IBM System/36

Program Service Information

IBM System/36

Program Service Information

File Number
S36-36

Order Number
LY21-0590-4

Fifth Edition (August 1986)

This major revision makes obsolete LY21-0590-3. See *About This Manual* for a summary of major changes.

This edition applies to Release 5, Modification Level 0, of IBM System/36 System Support Program Products (Program 5727-SS1 for the 5360 and 5362 System Units, and Program 5727-SS6 for the 5364 System Unit), and to all subsequent releases and modifications until otherwise indicated.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

The numbers at the bottom right of illustrations are publishing control numbers and are not part of the technical content of this manual.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to your IBM-approved remarketer.

This publication could contain technical inaccuracies or typographical errors. A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department 245, Rochester, Minnesota, U.S.A. 55901. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

About This Manual

Who should use this manual...

This manual provides the information that IBM service representatives need to perform program support services on System/36 programming and program products for which they are responsible. Information in this manual defines the general programming architecture of System/36. This information is at a more general level than information traditionally presented in program logic manuals. This manual should help you to quickly recall major functions of the System/36 programming you must support. It is intended to help you find the failing function or subfunction so that you can report the problem via an APAR.

To use this manual effectively, you should have completed either System/36 program support training or System/36 support-level CE/CSR training.

How this manual is arranged...

- *Sections 1, 2, and 4:* The introduction, system function/control flow, and directory sections help you recall major SSP functions. These sections enable you to provide on-site assistance to customers or other IBM personnel for circumventing programming problems.

- *Section 3:* The SSP programming interface section helps you isolate problems in SSP programming from problems in other IBM-supplied main storage or control storage programming.

- *Appendix A:* The data communications protocols help you diagnose problems in IBM-supplied data communications programming features.

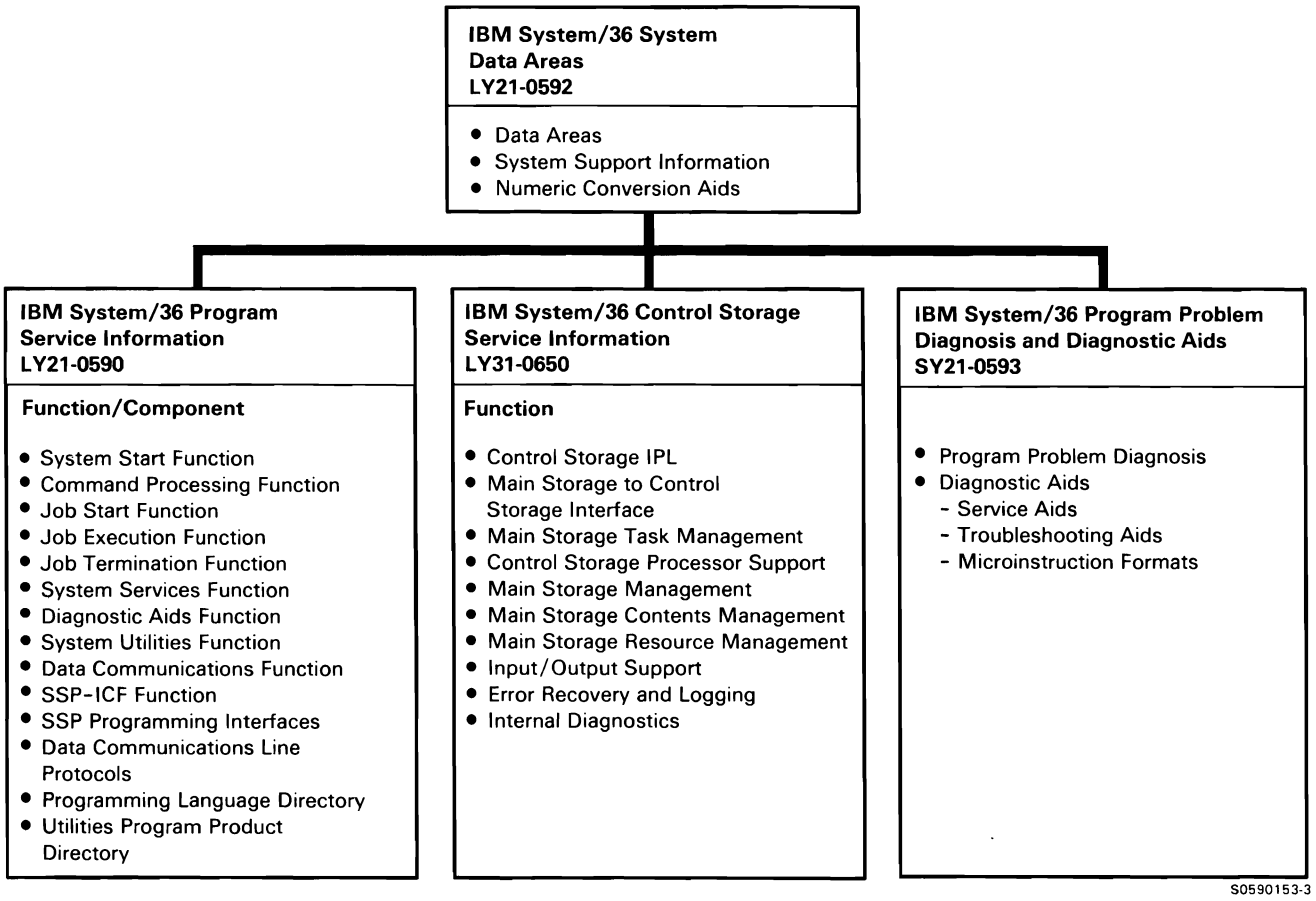
- *Appendix B:* The programming language directory provides reference information that you can use to isolate problems in the IBM-supplied compiler modules and execution-time subroutines from possible problems in user-supplied source code.

