GET SYSTEM POINTER TABLE ADDRESS (CODE > 32)

This SVC is not privileged, but it is only useful to privileged or system tasks. The SVC returns the memory address of a table of pointers to various system routines and data structures. The table, also called the overlay table, is used by disk-resident system tasks and overlays to provide run-time linkage with the operating system. The call block is six bytes long and has the following format:



2283084

Byte	Description
0	SVC code (> 32).
1	Error code returned by DX10.
2 – 3	Address return field.
4 – 5	Reserved. Initialize to zero.

The following code is an example of the Get System Pointer Table Address SVC:

```

GSPTA  EVEN
        BYTE  > 32      SVC CODE
        BYTE  0         ERROR CODE
        DATA 0         ADDRESS RETURN FIELD
        DATA 0         RESERVED
        .
        .
        .
        SVC   @GSPTA   GET SYSTEM POINTER TABLE ADDRESS
    
```

### 7.13 INITIALIZE DATE AND TIME (CODE > 3B)

This SVC stores the date and time information given in the specified block. The 6-byte call block has the following format:

BYTE			
0	<table border="1"><tr><td>SVC CODE</td><td>ERROR CODE</td></tr></table>	SVC CODE	ERROR CODE
SVC CODE	ERROR CODE		
2	DATE/TIME BLOCK ADDRESS		
4	RESERVED (=0)		

2283085

The SVC code, error code, and reserved bytes have the same use as in preceding SVC descriptions. Bytes 2 and 3 contain the address of a block of data that contains the date and time. The data block is a 6-word block of the form:

BYTE	
0	YEAR
2	DAY IN YEAR
4	HOUR (1-24)
6	MINUTE
8	SECOND
10	RESERVED (=0)

2283086

Each data word should be a binary value. The following is an example of an Initialize Data and Time SVC:

	EVEN		
IDATIM	BYTE	> 3B	SVC CODE
	BYTE	0	ERROR CODE
	DATA	BLOCK	ADDRESS OF DATA BLOCK
	DATA	0	RESERVED
BLOCK	DATA	1977	YEAR (1977)
	DATA	84	DAY (MARCH 25)
	DATA	8	HOUR (8 A.M.)
	DATA	3	MINUTE (8:03)
	DATA	15	SECOND (8:03:15)
	DATA	0	RESERVED
	.		
	.		
	.		
	SVC	@IDATIM	

#### 7.14 DISK MANAGER (CODE > 22)

This SVC calls the disk manager. The disk manager allocates and deallocates disk space in allocatable disk units (ADUs). DX10 uses ADUs strictly for allocating and deallocating disk space. An ADU is an integer number of sectors on the disk, and is defined to be the smallest amount of disk storage which can be allocated by the disk manager. An ADU is defined such that the maximum number of ADUs per disk is less than 65,536 (that is, any ADU may be addressed in a 16-bit word). Since different disk types vary in the number of sectors (and the amount of information) they contain, ADU size varies between disks. Table 7-1 shows the ADU sizes on disks supported by DX10.

Table 7-1. Disk Descriptions

Disk Type	Available Space (M Bytes)	No. ADUs	No. Heads	No. Cylinder	Sec./Track	Sec./ADU	Bytes/Sec.
DS10	4.7	16,320	2	408	20	1	288
DS25	22.3	25,840	5	408	38	3	288
DS50	44.6	51,616	5	815	38	3	288
DS80	62.7	40,819	5	803	61	6	256
DS200	169.5	65,381	19	815	38	9	288
DS300	238.3	62,045	19	803	61	15	256
FD1000	1.15	4,004	2	77	26	1	288
CD1400/32							
Removable	13.5	52,544	1	821	64	1	256
Fixed	13.5	52,544	1	821	64	1	256
CD1400/96							
Removable	13.5	52,544	1	821	64	1	256
Fixed	67.3	43,786	5	821	64	6	256
WD500	4.9	19,200	4	150	32	1	256
WD500A	17.0	22,208	3	694	32	3	256
WD800-18	18.5	24,087	3	651	37	3	256
WD800-43	43.2	56,203	7	651	37	3	256
WD800A/38	38.4	50,105	5	911	33	3	256
WD800A/68	69.2	45,094	9	911	33	6	256
WD800A/114	114.5	49,720	15	904	33	9	256
WD900-138	138.1	59,928	10	805	67	9	256
WD900-138/2	69.0	44,946	5	805	67	6	256
WD900-425	425.8	61,600	24	693	100	27	256
WD900-425/2	212.9	55,440	12	693	100	15	256

When the disk manager is called via SVC code >22, the calling task is suspended and made eligible for rollout. The call block has the following format:

0	SVC CODE	ERROR CODE
2	OPCODE	RESERVED
4	DISK PDT ADDRESS	
6	BLOCK SIZE (IN BYTES)	
8	MAXIMUM NUMBER OF BLOCKS	
10		
12	ADU NUMBER	

2283087

Byte	Description
0	SVC code (> 22).
1	Error code returned by DX10.
2	Opcode: 0 — Deallocate 1 — Allocate the whole area or fail 2 — Allocate as much as possible 3 — Allocate as much as possible where requested or fail
3	Reserved. Initialize to zero.
4 – 5	Physical device table (PDT) address of the disk unit on which space is to be allocated/deallocated.
6 – 7	Size of blocks to be allocated, in bytes. Ignored for deallocation.
8 – 11	Maximum number of blocks to be allocated. Bytes 8–9 ignored for deallocation, 10–11 is number of ADUs to deallocate.
12 – 13	If deallocating: starting ADU of area to be released. If allocating: try to allocate as close to this ADU as possible; if the value is 0 and the opcode is 1 or 2, the lowest area is allocated.

Note that specifying the area to be allocated in blocks, and a block size given in bytes, makes it unnecessary for the caller to know the disk format. Allocations are made in terms of ADUs since file allocation tables within DX10 are kept in terms of ADUs.



*Privileged Supervisor Calls*

In addition to the error code, the disk manager returns allocation information if the opcode was a 1, 2, or 3. The returned information is placed in the call block as shown:

0	*		ERROR CODE
2	*		
4	*		
6	*		
8		ADUS/BLOCK	BLOCKS/ADU
10	NUMBER OF ADUS ALLOCATED		
12	STARTING ADU NUMBER		

\*NOT CHANGED FROM INPUT VALUES

2283088

Byte	Description
0	Not changed
1	Error code returned by DX10.
2 – 7	Not changed
8	Number of ADUs per block (block size specified in call). Set to 1 if less than 1.
9	Number of blocks per ADU. Set to 0 if less than 1.
10 – 11	Number of ADUs allocated.
12 – 13	ADU number of the first ADU in the allocated area.

The disk manager allocation algorithm is designed to reduce head movement and fragmentation of files and disk space. The priorities used when allocating disk space are:

1. Start at requested ADU (coalesce disk space)\*
2. Size requested\*
3. Proximity to requested ADU (reduce head movement)
4. Lower ADU first (reduce fragmentation of disk space)

The following is an example of a disk manager call to allocate.

```

EVEN
DSCMGR BYTE > 22      SVC CODE
ERRCOD  BYTE 0        ERROR CODE
        BYTE 2        ALLOCATE AS MUCH AS POSSIBLE
        BYTE 0        RESERVED
        DATA DPDT    DISK PDT ADDRESS
        DATA 128     128-BYTE BLOCKS
APBPA   DATA 0       NEED 64
SIZE    DATA 64     BLOCKS
START   DATA 0      ALLOCATE AS LOW AS POSSIBLE
        .
        .
        .
        SVC    @DSCMGR
    
```

### 7.15 ASSIGN SPACE ON PROGRAM FILE (CODE > 37)

This SVC is used by the Link Editor and other system utilities. The call assigns a starting record on the specified program file to be loaded with a task, procedure, or overlay image. The call block has the following form:

0	SVC CODE	ERROR CODE
2	PFLUNO	RESERVED (=0)
4	LENGTH	
14	RECORD NUMBER	
16	RESERVED (=0)	

2283089

\* Priority for opcodes 2, 3; reversed for opcode 1.

*Privileged Supervisor Calls*

Byte	Description
0	SVC code (> 37).
1	Error code returned by DX10.
2	LUNO assigned to the program file. The LUNO should have been opened with exclusive write access by the calling task.
3	Reserved. Initialize to zero.
4 – 5	The length in bytes of the image to be written.
6 – 7	Returned record number which begins the area allocated on the program file.
8 – 9	Reserved. Initialize to zero.

The following code is an example of the Assign Space on Program File SVC:

```
ASOPF    EVEN
ERRCOD   BYTE    > 37      SVC CODE
          BYTE    0        ERROR CODE
          BYTE    PFLUN    LUNO OF THE PROGRAM FILE
          BYTE    0        RESERVED
RECORD   DATA   100      NEED 100 BYTES
          DATA   0        RESERVED FOR RETURNED RECORD #
          DATA   0        RESERVED
          .
          .
          .
          SVC    @ASOPF
```

**7.16 INITIALIZE NEW VOLUME (CODE > 38)**

This SVC is used to initialize a new disk volume. It also installs the volume (see paragraph 7.17). The Initialize New Volume SVC performs the following functions:

- Formats all tracks on the disk one sector per record
- Builds track 0, sector 0 (special volume overhead)
- Builds the bad ADU list on track 0, sector 1
- Builds the ADU availability partial bit maps which fill the remainder of track 0
- Builds the volume directory on the disk
- Installs the new volume by issuing an Install Volume SVC (see paragraph 7.17)

The call block has the following format:

0	SVC CODE	ERROR CODE
2	DISK DRIVE NAME (4 ASCII CHARACTERS)	
4		
6	VOLUME NAME (8 ASCII CHARACTERS)	
8		
10		
12		
14	NUMBER OF DIRECTORY ENTRIES	
16	BAD TRACK LUNO	FLAGS
18	DEFAULT PHYSICAL RECORD SIZE	
20	HARDWARE INTERLEAVING FACTOR	
22	TRACK #1 LOADER LUNO	RESERVED (=0)

2283090

Byte	Description
0	SVC Code (> 38).
1	Returned error code.
2 – 5	Device name of disk unit (for example, DS02, DS03).
6 – 13	Name of volume to be initialized.
14 – 15	Number of entries to be included in the volume directory. (If equal to zero, > 360 is assumed.)
16	LUNO assigned to a file which contains a list of the bad tracks for the new disk. The form of the list is as follows:  <div style="text-align: center;"> <p>HEAD CYLINDER; OR HEAD, CYLINDER; HEAD, CYLINDER; etc.</p> </div> <p>The last entry in the list need not be followed by a semicolon. The list is terminated by an empty line. A zero in this field implies no list is present.</p>
17	Flags:  <div style="margin-left: 20px;"> <p>&gt; 00 Use the loader on the system disk.</p> <p>&gt; 80 No loader will be placed on track 1 (that is, VCATALOG starts on track 1, sector 0).</p> <p>&gt; C0 Byte 22 identifies track 1 loader.</p> </div>
18	Default physical record size used in VCATALOG for subsequent file creation. If the value is zero, the default physical record size is taken from the system generation parameter for that disk drive.
20	Hardware interleaving factor used to format the sectors of the disk. The value must be N; $0 \leq N \leq (\text{number of sectors per track} - 2)$ . If $N = 0$ , a value of 1 is assumed (this applies only to double-sided, double-density (DSDD) diskettes).
22	LUNO assigned to an image file that is to be placed on track 1. The first word of the file is assumed to be an entry point and is placed in track 0, sector 0, byte > 18. Byte 17 must contain > C0 for this option.

The following is an example of an Initialize New Volume SVC:

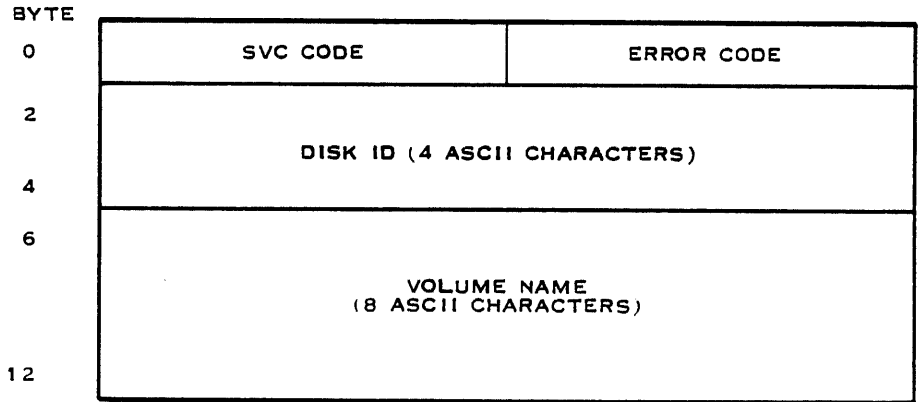
INIUT	EVEN		SVC CODE
	BYTE	> 38	RETURN ERROR CODE
	BYTE		DEVICE NAME
	TEXT	DS04	VOLUME NAME
	TEXT	NEWVOLbb	20 ENTRIES IN
	DATA	20	VOLUME DIRECTORY
*			BAD TRACK LIST LUNO
*	BYTE	BTLLUN	(See Volume III)
	BYTE	> C0	FLAGS (BYTE 22 CONTAINS
*			TRACK 1 LUNO)
	DATA	0	DEFAULT PHYSICAL
*			RECORD SIZE (768 is implied)
	DATA	0	HARDWARE INTERLEAVING
*			(1 is implied)
	BYTE	LDRLUN	LUNO FOR TRACK 1
*			LOADER (See Volume III)
	BYTE	0	RESERVED (must be 0)

### 7.17 INSTALL DISK VOLUME (CODE > 20)

The Install Volume SVC must be used in order to make a disk volume available for file storage. The Install Volume SVC performs the following functions:

- Verifies that the given volume name matches the disk drive specified
- Verifies that the specified drive is available
- Builds the file control block for the volume directory
- Initializes memory to be used by the disk manager
- Deletes any temporary files on the disk

The call block is the same as the first 14 bytes of the Initialize Volume SVC, and has the following format:



2283091

Byte	Description
0	SVC code (> 20).
1	Returned error code.
2-5	Disk device name (for example, DS01, DS03).
6-13	Name of volume to be installed. If another volume is installed on the disk drive, the installed volume name is returned in this field, and byte 1 will contain an error code.

The following code is an example of an Install Volume SVC:

```

EVEN
INSVOL  BYTE > 20      SVC CODE
        BYTE 0         RETURNED ERROR CODE
        TEXT 'DS02'    DISK UNIT 02
        TEXT 'NEWVOL ' VOLUME TO BE INSTALLED
        .
        .
        .
        SVC @INSVOL   INSTALL THE VOLUME
    
```

### 7.18 UNLOAD DISK VOLUME (CODE > 34)

This SVC unloads a volume that is no longer being used, usually with the intent of installing a new volume on the disk drive. The Unload Volume SVC performs the following functions:

- Determines the device on which the specified volume is installed
- Verifies that no LUNOs are assigned to files in this volume
- Releases all memory allocated to support this volume
- Updates the disk PDT to show that the volume is unloaded
- Returns the physical device name

The 14-byte call block has the same format as the Install Volume SVC, except that the disk ID in bytes 2 through 5 is returned by the system. The following is an example of an Unload Volume SVC:

UNLVOL	EVEN		
	BYTE	> 34	SVC CODE
	BYTE	0	RETURNED ERROR CODE
DISKID	BSS	4	RETURNED DISK ID
	TEXT	'OLDVOL'	VOLUME TO BE UNLOADED

### 7.19 DIRECT DISK I/O

In order to maintain disk file integrity and security, and to control disk allocation, direct disk (or diskette) I/O is reserved for use by privileged or system tasks, which have direct and global access to disk units.

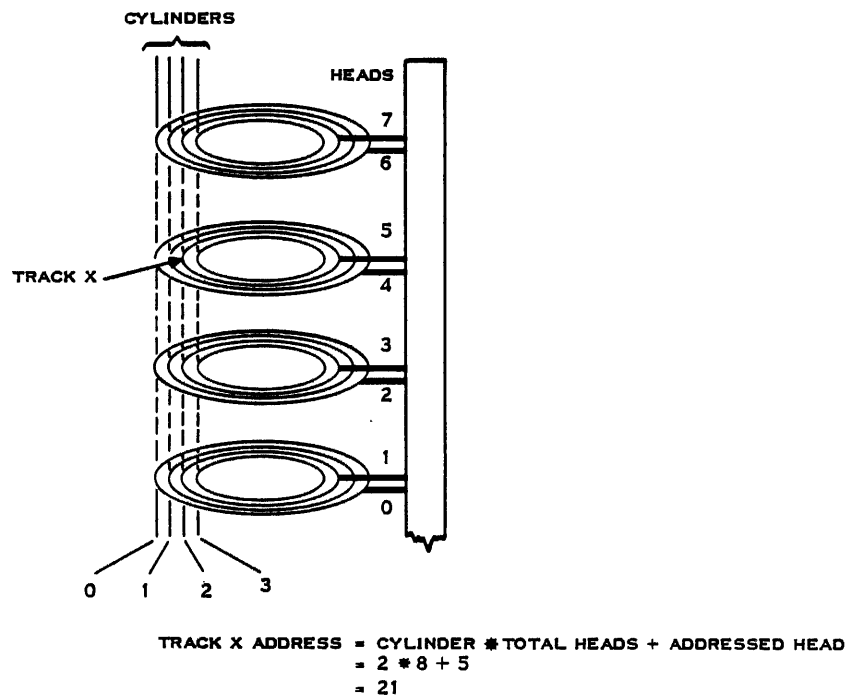
I/O to disk or diskette is controlled by an extended call block as described in paragraph 7.19.1. As with any other device, access is through a LUNO assigned to the disk. DX10 has a reserved system (nonreleasable) LUNO for each disk unit configured in the system. Action taken on the standard I/O opcodes for disk is described in paragraph 7.19.2, while diskette is described in paragraph 7.19.3. Disks and diskettes are classified as record-oriented devices by DX10 and need not be opened before I/O operations or closed afterward.



Before tracks on a disk cartridge or a diskette can be used for data storage, they must be formatted. That is, the physical record length of the records on a track must be defined on the disk pack. This is done when a system disk is built or when a secondary disk is initialized (see Section 2). The actual physical organization of a disk pack is by heads (recording surfaces), cylinders (recording bands on all surfaces), and sectors (timing marks dividing each cylinder). A track is one surface of a cylinder and is the smallest unit which can be independently formatted. Tracks are addressed by the expression:  $c * H + h = T$ .

T is the addressed track number  
H is the number of heads for the disk unit  
c is the addressed cylinder number  
h is the addressed head

Figure 7-1 shows the manner by which tracks are addressed.



2279565

Figure 7-1. Addressing a Track

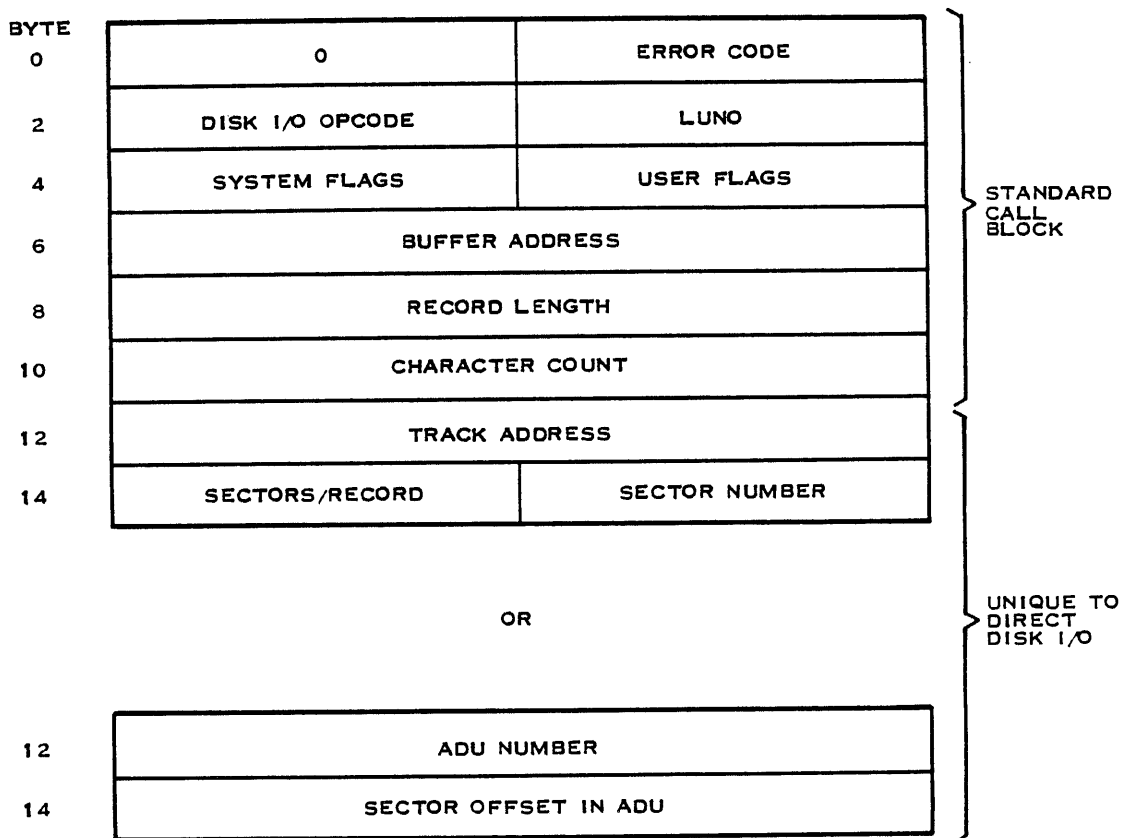
Track addressing runs sequentially from zero to the maximum track address. Each track is formatted for a given physical record length (a physical record is the smallest unit of data transfer from the disk) with resulting records per track and sectors per record. Physical records must start at sector boundaries but need not fill entire sectors and may cross sector (but not track) boundaries. Each record defined on a track requires a fixed amount of overhead, thus the larger the records, the fewer records per track, the less overhead required, resulting in more usable bytes per formatted track. Note that all disks initialized under DX10, Release 3.2 or later, (using the Initialize Volume command or SVC) are formatted one sector per physical record and that the FD800 diskette supported by DX10 is also formatted one sector per record.

**CAUTION**

**Reformatting a track after the disk has been initialized for DX10 use will destroy any data that is on the specified track. Also, if the format is not one SECTOR/RECORD, DX10 will not be able to access that track except through user direct disk I/O.**

**7.19.1 Direct Disk I/O Call Block**

The SVC block used for the Direct Disk I/O SVC is the same as a normal I/O call block, with a two-word extension used to contain a disk address. The call block has the following format:



2283105

**NOTE**

DX10 supports addressing by ADU/Sector offset or Track/Sector offset on Release 3 or later disk units. Direct disk I/O to CRU diskette must use the track/sector method of addressing.

Byte	Description
0	SVC code (0).
1	Error code returned by DX10.
2	I/O opcode (see paragraphs 7.19.2 and 7.19.3).
3	LUNO assigned to disk.
4	Flags set by the system — when set high the following conditions are indicated:  Bit 0 — LUNO is busy.  Bit 1 — Error.  Bit 2 — End-of-file condition (not applicable).  Bits 3 – 7 — Reserved.
5	Flags set by user — when set high the following conditions are indicated:  Bit 0 — Initiate I/O.  Bit 1 — User supplied format information pointed to by TRBDBA.  Bit 2 — Reserved (set to 0).  Bit 3 — Offset enable.  Bit 4 — Offset forward.  Bit 5 — Transfer inhibit.  Bit 6 — Reserved (set to 0).  Bit 7 — Retry flag; 1 = no retries are to be done by the disk DSR.

Byte	Description
6 – 7	The address of the buffer to which data is to be read or from which data is to be written. Must begin on a word boundary.
8 – 9	The length in bytes of a physical record. This number determines the maximum number of bytes transferred during a read. Must be an even count.
10 – 11	The number of characters to be written. Must be an even count. DX10 returns the number of characters read by a read operation in these bytes.

The last two words of the call block vary according to the addressing mode of the I/O operation. If the address of the logical record is given as track and sector, then:

Byte	Description
12 – 13	The track number in which the record is located (see Figure 7-1 for track addressing).
14	The number of sectors per physical record.
15	The starting sector number of the record.

If the address is given as ADU/sector offset, then:

Byte	Description
12 – 13	The ADU number in which the physical record starts.
14 – 15	The starting sector number of the physical record within the ADU. This must be specified because ADUs may span physical records (that is, a record may start in the middle of an ADU).

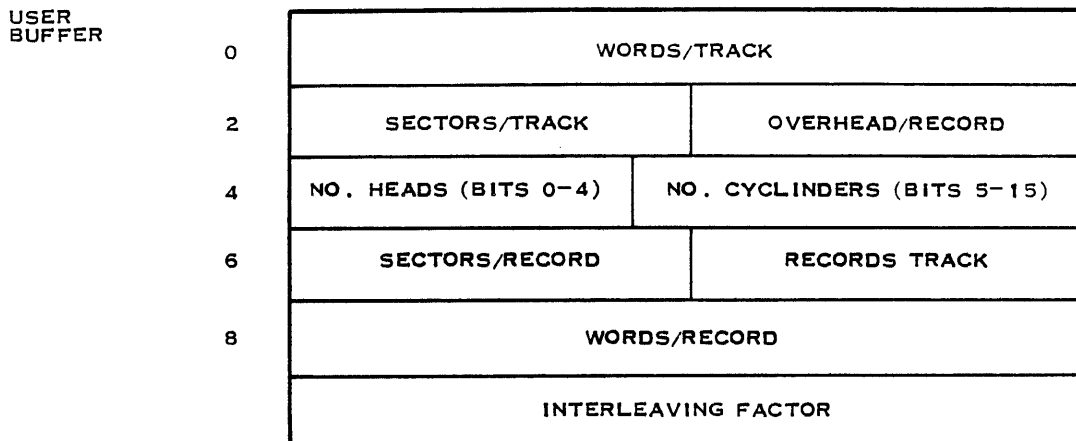
### 7.19.2 Direct Disk I/O Opcodes

I/O opcodes for direct I/O to disks do not have the same meanings as for other device I/O. (Diskette opcodes are described in paragraph 7.19.3.) Those I/O opcodes that are defined for the disk are defined as shown in Table 7-2 (all other opcodes are ignored by the disk).

**Table 7-2. Direct Disk I/O Opcodes**

Hexadecimal Opcode	I/O Function	I/O Action
5,F	Read Format	Returns disk and track format.
8	Write Format	Formats a track.
9	Read by ADU	Inputs data starting in the specified ADU and sector.
A	Read by Track	Inputs data starting in the specified track and sector.
B	Write by ADU	Outputs data starting in the specified ADU and sector.
C	Write by Track	Outputs data starting in the specified track and sector.
E	Store Registers	Returns the disk parameter registers, which are the same as the first three words of the read format data.
12	Write Interleaved Factor	Writes format for interleaving to a double-sided, double-density (DSDD) track.

**7.19.2.1 Read Format (Opcodes > 5 and > F).** The Read Format command requires specification of a buffer address in bytes 6 and 7 of the call block and a track address in bytes 12 and 13. The specified buffer must begin on a word boundary. The record length specifies the number of bytes to be returned (must be even) up to 12 bytes. The character count, sectors per record, and sector number fields of the call block are ignored. Twelve bytes of disk and track data are returned in the user buffer as shown in Figure 7-2.



2277819

**Figure 7-2. Store Registers and Read Format Data**

**7.19.2.2 Write Format (Opcode > 8).** The Write Format command requires specification of a physical record length in bytes 8 and 9 of the call block, and a track address in bytes 12 and 13. The SCB buffer address, character count, and sector number fields are ignored. The addressed track is formatted to the given record length and the resulting number of sectors per record is returned in byte 14 of the call block.

**7.19.2.3 Store Registers (Opcode > E).** The Store Registers command transfers the disk parameters registers into the specified user buffer. The buffer address is specified in bytes 6 and 7 of the call block and must begin on a word boundary. The record length specifies the number of bytes to be returned (must begin on even boundary) up to 6 bytes. Bytes 10 through 15 of the SCB are ignored. Six bytes of disk format data are returned in the buffer as shown in Figure 7-2.

Bytes 0 through 11 are returned for read format opcodes (> 5 and > F), and bytes 0 through 5 are returned for store register opcode (> E).

**7.19.2.4 Read by ADU (Opcode > 9).** The Read by ADU command transfers data from the disk to the specified buffer. The data buffer address is specified in bytes 6 and 7 of the SVC call block and must begin on a word boundary. The length of the physical record to be read is specified in bytes 8 and 9 and must be an even number. The ADU number and sector offset from which to read the record are specified in bytes 12 through 15. The number of characters actually read is placed in bytes 10 and 11 by the system.

**7.19.2.5 Read by Track (Opcode > A).** The Read by Track command transfers data from the disk to the specified buffer. The buffer address is specified in bytes 6 and 7 of the call block and must begin on a word boundary. The record length is specified in bytes 8 and 9, and must be an even number. The track address is specified in bytes 12 and 13 and the sectors per record and starting sector number are specified in bytes 14 and 15. The number of characters read is returned in bytes 10 and 11.

Bits 3, 4, 5, and 7 of byte 5 control alternative disk operations. When set: bit 3 enables head offset; bit 4 enables head offset forward; bit 5 enables transfer inhibit; and bit 7 disables retries.

**7.19.2.6 Write by ADU (Opcode > B).** The Write by ADU command transfers data from a specified buffer to a specified address on disk. The buffer address is specified in bytes 6 and 7 of the call block and must begin on a word boundary. The number of characters to write is specified in bytes 10 and 11 and must be an even number. The disk address to which the data is to be written is specified by the ADU number and sector offset in bytes 12 through 15 of the call block. If the character count is not an integral number of sectors, the remainder of that sector will be zeroed out.

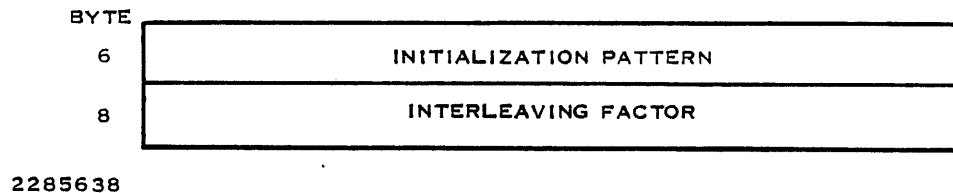
**7.19.2.7 Write by Track (Opcode > C).** The Write by Track command transfers data from a specified buffer to a specified address on disk. The buffer address is specified in bytes 6 and 7 of the call block and must begin on a word boundary. The number of characters to write is specified in bytes 10 and 11 and must be an even number. The disk address to which the data is to be written is specified by the track address in bytes 12 and 13 and the sectors per record and starting sector number in bytes 14 and 15.

**7.19.2.8 Write Interleaved Factor (Opcode > 12).** The Write Interleaved Factor command specifies an interleaving factor for a disk or DSDD diskette track. The operation can specify the data pattern used in formatting the disk or diskette. The operation is otherwise identical to the Write Format operation, opcode >08. Flags in the user flag field apply as follows:

Byte	Description
5	<p>Flags:</p> <p>Bit 0 — Initiate flag. If 1, the system initiates the operation and returns control to the calling task. If 0, the system suspends the calling task until the operation has completed.</p> <p>Bit 1 — Format data location flag. If 1, data is in the buffer block. If 0, data is in the call block.</p>

The input character count field must be specified as the number of bytes per record.

When the format data location flag is set to 1, a two-word buffer with the following contents is required:



When the format data location flag is set to zero, the write character count field (bytes 10 and 11) of the supervisor call block contains the interleaving factor.

The following is an example of the source code for a supervisor call block to format track 40 on a diskette with an interleaving factor:

```

FMTI   DATA   0           FORMAT TRACK 40 ON DISKETTE
        BYTE   > 12,> D4   ASSIGNED TO LUNO > D4 WITH
        DATA   0           INTERLEAVING FACTOR OF 3
        DATA   0
        DATA   256
        DATA   3
        DATA   40
        DATA   0
        .
        .
        .
        SVC    @FMTI

```

The buffer address is specified in bytes 6 and 7 of the call block and must begin on a word boundary. The interleaving factor is specified in bytes 10 and 11. The track address is specified in bytes 12 and 13.

### 7.19.3 Direct Diskette I/O

Some of the I/O opcodes for diskette operations, shown in Table 7-3, are the same as for disk, although DX10 does not support diskette addressing by ADU/sector offset. In addition, diskette I/O allows some special functions, such as specifying logical track numbers and sector sizes. The following paragraphs describe the I/O operations for diskette which are supported by DX10.

**7.19.3.1 Read Format (Opcodes > 5 and > F).** This command is exactly the same as for disk, except that a track address is required in bytes 12 and 13 of the I/O call block. Bytes 6 and 7 must contain the address of a 5-word buffer, which must begin on a word boundary, in which the system returns the same information as returned for disk (shown in Figure 7-2). All other fields in the call block are ignored.



Table 7-3. Diskette Direct Disk I/O Opcodes

Hexadecimal Opcode	I/O Function	I/O Action
0	Open	Ignored (device type returned)
1	Close	Ignored
2	Close EOF	Ignored
3	Open Rewind	Ignored (device type returned)
4	Close Unload	Ignored
5	Read Format	Return disk/track format
6	Forward Space	Ignored
7	Back Space	Ignored Formats Track
8	Write Format	Formats Track
9	Read ASCII	Read data
A	Read Direct	Read data
B	Write ASCII	Write data
C	Write Direct	Write data
D	Write EOF	Ignored
E	Rewind	Ignored
F	Read Format	Return disk/track format
10	Write Deleted Sector	Writes delete code on sector
11	Read Deleted Sector	Reads delete code on sector

**7.19.3.2 Write Format (Opcode > 8).** This command requires a track address in bytes 12 and 13 of the call block, and a record length, given in bytes 8 and 9. All other fields in the call block are ignored. DX10 formats the specified track, and returns the number of sectors per record in byte 14.

**7.19.3.3 Read ASCII (Opcode > 9), Read Direct (Opcode > A).** These two commands transfer data from the diskette to the user buffer whose address is specified in bytes 6 and 7 of the I/O call block. The length of the buffer is given in bytes 10 and 11, and must be an even number. Bytes 12 and 13 specify the track from which the data is to be read, byte 14 is the number of sectors per record on that track (should be 1), and byte 15 is the sector number on the track at which the read operation should begin. Under DX10, there is no difference between the Read ASCII and Read Direct commands. The system returns the actual number of characters read in bytes 10 and 11.

**7.19.3.4 Write ASCII (Opcode > B), Write Direct (Opcode > C).** These commands transfer data from the user buffer, pointed to by bytes 6 and 7 of the call block, to the diskette. Bytes 12 through 15 contain the track/sector address, as described in paragraph 7.19.3.3. Bytes 10 and 11 specify the number of characters to be written (must be an even number). DX10 makes no distinction between the Write ASCII and Write Direct commands.

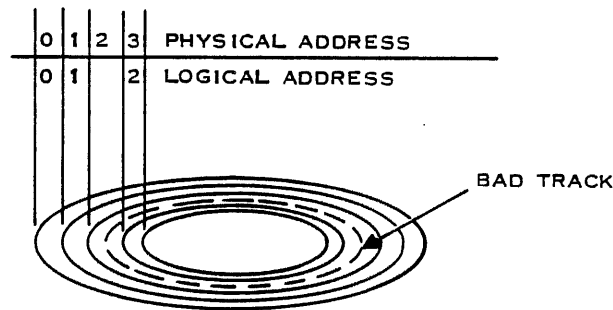
**7.19.3.5 Write Deleted Sector (Opcode > 10).** The diskette controller can delete a sector by writing a special data pattern on it. This operation can be performed by specifying a > 10 opcode (byte 2) and a track/sector address (bytes 12 through 15) in a normal I/O call block.

**7.19.3.6 Read Deleted Sector (Opcode > 11).** This operation is used to read the special data pattern written on a sector by the Write Deleted Sector operation. The Read Deleted Sector operation may be performed by specifying > 11 opcode (byte 2) and a track/sector address (bytes 12 through 15) in a normal I/O call block.

**7.19.3.7 Special Diskette Options.** In addition to the operations described above, the diskette controller allows two options, variable sector lengths and logical track numbering.

For any Read or Write operation, the user can specify the size of a sector, in the range of 2 to 128 bytes. The sector length must be even, and is assumed to be 128 bytes unless otherwise specified. A smaller sector size for an I/O operation results in unused disk space (the difference between 128 and the specified sector size). To specify the sector length, bit 6 of the user flags (byte 5 of the call block) must be set to one and the desired sector size placed in byte 14 (normally used for sectors/record).

Logical track numbering allows the user to assign an address to a track different from the physical address of that track. This option enables the user to skip bad tracks in the track numbering. Logical numbers should still increase from zero as the head travels inward on the diskette, as do the normal addresses. Figure 7-3 shows a logically numbered disk.



2283092

**Figure 7-3. Logical Track Addressing**

To assign a logical address to a track, a write format operation must be performed. Bit 6 of the user flags (byte 5) must be set to one, and the address to be assigned to the track must be in byte 14.

# DX10 2.X Compatibility

---

## 8.1 INTRODUCTION

New releases of DX10 incorporate improved methods of operation. These improvements include changes to techniques of formatting and accessing mass storage devices. Data files stored by systems running under releases of DX10 prior to Release 3.0 are not compatible with later releases of DX10. If you possess any such files, you must convert them to the current format. This section discusses the methods to perform this conversion.

## 8.2 DX10 2.X DISK CONVERSION

The procedure for transferring data files from a DX10 2.X disk to a Release 3 or later version disk varies depending on the hardware configuration of the system. The following paragraphs describe such procedures for the two types of systems: single disk with magnetic tape, and multiple disk systems.

### 8.2.1 Disk/Magnetic Tape Configuration

Transferral of sequential files and sequential library members from a 2.X disk to a Release 3 or later version disk involves no more than copying the 2.X files to an unformatted magnetic tape (operating under the 2.X system), installing the new version disk and loading the new system, and then copying the magnetic tape to a new directory, using the Copy/Concatenate (CC) command. The following is a step-by-step procedure for converting 2.0 sequential files and library members:

1. Copy all 2.0 sequential files and sequential library members to an unformatted tape (or tapes) using the CC command. Write down each file name as it is copied, maintaining a list of the files.
2. Install a Release 3 or later version system disk, perform an initial program load (IPL), and bid SCI.
3. Rewind the starting tape.
4. Create a directory under which the 2.X files are to be cataloged, using the Create Directory File (CFDIR) command (this step is optional).
5. Copy the files from tape to disk using the CC command. Use the written list in order to get the files from the tape in the correct order.

## 8.2.2 Multiple Disk Configuration

For systems with more than one disk drive, 2.X file conversion is much simpler. Once a Release 3 or later version system is running, you can convert the 2.0 files by executing the disk conversion utility. The utility runs under the DX10 Release 3 or later version operating system and requires at least two disk drives, one for the 2.X disk and one for a new version disk. The 2.X disk may remain write protected since the conversion utility only reads it. The new version disk must be initialized and installed, though not necessarily empty. The utility converts any or all of the 2.X files (except system files and indexed files) and writes them on the new version disk. Sequential and relative record files (both contiguous and noncontiguous) on the 2.X disk are converted to unbounded sequential and relative record files (respectively) on the Release 3 or later version disk. User file directories and libraries on the 2.X disk are converted into the new version directory files. If a 2.X file has the same pathname as a Release 3 or later version file, it is not converted, in order to avoid pathname conflicts.

**8.2.2.1 Operating Procedure.** When converting 2.X files, you have the option of placing the 2.X files under a Release 3 or later version directory or simply under the volume catalog of the disk receiving the files. By being cataloged under the volume catalog, the 2.X files retain the same pathnames as on the 2.X disk, a valuable characteristic when using programs developed under DX10 2.X. However, placing the 2.X files under a newly created Release 3 or later version directory avoids the possibility of pathname conflicts.

If the 2.X files are to be cataloged under a new version directory, the directory must exist before executing the conversion utility. In addition, the pathname of the directory may not exceed 27 characters in length (including periods).

To activate the conversion utility, issue the Execute 2.2 to 3.0 Disk Conversion (XCU) command. The following prompts appear:

```
[ ]XCU
EXECUTE 2.2 TO 3.0 DISK CONVERSION
      DISK TO BE CONVERTED: < acnm>
      LISTING DEVICE:      < acnm>
```

You must specify the disk drive that contains the 2.X disk to be converted and a listing device to which the utility should send any error messages and the log of converted files (see paragraph 8.2.2.2).

After you answer these prompts, the conversion utility prompts for the access name of a disk or a directory (if desired) under which the 2.X files are to be cataloged, as follows:

```
OUTPUT DISK NAME?: < acnm>
```

The default value for < acnm> is the system disk, with all 2.X files and libraries cataloged under the volume catalog (VCATALOG) with their 2.X pathnames. You can specify any disk in the system. You can specify any existing Release 3 or later version directory hierarchy under which the 2.X files should be cataloged, provided that the directory hierarchy has already been defined (as explained previously).

After you specify the output disk name, the conversion utility begins converting files. At this point, you have the option of converting all 2.X files or only specific files. The following prompt appears:

CONVERT FILE < pathname> (Y/N/A/S)?

You should respond with one of the following:

Response	Action
Y (yes)	Converts this file (< pathname>). The file is converted (if possible), and the utility prompts for the next file.
N (no)	Does not convert this file. The utility prompts for the next file.
A (all)	Converts all files on the directory specified by < pathname>. The utility terminates.
S (stop)	Terminates the conversion utility.

Whenever the conversion utility converts a 2.X file, the pathname of the Release 3 or later version file is the old 2.X pathname appended to whatever you responded to the OUTPUT DISK NAME? prompt. The following table shows how the conversion utility names converted files:

User Response to OUTPUT DISK NAME?	New Pathname of 2.X File UFD1.LIB1 (MEM1)
< default>	.UFD1.LIB1.MEM1
.MYCAT	.MYCAT.UFD1.LIB1.MEM1
DS03.MYCAT.DX2XFILE	DS03.MYCAT.DX2XFILE.UFD1.LIB1.MEM1
DS02	DS02.UFD1.LIB1.MEM1

#### NOTE

For compatibility purposes, DX10 Release 3 or later versions supports the 2.X convention of enclosing a library member name in parentheses, rather than delimiting it with a period. For example, the file in Example 2 can be referred to as the following:

.MYCAT.UFD1.LIB(FILEA)

**8.2.2.2 Converted File Log.** In addition to writing the converted 2.X files to the new version disk, the conversion utility writes a log of the files converted to the listing device. Figure 8-1 shows a sample log.

## DX10 R2.2 DISK CONVERSION

R2.2 DISK: TPLHOSE

PATHNAME	OT	NT	CC
.JON.TPL	U	D	
.JON.TPL.SRC	L	D	
.JON.TPL.SRC.IOF	BM	NS	
.JON.TPL.SRC.LNK	BM	NS	
.JON.TPL.SRC.DRV	BM	NS	
.JON.TPL.SRC.STK	BM	NS	
.JON.TPL.SRC.DSP	BM	NS	
.JON.TPL.SRC.LGC	BM	NS	
.JON.TPL.SRC.MAN	BM	NS	
.JON.TPL.SRC.NTR	BM	NS	
.JON.TPL.SRC.CHR	BM	NS	
.JON.TPL.OBJ	L	D	
.JON.TPL.OBJ.IDS	NM	NS	
.JON.TPL.OBJ.IOF	NM	NS	

END R2.2 DISK CONVERSION

Figure 8-1. Sample Log of Converted Files

The PATHNAME log entry is the new Release 3 or later version pathname of the file, with the old 2.X pathname appended to the new pathname specified in response to the OUTPUT DISK NAME?: prompt.

The OT entry is the old 2.X file type. NT is the new version file type. The file types are:

U	User file directory (old type only)
L	Library (old type only)
D	Directory (new type only)
BM	Blank suppressed library member (old type only)
NM	Nonblank suppressed library member (old type only)
CR	Contiguous relative record file (old type only)
NR	Noncontiguous relative record file
CS	Contiguous sequential file (old type only)
NS	Noncontiguous sequential file
CI	Contiguous indexed file (old type only)
NI	Noncontiguous indexed file (old type only)
A	Alias

The CC log entry is a completion code. A blank CC field indicates no error. An S means that the 2.X file was a system file and was not converted. If an alias was successfully converted, this field contains the name of the file to which the alias is assigned. Other codes indicate that a file conversion was aborted. They are preceded by an asterisk (\*) and have three fields.

Field 1 specifies what was aborted and may be one of the following:

Value	Meaning
X	The whole 2.X disk was aborted.
C	The whole catalog
U	Everything under a 2.X user file directory
L	The whole library
M	A library member
F	A file
A	An alias

Field 2 specifies the source of the error:

Value	Meaning
D	Error detected at the DX10 Release 3 or later version disk
2	Error detected at the 2.X disk
0	Error detected by the conversion utility

Field 3 specifies what error was detected. If the error was detected by the utility (for example, Field 2 was 0), it may be one of the following:

Value	Meaning
01	Physical record size of the 2.X file is too large for the utility buffers (greater than 1000 bytes).
02	Blank suppressed logical record size is too large for the utility buffers (greater than 200 bytes).
03	Sequential file EOF block error.

If the error was detected at either disk, Field C is the error code returned by an I/O SVC processor. Volume VI documents these error codes.

### 8.3 DX10 2.X SVC COMPATIBILITY

In addition to the SVCs described in Section 7 and in Volume III, DX10 Release 3.0 supports some SVCs strictly for the purpose of compatibility with programs developed under a 2.X system. Other SVCs are no longer supported. The following paragraphs describe SVC differences between DX10 2.X and Release 3 or later versions.

### **8.3.1 SVCs Not Supported**

The following 2.X SVCs are not supported by DX10 Release 3 or later versions:

- Abort I/O (by PRB — code > 1E)
- Set Condition Bit (code > 19)
- Get Character (code > 8)
- Conditional Get Character (code > 18) used with VDT character mode
- 913 VDT Utility (code > 1A)

The three SVCs used to support the 913 VDT in character mode have been replaced by an extension to the I/O supervisor call block, as explained in Volume III.

### **8.3.2 SVCs Supported Only for Compatibility**

The following SVCs are supported by DX10 Release 3 or later versions only to be compatible with 2.X programs:

- FUR Utility Call (code > 15)
- Librarian I/O Opcodes (SVC code > 00, opcodes > 80 through > 8F)
- End of Program (code > 16)
- Bid Task (code > 5)

The call blocks are the same in DX10 Release 3 or later versions as in 2.X. Support for these SVCs will be discontinued in a future release.

### **8.3.3 SVCs Still Used But Different**

Some SVCs under DX10 Release 3 or later versions have the same form (call block, opcode) as in 2.X but are implemented differently and have new uses:

- Get Memory (code > 12) — A Get Memory SVC in Release 3 or later versions automatically causes the task to be rolled out and therefore may not be used by memory-resident tasks or tasks with system common mapped in.
- Bid Task (code > 5) and Scheduled Bid Task (code > 1F) — These two SVCs may only bid nonreplicable tasks installed on the system program file.
- Activate Time Delayed Task (code > E) and Activate Suspended Task (code > 7) — The specified task ID must be the run-time ID (not the installed ID).
- Load Overlay (code > 14) — The LUNO specified should be assigned to a program file, and overlay numbers must be in the range from 0 through 255.



# 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.




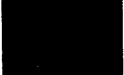








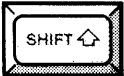



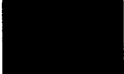
















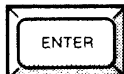









Multiple key sequences and simultaneous keystrokes can also be described in generic keycap names that are applicable to all terminals. For example, you use a multiple key sequence and simultaneous keystrokes with the log-on function. You log on by *pressing the Attention key, then holding down the Shift 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 Keypcap 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					 




























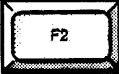





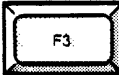











Notes:

<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 <sup>1</sup> KSR
Erase Input					 
Exit			 	 	
Forward Tab	 			 	 
F1					 
F2					 
F3					 
F4					 

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 Keypcap Names (Continued)































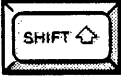


















Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820' KSR
F5					 
F6					 
F7					 
F8					 
F9	 			 	 
F10	 			 	 

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.

228 473 4 (4/1 4)

Table A-1. Generic Keycap Names (Continued)

Generic Name	911 VDT	940 EVT	931 VDT	Business System Terminal	820' KSR
F11	 			 	 
F12	 			 	 
F13	 	 	 	 	 
F14	 	 	 	 	 
Home					 
Initialize Input		 			 

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.

2284734 (5/14)














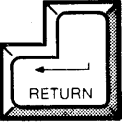




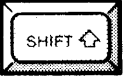


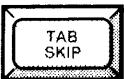








Table A-1. Generic Keycap Names (Continued)

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>3</sup>The keyboard is typamatic, and no repeat key is needed.

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

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

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

911 Phrase	Generic Name
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

228 4734 (8/14)



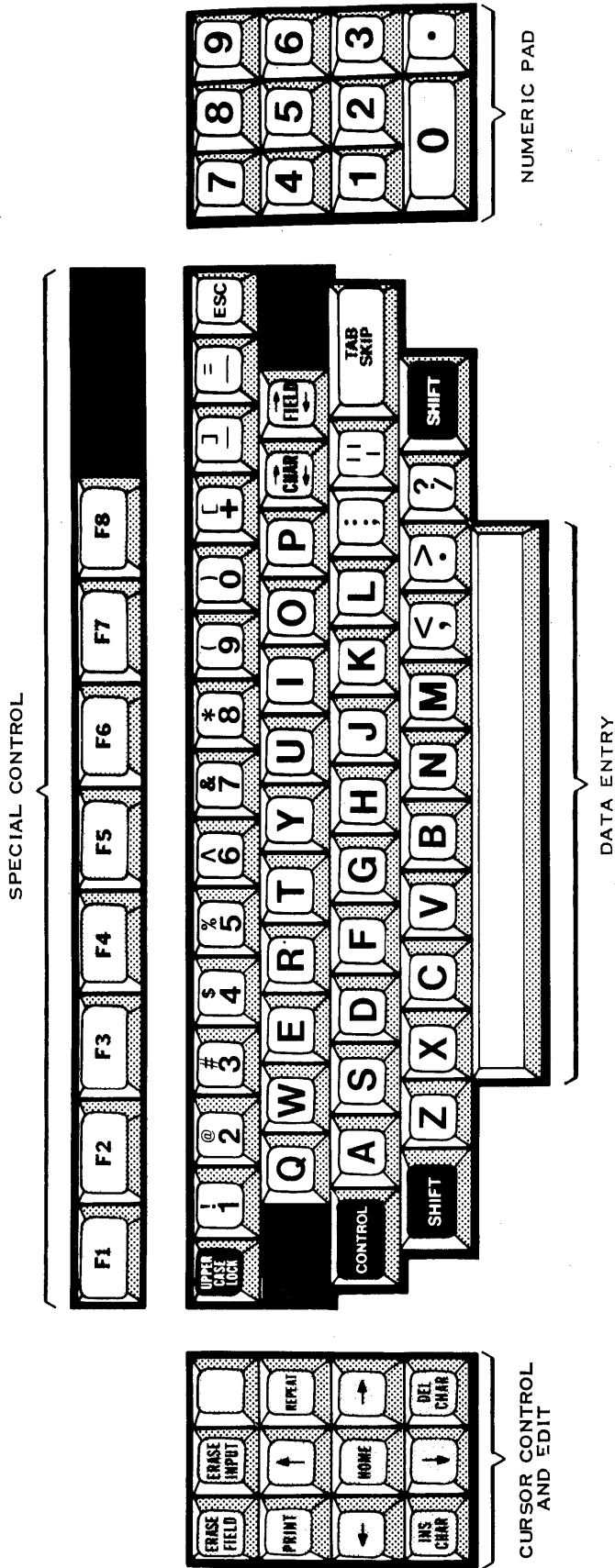


Figure A-1. 911 VDT Standard Keyboard Layout

2284734 (9/14)

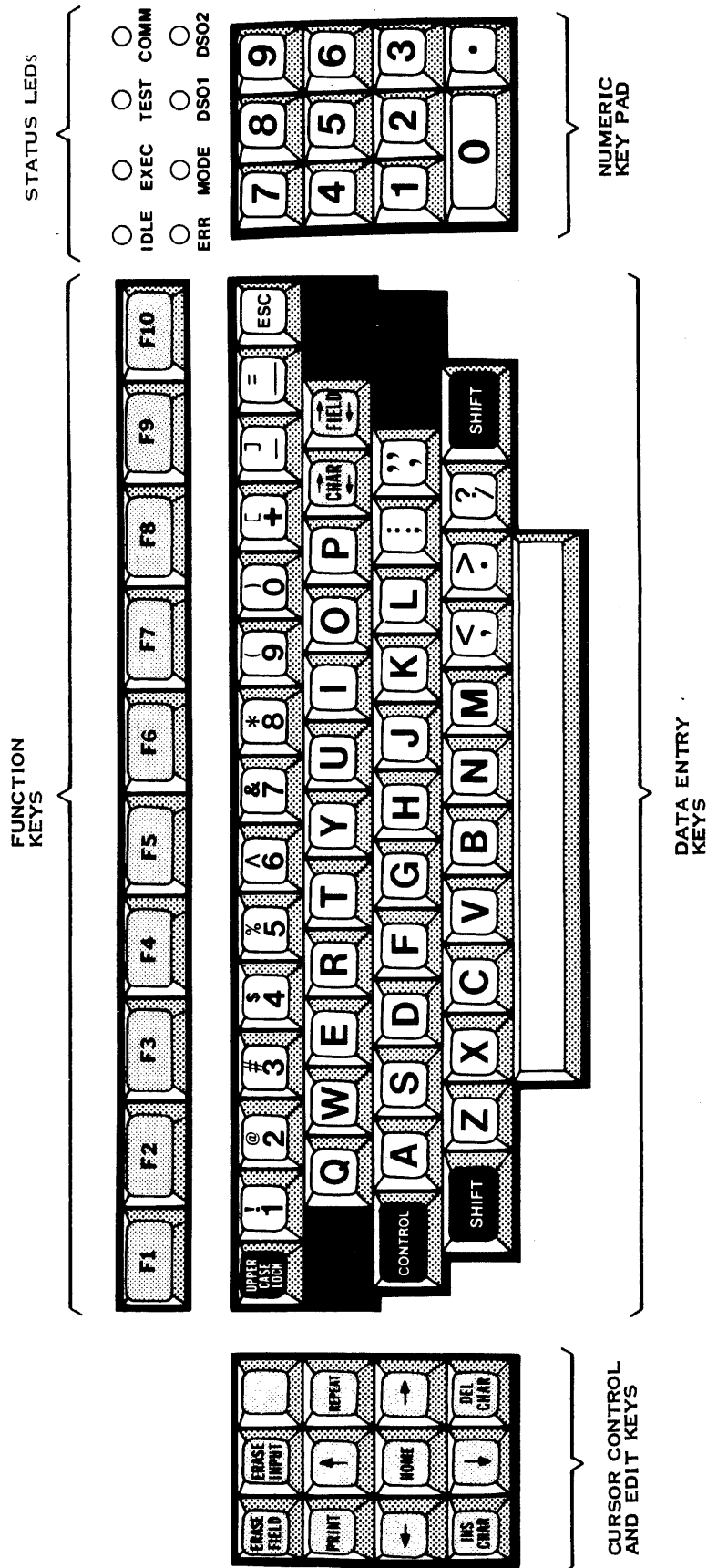


Figure A-2. 915 VDT Standard Keyboard Layout

2284734 (10/14)

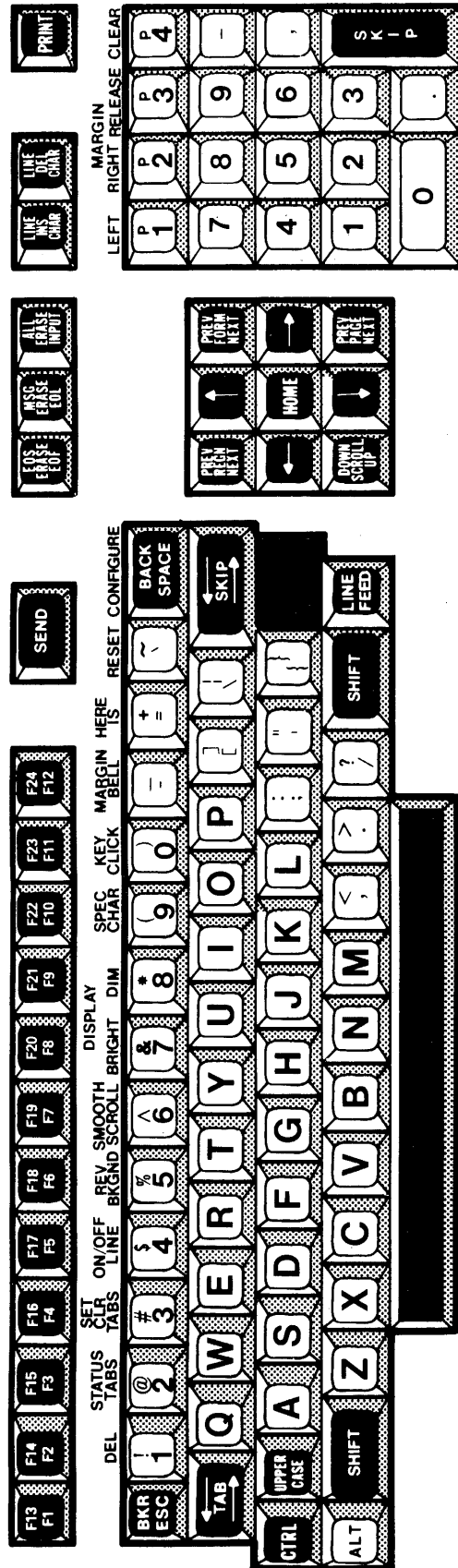


Figure A-3. 940 EVT Standard Keyboard Layout

2284734 (11/14)



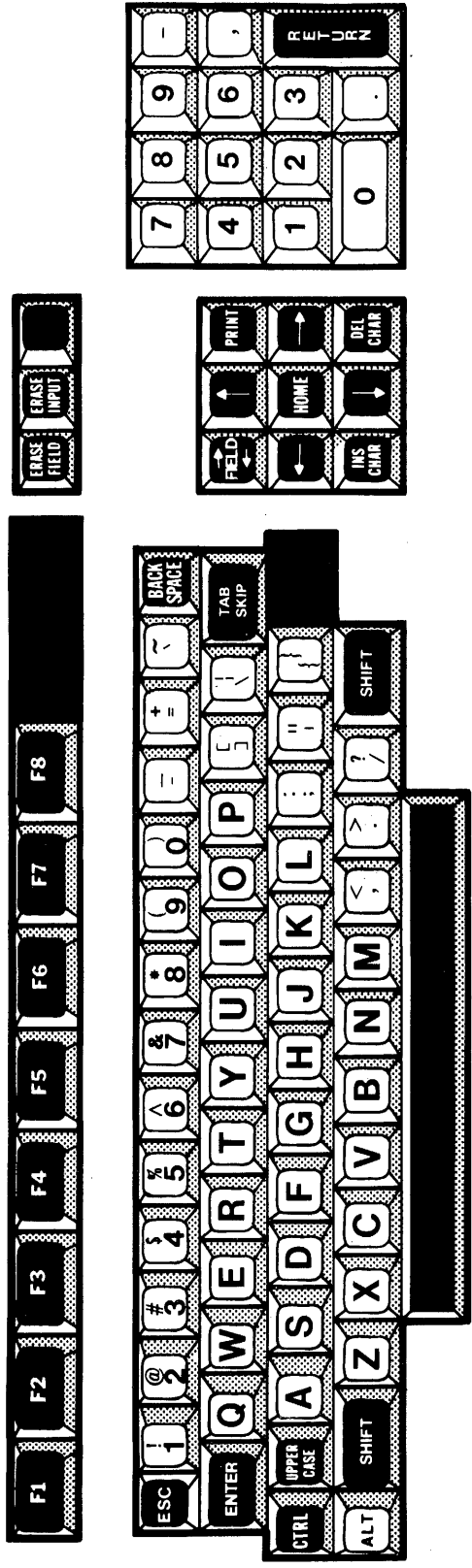
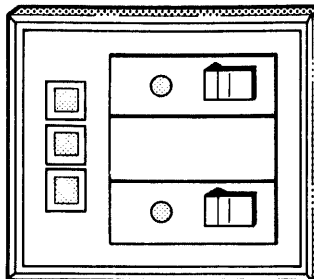
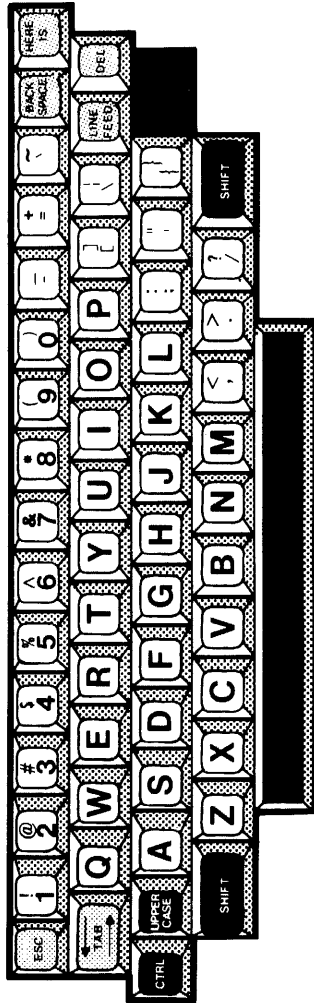
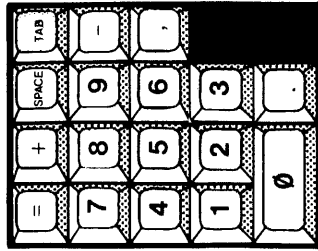


Figure A-5. Business System Terminal Standard Keyboard Layout

2284734 (13/14)



2284734 (14/14)

Figure A-6. 820 KSR Standard Keyboard Layout

# Appendix B

## System Task IDs

---

Table B-1 provides a list of the system task IDs.

**Table B-1. System Task IDs**

Code	Name	Purpose
0	RES00	Reserved
1	SMMAP	Display Memory Map
2	SCILMT	SCI Foreground Limitation Task
3	MLP	Modify LUNO Protect
4	SLMFOT	Format System Log
5	TM\$SBD	Scheduled Bid Task
6	TXSF	TX Set System File
7	DCOPY	Disk to Disk Copy
8	CPYSEQ	Copy Sequential Media
9	PF\$LIN	Install Task
A	TM\$DGN	Diagnostic Task
B	FUTIL	File Utility Task
C	SD	Disk Scan
D	OQ\$COPY	SCI Output Queuer
E	PATCHSYN	Patch Synonym Generator
F	SSTM	Show System Table Map
10	PF\$LDE	Delete Task
11	CVD	Copy and Verify Disk
12	TRCTSK	Reserved
13	PF\$LMN	Map Name to ID/ID to Name
14	SYSRST	System Restart Task
15	SVCKIL	Kill Task
16	INSTAL	Install/Unload Disk Volume
17	SIS	Show I/O Status
18	MADSAD	Modify/Show Absolute Disk
19	IDT	Initialize Date and Time
A1	CHKDSK	Check Disk Structures
1B	RTSTSK	Remote Terminal Subsystem Download Task
1C	MPISPI	Map/Show Program Image
1D	STS	Show Task Status
1E	SLCMDP	Start Log Command Processor
1F	PF\$LAS	Assign Space on a Program File
20	SCI990	System Command Interpreter
21	MRFSRF	Modify/Show Relative to File
22	MAPDSC	Map Disk
23	BRM	SCI Background Resource Manager
24	QBID	SCI Queue Bidder
25	OQ\$MGR	SCI Output Queuer
26	BD	Backup Directory

Table B-1. System Task IDs (Continued)

Code	Name	Purpose
27	RD	Restore Directory
28	CD	Copy Directory
29	DD	Delete Directory
2A	SORTMG	Sort/Merge
2B	LLR	List Logical Record
2C	CKS	Copy KIF to Sequential File
2D	DES	DES (Sort/Merge)
2E	OCPRAL	Release All LUNOs
2F	CONVRT	Convert 2.2 Disk to 3.X Disk
30	ISOLNK	Install System Overlay
31	MVIS	Modify Volume Information
32	LD	List Directory
33	GEN990	System Generation
34	S\$CCAF	Copy/Concatenate and Append File
35	AT	Activate Suspended Task
36	LSTSYN	List Synonyms
37	SVS	Show Volume Status
38	FILMAP	Map Program File
39	CPI	Copy Program Image
3A	TXMD	TX Map Disk
3B	TXFD	TX Format Diskette
3C	TXCP	TX Change Protection
3D	TXDX	TX to DX Conversion
3E	DXTX	DX to TX Conversion
3F	VC	Verify Copy
40	VB	Verify Backup
41	IBMUTL	IBM Conversion Utility
42	MKL	Modify Key Indexed File Logging
43	ERRADD	Add Error Message
44	ITGS	Install Generated System
45	MMPACK	Memory Packing
46	ERRSHW	Show Error Message
47	RES47	Reserved
48	CKSR	Copy KIF to Sequential Randomly
49	RES49	Reserved
4A	GETLOG	Determine Log File to Use
4B	KIFPRC	Reserved
4C	INVOL	Install New Volume
4D	RWCRU	Read/Write From/To CRU Address
4E	LDC	List Device Configuration
4F	MDS	Modify Device Status
50	MCCSCC	Modify/Show Country Code
51	TAMER	Reserved
52	SPL	Print File from Task
53	RCD	Recover Disk
54	DPLNKERR	
55	RES55	Reserved
56	IDS	Initialize Disk Surface
57	RES57	Reserved
58	RES58	Reserved
59	RES59	Reserved



Table B-1. System Task IDs (Continued)

Code	Name	Purpose
5A	CALANS	Call Terminal/Answer Incoming Call
5B	DISC	Disconnect Terminal
5C	MHPC	Modify Hard-copy Terminal Port Characteristics
5D	LHPC	List Hard-copy Terminal Port Characteristics
5E	SLA1	System Log Analysis — Level 1
5F	SLA2	System Log Analysis — Level 2
60	IPFTSK	International Print File Task
61	RES61	Reserved
62	CKR	Copy KIF Randomly
63	EDITOR	Text Editor
6B	ANALZ	Analyze Crash Dump
6C	DEBUG	DX10 Debugger
7B	IFSVC	SVC Processor
82	MODSYN	Modify Synonyms
83	TINFO	Terminal Information
84	OUTQUE	Print Output Queuer
85	MTMPMO	MTE, MPE, MOE
87	CRV	Check and Reset Volume
A0	SMS	Show Memory Status
A3	QSCOMM	DX10 HDLC Queue Server (> 4D SVC)
A4	PCDUMP	PC counter
A5	RESA5	Reserved
A6	RESA6	Reserved
DB	DBM	Data Base Manager
E0 – EF		Reserved for User Tasks
F0 – FD		Reserved for Communication Tasks



# Appendix C

## System Overlay IDs

Table C-1 provides a list of the system overlay ID numbers in the system overlay file. Table C-2 provides a list of the system task and utility overlay ID numbers in the system program file.

**Table C-1. System Overlay Numbers (.S\$0VLYA)**

Code	Name	Purpose
0	TM\$OVI	TM\$BID error recovery
1	OLN005	KIF, split a B-Tree
2	OLN006	KIF, insert a record
3	OLN007	KIF, rewrite a record
4	OLN00A	KIF, open/close processor
5	OLN00B	KIF, delete a record
6	OLN00C	KIF, delete B-tree entry overlay
7	FMOV10	File Management, rewrite, space
8	FMOV11	File Management, write EOF, rewind, unlock
9	FMOV12	File Management
A	FMOV13	File Management, open extend
B	DMOV0D	Disk Manager
C	DMOV0E	Disk Manager
D	DMOV0F	Disk Manager
E	DMOV14	Disk Manager
F	TM\$RWO	Task Manager
11	PLGERR	KIF, error recovery

**Table C-2. System Task and Utility Overlay Numbers (.S\$PROGA)**

Code	Name	Purpose
0	RES00	Reserved
2	PARSER	System Command Interpreter PARSER
3	E\$EDIT	Text Editor Edit Mode
4	E\$CMD\$	Text Editor Command Mode
5	INFT	System Generation
6	INTACT	System Generation
7	BUILD	System Generation
1B	IFROOT	SCI SVC Processor Main Routine
1C	ISOVC	SCI SVC Processor SVC Section
1D	ISOUCB	SCI SVC Processor UCB Section
1E	ISOERR	SCI SVC Processor Errors

Table C-2. System Task and Utility Overlay Numbers (.\$\$PROGA) (Continued)

Code	Name	Purpose
21	DERROR	System Command Interpreter Errors
22	MODSYN	SCI Modify Synonym
23	TINFO	Terminal Information
24	OUTQUE	CMD-990 Output Queuer
25	MTMPMO	SCI Modify Task/Overlay/Procedure Entry
26	XSRTGEN	Sort/Merge
27	XTRANSL	Sort/Merge
28	XHEADER	Sort/Merge
29	XRECORD	Sort/Merge
2A	XFIELD	Sort/Merge
2B	XSORT	Sort/Merge
2C	XSRTINT	Sort/Merge
2D	XSRT	Sort/Merge
2E	XLASPAS	Sort/Merge
2F	XFILMGR	Sort/Merge
30	XMRGINT	Sort/Merge
31	XMERGER	Sort/Merge
32	ISOVC	SCI SVC Processor SVC Section
32	COBOLOV1	COBOL (DX5 only)
33	ISOUCB1	SCI SVC Processor UCB Section
33	COBOLOV2	COBOL (DX5 only)
34	ISOERR2	SCI SVC errors
34	COBOLOV3	COBOL (DX5 only)
40	SMS	SCI Show Memory Status
41	XKDATA	SCI Procedure/Data Support
5E	SLA1	System Log Analyzer
5F	SLA2	Online Diagnostics
60	FUTOV0	File Utility Overlay
61	FUTOV1	File Utility Overlay
62	FUTOV2	File Utility Overlay
63	FUTOV3	File Utility Overlay
64	FUTOV4	File Utility Overlay
65	FUTOV5	File Utility Overlay
66	FUTOV6	File Utility Overlay
67	FUTOV7	File Utility Overlay
68	FUTOV8	File Utility Overlay
69	FUTOV9	File Utility Overlay
6A	FUTOVA	File Utility Overlay
6B	FUTOVB	File Utility Overlay
A3	QSCOMM	DX10 HDLC Communications QSERVER
B0	K\$EDIT	Japan-Kanji Text Editor
B1	K\$CMD\$	Japan-Kanji Text Editor
E0 - EF		Reserved
F0 - FD		Reserved

# Appendix D

## System Procedure IDs

---

Table D-1 provides a list of the system procedure IDs.

**Table D-1. Procedure IDs**

---

Hexadecimal Code	Name	Purpose
00		Reserved
01	RESERVED	Reserved for System Task
02	SCA	CMD-990 System Communication Area
03	SCI	CMD-990 System Command Interpreter
04	BCA	CMD-990 Background Communication Area
05	TXDXFC	TXDX File Conversion
06	DBMS	Data Base Management System
07	DBINFACE	Data Base Interface
08		Reserved
09	C\$HDLC	Industrial HDLC Procedure
0A	RTP	TPL Run Time
0B		Reserved
0C	S\$MODS	S\$ Routines
0D	S\$MODSCA	S\$ Routines for Tasks with SCA
0E	S\$OCP	OCP Routines (after S\$MODS)
0F	S\$MODOCP	S\$/OCP Routines for System Tasks
10	RCOBOL	COBOL Run Time (only to be used when COBOL has been installed on the system)
11	RPGII	RPGII Run Time
1B	LOWCOUNT	Reserved
1C	HICOUNT	Reserved
50	TIGRESS	Reserved
51	TAMER	Reserved
8B	BASIC	BASIC Run Time
DD		Data Dictionary
E0 – EF		Reserved
F0 – FD		Reserved

---



# Appendix E

## Global LUNOs

---

Table E-1 provides a list of the global LUNO assignments. When the words System LUNO appear in parentheses beside an assignment, that means the assignment cannot be released.

**Table E-1. Global LUNO Assignments**

LUNO	Assignment
00	System Console (ST01)
01	S\$FGTCA — Foreground TCA File
02	S\$BG TCA — Background TCA File
03	S\$TCALIB — TCA Library File
04	Reserved
05	Reserved
06	Reserved
07	News File
08	Reserved
09	System Disk
0A	S\$OVLYA — Overlay File (System LUNO)
0B	S\$PROGA — System Program File (System LUNO)
0C	Reserved
0D	S\$PROGA — System Program File (System LUNO)
0E	S\$ROLLA — Roll File (System LUNO)
0F	S\$PROGA — System Program File (System LUNO)
10	S\$SDS\$ — Languages Program File (System LUNO)
11	Reserved/WT
12	S\$SLG1 — System Log File
13	S\$SLG2 — System Log File
14	System Log Device
15	System Log Attention Device
16	.\$SWP — Word Processing Program File
17	.\$SODIAG.PGM — Online Diagnostics Program File
18 – 20	Reserved
21	Pascal Program File
22	.\$\$COMMPF — Communications Program File
23	DX10 RFT Program File (I) (X.25)
24	DX10 RFT Interprocess Device (X.25)
25	DX10 RFT Program File (II) (X.25)

**Table E-1. Global LUNO Assignments (Continued)**

---

<b>LUNO</b>	<b>Assignment</b>
26 – 3F	Reserved for Expansion
40	DX10 HDLC User Program File
90	DX10 HDLC System Program File
91 – CF	Reserved for User Files
D0 – D1	Reserved
D2	Disk 2
D3	Disk 3
D4 – FF	Reserved for System Assignment

---



# Appendix F

## System Generation Examples

---

### F.1 INTRODUCTION

This appendix helps you complete the system generation process for any installation. The examples given here follow the format required to perform all system generations; however, the details for a specific installation vary with the needs of that installation.

These examples demonstrate the terminal display generated during the execution of GEN990, the System Generation utility. Although system generation consists of a sequence of utilities (GEN990, ALGS, PGS, and IGS), only GEN990 requires extensive interaction with you. Since Section 3 provides examples of ALGS, PGS, and IGS commands, this appendix only provides examples of GEN990.

These examples copy the format that you actually see on the screen. Each line begins with a word or phrase followed by a question mark (?). This word or phrase is the prompt displayed by GEN990. Respond to the prompt by keying in an appropriate value for the prompt and pressing the Return key as described in Section 3.

The prompt may be followed by an initial value in parentheses. You can accept this value by pressing the Return key without entering another value. Alternatively, you can enter your own choice of values in the space following the initial value.

You can respond to some prompts by simply pressing the Return key. The DX10 operating system supplies an appropriate default value for the parameter. If a default value is not available, the cursor remains in its location following the prompt until you enter a value.

In these examples, some prompts are followed by responses. Other prompts are followed by an initial value and no other response. Still other prompts are followed by no response at all. If a prompt is not followed by a response, it is clear that the default value or initial value is accepted.

## F.2 EXAMPLE ONE — FIRST TIME SYSTEM GENERATION

In this example, suppose you want to include the following devices in your configuration:

- Two DS50 disk drives
- One 733 ASR
- Two 911 VDTs
- One 931 VDT
- One 940 EVT
- One 810 line printer

In this example, an ASR is the attention device for the system log. No user-written tasks are bid when an IPL is performed. KIF logic is also included. The interrupt levels used are those for a 13-slot chassis. Note that this example does not include an expansion chassis or user-written SVCs.

Figure F-1 shows the desired configuration for this example. The values in parentheses are the default values displayed by GEN990, the System Generation utility.

Initiate system generation by issuing the XGEN command in response to the command prompt ([ ]). The prompts for the XGEN command appear on the terminal screen:

```
[ ]XGEN
EXECUTE AUTO-SYSGEN
      DATA DISK/VOLUME:  DS01
      INPUT CONFIGURATION:
      OUTPUT CONFIGURATION:  EXAMP
```

After you respond to the preceding prompts, the screen becomes clear temporarily. The remaining GEN990 prompts and messages appear on the terminal screen in TTY mode.

LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1	SMI				1	SMI			
2	AU				2	AU			
3	MEMORY CONTROL				3	MEMORY CONTROL			
4	MEMORY				4	MEMORY			
5	MEMORY				5	MEMORY			
6					6				
7	DS50		>F800	13	7	DS50		>F800	13
8					8				
9					9				
10	931 VDT #4	>0E0	>F980	12	10	940 EVT #3	>0C0	>F980	11
11	911 VDT #2	>0A0		3	11	911 VDT #1	>080		7
12	810 LP	>060		14	12				
13					13	733 ASR	>000		6

2283094

Figure F-1. Example 1 — Configuration Table

GEN990-AUTO SYSGEN RELEASE 3.6.0

LINE? (60)  
TIME SLICING ENABLED? (YES)  
TIME SLICE VALUE? (1)  
TASK SENTRY ENABLED? (YES)  
TASK SENTRY VALUE? (60)  
TABLE? 6000  
COMMON?  
INTERRUPT DECODER?  
FILE MANAGEMENT TASKS? (2)  
CLOCK? (5)  
RESTART ID?  
OVERLAYS? (2)  
ONLINE DIAGNOSTIC SUPPORT? (NO)  
SYSLOG? (6)  
BUFFER MANAGEMENT (BYTES)? (1024)  
I/O BUFFERS?  
INTERTASK? (100)  
ITC MESSAGES? (3)  
KIF? (YES)  
COUNTRY CODE? (US)  
POWER FAIL? (NO)  
AUTO MEDIA CHANGE RECOVERY? (YES)  
SCI BACKGROUND? (2)  
SCI FOREGROUND? (8) 5  
BREAKPOINT? (16)  
PANEL DISPLAY(BAR CHART)? (YES)  
CARD 1?  
CARD 2?  
NEXT? DVC

If unsure of the proper response to a parameter, you can enter a question mark (?) and press the Return key for a message (similar to the one below). Messages also appear if you enter an invalid response.

For example, to receive information on the DEVICE TYPE? prompt, enter a question mark in response and press the Return key.

DEVICE TYPE?

The following information appears on the screen:

EACH TYPE OF DEVICE OR PERIPHERAL IN YOUR SYSTEM MUST BE DEFINED.  
THE FOLLOWING MNEMONICS ARE USED:

CRDR — CARD READER	LP — LINE PRINTER
DS — DISK	VDT — VIDEO DISPLAY TERMINAL
ASR — ASR/733	KSR — KSR
MT — MAG TAPE	COM — COMMUNICATIONS
SD — SPECIAL DEVICE	

IF MORE THAN ONE DEVICE OF THE SAME TYPE EXISTS, EACH MUST BE DEFINED SEPARATELY. TO STOP ADDING DEVICES ENTER "RETURN" WHEN PROMPTED.

After the message, continue building the system. The next set of parameters defines the two disk drives. The type is determined by the type of disk controller installed in the system and has no bearing on the system generation process.

```

DEVICE TYPE? DS
TILINE ADDRESS? (> F800)
DRIVES? (1) 2
DEFAULT RECORD SIZE? (768)
INTERRUPT? (13)

```

The next set of parameters defines an ASR:

```

DEVICE TYPE? ASR
CRU ADDRESS? 0
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
CASSETTE OPENS VALIDATED? (YES)
INTERRUPT? (6)

```

The next set of parameters defines the VDTs:

```

DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100) 080
INTERRUPT? (10) 7

```

```

DEVICE TYPE? VDT
*** WARNING : THAT ADDRESS HAS BEEN PREVIOUSLY DEFINED ***
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100) 080
INTERRUPT? (10) 3

```

```
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 940
ASSOCIATED PRINTER? (NO)
SWITCHED LINE? (NO)
SPEED? (1200)
INTERFACE TYPE? CI403
TILINE ADDRESS? (> F980)
CHANNEL NUMBER? (0)
INTERRUPT? (10) 11
```

```
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 931
ASSOCIATED PRINTER? (NO)
SWITCHED LINE? (NO)
SPEED? (1200)
INTERFACE TYPE? CI403
TILINE ADDRESS? (> F980)
CHANNEL NUMBER? (0) 1
INTERRUPT? (10) 11
```

In the example, you made an error when you defined the CRU address for the second VDT. You correct it shortly.

The next set of parameters defines an 810 line printer. This example and all others that define 810 printers assume that the extended character set is present.

```
DEVICE TYPE? LP
ACCESS TYPE? (FILE)
PRINT MODE? (SERIAL)
EXTENDED? (YES)
SPEED? (4800)
TIME OUT? (60)
INTERFACE TYPE? TTY/EIA
CRU ADDRESS? (> 60)
INTERRUPT? (14)
```

After you define all the devices, press the Return key. The System Generation utility prompts for another device. At this time, correct the error that you made earlier. You must correct errors before terminating the System Generation utility. Press the Command key. This places the System Generation utility in command mode, and you can change or delete any device. This is the format:

```

DEVICE TYPE? < Press the Command key>
COMMAND? CHANGE
PARAMETER TO BE CHANGED? DVC
DEVICE NAME? ST03
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100) > A0
INTERRUPT? (10) 3
COMMAND?

```

To change a device, you must know the device name. The device in error was the second VDT defined, but it was the third station defined to the system (the ASR is ST01 and the first VDT is ST02). Hence, its device name is ST03. You can enter the LIST command when in the command mode to display the device names and the current parameter values for the configured system. Once all entities have been defined, you can terminate this system generation session.

```

COMMAND? STOP
BUILD? (YES)
***** CONFIGURATION FILE SAVED *****
***** D$DATA SOURCE FILE IS NOW BEING BUILT *****
***** THE LINK EDIT COMMAND STREAM SOURCE FILE IS BEING BUILT *****
***** BATCH FILE FOR SYSGEN COMPLETION IS NOW BEING BUILT *****
***** SYSGEN COMPLETE *****
***** GEN990 TERMINATED *****:

```

With GEN990 completed successfully, press the Command key to return to SCI. The data provided to GEN990 must be further processed into a form that is suitable for use by DX10. The two system utilities that perform this processing are as follows:

- Assemble and Link Generated System
- Patch Generated System

Two more system utilities test the generated system and install it as the system that loads whenever an initial program load (IPL) occurs:

- Test Generated System
- Install Generated System

These utilities are initiated by SCI commands. Since they do not require extensive interaction with you, they are not demonstrated in these examples. Refer to Section 3 for a description of these utilities.

### F.3 EXAMPLE TWO — MODIFYING AN EXISTING CONFIGURATION FILE

This example uses the configuration developed in Example One for the input configuration and uses it as a framework to build on. The system configuration for this example contains the following:

- Two DS50 disk drives
- One 820 KSR (replacing the 733 ASR)
- Four 911 VDTs (two VDTs added)
- One 931 VDT
- One 940 EVT
- One 810 line printer
- One 2260 parallel line printer (added)
- One 979A tape drive (added)

Figure F-2 shows the desired configuration for this example.

Initiate GEN990 by issuing the XGEN command in response to the command prompt ([ ]). The prompts for the XGEN command appear on the terminal screen:

```
[ ]XGEN
EXECUTE AUTO-SYSGEN
      DATA DISK/VOLUME: DS01
      INPUT CONFIGURATION: EXAMP
      OUTPUT CONFIGURATION: EXAMP2
```

After you respond to the preceding prompts, the screen becomes clear temporarily. The remaining GEN990 prompts and messages appear on the terminal screen in TTY mode.

```
GEN990-AUTO SYSGEN RELEASE 3.6.0
```

GEN990 pauses for a few seconds while it reads the input configuration file and copies it to the output configuration file. When ready for further input from you, GEN990 enters the command mode and displays the following prompt:

```
COMMAND?
```

To modify the system configuration of EXAMP to match that of the second example, you must delete the 733 ASR and add an 820 KSR. You can accomplish this in two steps, first deleting the ASR and, then, adding the KSR as a new device. You can also accomplish this in a single step using the CHANGE command. The following example demonstrates the use of the CHANGE command. To change, delete or add a parameter, first enter the command mode by pressing the Command key on your terminal. For this example, respond to the COMMAND? prompt with CHANGE.



LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1	SMI				1	SMI			
2	AU				2	AU			
3	MEMORY CONTROL				3	MEMORY CONTROL			
4	MEMORY				4	MEMORY			
5	MEMORY				5	MEMORY			
6					6				
7	DS50		>F800	13	7	DS50		>F800	13
8	979A TAPE		>F880	9	8	979A TAPE		>F880	9
9	911 VDT #6	>120		8	9	911 VDT #5	>100		10
10	931 VDT #4	>0E0	>F980	12	10	940 EVT #3	>0C0	>F980	11
11	911 VDT #2	>0A0		3	11	911 VDT #1	>080		7
12	810 LP	>060		14	12	2260 LP	>040		4
13					13	820 KSR	>000		6

2283095

Figure F-2. Example 2 — Configuration Table

```
COMMAND? CHANGE
PARAMETER TO BE CHANGED? DVC
DEVICE NAME? ST01
DEVICE TYPE? KSR
DSR TYPE(TPD/KSR/K820)? TPD
TERMINAL TYPE? (743) 820
INTERFACE TYPE? 9902
SWITCHED LINE? (NO)
BAUD RATE? (300) 4800
ACU PRESENT? (NO)
FULL DUPLEX MODEM? (YES)
ECHO? (YES)
CRU ADDRESS? (> 00)
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
INTERRUPT? (6)
COMMAND?
```

Next, you must add the two 911 VDTs. Enter the inquire mode by entering INQUIRE in response to the COMMAND? prompt.

```
COMMAND? INQUIRE
NEXT? DVC
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100)
INTERRUPT? (10)
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100) 0120
INTERRUPT? (10) 8
```

This example stops GEN990 at this point to demonstrate that a configuration need not be completed in one session. However, you must direct GEN990 to save the configuration file if you want to preserve the data that has been entered.

```

DEVICE TYPE? < Press the Command key>
COMMAND? STOP
BUILD? (YES) NO
DO YOU WANT TO SAVE THE CONFIGURATION FILE? (NO) YES
***** CONFIGURATION FILE SAVED *****
***** SYSGEN COMPLETE *****
***** GEN990 TERMINATED *****;

```

When you want to finish defining the devices in this system, reenter GEN990 as shown below. Note that the input configuration is EXAM2.

```

[ ]XGEN
EXECUTE AUTO-SYSGEN
  DATA DISK/VOLUME: DS01
  INPUT CONFIGURATION: EXAM2
  OUTPUT CONFIGURATION: EXAM2

```

GEN990 SYSTEM GENERATION DX10 3.6.0

GEN990 pauses for a few seconds while it reads the input configuration file and copies it to the output configuration file. When ready for further input from you, GEN990 enters the command mode and displays the following prompt:

COMMAND?

The only devices that you still must define are the additional line printer and the magnetic tape drive.

```

COMMAND? INQUIRE
NEXT? DVC
DEVICE TYPE? LP
ACCESS TYPE? (FILE)
PRINT MODE? (SERIAL) PARALLEL
EXTENDED? (YES)
TIME OUT? (60)
CRU ADDRESS? (> 60) > 40
INTERRUPT? (14) 4

```

```
DEVICE TYPE? MT
TILINE ADDRESS? (> F880)
DRIVES? (1)
INTERRUPT? (9)
DEVICE TYPE? < Press the Command key>
COMMAND? STOP
BUILD? (YES)
***** CONFIGURATION FILE SAVED *****
***** D$DATA SOURCE FILE IS NOW BEING BUILT *****
***** THE LINK EDIT COMMAND STREAM SOURCE FILE IS BEING BUILT *****
***** BATCH FILE FOR SYSGEN COMPLETION IS NOW BEING BUILT *****
***** SYSGEN COMPLETE *****
***** GEN990 TERMINATED *****:
```

This configuration is now complete and ready to be assembled and linked, patched, tested, and installed.

#### F.4 EXAMPLE THREE — DEFINING SPECIAL DEVICES

This example demonstrates the definition of a special device. This example is not associated with the previous configuration charts. You must supply the device service routine (DSR) for the device and a physical device table (PDT). Refer to Section 5 for details on DSRs and PDTs.

```
DEVICE TYPE? SD
INTERFACE TYPE? (NONE)
SPECIAL DEVICE ADDRESS? (> 100)
DEVICE NAME? PL01
INTERRUPT BIT? (15)
KSB ADDRESS? (NONE)
DSR WORKSPACE? DSRPIO
INTERRUPT ENTRY? PLINT
PDT FILE? .S$SYSGEN.PLPDT01
DSR FILE? .S$SYSGEN.PLDSR
OVERRIDE? YES
INTERRUPT? (15) 10
NEXT? DVC
```

If the device contains a keyboard, GEN990 names it using the ST prefix according to how it named all previous keyboard devices. However, this is not a keyboard device; it is a plotter device.

The next example shows how to define a second plotter:

```

DEVICE TYPE? SD
INTERFACE TYPE? (NONE)
SPECIAL DEVICE ADDRESS? (> 100) > 120
DEVICE NAME? PL02
INTERRUPT BIT? (15)
KSB ADDRESS? (NONE)
DSR WORKSPACE? DSRPIO
INTERRUPT ENTRY? PLINT
PDT FILE? .$$$SYSGEN.PLPDT02
DSR FILE? .$$$SYSGEN.PLDSR
OVERRIDE? YES
INTERRUPT? (15) 10

```

A special device can be a TILINE device. The following example displays the portion of the System Generation utility required to define two special TILINE devices, with device names SS01 and SS02:

```

DEVICE TYPE? SD
INTERFACE TYPE? (NONE)
SPECIAL DEVICE ADDRESS? (> 100) > F850
DEVICE NAME? SS01
INTERRUPT BIT? (15)
KSB ADDRESS? (NONE) SSKSB01
DSR WORKSPACE? DRSIO
INTERRUPT ENTRY? SSINT
PDT FILE? .$$$SYSGEN.SSPDT01
DSR FILE? .$$$SYSGEN.SSDSR
OVERRIDE? YES
INTERRUPT? (15)

```

```

DEVICE TYPE? SD
INTERFACE TYPE? (NONE)
SPECIAL DEVICE ADDRESS? (> 100) > F860
DEVICE NAME? SS02
INTERRUPT BIT? (15)
KSB ADDRESS? (NONE) SSKSB02
DSR WORKSPACE? DRSIO
INTERRUPT ENTRY? SSINT
PDT FILE? .$$$SYSGEN.SSPDT02
DSR FILE? .$$$SYSGEN.SSDSR
OVERRIDE? YES
INTERRUPT? (15) 14

```

## F.5 EXAMPLE FOUR — DEFINING AN EXPANSION CHASSIS

This example describes the steps required to perform a system generation for an installation that requires at least one expansion chassis. This example also describes the inclusion of the 3780 communications emulator with an auto-call unit (ACU). The system configuration for this example includes the following:

- Two DS50 disk drives
- One DS10 disk drive
- One FD1000 double-sided, double-density (DSDD) flexible diskette
- One 979A tape drive
- One communications interface board with 3780 protocol
- Two 810 line printers
- One 2260 line printer
- One 733 KSR
- Seven 911 VDTs
- Two 820 KSRs
- Two special devices

Figure F-3 shows the devices as they are configured in the main chassis (a 17-slot chassis in this example). Figure F-4 is the configuration chart for the expansion chassis.

The steps for defining this configuration are as follows:

Initiate system generation by issuing the XGEN command in response to the command prompt ([ ]). The prompts for the XGEN command appear on the terminal screen:

```
[ ] XGEN
EXECUTE AUTO-SYSGEN
      DATA DISK/VOLUME: DS01
      INPUT CONFIGURATION:
      OUTPUT CONFIGURATION: EXAM4
```

After you respond to the preceding prompts, the screen becomes clear temporarily. The remaining GEN990 prompts and messages appear on the terminal screen in TTY mode.

TOP OF CHASSIS (P1)					BOTTOM OF CHASSIS (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1	SMI				1	SMI			
2	AU				2	AU			
3	MEMORY CONTROL				3	MEMORY CONTROL			
4	MEMORY				4	MEMORY			
5	MEMORY				5	MEMORY			
6	MEMORY				6	MEMORY			
7	DS50		>F800	15	7	DS50		>F800	15
8	979A TAPE		>F880	12	8	979A TAPE		>F880	12
9	DS10		>F820	8	9	DS10		>F820	8
10	FD1000		>F840	3	10	FD1000		>F840	3
11	MODEM #2				11	2260 LP	>180		13
12	ACU #2	>160			12				
13	CRU EXPANSION BOARD			10	13	CRU EXPANSION BOARD			10
14					14				
15	ACU #1	>0A0			15	CI401 (COMM I/F)	>080		7
16	810 LP #2	>060		14	16	810 LP #1	>040		4
17	MODEM #1				17	733-ASR	>000		6

2283096

Figure F-3. Example 4 — Main Chassis

LEFT SIDE (P1)					RIGHT SIDE (P2)				
SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT	SLOT	DEVICE	CRU BASE	TILINE	INTER- RUPT
1	CRU BUFFER BOARD								
2					2				
3					3				
4					4				
5					5				
6					6				
7					7	SPECIAL TM	>580		13
8					8	SPECIAL EM	>540		9
9	820 KSR	>520		8	9	820 KSR	>500		10
10	911 VDT #8	>4E0		12	10	911 VDT #7	>4C0		11
11	911 VDT #6	>4A0		3	11	911 VDT #6	>480		7
12	911 VDT #4	>460		14	12	911 VDT #3	>440		4
13	911 VDT #2	>420		15	12	820 KSR	>400		6

2283097

Figure F-4. Example 4 — Expansion Chassis



## GEN990-AUTO SYSGEN RELEASE 3.6.0

LINE? (60)  
 TIME SLICING ENABLED? (YES)  
 TIME SLICE VALUE? (1)  
 TASK SENTRY ENABLED? (YES)  
 TASK SENTRY VALUE? (60)  
 TABLE? 8000  
 COMMON?  
 INTERRUPT DECODER?  
 FILE MANAGEMENT TASKS? (2)  
 RESTART ID?  
 OVERLAYS? (2)  
 ONLINE DIAGNOSTIC SUPPORT? (NO)  
 SYSLOG? (6)  
 BUFFER MANAGEMENT (BYTES)? (1024)  
 I/O BUFFERS?  
 INTERTASK? (100)  
 ITC MESSAGES? (3)  
 KIF? (YES)  
 COUNTRY CODE? (US)  
 POWER FAIL? (NO)  
 AUTO MEDIA CHANGE RECOVERY? (YES)  
 SCI BACKGROUND? (2)  
 SCI FOREGROUND? (8) 10  
 BREAKPOINT? (16)  
 PANEL DISPLAY(BAR CHART)? (YES)  
 CARD 1? 10  
 CARD 2?  
 NEXT? DVC  
  
 ENTITY? DVC  
 DEVICE TYPE? DS  
 TILINE ADDRESS? (> F800)  
 DRIVES? (1) 2  
 DEFAULT RECORD SIZE? (768)  
 INTERRUPT? (13) 15  
  
 DEVICE TYPE? MT  
 TILINE ADDRESS? (> F880)  
 DRIVES? (1)  
 INTERRUPT? (9) 12

The next disk drive to be defined is the DS10. (Refer to Figure F-3.) Note that a DS10 disk drive consists of two drives that share a single cabinet and a single interface to a single controller board. GEN990 treats each drive as a separate logical device.

DEVICE TYPE? DS  
TILINE ADDRESS? (> F800) > F820  
DRIVES? (1) 2  
DEFAULT RECORD SIZE? (768) 288  
INTERRUPT? (13) 8

The next device is the Model FD1000 flexible diskette drive.

DEVICE TYPE? DS  
TILINE ADDRESS? (> F800) > F840  
DRIVES? (1) 2  
DEFAULT RECORD SIZE? (768) 256  
INTERRUPT? (13) 3

DEVICE TYPE? ASR  
CRU ADDRESS? 0  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
CASSETTE OPENS VALIDATED? (YES)  
CASSETTE ACCESS TYPE? (FILE)  
INTERRUPT? (6)

DEVICE TYPE? LP  
ACCESS TYPE? (FILE)  
PRINT MODE? (SERIAL)  
EXTENDED? (YES)  
SPEED? (4800)  
TIME OUT? (60)  
INTERFACE TYPE? TTY/EIA  
CRU ADDRESS? (> 60) > 40  
INTERRUPT? (14) 4

DEVICE TYPE? LP  
ACCESS TYPE? (FILE)  
PRINT MODE? (SERIAL)  
EXTENDED? (YES)  
SPEED? (4800)  
TIME OUT? (60)  
INTERFACE TYPE? TTY/EIA  
CRU ADDRESS? (> 60)  
INTERRUPT? (14)

DEVICE TYPE? LP  
ACCESS TYPE? (FILE)  
PRINT MODE? (SERIAL) PARALLEL  
EXTENDED? (YES)  
TIME OUT? (60)  
CRU ADDRESS? (> 60) > 180  
INTERRUPT? (14) 13

You must define the communications emulators being used. Enter the interrupt level and CRU or TILINE address associated with the communications board. You do not need to define the ACU or the modem associated with this device during GEN990's execution. Define the ACU and the associated 3780 emulator after executing GEN990 and before assembling and linking the generated system (ALGS) as part of the emulator's software installation. (Consult the appropriate emulator object installation guide.) The modem is in the chassis at this time for the sake of convenience only. The modem does not receive power until it is physically connected to an ACU and even then its presence is transparent to the operating system.

```

DEVICE TYPE? COM
COM DEVICE ADDRESS? > 80
NUMBER OF CHANNELS? (1)
CHANNEL NUMBER 00 PROTOCOL? 3780
BUFFER SIZE? (528)

```

Since all of the 820s are installed in the expansion chassis, the interrupt level assigned to each is the same. The parameter EXPANSION POSITION refers to the position or interrupt level associated with the device in the expansion chassis. The DSR type is specified as TPD although it could be the K820 DSR. Refer to the note in paragraph 3.4.5.6 for a discussion of the available DSRs. The following examples define the 820 KSRs:

```

DEVICE TYPE? KSR
DSR TYPE(TPD/KSR/K820)? TPD
TERMINAL TYPE? (743) 820
INTERFACE TYPE? CI401
SWITCHED LINE? (NO)
BAUD RATE? (300) 1200
ACU PRESENT? (NO) YES
ACU CRU? (0) > 0160
FULL DUPLEX MODEM? (YES)
ECHO? (YES)
CRU ADDRESS? (> 00) > 0520
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
INTERRUPT? (6) 10
EXPANSION CHASSIS? 1
EXPANSION POSITION? 8

```

DEVICE TYPE? KSR  
DSR TYPE(TPD/KSR/K820)? TPD  
TERMINAL TYPE? (743) 820  
INTERFACE TYPE? CI401  
SWITCHED LINE? (NO)  
BAUD RATE? (300) 1200  
ACU PRESENT? (NO)  
FULL DUPLEX MODEM? (YES)  
ECHO? (YES)  
CRU ADDRESS? (> 00) > 0500  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
INTERRUPT? (6) 10  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 10

DEVICE TYPE? KSR  
DSR TYPE(TPD/KSR/K820)? TPD  
TERMINAL TYPE? (743) 820  
INTERFACE TYPE? TTY/EIA  
SWITCHED LINE? (NO)  
BAUD RATE? (300)  
ACU PRESENT? (NO)  
FULL DUPLEX MODEM? (YES)  
ECHO? (YES)  
CRU ADDRESS? (> 00) > 0400  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
INTERRUPT? (6) 10  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 6

Since all of the VDTs are installed in the expansion chassis, the interrupt level assigned to each is the same. The parameter EXPANSION POSITION refers to the position or interrupt level associated with the device in the expansion chassis.

DEVICE TYPE? VDT  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
VDT TYPE? 911  
CRU ADDRESS? (> 100) > 420  
INTERRUPT? (10)  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 15

DEVICE TYPE? VDT  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
VDT TYPE? 911  
CRU ADDRESS? (> 100) > 440  
INTERRUPT? (10)  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 4

DEVICE TYPE? VDT  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
VDT TYPE? 911  
CRU ADDRESS? (> 100) > 460  
INTERRUPT? (10)  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 14

DEVICE TYPE? VDT  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
VDT TYPE? 911  
CRU ADDRESS? (> 100) > 480  
INTERRUPT? (10)  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 7

DEVICE TYPE? VDT  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
VDT TYPE? 911  
CRU ADDRESS? (> 100) > 4A0  
INTERRUPT? (10)  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 3

DEVICE TYPE? VDT  
ACCESS TYPE? (RECORD)  
TIME OUT? (0)  
CHARACTER QUEUE? (6)  
VDT TYPE? 911  
CRU ADDRESS? (> 100) > 4C0  
INTERRUPT? (10)  
EXPANSION CHASSIS? 1  
EXPANSION POSITION? 11

```
DEVICE TYPE? VDT
ACCESS TYPE? (RECORD)
TIME OUT? (0)
CHARACTER QUEUE? (6)
VDT TYPE? 911
CRU ADDRESS? (> 100) > 4E0
INTERRUPT? (10)
EXPANSION CHASSIS? 1
EXPANSION POSITION? 12
```

The special devices are defined last in this example, but you can define them any time during the system generation process.

```
DEVICE TYPE? SD
INTERFACE TYPE? (NONE)
SPECIAL DEVICE ADDRESS? (> 100) > 540
DEVICE NAME? EM01
INTERRUPT BIT? (15)
KSB ADDRESS? (NONE)
DSR WORKSPACE? PSEM01
INTERRUPT ENTRY? EMINT
PDT FILE? .$$$SYSGEN.EMPDT
DSR FILE? .$$$SYSGEN.EMDSR
OVERRIDE? YES
INTERRUPT? (15) 10
EXPANSION CHASSIS? (1)
EXPANSION POSITION? (0) 9
```

```
DEVICE TYPE? SD
INTERFACE TYPE? (NONE)
SPECIAL DEVICE ADDRESS? (> 100) > 580
DEVICE NAME? TM01
INTERRUPT BIT? (15)
KSB ADDRESS? (NONE)
DSR WORKSPACE? PSTM01
INTERRUPT ENTRY? TMINT
PDT FILE? .$$$SYSGEN.TMPDT
DSR FILE? .$$$SYSGEN.TMDSR
OVERRIDE? YES
INTERRUPT? (15) 10
EXPANSION CHASSIS? (1)
EXPANSION POSITION? (0) 9
```

Once all devices and user-written software are included, the configuration is complete. You can now terminate the System Generation utility.

```
DEVICE TYPE? < Press the Command key>
COMMAND? STOP
BUILD? (YES)
***** CONFIGURATION FILE SAVED *****
***** D$DATA SOURCE FILE IS NOW BEING BUILT *****
***** THE LINK EDIT COMMAND STREAM SOURCE FILE IS BEING BUILT *****
***** BATCH FILE FOR SYSGEN COMPLETION IS NOW BEING BUILT *****
***** SYSGEN COMPLETE *****
***** GEN990 TERMINATED *****:
```

With successful execution of GEN990, the remaining steps of system generation are assembling and linking, patching, testing, and installing the system. Refer to Section 3 for instructions and examples of these steps.





# Appendix G

## SVC Codes

Table G-1 contains a list of the codes associated with the calls used in DX10.

**Table G-1. SVC Codes**

Call	Hexadecimal Code	Call Block Length (Bytes) Decimal
<b>I/O Calls</b>		
Perform I/O Operation	00	14
Wait for I/O	01	4
Wait for Multiple Initiate I/Os	36	4
Get Event Character	30	4
Abort I/O	0F	2
Get Event Character by LUNO	39	4
File Utility Request	15	24
<b>Program Control Calls</b>		
Execute Task	2B	12
Activate Suspended Task	07	3
Scheduled Bid Task	1F	14
Load an Overlay	14	7
End of Task	04	1
End of Program	16	1
Time Delay	02	4
Change Priority	11	2
Poll Status of Task in Terminal Task Set	35	8
Unconditional Wait	06	1
Activate Time Delay Task	0E	3
Get Parameters	17	6
Self Identification	2E	6
End Action Status	2F	10
Reset End Action	3E	4
Map Program Name to ID	31	16
Bid Task	05	8
Do Not Suspend	09	2

Table G-1. SVC Codes (Continued)

Call	Hexadecimal Code	Call Block Length (Bytes) Decimal
<b>Memory Control Calls</b>		
Get Memory	12	4
Release Memory	13	4
Get Common Data Address	10	2
Return Common Data Address	1B	1
<b>Miscellaneous Calls</b>		
Getdata	1D	12
Putdata	1C	12
Date and Time	03	8
System Log	21	7
Retrieve System Information	3F	6
<b>Data Conversion Calls</b>		
Convert Binary to Decimal	0A	8
Convert Decimal to Binary	0B	8
Convert Binary to Hexadecimal	0C	6
Convert Hexadecimal to Binary	0D	6
<b>Privileged Supervisor Calls</b>		
Install Task	25	20
Install Procedure	26	16
Install Overlay	27	18
Delete Task	28	6
Delete Procedure	29	6
Delete Overlay	2A	6
Kill Task	33	8
Suspend Awaiting Queue Input	24	1
Read/Write TSB	2C	0
Read/Write Task	2D	40
Get System Pointer Table Address	32	6
Initialize Date and Time	3B	6
Disk Manager	22	14
Assign Space on Program File	37	10
Initialize New Volume	38	24
Install Disk Volume	20	14
Unload Disk Volume	34	14
Direct Disk I/O	00	16

# Appendix H

## Bidding a Task from a DSR

---

The following is an example of task bidding from a device service routine (DSR). It should be emphasized that this is an example only. The 913 VDT does not have this capability programmed into it. This is to illustrate some of the problems encountered in programming a DSR to have tasks bid from it and to reach a possible solution.

This specific example shows how a task installed as task > 99 on the system program file would be bid whenever the command key F4 is pressed on the keyboard of a 913 VDT. Since several registers are needed to call the subroutine TMBID0, a context switch was performed using the Scheduler Stack (SHDSTK) as the workspace. The Scheduler Stack also provided the necessary stack area for the call to subroutine TMBID0. Bit 7 of the device status flags in the physical device table (PDT) was used as an indicator that a task bid is in progress. It is not recommended that this bit be used, as this bit is reserved to be defined in future releases of DX10. A feasible solution would be to define an additional word in the PDT when writing the DSR. This can be used as a flag word.

One problem that must be addressed is the handling of multiple requests to bid a task before a previous bid is complete. In this example, while the Bid Task in Progress flag is set, all subsequent requests to bid a task are ignored.

DSR913 11:38:31 WEDNESDAY, JUL 05, 1978.
DSR913 - CRT 913 HANDLER PAGE 0037

1165 \*\$0=0
1166 \* ABSTRACT:
1167 \* TDI913 - ENTRY POINT FOR TIMER DSR AND KBI'S.
1168 \* R6 CONTAINS THE INTERRUPT VECTOR.
1169 \*
1170 \*\$0=0
1171 \*
1172 0670 TDI913
1173 0670 C924 MOV @PDT\$M1(4),@PDT\$M2(4) LET TIMEOUT=RESET
0672 FFFA
0674 FFFC

1174 \*\*\*\*\*
1175 \*
1176 \* ABSTRACT:
1177 \* BIDTASK - THIS ROUTINE WILL BID A SPECIFIED TASK
1178 \* FROM THE DSR. TDI913 IS THE RE-ENTER ME VECTOR IN
1179 \* THE DSR. THIS ROUTINE CHECKS FOR A BID IN PROGRESS.
1180 \* IF THERE IS ONE, A CONTEXT SWITCH IS PERFORMED
1181 \* IN ORDER TO GAIN THE NECESSARY REGISTERS FOR THE
1182 \* CALL TO SUBROUTINE 'TMBIDO'. THE SCHEDULER STACK
1183 \* IS USED FOR A WORKSPACE.
1184 \* AFTER THE CONTEXT SWITCH IS PERFORMED, A
1185 \* STACK IS SET UP FOR THE SUBROUTINE CALL. THEN THE
1186 \* PARAMETER FOR THE CALL TO 'TMBIDO' IS SET AND THE
1187 \* SUBROUTINE IS CALLED.
1188 \* IF THERE IS AN ERROR, THE ERROR IS REPORTED
1189 \* TO THE LOG. (NOTE: ERROR PROCESSING IS DEPENDENT ON
1190 \* THE APPLICATION). IN EITHER ERROR OR NORMAL RETURN
1191 \* A RETURN FROM THE CONTEXT SWITCH IS PERFORMED, THE
1192 \* BID IN PROGRESS FLAG IS RESET, AND NORMAL PROCESSING
1193 \* IS RESUMED.
1194 \*

1195 \*\*\*\*\*
1196 REF TMBIDO TM\$ROT - BID TASK
1197 REF BYTE01 GLOBAL VARIABLE
1198 REF SHDSTK SCHEDULER STACK
1199 REF BIDTSK GLOBAL BID TASK FLAG
1200 REF TM\$DFR ACTIVATE SCHEDULER FLAG
1201 REF TMESLC TIME SLICE EXTENDED FLAG
1202 0676 C1C2 MOV R2,R7 SET FLAGS
1203 0678 0A87 SLA R7,8 BID IN PROGRESS?
1204 067A 1709 JNC TD12 NO - CONTINUE
1205 067C 0207 LI R7,SHDSTK YES-USE SCHEDULER STACK AS WSP
067E 0000
1206 0680 0208 LI R8,BID1 SET UP PROGRAM COUNTER
0682 0696
1207 0684 0407 BLWP R7 PERFORM CONTEXT SWITCH
1208 \* RETURN FROM CONTEXT SWITCH
1209 0686 0620 DEC @BIDTSK RESET GLOBAL BID IN PROGRESS
0688 0000



```

DSR913                               11:38:31 WEDNESDAY, JUL 05, 1978.
DSR913 - CRT 913 HANDLER                               PAGE 0039

1252 06C6          KBI913
1253 06C6 04CA          CLR R10
1254 06C8 35CA          STCR R10,7          ENTER INTO CHAR(7-BIT)
1255 06CA 1F14          TB KBPERR-8          IF PARITYERR EQ 1 THEN
1256 06CC 1602          JNE KBIC
1257 06CE 020A          LI R10, 7F00          LET CHAR=DELETE          END
      06D0 7F00
1258 06D2 1D11          KBIC SB0 KBACK-8          ! ACK INTERRUPT
1259 *****
1260 *
1261 *          CHECK FOR BID TASK INTERRUPT
1262 *          AT THIS POINT, A CHECK IS MADE TO SEE IF THE 'F4'
1263 *          HAS BEEN TYPED IN, INDICATING THAT TASK 99 IS TO BE
1264 *          BID UP. THE 'F4' KEY SENDS AN ASCII CODE OF 75.
1265 *          A CHECK IS MADE TO SEE IF A BID TASK IS ALREADY
1266 *          IN PROGRESS. IF SO, THIS REQUEST IS IGNORED. (NOTE:
1267 *          HANDLING OF MULTIPLE BID TASK REQUESTS ON THE SAME
1268 *          DEVICE IS APPLICATION DEPENDENT). OTHERWISE, THE
1269 *          LOCAL AND GLOBAL TASK FLAGS ARE SET. SCHEDULING
1270 *          IF FORCED IF POSSIBLE.
1271 *****
1272 06D4 028A          CI R10, 7500          BID TASK REQUESTED?
      06D6 7500
1273 06D8 160F          JNE KBIB          NO - CONTINUE
1274 06DA C250          MOV *R0,R9          NES-WORKSPACE POINTER (PDT ADDR
1275 06DC C1E9          MOV @PDTDSF(R9),R7  BID TASK ALREADY IN PROGRESS?
      06DE 0008
1276 06E0 0A87          SLA R7,8
1277 06E2 180A          JOC KBIB          YES - IGNORE THIS REQUEST
1278 06E4 FA60          SOCB @BYTE01,@PDTDSF(R9)NO - SET LOCAL BID TASK FLA
      06E6 068C
      06E8 0008
1279 06EA 05A0          INC @BIDTSK          SET GLOBAL BID TASK FLAG
      06EC 0688
1280 06EE C1E0          MOV @TMESLC,R7          TIME SLICE EXTENDED OF EXECUTING
      06F0 0000
1281 06F2 1602          JNE KBIB          TASK? YES, CONTINUE
1282 06F4 0720          SETO @TM$DFR          NO, FORCE SCHEDULING NOW
      06F6 0000
1283 *****
1284          06F8^ KBIB EQU $
1285 06F8 C186          MOV R6,R6          IF NOT IN CHARACTER MODE
1286 06FA 1131          JLT KBIW
1287 06FC 06A0          BL @KEYFUN          CALL KEYFUN(R10,RESET,KILL,
      06FE 0000
1288 0700 700D          DATA >700D,>6100,KBIU          BID,00,KBIU)
      0702 6100
      0704 0754^
1289 0706 028A          CI R10,>3F00          IF CHAR LT 3F THEN
      0708 3F00

```

DSR913  
DSR913 - CRT 913 HANDLER

11:38:31 WEDNESDAY, JUL 05, 1978  
PAGE 0040

```

1290 070A 1BOB      JH  KBII
1291 070C 0209      LI  R9,-LOWEND-2    LET COUNT=-LOWEND-2
      070E FFDC
1292 0710 05C9      KBIF INCT R9          REPEAT LET COUNT=COUNT+2
1293 0712 130E      JEQ  KBIL          UNTIL COUNT EQ 0
1294 0714 92A9      CB  @MAPLOW+LOWEND(9),R10 IF MAPLOW(COUNT)EQ CHAR
      0716 0796
1295 0718 1BOB      JH  KBIL
1296 071A 16FA      JNE  KBIF
1297 071C D2A9      MOVB @MAPLOW+LOWEND+1(9),R10 LET CHAR=MAPLOW(COUNT)
      071E 0797
1298 0720 1007      JMP  KBIL                      END
1299 0722 028A      KBII CI  R10,>5A00    ELIF CHAR GT 5A THEN
      0724 5A00
1300 0726 1204      JLE  KBIL
1301 0728 C24A      MOV  R10,R9          LET CHAR=REMAP(CHAR)
1302 072A 0989      SRL  R9,8
1303 072C D2A9      MOVB @REMAP->5B(9),R10      END END
      072E 073C
1304 0730 028A      KBIL CI  R10,>7F00    IF CHAR GT CLEAR
      0732 7F00
1305 0734 120A      JLE  KBIO
1306 0736 028A      CI  R10,LO$EDT    AND (CHAR LT LO$EDT)
      0738 8800
1307 073A 1A03      JL   KBIM
1308 073C 028A      CI  R10,HI$EDT    OR CHAR GT HI$EDT) THEN
      073E 9500
1309 0740 1204      JLE  KB10
1310 9742 06A0      KBIM BL  @PUTEBF      CALL PUTEBF(R10,ERR=KBIR) END
      0744 0000
1311 0746 0752      DATA KBIR          !PUT CHAR IN EVENT BUFFER
1312 0748 1005      JMP  KBIU                      END
1313 074A 06A0      KBIO BL  @PUTCBF      CALL PUTCBF(R10)
      074C 0000
1314 074E 0752      DATA KBIR          !PUT CHAR IN DATA BUFFER
1315 0750 1001      JMP  KBIU
1316 0752 1D07      KBIR SBO BEEEEP-8    DISPLAY .BEEP
1317 0754 C250      KBIU MOV  *R0,R9          CALL DSR
1318 0756 FA60      SOCB @BYTE04,@PDTDSF(9) !SET RE-ENTER-ME FLAG
      0758 0694
      075A 0008
1319 075C 0380      KBIZ RTWP          RETURN
1320 *
1321 075E      KBIW EQU  $
1322 075E 06A0      BL  @CMODE          CALL CHARACTER MODE ROUTINE
      0760 0000
1323 0762 0752      DATA KBIR
1324 0764 10F7      JMP  KBIU

```

```

1326          * 0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=
1327          *          WAIT:
1328          *          THIS ROUTINE DOES THE TIMING LOOP FOR ALL
1329          *          CURSOR MOVES AND STROBES. IF THIS IS A
1330          *          /10 MACHINE, THE POWER UP CODE WILL MODIFY
1331          *          THE ROUTINE BY PLACING THE RT INSTRUCTION
1332          *          AT THE START OF THE MODULE.
1333          * 0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=0=
1334          0766  WAIT      EQU      $
1335          0766  C28A      MOV      R10,R10          TIMING LOOP,1 INSTR=550 NANOS
1336          0768  C28A      MOV      R10,R10
1337          076A  C28A      MOV      R10,R10
1338          076C  C28A      MOV      R10,R10
1339          076E  C28A      MOV      R10,R10
1340          0770  C28A      MOV      R10,R10
1341          *NOTE:  THE RETURN WILL BE MOVED AFTER "WAIT" ON THE /10
1342          0772  045B  WAITRT  RT
1343          *
1344          0774  028C  MAPLOW  DATA >028C,>0888,>0989,>0A8A
1345          0776  0888
1345          0778  0989
1345          077A  0A8A
1345          077C  0D8D          DATA >0D8D,>1A95,>1C92,>233A
1345          077E  1A95
1345          0780  1C92
1345          0782  233A
1346          0784  243C          DATA >243C,>262F,>2F8F,>8A8B
1346          0786  262F
1346          0788  2F3F
1346          078A  3A3B
1347          078C  3B8B          DATA >3B8B,>3C7F,>3D91,>3E90
1347          078E  3C7F
1347          0790  3D91
1347          0792  3E90
1348          0794  3F5E          DATA >3F5E
1349          0022  LOWEND  EQU      $-MAPLOW
1350          0796  00          BYTE  00
1351          0797  8F          REMAP  BYTE          >8F,>94,>8E,>00,>00 5B - 5F
1351          0798  94
1351          0799  8E
1351          079A  00
1351          079B  00
1352          079C  40          BYTE  >40,>21,>22,>23,>24,>3D,>26,>27 60 - 67
1352          079D  21
1352          079E  22
1352          079F  23
1352          07A0  24
1352          07A1  3D
1352          07A2  26
1352          07A3  27

```



DSR913  
DSR913 - CRT 913 HANDLER

11:38:31 WEDNESDAY, JUL 05, 1978.

PAGE 0042

1353	07A4	28	BYTE >28,>29,>2A,>3E,>00,>00,>00,>00	68 - 6F
	07A5	29		
	07A6	2A		
	07A7	3E		
	07A8	00		
	07A9	00		
	07AA	00		
	07AB	00		
1354	07AC	9B	BYTE >9B,>80,>81,>82,>83,>84,>85,>86	70 - 77
	07AD	80		
	07AE	81		
	07AF	82		
	07B0	83		
	07B1	84		
	07B2	85		
	07B3	86		
1355	07B4	96	BYTE >96,>98,>9E,>9F,>93,>99,>00,>7F	78 - 7F
	07B5	98		
	07B6	9E		
	07B7	9F		
	07B8	93		
	07B9	99		
	07BA	00		
	07BB	7F		
1357			END	

NO ERRORS



# Appendix I

## DX10 Memory Estimator

---

### I.1 INTRODUCTION

This appendix contains the DX10 memory estimator and an explanation of the entries of the memory estimator. This appendix concludes with a paragraph of hints on determining the number of terminals that a particular generated system can support.

The memory estimator and accompanying material are guides to aid in configuring a system. Cautions are provided about terminal capacity and application's requirements; the final test is whether the system can be successfully generated and run. Results derived from the memory estimator should not be construed as an assurance by Texas Instruments that a given configuration will or will not work.

### I.2 DX10 MEMORY ESTIMATOR

The memory estimator is used for four purposes:

- To estimate the total physical memory requirements of a system.
- To determine that the logical address space requirements for a given configuration do not exceed > F800.
- To determine that the system root does not exceed 48K bytes in size.
- To estimate system table area capacity and requirements. The DX10 Memory Estimator is a tool to aid the calculation of a system's total length and the allocation of system memory in terms of operating system overhead and space for application programs.

Advance through the items of the memory estimator entering values that reflect your system's configuration. The SIZE column gives the size of an item used by DX10. The QTY column is available for you to write in how many of a particular thing you have in your system. If a 1 is written into the table, or if an item is blank, then DX10 only requires one of the item. The ITEM SIZE column is available for you to write in the result of your SIZE times QTY calculation. On some of the items, a blank space is included in the SIZE column for you to write in a subtotal. Constants appear in the column in which the other numbers you write in should be added. The SUM OF ITEMS column contains values obtained by computing sums and differences of other items. Constants used in computing such sums and differences appear in that column. Paragraph I.3 provides an explanation of items listed in the memory estimator.

DX10 Memory Estimator

ITEM NO.	DESCRIPTION	SIZE	X	QTY	ITEM SIZE	SUM OF ITEMS	REQUIRED (YES/NO)
----------	-------------	------	---	-----	-----------	--------------	-------------------

-----  
 \*\* ALL NUMBERS ARE IN DECIMAL BYTES \*\*  
 -----

MAP SEGMENT 1

REQUIRED PARTS AND DEVICE SUPPORT

1.	OPERATING SYSTEM	12276				12276	YES
2.	POWERFAIL SUPPORT	520					NO
3.	DISK(S)	186	x	_____	=	_____	YES
4.	DISK CONTROLLER(S)	32	x	_____	=	_____	YES
5.	911/913 VDT(S)	160	x	_____	=	_____	NO
6.	931/940 VDT(S)	174	x	_____	=	_____	NO
7.	931/940 LINE PRINTER	86	x	_____	=	_____	NO
8.	LINE PRINTER (CI403 OR CI404)	150	x	_____	=	_____	NO
9.	LINE PRINTER (SERIAL)	100	x	_____	=	_____	NO
10.	LINE PRINTER (PARALLEL)	238	x	_____	=	_____	NO
11.	733 ASR W/CASS(S)	324	x	_____	=	_____	NO
12.	733 KSR(S)	156	x	_____	=	_____	NO
13.	820 KSR	160	x	_____	=	_____	NO
14.	TELEPRINTER DEVICE	178	x	_____	=	_____	NO
15.	MAG TAPE (979 OR CARTRIDGE)	174	x	_____	=	_____	NO
16.	MAG TAPE CONTROLLER(S)	32	x	_____	=	_____	NO
17.	DISKETTE(S) FD800	108	x	_____	=	_____	NO
18.	DISKETTE CONTROLLER(S)	34	x	_____	=	_____	NO
19.	CARD READER(S)	100	x	_____	=	_____	NO
20.	IBM 3270 STATION	160	x	_____	=	_____	NO
21.	3270 LINE PRINTER	170	x	_____	=	_____	NO
22.	IBM 3780 COMM CHANNEL	160	x	_____	=	_____	NO
23.	IBM 2780 COMM CHANNEL	160	x	_____	=	_____	NO
24.	FILE MANAGERS	366	x	_____	=	_____	YES
25.	SYSTEM OVLY AREA(S)	806	x	_____	=	_____	YES
26.	SUM OF ITEMS 1 THRU 25					_____	

SYSTEM TABLE AREA

27.	911/913 VDT BUFFERS (QTY FROM ITEM 5)	86	x	_____	=	_____	NO
28.	931/940 VDT BUFFERS (QTY FROM ITEM 6)	86	x	_____	=	_____	NO
29.	LP BUFFERS (QTY FROM ITEMS 7+8+9+10)	512	x	_____	=	_____	NO
30.	733 ASR BUFFERS (QTY FROM ITEM 11)	258	x	_____	=	_____	NO

31.	733 KSR BUFFERS (QTY FROM ITEM 12)	86	x	_____	=	_____		NO
32.	820 BUFFERS (QTY FROM ITEM 13)	160	x	_____	=	_____		NO
33.	TELEPRINTER BUFFERS (QTY FROM ITEM 14)	512	x	_____	=	_____		NO
34.	DISKETTE BUFFERS (QTY FROM ITEM 17)	256	x	_____	=	_____		NO
35.	CARD READER BUFFERS (QTY FROM ITEM 19)	160	x	_____	=	_____		NO
36.	SUM OF ITEMS 27 THRU 35						_____	
37.	ADD. I/O BUFFERS						_____	NO
38.	SYSTEM TABLE CONSTANT	2200				2200		YES
39.	ACTIVE FOREGROUND TASKS	600	x	_____	=	_____		YES
40.	ACTIVE BACKGROUND TASKS	1000	x	_____	=	_____		NO
41.	SYSTEM TABLE SIZE (SUM OF ITEMS 38, 39, AND 40)						_____	
42.	TOTAL SYSTEM TABLE SIZE (SUM OF ITEMS 36, 37, AND 41)						_____	
43.	SYSTEM ROOT LIMIT						49152	
44.	ROOT SIZE (SUM OF ITEMS 26 AND 42)						_____	
45.	DIFFERENCE OF ITEM 43 MINUS ITEM 44						_____	
MAP SEGMENT 2								
46.	DEVICE I/O SUPPORT	32				32		YES
47.	COMM I/O SUPPORT TABLES	1160	x	1	=	_____		NO
48.	IBM 3270 STATION	75	x	_____	=	_____		NO
49.	IBM 2780/3780 COMM SUPPORT TABLES	600	x	_____	=	_____		NO
50.	I/O COMMON SUPPORT (SUM OF ITEMS 46 THRU 49)						_____	

MAP SEGMENT 3

51.	SVC SUPPORT	7688				YES
52.	ONLINE DIAG SUPPORT	192				NO
53.	SVC 1 SUPPORT					
	(SUM OF ITEMS 51 AND 52)	_____				YES
54.	SVC 2 SUPPORT	2804			2340	YES
55.	DX10 2.2 SUPPORT	200				NO
56.	USER SVCS					NO
57.	LARGEST OF ITEMS					
	53, 54, 55, AND 56					_____
58.	DISK	2976				YES
59.	ONLINE DIAG SUPPORT	288				NO
60.	AUTO MEDIA CHANGE	416				NO
61.	DISK SUPPORT (SUM OF					
	ITEMS 58, 59, AND 60)	_____	x	1 =	_____	YES
62.	911 VDT	3936	x	1 =	_____	NO
63.	913 VDT	2112	x	1 =	_____	NO
64.	LINE PRINTER ON EIA/TTY	1088	x	1 =	_____	NO
65.	733 ASR W/CASS	2864	x	1 =	_____	NO
66.	733 KSR (NO ASR)	1808	x	1 =	_____	NO
67.	MAG TAPE (979 OR					
	CARTRIDGE)	1376				NO
68.	ONLINE DIAG SUPPORT	96				NO
69.	MAG TAPE SUPPORT					
	(SUM OF ITEMS 67 AND 68)	_____	x	1 =	_____	NO
70.	LP ON 403/404	4378	x	1 =	_____	NO
71.	940/931 ON CI401	11092	x	1 =	_____	NO
72.	940/931 ON 9902	11146	x	1 =	_____	NO
73.	940/931 ON CI403/4	11114	x	1 =	_____	NO
74.	DISKETTE	1174	x	1 =	_____	NO
75.	CARD READER	952	x	1 =	_____	NO
76.	820 TERMINAL	1986	x	1 =	_____	NO
77.	TELEPRINTER DEVICE	5696	x	1 =	_____	NO
78.	IBM 3270 COMM LINK	9000	x	1 =	_____	NO
79.	IBM 3780 COMM LINK	9000	x	1 =	_____	NO
80.	IBM 2780 COMM LINK	9000	x	1 =	_____	NO
81.	911 VDT POWERFAIL	3936	x	1 =	_____	NO
82.	LARGEST DSR (OF ITEMS					
	61 THRU 66, 69 THRU 81)					_____
83.	LARGEST SEGMENT 3					_____
	(OF ITEMS 57 AND 82)					_____
84.	911 POWERFAIL					
	SCREEN BUFFERS					
	(QTY FROM ITEM 5)	2080	x	_____ =	_____	
85.	TOTAL SEGMENT 3 NEEDS					
	(SUM OF ITEMS 53 THRU					
	56, 61 THRU 66,					
	69 THRU 81, AND 84)					_____

## FILE MANAGEMENT (COMBINED WITH MAP SEGMENT 1)

86.	FILE MANAGEMENT	4160		4160		YES
87.	KEY INDEXED FILES	5354				NO
88.	FILE MANAGER SUPPORT (SUM OF ITEMS 86 AND 87)			<u>          </u>		
89.	LARGEST PHYSICAL RECORD ON DISK			<u>          </u>		YES
90.	SIZE OF COMMON			<u>          </u>		NO
91.	MEM RES SYS TASKS	4412			4672	YES
92.	TOTAL ADDRESS SPACE FOR FILE MANAGER	65440			65440	
93.	SUM OF ITEMS 44 AND 88			<u>          </u>		
94.	DIFFERENCE OF ITEM 92 MINUS ITEM 93			<u>          </u>		
95.	DX10 LIMIT	63488			63488	
96.	SIZE OF LONGEST OVERLAY PATH (SUM OF ITEMS 44, 50, AND 83) IF ITEM 83 IS 931 DSR, USE 46 INSTEAD OF 50.			<u>          </u>		
97.	DIFFERENCE OF ITEM 95 MINUS ITEM 96			<u>          </u>		
98.	SMALLER OF ITEMS 45 AND 94			<u>          </u>		
99.	LP PDT EXTENSION (CI403 OR CI404; QTY FROM ITEM 8)	192	x	<u>          </u>	=	
100.	931/940 SCREEN BUFFERS (QTY FROM ITEM 6)	2112	x	<u>          </u>	=	
101.	TOTAL SIZE OF MEMORY-RESIDENT TASKS AND PROCEDURES ON SSPROGA (YOU MUST INCLUDE ALL, INCLUDING THOSE MEMORY- RESIDENT AT THE FACTORY.)			<u>          </u>		
102.	INTERTASK BUFFERS			<u>          </u>		NO
103.	GENERATED DISK(S)	480	x	<u>          </u>	=	YES
104.	INTERRUPT HANDLER FOR 940/931 ON CI401	3302				NO
105.	INTERRUPT HANDLER FOR 940/931 ON CI402	3356				NO
106.	INTERRUPT HANDLER FOR 940/931 ON CI403	3324				NO
107.	SUM OF ITEMS 44, 50, 85, 88, 89, 90, 91, 99, 100, 101, 102, 103, 104, 105, AND 106			<u>          </u>		

### **I.3 EXPLANATION OF ITEMS OF THE MEMORY ESTIMATOR**

This paragraph provides an additional explanation of the line items on the DX10 Memory Estimator.

#### **ITEM 1. OPERATING SYSTEM**

This is the base portion of the operating system. It contains the anchors for the data structures and the common routines that are necessary for the operating system to run.

#### **ITEM 2. POWERFAIL SUPPORT**

If powerfail is selected, this number represents the routines necessary to provide that function.

#### **ITEM 3 THROUGH ITEM 23. PHYSICAL DEVICE SUPPORT**

These line items cover the memory requirements for the anchors of the actual physical devices that are included in the system. The QTY value should be the number of devices included during system generation. The note, "931/940," means that the number of 931 terminals is added to the number of 940 terminals and the result entered in the QTY column.

#### **ITEM 24. FILE MANAGERS**

The file management task processes all the file I/O requests. By including additional copies of this task, several file I/O requests may be processed concurrently, yielding faster overall processing of requests. A suggested value for this item is 2 tasks. The quantity entered for this item is the response to the system generation prompt FILE MANAGEMENT TASKS:(2). At least one must be provided.

#### **ITEM 25. SYSTEM OVERLAY AREA(S)**

DX10 uses overlays. The number of overlay areas affects the response time for non-memory resident functions of DX10. The smaller number of overlay areas requires more disk accesses to perform an operation that uses the system overlay area. A suggested minimal value for this item is 2 overlay areas. The quantity entered for this item is the answer to the system generation question OVERLAYS:(2). At least one must be provided.

#### **ITEM 26. SUM OF THE BASE OPERATING SYSTEM SUPPORT**

This item represents the base operating system tables for devices and necessary system sub-routines. Factors not included in this number are the table space for task support, additional I/O buffers, and intertask buffer area.

#### **ITEM 27 THROUGH ITEM 35. CRU DEVICE BUFFERS**

These items represent the estimated amount of table area needed to process I/O requests for devices interfaced through the CRU. The QTY should be the number of devices of the specified type(s) included in the system.

#### **ITEM 36. SUM OF CRU DEVICE BUFFERS**

This number represents the total estimated needs for device buffers. System generation will automatically include this amount in the total system table area.



**ITEM 37. ADDITIONAL I/O BUFFERS**

The buffers for I/O operations through physical devices are part of the system table area and support CRU device I/O, not disk or tape I/O of any sort. (Note that FD800 diskettes and ASR cassette tapes are CRU devices.) A value is needed here if communications or special devices are used, or if an application will be run that calls Initiate I/O frequently. This number is the correct response to the system generation prompt "I/O BUFFERS? (0)." Note that GEN990, the System Generation utility, wants this value expressed in bytes, as it is in this item.

**ITEM 38. SYSTEM TABLE CONSTANT**

A certain amount of system table space is required just to perform an initial program load (IPL) on the system. It is difficult to accurately define a simple equation for determining the dynamic table area required. The equation  $X = 2200 + N * 600 + B * 1000$  has been empirically defined using multiple terminals doing simple text edits and assembly language assemblies. ITEM 39 supplies the constant of the equation while ITEM 40 determines the  $N * 600$  term and ITEM 40 determines the  $B * 1000$  term.

**ITEM 39. ACTIVE FOREGROUND TASKS**

This item is the crux of the table area calculation. An active foreground task in the DX10 operating system is a process which requires operator interaction and does not allow other tasks to have operator interaction at the same time. This item is an estimate of the amount of system table area required to support an active foreground task. The true requirements depend on the specific application environment. Any given application could require more or less, depending on the number of tasks, open files, assigned LUNOs, and locked records that it uses. For full support of all terminals, the quantity specified for this item should be equal to the sum of the quantity of items 5, 6, 11, 12, 13, and 14, which is the number of terminals. A number less than this sum may be used, and DX10 will enforce this specified amount. In systems where less than 100% of the terminals are active at any given time, specifying a number less than the total number of terminals on the system may be satisfactory. If the maximum number of terminals is active and a user attempts to activate another terminal, the DX10 operating system will reject the request with a message informing the user that he cannot be supported at this time. The quantity entered for this item is the answer to the system generation question SCI FOREGROUND? (8).

**ITEM 40. ACTIVE BACKGROUND TASKS**

An active background task in the DX10 operating system is a process associated with a terminal that is not a foreground process (that is, a process executed via an SCI .QBID primitive). This item is an estimate of the amount of system table area required to support an active background task. As with the estimate for the foreground task described in ITEM 39, the true requirement depends on specific details of the background process. The quantity entered for this item is the answer to the system generation question SCI BACKGROUND? (2). You should carefully evaluate whether you need more than the default value. Most systems run satisfactorily with 2 backgrounds, and more than that will allow backgrounds to use more table area, leaving less to support foreground activities.

**ITEM 41. SYSTEM TABLE SIZE (SUM OF ITEMS 38, 39, AND 40)**

This is the dynamic table area out of which the operating system allocates work areas to service requests from active tasks in the system. This item is a composite of estimates, so the result given by the estimator may vary from the actual use by the running system. To provide a better number for system generation, it is necessary to monitor usage of the system table area on a running system. This can be done by using the Show Memory Status (SMS) command. This number is the answer to the system generation prompt TABLE?. The number in this item is expressed in bytes, so you must convert to words (by dividing by two) to specify it during system generation.

**ITEM 42. TOTAL SYSTEM TABLE AREA (SUM OF ITEMS 36, 37, AND 41)**

This item is the total space allocated for the system table area.

**ITEM 43. SYSTEM ROOT LIMIT**

The system tasks of DX10 are loaded at address 49152 (> C000). If the root portion of DX10 is larger than 49152, some overlapping of address space occurs with unpredictable results. To prohibit error conditions, the root of the DX10 operating system is limited to less than 49152 bytes in total length.

**ITEM 44. ROOT SIZE (SUM OF ITEMS 26 AND 42)**

The base operating system table support and the system table area comprise the system root.

**ITEM 45. DIFFERENCE OF ITEM 43 MINUS ITEM 44**

Subtract item 44 from item 43. The result, if positive, represents the amount by which the system root is less than the 49152-byte limit. If this number is negative, the system will exceed the allowed root size and either the system table area or the base operating system support tables must be decreased. The system generation process (PGS) will not allow the system root to exceed the 49152-byte upper limit.

**ITEM 46. DEVICE I/O SUPPORT**

DX10 provides common routines that support device I/O. This number includes the size of the routines necessary to support physical devices, the clock, and error interrupts.

**ITEM 47. COMMUNICATIONS I/O SUPPORT TABLES**

The IBM 3270 and/or 2780/3780 Communications Emulator package under DX10 requires that a module COMMCOM be included in the DEVICE I/O SUPPORT phase of the system structure. Details concerning the COMMCOM are documented in the communications package manuals. COMMCOM contains data structures and code to interface between the communications physical device I/O support and the standard DX10 I/O modules.

As a word of caution, generating a DX10 operating system with the 3270, 3780 and 2780 communications packages or combinations of a pair of the packages does not result in a memory size requirement equal to the sum of the memory required for individual packages. Some modules are shared and some modules reside in different portions of DX10 when two or more packages exist.

**ITEM 48. IBM 3270 STATION**

For each IBM 3270 emulated station (either a printer or VDT), table area is required.

**ITEM 49. IBM 2780/3780 COMMUNICATIONS SUPPORT TABLES**

Table areas are required for each IBM 2780/3780 communications link supported by the DX10 operating system.

**ITEM 50. I/O COMMON SUPPORT (SUM OF ITEMS 46 THROUGH 49)**

Routines common to all I/O requests for all files and devices are represented in this number and are referred to as I/O COMMON.

**ITEM 51. SUPERVISOR CALL (SVC) SUPPORT**

DX10 provides routines that are used to support SVCs. This number includes the size of the routines that support the standard SVCs used by DX10 software.

**ITEM 52. ONLINE DIAGNOSTICS SUPPORT**

If Online Diagnostics Support is requested, routines needed by DX10 and its DSRs are represented in this number. If Online Diagnostics Support is requested, you must include items 59 and 68.

**ITEM 53. SVC 1 SUPPORT (SUM OF ITEMS 51 AND 52)**

The total size of one of the SVC support segments required by DX10 is computed in this item.

**ITEM 54. SVC 2 SUPPORT**

DX10 routines necessary to support SVCs not supported in item 53 are represented by this number.

**ITEM 55. DX10 2.2 SUPPORT**

Routines necessary to support the character mode SVCs are represented in this number. Include them if you selected these SVCs in your system.

**ITEM 56. USER SVCS**

The total size of all modules included for user-written SVCs must be represented in this number. If there are none, enter zero.

**ITEM 57. LARGEST OF ITEMS 53, 54, 55, AND 56**

This item selects the largest of the modules used for SVC support. It is used in calculating the longest overlay path of the system.

**ITEMS 58 THROUGH 81. PHYSICAL DEVICE SERVICE ROUTINES (DSRS)**

These values are the sizes for the standard DX10 supported devices. Only one copy of the necessary DSR will be included in the system even when more than one physical device of the same type is connected to the system. The DISK term includes the DSDD of the FD1000 and the WD500 drives, while the DISKETTE term is used only for the FD800 drive.

The communications physical device support for DX10 contains modules to control a communications link using the IBM binary synchronous communications line discipline. They interface with standard DX10 I/O support via specialized modules included in the DEVICE I/O SUPPORT portion of DX10. One DSR module will support multiple communications links of the same type. Refer to the communications package manuals for further details.

The 733 terminal uses the same keyboard and printer handler routine whether cassettes are present or not. Therefore only one of ITEM 65 or ITEM 66 should be included in a system estimate.

The 911 support either includes the powerfail option or does not. The powerfail option cannot be chosen for some 911 terminals in the system and rejected for others. If powerfail is selected, see item 81 for more information.

**ITEM 81. 911 VDT POWERFAIL**

If powerfail is selected and there are any 911 terminals included in the system, include this item in the sum of line 85. You must also omit item 62 and include item 84.

**ITEM 82. LARGEST DSR (OF ITEMS 61 THROUGH 66 AND 69 THROUGH 81)**

The manner in which DX10 is constructed allows each DSR to be in the last segment of the map file describing the overlay of DX10 containing the DSR. The length of DX10 actually describes the longest overlay of DX10. The longest DSR will determine the longest overlay. If special devices (SD to system generation) are to be included, check the length of your longest special device DSR against the value obtained for this item, and substitute it if it is longer than any of the standard DX10 DSRs.

**ITEM 83. LARGEST SEGMENT 3 (OF ITEMS 57 AND 82)**

This number is used to compute the longest overall address path that must be used by DX10.

**ITEM 84. 911 POWERFAIL SCREEN BUFFERS**

If powerfail recovery is selected, 911 terminals require a screen buffer. The number of 911 terminals included during system generation (quantity of item 5) is entered in the QTY column to compute the total amount of buffer space needed.

**ITEM 85. TOTAL SEGMENT 3 NEEDS**

This total is the amount of memory needed to hold the DSRs, SVC support, and any screen buffers needed. This number is not part of the longest overlay path.

**ITEM 86. FILE MANAGEMENT SUPPORT**

DX10 supports different file types: relative record, sequential, program, and image. This item is the size required to support the standard files DX10 uses.

**ITEM 87. KEY INDEXED FILES (KIFS)**

DX10 optionally supports key indexed files (KIFs). If you select KIF support, include this item and respond YES to the system generation prompt KIF? (NO) to include KIF logic in the newly generated system.

**ITEM 88. FILE MANAGER SUPPORT (SUM OF ITEMS 86 AND 87)**

This total is the amount of memory required to hold the total file management support.

**ITEM 89. LARGEST PHYSICAL RECORD ON DISK**

DX10 requires a memory-resident buffer as a backup buffer for handling blocked files. This item reflects the largest physical record size of all blocked files that will be used in your system. This number is entered for the system generation parameter BUFFER MANAGEMENT: (1K) and should only be the size of the largest blocked physical record.

**ITEM 90. SIZE OF COMMON**

The task common area is a region of system memory accessible by all tasks. It is used for communication and coordination between tasks and its size is directly dependent on the needs of the application tasks using this area. This is an optional item. Determine the length by inspecting the assembly listing for the object module you have included for the prompt during system generation.

**ITEM 91. MEMORY-RESIDENT SYSTEM TASKS**

DX10 requires some tasks to be in memory at all times. These tasks perform specific functions necessary to the operation of the operating system. This item reflects the size of these tasks, which are linked into the memory-resident portion of the system.

**ITEM 92. TOTAL ADDRESS SPACE FOR FILE MANAGER**

This number represents the practical limit on address space usage by the file manager task.

**ITEM 93. SUM OF ITEMS 44 AND 88**

This item represents the total address space occupied by the system root and the routines necessary to support file management functions.

**ITEM 94. DIFFERENCE OF ITEM 92 MINUS ITEM 93**

Subtract item 93 from item 92. The result represents the total amount of address space available to the file manager for handling blocking of user records in and out of physical records. It can be divided into two parts, one for the logical record and one for the physical record, in any ratio needed by the application. However, large physical records (over half the space available) can create problems with utilities such as CKS, BD, and RD. Therefore, it is wise to plan your physical records to be, at most, half this number. Also consider that future system generations may reduce this number. See Volume III of this set for more information on physical records, and Section 4 of this manual for more information on how the file manager handles available address space.

**ITEM 95. DX10 LIMIT**

DX10 is bounded by hardware limitations. The number given here is the maximum address usable by a DSR or SVC segment.

**ITEM 96. SIZE OF LONGEST OVERLAY PATH (SUM OF ITEMS 44, 50, AND 83)**

With the structure of DX10, the maximum size that DX10 can become depends on the length of the longest overlay path and not on the total sum of all the parts. Therefore, the DX10 operating system can occupy more than the address space limit of 63,488 bytes. This item defines the combined length of the first, second, and the longest of the third segments of DX10 that gives the longest overlay path. (The longest overlay path cannot exceed 63,488 bytes.)

**ITEM 97. DIFFERENCE OF ITEM 95 MINUS ITEM 96**

Subtract item 96 from item 95. The result, if positive, is the amount of memory by which the longest overlay path could be larger. If negative, the system must be made smaller by reducing items 3 through 40 and items 46 through 50, or identify and remove the item that defined item 83 so that it becomes smaller.

**ITEM 98. SMALLER OF ITEMS 45 AND 94**

The system table area can be made larger by the smaller of these two numbers. If either of the numbers included here is negative, the system will have to be made smaller. Bear in mind that these are estimates, and that if it is within several hundred bytes of zero only a trial system generation will inform you whether the system can be made to work. The PGS command produces an error message if the system is too large.

**ITEM 99. LP PDT EXTENSION (CI403 OR CI404)**

Any serial line printer interfaced through a CI403 or CI404 TILINE multiplexing controller will have a PDT extension that is not included in any of the mapped parts of the operating system. This item includes them in the total size of the memory-resident portion of the operating system.

**ITEM 100. 931/940 SCREEN BUFFERS**

The screen buffers for handling 931 and 940 terminals do not appear in any of the mapped portions of the system. Incorporate here the number of 931 and 940 terminals included in the system generation. The buffers are included in the total memory-resident portion of the operating system.

**ITEM 101. TOTAL SIZE OF MEMORY-RESIDENT ROUTINES**

This item includes the size of all tasks and procedures marked memory-resident in the system program file. A standard DX10 has one memory-resident task and two memory-resident procedures. These and any additional procedures and tasks that you make memory resident become part of the static memory size displayed by the Show Memory Map (SMM) command and should be added before comparing actual total memory with predicted total memory. To compute the amount of memory needed by a memory-resident segment, determine its length from a Map Program File (MPF) command on the system program file, add >3F, and then round down to the next lower integer value that is evenly divisible by >20. For example, a task listed as >3A8 bytes in length will occupy >3E0 bytes of memory when loaded, as shown by the following formula:

$$((>3A8 + >3F) / >20) * >20 = >3E0$$

**ITEM 102. INTERTASK BUFFERS**

The intertask communication buffers, different than COMMON, are not taken from the system table area. This number is the maximum bytes of memory that the DX10 operating system can use for intertask communications buffers. If you are using TIFORM or Sort/Merge, refer to the appropriate manual. This number is the response to the system generation prompt INTERTASK? (100). Note that the value included here must be in bytes and that for GEN990 it must be expressed in words.

**ITEM 103. GENERATED DISK(S)**

The information necessary to access a disk when it is installed is maintained in the system table area and in a separate disk information area. System table area needs are included in the estimates for items 39 and 40. This item includes the space needed for bit maps which are not stored in the system table area. Memory is set aside by the system loader for each disk (except FD800) included in the system generation. The quantity entered here is the total number of disks generated.

**ITEM 104 THROUGH 106. INTERRUPT HANDLING**

This is the interrupt handling portion of the respective terminal support routine. It is required when that device is defined to the system during system generation.

**ITEM 107. SUM OF ITEMS 44, 50, 85, 88, 89, 90, 91, 99, 100, 101, 102, AND 103.**

This is the total memory-resident size of the system. Subtract this number from the total available memory on the computer. The result is the amount of memory available for user tasks or for SCI. If the result is small enough to cause excessive rollin/rollout when running the application programs, response time will be adversely affected. Do not include tasks that will be rolled out most of the time the application will be running, such as SCI. If response is slow because of rollin/rollout, more memory will help.

Response time is also affected by the compute load imposed by the application program. The only helps for this condition are to reduce the compute load and/or get a faster processor.

The estimator can be expected to yield an estimate of system size that is accurate within 300 bytes most of the time. In the above items, reference is made to limits not being exceeded. In practice, you will find that if the estimator shows that the system is too large, the system may still be successfully generated. However, this is an important clue that the system's limits are being reached; special attention in application design may be required to meet the projected table area use. Conversely, the estimator may show that a configuration can be successfully generated when, in fact, it cannot. The following paragraphs give some indications on how to handle the estimation better.

#### I.4 HINTS ON DETERMINING TERMINAL CAPACITY

Many factors affect the terminal capacity of DX10, but the greatest one is system table area. The number of devices in the configuration is the primary adverse effect on the maximum table area that can be specified. Also of importance is the number of communications lines, RTS devices, and RTC devices generated. These devices require space larger than the size of ITEMS 3 through 19 and ITEMS 27 through 35. Since applications and SCI that are needed to run a terminal use table space, it follows that the more devices there are, the more economical the use of table area per terminal must be. The sizes in ITEMS 39 and 40 are based on economical use; the values in Table 3-8 of Volume V are based on heavier development use where the user of the system is doing a typical interactive application requiring a number of files to be open, which takes more table area. These two numbers have the greatest unpredictable effect on the estimation of system size and capacity. The guide is designed so you can compute all other terms and then experiment with various values for these two items as noted in the following paragraph.

Two things can be done with the guide. You can determine if an application program with known table requirements will run on a given configuration, or you can determine the amount of table area available per terminal. The latter information is a guide for a design-to-fit exercise in which you determine the available table area per terminal and check whether it is possible to design an application to operate within that limit. Note that whatever basis is used, the root size in ITEM 44 cannot exceed the value in ITEM 43.

Determining table area requirements for one terminal for an application is not a simple matter. The best way is to benchmark it on an existing DX10 system. It may not be necessary to write the entire application to do this; dummy programs that open the required number of files and do a similar amount and kind of I/O can be written. The table area use for at least two, preferably more, can be observed using the SMS command after each invocation of the task. Any data base files that need to be set up can be set up with dummy fields and records. The tasks do not have to actually call the data base, but the data base needs to be started and the number of files needed must be opened. The basic idea is to determine the base table area need that is required to start one terminal, then determine the incremental need (that is, the space required to bring up each additional terminal). Then, based on the information in the estimator, the terminal capacity for that configuration is obtained as follows:

1. Determine the maximum table area for your system. This can be done in one of several ways:
  - a. Add item 98 to item 41 and recompute the rest of the estimator;
  - b. Perform a trial system generation (including PGS) and maximize the table area according to instructions in Section 3. Then, either look in D\$SOURCE for the value giving the size of table area (identified by comments in that file) or boot the system and perform an SMS. Then, substitute the value obtained from SMS into item 42.
2. Subtract the value of ITEM 26 from the value of ITEM 42.
3. Subtract the base table area for the application from the result of step 2. (Remember that the SMS command presents values in words.)
4. Divide the incremental table area requirement for the application into the result of step 3. The result plus one is approximately the number of terminals that can be supported.

Exercise care not to overestimate terminal capacity. System table area fragmentation occurs because all pieces allocated out of it on a dynamic basis are not the same size, and the order of allocation is random. This has the effect of increasing the amount of table area needed for an application. Other things are also allocated out of table area, including record-lock table entries that may not show up in a dummy application. You can largely compensate for those effects by increasing the estimate of the incremental table area requirement by 15 percent. If the calculation shows over 15 terminals can be supported, be wary. Extra tuning may be necessary to actually achieve the calculated result. (In some environments, nine terminals may be the number that can be supported.) Such things as background execution consume table area and can cause the limit (in practice) to be lower than calculated. Be persistent in your search for all possible things that use table area and include them in your calculation. Use the Show System Table Map (SSTM) command on your running system to show what is taking up table area. You can determine whether reducing either the number of tasks, the number of LUNOs assigned, or the number of files assigned will affect the execution of your system with your application.

The available table area per terminal can be determined by the following steps:

1. Perform the computation outlined in items 1 and 2 in the preceding list.
2. Divide the result of step 1 by the number of terminals you desire to support. The result is the average table area that can be occupied by each terminal's needs and not exceed the maximum table area that can be specified.

Note that the average includes the extra amount needed for starting the application spread out over all the terminals. You can use this number when designing the application as a guide to the number of tasks that can be associated with each terminal and the number of files that can be open.



# Appendix J

## Example Patch File

---

Appendix J contains a listing of an example patch file created by the Assemble and Link Generated System (ALGS) command and applied to the system by the Patch Generated System (PGS) command. Refer to Section 3 for information on the ALGS and PGS commands.

Example Patch File

```
*****
BATCH LS=YES
*****
*** PATCH FILE MEMRES DX10 3.6.0 LAST UPDATE 04/22/83 ***
*****
*
* *** SPECIAL INSTRUCTIONS ***
*
* AFTER A SYSGEN IS DONE, THE DX10 IMAGE FILE (OBJECT OUTPUT FROM
* THE LINK EDITOR) WILL NEED TO HAVE THE PATCHES IN THIS FILE
* APPLIED TO IT. THIS FILE CANNOT BE RUN AS A BATCH STREAM. TO
* PATCH A SYSTEM, ONE MUST ASSIGN THE SYNONYM $$DSC$ TO THE
* VOLUME NAME OF THE DISK ON WHICH THE PATCHES ARE TO BE APPLIED.
* THEN EITHER (A) OR (B) BELOW MUST BE PERFORMED.
*
* A) RUN THE XPS PROC, WITH THE FOLLOWING INPUTS:
* LINK = <THE LINK MAP OF THE SYSTEM TO BE PATCHED>
* INPUT = <THE NAME OF THIS FILE>
* OUTPUT = <THE NAME OF A FILE, WHICH IS TO BE USED AS THE
* BATCH STREAM>
* ERROR = <THE NAME OF SOME FILE TO BE USED FOR ERRORS>
* THE FILE SPECIFIED FOR OUTPUT CAN BE RUN AS A BATCH STREAM
* TO PATCH THE SYSTEM.
*
* B) IF THE LINK MAP IS NO LONGER AVAILABLE ON DISK FILE, THIS FILE
* WILL HAVE TO BE TRANSLATED MANUALLY. COPY THIS FILE TO
* ANOTHER FILE AND EDIT THAT FILE DOING THE FOLLOWING.
*
* 1. FOR EACH #SYN COMMAND FOUND IN THE FILE, TWO OR MORE
* SYNONYMS MUST BE DEFINED. EACH #SYN COMMAND IS IN THE
* FORMAT:
* #SYN <OVERLAY OR TASK ID>,<MODULE NAME>,<MODULE NAME>, ...
* FOR EXAMPLE:
* #SYN SROOT,ROOT
* WHERE:
* SROOT IS THE <OVERLAY OR TASK ID> SYNONYM NAME, AND
* ROOT IS THE <MODULE NAME> OF THE MODULE TO BE PATCHED.
*
* 2. FIND THE <MODULE NAME>S IN THE LINK MAP. ALL MODULES
* ON ONE LINE OF A #SYN COMMAND WILL RESIDE IN THE SAME PHASE
* OF THE LINK. DETERMINE THE OVERLAY OR TASK ID FROM THE
* PHASE HEADER LINE OF THE PHASE WHERE THE SPECIFIED MODULES
* ARE FOUND. INSERT A #SYN COMMAND INTO THE PATCH FILE TO
* DEFINE THE OVERLAY OR ID, AS IN THE FOLLOWING EXAMPLE:
* .SYN SROOT=1
* WHERE
* SROOT IS THE <OVERLAY OR TASK ID> SYNONYM FOUND
* IN THE #SYN COMMAND, AND
* 1 IS THE OVERLAY OR TASK ID FROM THE PHASE
* IN WHICH THE LISTED MODULES RESIDE.
* (1 IS THE ID FOR THE FIRST SYSTEM. THIS WILL
* BE DIFFERENT FOR ADDITIONAL SYSTEMS.)
*
```

- ```

*      3.  NEXT, INSERT .SYN COMMANDS INTO THE PATCH FILE FOR EACH
*          <MODULE NAME> IN THE #SYN COMMAND.  FOR EXAMPLE:
*          .SYN ROOT=031DA
*          WHERE:
*              ROOT    IS THE <MODULE NAME> OF THE MODULE (FROM THE #SYN
*                   COMMAND) TO BE PATCHED, AND
*              031DA  IS THE HEXADEDECIMAL ADDRESS OF THE MODULE
*                   ORIGIN FOR TI-SUPPLIED SYSTEM.  (THIS MAY BE
*                   DIFFERENT FOR CUSTOM SYSTEMS.)
*
*      4.  DELETE THE #SYN LINE FROM THE PATCH FILE.
*
*      5.  REPEAT STEPS 1 - 4 FOR EACH #SYN COMMAND IN THE FILE.
*
*      6.  DELETE ALL THE %SYN LINES FROM THE PATCH FILE.
*
*      7.  QUIT THE TEXT EDIT SPECIFYING "YES" TO THE REPLACE OPTION.
*
*      8.  EXECUTE THE BATCH STREAM YOU JUST CREATED.

```

\*\*\*\*\*

\*\*\* SYNONYM ASSIGNMENT

```

*
Q$SYN
.SYN $ESC="0"
.SYN IMAGE="@$$DSC$.S$IMAGES"
%SYN SROOT,ROOT
%SYN SROOT,TM$PFN
%SYN SFILMG,FILMG,KIF3 ! FILMG REQUIRED FOR SYSTEM LENGTH CHECK
%SYN SMEM,MEMRES
%SYN SIO,SVCIO,TM$CKY
%SYN S2D,SVC2ND,SYSLOG !2ND SVC SEGMENT
%SYN IOC,IOCOM
%SYN SDDIO,DDIOSR,OLDDKY
%SYN S911,DSR911
%SYN S913,DSR913
%SYN SLP,LPDSR
%SYN S820,DSR820
%SYN SFPY,FPYDSR
%SYN S979,DSR979,OLDTPY
%SYN S733,DSR733
%SYN STPD,DSRTPD
%SYN S93C,DSR93C
%SYN S$PC,DSR$PC
%SYN S93B,DSR93B

```

\*\*\*\*\*  
\* THE FOLLOWING SYNONYMS ARE DEFINED BY TI SYSTEM ANALYSTS ONLY AT  
\* SYSTEM BUILD TIME. THEY SHOULD NOT BE MODIFIED BY THE USER.  
\*\*\*\*\*

\* INSERT PRE-LINK DEFINED SYNONYMS HERE:

```

#SYN SROOT,ROOT
.EVAL S$$PAT="@ROOT+>0000"

```

\*\*\*\*\*

Example Patch File

```
*
* CHECK TO SEE IF THE SYSTEM ROOT IS GREATER THAN >C000.
* THIS CAUSES SEVERAL PROBLEMS IN DISK RESIDENT SYSTEM TASKS.
*
#SYN SFILMG,FILMG
.IF @FILMG,GT,>C000
CM R=ME,M="PATCH STREAM FATAL ERROR"
CM R=ME,M="SYSTEM ROOT EXCEEDS >C000 "
CM R=ME,M="CURRENT SYSTEM ROOT SIZE = @FILMG "
CM R=ME,M="PATCHING TERMINATED"
.STOP
.ENDIF
.SYN SFILMG="",FILMG=""
*
*****
*
* CHECK TO SEE IF THE SYSTEM LENGTH IS GREATER THAN >F800.
* THIS CAUSES SEVERAL PROBLEMS IN EXECUTION AS ANY REFERENCE
* IN THE OS TO ADDRESSES ABOVE >F800 TALK TO DISKS INSTEAD
* OF MEMORY.
*
.IF @SROOT,EQ,>100
CM R=ME,M="PATCH STREAM FATAL ERROR"
CM R=ME,M="LONGEST OVERLAY PATH EXCEEDS >F800"
CM R=ME,M="PATCHING TERMINATED"
.STOP
.ENDIF
*
*****
* The following are the current versions of the templates for each
* type of patch. The column positions in the header are important.
* Note each comment starts in column 4. Each line extends no further
* than column 70, with the exception of star lines for separators.
* No more than four verify or data items allowed per MPI/MRF/MAD.
*****
*=Pnnnn INT MM/DD/YY STR #mmmmm v.r.e modulename DX10
*
* up to ten lines of description of the patch (each line of the form
* shown here - star, two blanks, then words)
*
*QSSYN
*#SYN ovlynam,modnam
*MPI PF=IMAGE,MT=tt,MN=modnam,ADR=>0000,
* V(> ,> ,> ,> ),
* D(> ,> ,> ,> ),C=>
*EC
*.SYN ovlynam="",modnam=""
*$
*****
```

```

*
*=Pnnnn INT MM/DD/YY STR #mmmmm v.r.e modulename DX10
*
* up to ten lines of description of the patch (each line of the form
* shown here - star, two blanks, then words)
*
*QSSYN
*MRF PN=pathname,RN=nnnn,FW=mmmm,
* V=(> ,> ,> ,> ),
* D=(> ,> ,> ,> ),C=>
*EC
*$
*****
*
*=Pnnnn INT MM/DD/YY STR #mmmmm v.r.e modulename DX10
*
* up to ten lines of description of the patch (each line of the form
* shown here - star, two blanks, then words)
*
*QSSYN
*MTE PF=programfile,MN=module,PRIORITY=
*EC
*$
*****
*
*=Pnnnn INT MM/DD/YY STR #mmmmm v.r.e modulename DX10
*
* up to ten lines of description of the patch (each line of the form
* shown here - star, two blanks, then words)
*
*QSSYN
*MAD DISK=diskname,TRACK=nnnn,SECTOR=mmmm,FW=www,
* V=(> ,> ,> ,> ),
* D=(> ,> ,> ,> ),C=>
*EC
*$
*****
* end of templates
*****
*
* BEGIN PATCHES FOR RELEASE 3.6.0
*
*****
*
* INSERT PATCHES HERE. BE SURE TO FOLLOW EACH MPI CALL WITH AN EC CALL.
*
*****

```

Example Patch File

```
*****
*
*=Pnnnn CIW/INT 04/28/83 KI$LOC3 STR #mmmmm 0.6.0 KIF3 DX10
*
* fix >A0 crash in fan #4
*
#SYN SFILMG,KIF3
.IF @KIF3, NE, "NONE"
Q$$SYN
MPI PF=IMAGE,MT=OV,MN=@SFILMG,ADR=@KIF3+>115C+>0190,
V=>06A0,D=>0460,>C=0460
EC
MPI PF=IMAGE,MT=OV,MN=@SFILMG,ADR=@KIF3+>115C+>0192,
V=@ROOT+>190C,D=@S$$PAT,C=@S$$PAT
EC
MPI PF=IMAGE,MT=TA,MN=@SROOT,ADR=@S$$PAT,
V=(>DEAD,>DEAD),
D=(>CE8B,>06A0),
C=>C82B
EC
MPI PF=IMAGE,MT=TA,MN=@SROOT,ADR=@S$$PAT+>0004,
V=>DEAD,D=@ROOT+>190C,C=@ROOT+>190C
EC
MPI PF=IMAGE,MT=TA,MN=@SROOT,ADR=@S$$PAT+>0006,
V=>DEAD,D=>460,C=>0460
EC
MPI PF=IMAGE,MT=TA,MN=@SROOT,ADR=@S$$PAT+>0008,
V=>DEAD,D=@KIF3+>115C+>0194,C=@KIF3+>115C+>0194
EC
.ENDIF
.SYN KIF3=""
*$
*****
*
* END OF 3.6.0 PATCHES
Q$$SYN
#SYN SROOT,ROOT
MPI PF=IMAGE,MT=TA,MN=@SROOT,ADR=@S$$PAT+>0240,V=>2D31,D=>2030,C=>2030
!EC
*****
*
CM R=ME,M="SYSTEM PATCH STREAM ERROR COUNT = @$E$C"
*
*****
```

```
*
*   ***   DELETE SYNONYMS THAT ARE UNIQUE TO THIS BATCH STREAM   ***
*
QSSYN
.SYN IMAGE="", $$$PAT=""
.SYN SROOT="", ROOT="", S820="", DSR820=""
.SYN SFILMG="", FILMG="", KIF="", KIF3="", HKIF=""
.SYN SMEM="", MEMRES="", SIO="", SVCIO=""
.SYN DDIOSR="", SDDIO="", OLDDKY="", IOC="", IOCOM=""
.SYN S911="", DSR911="", SLP="", LPDSR=""
.SYN S913="", DSR913=""
.SYN SFPY="", FPYDSR=""
.SYN STPD="", DSRTPD=""
.SYN SKIF3="", S2D="", SVC2ND="", SYSLOG=""
.SYN DSR979="", COMMCOM="", DSR733=""
.SYN SVCUSR="", SVCOPT="", T940=""
.SYN S733="", S979="", SCOM="", TM$CKY=""
.SYN OLDTPY="", TM$PFN=""
*
*****
SDT
EBATCH,LS=Y
*****
* THE NEXT AVAILABLE PATCH LOCATION IS $$$PAT+>000A
*****
```





# Alphabetical Index

## Introduction

---

### HOW TO USE INDEX

The index, table of contents, list of illustrations, and list of tables are used in conjunction to obtain the location of the desired subject. Once the subject or topic has been located in the index, use the appropriate paragraph number, figure number, or table number to obtain the corresponding page number from the table of contents, list of illustrations, or list of tables.

### INDEX ENTRIES

The following index lists key words and concepts from the subject material of the manual together with the area(s) in the manual that supply major coverage of the listed concept. The numbers along the right side of the listing reference the following manual areas:

- **Sections** — Reference to Sections of the manual appear as “Sections x” with the symbol x representing any numeric quantity.
- **Appendixes** — Reference to Appendixes of the manual appear as “Appendix y” with the symbol y representing any capital letter.
- **Paragraphs** — Reference to paragraphs of the manual appear as a series of alphanumeric or numeric characters punctuated with decimal points. Only the first character of the string may be a letter; all subsequent characters are numbers. The first character refers to the section or appendix of the manual in which the paragraph may be found.
- **Tables** — References to tables in the manual are represented by the capital letter T followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the table). The second character is followed by a dash (-) and a number.

Tx-yy

- **Figures** — References to figures in the manual are represented by the capital letter F followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the figure). The second character is followed by a dash (-) and a number.

Fx-yy

- **Other entries in the Index** — References to other entries in the index preceded by the word “See” followed by the referenced entry.

- Address:
  - CRU ..... 3.2.2
  - TILINE ..... 3.2.4
- ALGS Command ..... Appendix J, 3.5, 4.5
- Assemble and Link Generated System (ALGS) Command ..... Appendix J, 3.5, 4.5
- Assembling and Linking the System, System Generation ..... 3.5
- Assign Space on Program File
  - SVC > 37 ..... 7.15
- Asynchronous:
  - Data Structure Allocation ..... 5.5.5
  - Data Structure Linkages ..... F5-20
  - Device Support ..... T5-7
  - DSR Local PDT Extension ..... 5.3.6
  - DSR Long-Distance Device Extension ..... 5.3.7
  - DSR Structure ..... 5.5
  - Interrupt Service Routine (ISR) ..... 5.5.3
  - ISR ..... 5.5.3
- Backup Directory (BD) Command ..... 2.9
- Backup Directory to Device (BDD) Command ..... 2.9
- Base System Configuration ..... 2.2
- BATCH Command ..... 4.5
- BD Command ..... 2.9
- BDD Command ..... 2.9
- Begin Batch Execution (BATCH) Command ..... 4.5
- Boards, Interface ..... T3-9, T3-10
- Branch Table Call (BRCALL) Routine .. 5.7.1
- Branch Table Call with Statistics (BRSTAT) Routine ..... 5.7.10
- BRCALL Routine ..... 5.7.1
- BRSTAT Routine ..... 5.7.10
- Buffered I/O Request Block ..... 5.3.3
- Business System Terminal
  - Keyboard Layout ..... FA-5
- Busy Check (BZYCHK) Routine ..... 5.7.2
- BZYCHK Routine ..... 5.7.2
- Call Block, SVC ..... 6.2.1
- CD Command ..... 2.9
- CDM ..... F2-2
- Chassis, Expansion .... 3.4.4.20, 3.4.5.1, F.5
- Clear Secret Synonyms (Q\$SYN) Command ..... 4.5
- Codes, SVC ..... TG-1
- Command ..... See name or acronym of command
- Command Mode, GEN990 ..... 3.3.2.2
- Compatibility, DX10 2.X ..... Section 8, 1.7
- SVC ..... 8.3
- Control/Display Module (CDM) ..... F2-2
- Controller Interrupt Decoder
  - HSR Subroutine ..... 5.6.9
- Conversion, DX10 2.X:
  - Disk ..... 8.2
  - Disk/Magnetic Tape ..... 8.2.1
  - Log ..... 8.2.2.2
  - Multiple Disk ..... 8.2.2
- Copy and Verify Disk (CVD) Command ..... 2.9
- Copy Directory (CD) Command ..... 2.9
- Copy Volume (CV) Command ..... 2.5.2, 2.9
- Crash ..... 4.3.1
- CRU Address ..... 3.2.2
- CV Command ..... 2.5.2, 2.9
- CVD Command ..... 2.9
- Data Structures, DSR ..... 5.3
- DCOPY Command ..... 2.9
- Delete Overlay SVC > 2A ..... 7.7
- Delete Procedure SVC > 29 ..... 7.6
- Delete Task SVC > 28 ..... 7.5
- Device Service Routine (DSR) ..... Section 5, Appendix H, 1.4
- Device, Special ..... 3.4.5.10, F.4
- Direct Disk I/O ..... 7.19
- Disk Build ..... 1.2, 2.5.2
- Disk Build Utility ..... 2.6
- Disk Conversion, DX10 2.X ..... 8.2
- Disk Copy/Restore (DCOPY) Command ..... 2.9
- Disk I/O, Direct ..... 7.19
- Disk Manager SVC > 22 ..... 7.14
- Disk/Magnetic Tape Conversion, DX10 2.X ..... 8.2.1
- DSR ..... Section 5, Appendix H, 1.4
- Data Structures ..... 5.3
- Enable/Disable Status Change
  - Notification HSR Subroutine ..... 5.6.4
- End-of-Record (ENDRCD) Routine ..... 5.7.3
- ENDRCD Routine ..... 5.7.3
- Examples, System
  - Generation ..... Appendix F
- Execute System Generation Utility (XGEN) Command ..... 3.4, 4.5
- Execute 2.2 to 3.0 Disk Conversion (XCU) Command ..... 8.2.2.1
- Expansion Chassis .... 3.4.4.20, 3.4.5.1, F.5
- Extension for Terminals with Keyboards (XTK) ..... 5.3.5
- Factor, Interleaving ..... 2.5.2
- FD1000 ..... 2.6.3
- File, Patch ..... Appendix J
- Files:
  - GEN990 ..... 3.3.1
  - System ..... 2.11
- Generic Keycap
  - Names ..... Appendix A, TA-1
- GEN990 ..... 1.3, 3.3
- Command Mode ..... 3.3.2.2
- Files ..... 3.3.1
- Inquire Mode ..... 3.3.2.1
- Get Character (GETC) Routine ..... 5.7.5
- Get System Pointer Table
  - Address SVC > 32 ..... 7.12

- GETC Routine ..... 5.7.5
- Global LUNOs ..... TE-1
- Hang ..... 4.3.1
- Hardware Controller Service
  - Routine (HSR) ..... 5.5.4
- HSR ..... 5.5.4
  - Baud Rate Code ..... T5-10
  - Controller Type Code ..... T5-11
- HSR Common Subroutine ..... 5.6
  - Controller Interrupt Decoder ..... 5.6.9
  - Enable/Disable Status Change
    - Notification ..... 5.6.4
  - Output a Character ..... 5.6.5
  - Power-Up Initialization ..... 5.6.1
  - Read Input Signal or Function ..... 5.6.3
  - Read Operational Parameters
    - and Information ..... 5.6.7
  - Request Time Interval Notification .. 5.6.8
  - Write Operational Parameters ..... 5.6.6
    - Set Channel Speed
      - (Baud Rate) ..... 5.6.6.1
      - Set Data Character Format ..... 5.6.6.2
    - Write Output Signal or Function .... 5.6.2
- IGS ..... 3.8
- Initialize Date and Time SVC > 3B ..... 7.13
- Initialize New Volume (INV)
  - Command ..... 2.5.2, 4.6
- Initialize New Volume SVC > 38 ..... 7.16
- Initialize System (IS)
  - Command ..... 2.6.1, 2.13.3
- Inquire Mode, GEN990 ..... 3.3.2.1
- Install Disk Volume SVC > 20 ..... 7.17
- Install Generated System (IGS)
  - Command ..... 3.8
- Install Overlay SVC > 27 ..... 7.4
- Install Procedure SVC > 26 ..... 7.3
- Install Real-Time Task SVC > 25 ..... 7.2.2
- Install Task:
  - Special Form SVC > 25 ..... 7.2.1
  - SVC > 25 ..... 7.2
- Interface Boards ..... T3-9, T3-10
- Interleaving Factor ..... 2.5.2
- Interrupt Service Routine (ISR) ..... 5.2.5
  - Asynchronous ..... 5.5.3
- Interrupts ..... 3.2.3, 5.2.5
- INV Command ..... 2.5.2, 4.6
- I/O Call Routine ..... 5.2.1
- IS Command ..... 2.6.1, 2.13.3
- ISR ..... 5.2.5
  - Asynchronous ..... 5.5.3
- JMCALL Routine ..... 5.7.6
- Jump Table Call (JMCALL) Routine ... 5.7.6
- Key Sequences ..... TA-2
- Keyboard Function (KEYFUN)
  - Routine ..... 5.7.4
- Keyboard Layout:
  - Business System Terminal ..... FA-5
  - 820 KSR ..... FA-6
  - 911 VDT ..... FA-1
  - 915 VDT ..... FA-2
  - 931 VDT ..... FA-4
  - 940 EVT ..... FA-3
- Keyboard Status Block (KSB) ..... 5.3.2
- Keycap Name Equivalents, 911 VDT ... TA-3
- Keycap Names,
  - Generic ..... Appendix A, TA-1
- KEYFUN Routine ..... 5.7.4
- Kill Task SVC > 33 ..... 7.8
- KSB ..... 5.3.2
- Log, DX10 2.X Conversion ..... 8.2.2.2
- LUNOs, Global ..... TE-1
- Memory Estimator ..... Appendix I
- Modified SVC, DX10 2.X ..... 8.3.3
- Modify Terminal Status (MTS)
  - Command ..... 2.13.2
- Modify Volume Information (MVI)
  - Command ..... 2.12
- MTS Command ..... 2.13.2
- Multiple Disk, DX10 2.X Conversion ... 8.2.2
- MVI Command ..... 2.12
- M\$00 ..... 4.5
- Near Equality Algorithm ..... 3.3.3.2
- Obsolete SVC, DX10 2.X ..... 8.3.1, 8.3.2
- Output a Character HSR Subroutine ... 5.6.5
- Overlay IDs, System ..... Appendix C
- Patch File ..... Appendix J
- Patch Generated System (PGS)
  - Command ..... Appendix J, 3.6, 4.5
- Patching the System,
  - System Generation ..... 3.6
- PDT ..... 5.3.1
- PGS Command ..... Appendix J, 3.6, 4.5
- Physical Device Table (PDT) ..... 5.3.1
- Power-Up Initialization HSR
  - Subroutine ..... 5.6.1
- Privileged SVC ..... Section 7, 1.6
- Procedure IDs, System ..... TD-1
- Programmer Panel ..... F2-1
- Put Character in Buffer (PUTCBF)
  - Routine ..... 5.7.7
- Put Event Character in Buffer (PUTEBF)
  - Routine ..... 5.7.8
- PUTCBF Routine ..... 5.7.7
- PUTEBF Routine ..... 5.7.8
- Q\$SYN Command ..... 4.5
- Read Input Signal or Function HSR
  - Subroutine ..... 5.6.3
- Read Operational Parameters
  - and Information HSR Subroutine ... 5.6.7
- Read/Write Task SVC > 2D ..... 7.11
- Read/Write TSB SVC > 2C ..... 7.10

- Request Time Interval Notification
  - HSR Subroutine ..... 5.6.8
  - Routine, Common ..... 5.7, See also name  
or mnemonic of routine
- Set Channel Speed (Baud Rate) HSR
  - Subroutine ..... 5.6.6.1
- Set Data Character Format HSR
  - Subroutine ..... 5.6.6.2
- Set Interrupt Mask (SETWPS)
  - Routine ..... 5.7.9
- SETWPS Routine ..... 5.7.9
- Special Device ..... 3.4.5.10, F.4
- Structure, SVC ..... 6.2.2
- Subroutine, HSR Common ..... 5.6
  - Controller Interrupt Decoder ..... 5.6.9
  - Enable/Disable Status Change  
Notification ..... 5.6.4
  - Output a Character ..... 5.6.5
  - Power-Up Initialization ..... 5.6.1
  - Read Input Signal or Function ..... 5.6.3
  - Read Operational Parameters  
and Information ..... 5.6.7
  - Request Time Interval  
Notification ..... 5.6.8
  - Write Operational Parameters  
Set Channel Speed  
(Baud Rate) ..... 5.6.6.1
  - Set Data Character Format ..... 5.6.6.2
  - Write Output Signal or Function ..... 5.6.2
- Supervisor Call ..... See SVC
- Suspend Awaiting Queue Input
  - SVC > 24 ..... 7.9
- SVC ..... Section 6, Section 7, 1.5
  - Assign Space on Program File,  
> 37 ..... 7.15
  - Call Block ..... 6.2.1
  - Codes ..... TG-1
  - Compatibility, DX10 2.X ..... 8.3
  - Delete Overlay, > 2A ..... 7.7
  - Delete Procedure, > 29 ..... 7.6
  - Delete Task, > 28 ..... 7.5
  - Disk Manager, > 22 ..... 7.14
  - Error:
    - > 05 ..... 4.4.1
    - > 0E ..... 4.4.2
    - > 89 ..... 4.4.3
  - Get System Pointer Table Address,  
> 32 ..... 7.12
  - Initialize Date and Time, > 3B ..... 7.13
  - Initialize New Volume, > 38 ..... 7.16
  - Install Disk Volume, > 20 ..... 7.17
  - Install Overlay, > 27 ..... 7.4
  - Install Procedure, > 26 ..... 7.3
  - Install Real-Time Task, > 25 ..... 7.2.2
  - Install Task Special Form, > 25 ..... 7.2.1
  - Install Task, > 25 ..... 7.2
  - Kill Task, > 33 ..... 7.8
  - Modified, DX10 2.X ..... 8.3.3
  - Obsolete, DX10 2.X ..... 8.3.1, 8.3.2
  - Privileged ..... Section 7, 1.6
  - Read/Write Task, > 2D ..... 7.11
  - Read/Write TSB, > 2C ..... 7.10
  - Structure ..... 6.2.2
  - Suspend Awaiting Queue Input,  
> 24 ..... 7.9
  - Writing ..... Section 6
  - Unload Disk Volume, > 34 ..... 7.18
- System:
  - Files ..... 2.11
  - Logbook ..... 4.2
  - Overlay IDs ..... Appendix C
  - Procedure IDs ..... TD-1
  - Task IDs ..... TB-1
- System Generation ..... Section 3, 1.3
  - Assembling and Linking the  
System ..... 3.5
  - Examples ..... Appendix F
  - Expansion Chassis .. 3.4.4.20, 3.4.5.1, F.5
  - Patching the System ..... 3.6
- Tape Build Utility ..... 2.7
- Task IDs, System ..... TB-1
- Task Status Block (TSB) ..... 5.3.4
- Terminal Service Routine (TSR) ..... 5.5.2
- Test Generated System (TGS)
  - Command ..... 3.7
  - TGS Command ..... 3.7
  - TILINE Address ..... 3.2.4
- TSB ..... 5.3.4
- TSR ..... 5.5.2
- Unload Disk Volume SVC > 34 ..... 7.18
- Utility:
  - Disk Build ..... 2.6
  - Tape Build ..... 2.7
- Write Operational Parameters
  - HSR Subroutine ..... 5.6.6
- Write Output Signal or Function
  - HSR Subroutine ..... 5.6.2
- Writing, SVC ..... Section 6
- XCU Command ..... 8.2.2.1
- XGEN Command ..... 3.4, 4.5
- XTK ..... 5.3.5
- 2.X Conversion, DX10:
  - Disk ..... 8.2
  - Disk/Magnetic Tape ..... 8.2.1
  - Log ..... 8.2.2.2
  - Multiple Disk ..... 8.2.2
- 820 KSR Keyboard Layout ..... FA-6
- 911 VDT:
  - Keyboard Layout ..... FA-1
  - Keypad Name Equivalents ..... TA-3
- 915 VDT Keyboard Layout ..... FA-2
- 931 VDT Keyboard Layout ..... FA-4
- 940 EVT Keyboard Layout ..... FA-3
- > 05, SVC Error ..... 4.4.1
- > 0E, SVC Error ..... 4.4.2
- > 89, SVC Error ..... 4.4.3

# USER'S RESPONSE SHEET

Manual Title: DX10 Operating System Systems Programming Guide, Volume V (946250-9705)

Manual Date: January 1985 Date of This Letter: \_\_\_\_\_

User's Name: \_\_\_\_\_ Telephone: \_\_\_\_\_

Company: \_\_\_\_\_ Office/Department: \_\_\_\_\_

Street Address: \_\_\_\_\_

City/State/Zip Code: \_\_\_\_\_

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

CUT ALONG LINE

| Location in Manual | Comment/Suggestion |
|--------------------|--------------------|
| _____              | _____              |
|                    | _____              |
|                    | _____              |
|                    | _____              |
|                    | _____              |
| _____              | _____              |
|                    | _____              |
|                    | _____              |
|                    | _____              |
|                    | _____              |
| _____              | _____              |
|                    | _____              |
|                    | _____              |
|                    | _____              |

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

FOLD



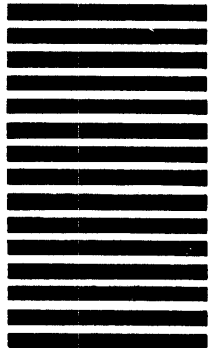
NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**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