- *Appendix C:* The program product utilities directory provides reference information that you can use to isolate problems in the IBM-supplied program product utilities modules from possible problems in user-supplied source code.

This manual assumes that almost all of your emergency work will involve the SSP programming. With customers becoming more and more dependent on online data to make minute-by-minute business decisions, SSP programming failures can be almost as catastrophic as hardware failures. Therefore, most of the information in this manual relates to the SSP portion of the System/36 programming. This manual places less emphasis on documenting IBM utility program products and languages because problems involving these products, whether valid or invalid, are usually dependent on the customer's source code. Thus problems involving these products can usually be temporarily circumvented by changing the source code or OCL.

If you need more information. . .

The chart below shows all program service documentation available for System/36:



In addition to the other service manuals, you may need some or all of the following System/36 information while using this manual:

- *Functions Reference Manual*, SA21-9436
- *System Reference*, SC21-9020
- *System Messages*, SC21-7938
- *Changing Your System Configuration*, SC21-9052
- *Interactive Communications Feature: Reference*, SC21-7910
- *Multiple Session Remote Job Entry Guide*, SC21-7909
- *3270 Device Emulation Guide*, SC21-7912
- *Overlay Linkage Editor Guide*, SC21-9041
- *Programming with Assembler*, SC21-7908

- *Programming with RPG II*, SC21-9006
- *Programming with COBOL*, SC21-9007
- *Programming with BASIC*, SC21-9003
- *Programming with FORTRAN IV*, SC21-9005
- *Sort Guide*, SC21-7903
- *Work Station Utility Guide*, SC21-7905
- *Data File Utility Guide*, SC21-7900
- *Source Entry Utility Guide*, SC21-7901
- *Creating Displays: Screen Design Aid and System Support Program*, SC21-7902
- *Ideographic Sort Guide*, SC09-1054
- *Character Generator Utility Guide*, SC09-1055
- *Communications and Systems Management Guide*, SC21-8010
- *Distributed Data Management Guide*, SC21-8011

How this manual has changed. . .

Changes and additions have been made to support the following Release 5 enhancements: Source distributed data management (SDDM), Release 3 and Release 4, and Release 5 target distributed data management (TDDM) mainline, Release 3 and Release 4 TDDM VTOC extract, DDM trace (SDDM and TDDM), disk copy record mode, network resource directory (NRD) LISTNRD utility, folder management services (FMS) utility, SNA-to-local area network (LAN) posting, multiline communications attachment/eight line communications attachment (MLCA/ELCA), remote work station (RWS) support subfunction, support subfunction for the LAN, SSP-ICF function, SSP-ICF advanced peer-to-peer networking (APPN), and the personal computer support subsystem. Miscellaneous changes and additions have also been made throughout the manual.

ACTLU	activate logical unit
ACTPU	activate physical unit
AFA	active format-1 area
APAR	authorized program analysis report
APPC	advanced program-to-program communications (SSP-ICF subsystem)
APPN	advanced peer-to-peer network
AQE	allocation queue element
ATR	address translation register
BSC	binary synchronous communications
BSCEL	binary synchronous communications equivalence link
BUB	binary synchronous unit block
C & SM	Communications and Systems Management
C/SNA	combined SNA
CE/CSR	customer engineer/customer service representative
CFB	command function block
CICS	customer information control system
CCP	communications control program
CNOS	change number of sessions (command)
config	configuration
CPI	characters per inch
CQE	console/subconsole queue element
CUB	controller (RWS) unit block
DACTLU	deactivate logical unit
DACTPU	deactivate physical unit
DCE	data communications equipment or X.25 data circuit terminating equipment
DDM	distributed data manager
DFA	Dump File Analysis (utility)
DFC	data flow control (a major group of SNA components that correlates requests and responses between SNA partners)
DISC	disconnect command
DSNX	distributed systems node executive
DSX	distributed systems executive
DTE	data terminal equipment
DTF	define the file (control block)
DUB	device unit block (includes TUBs and PUBs)
EOF	end of file
EOV	end of volume
ERAP	error recording analysis procedure
EXTN	(task) ideographic character processing task
FAB	file access block (for file subsystem processor)
FMS	folder management services
FSB	file specification block
F1	format 1
F5	format 5
HDR1	tape header label 1
HDR2	tape header label 2
IGC	ideographic character
IMS	information management system
IOCH	I/O control handler (in control storage)
IOB	I/O block
IOS	I/O supervisor (in control storage)
IT	intermediate text (in BASIC)
JCB	job control block
LCB	library control block
LCS	library control sector
LPI	lines per inch
MBB	MSRJE BSC block
MCR	(1255) magnetic (ink) character reader
MIC	message identification code
MLCA/ ELCA	multiline communications attachment/eight line communications attachment
MSRJE	Multiple Session Remote Job Entry feature
NEL	NRD entry lock
NMVT	network management vector transport (common data structure used in network problem management)
NRD	network resource directory
NR-NS	SDLC number received-number sent counts
NSI	next sequential instruction
NUPD	
op code	operation code
PB	program block
PSDN	packet switched data network
PSR	program support representative
PTF	program temporary fix
PVC	permanent virtual circuit
PUB	printer unit block
RB	request block
RIB	request indicator byte
RJE	remote job entry
RRN	relative record number
RWS	remote work station
SABM	set asynchronous balance mode
SDDM	source distributed data management
SDLC	synchronous data link control
SLCA	single line communications attachment
SMF	System Measurement Facility
SNA	Systems Network Architecture

SNRM	set normal response mode command
SNUB	SNA unit block
SNUF	SNA Upline Facility
SQS	system queue space
SSP	System Support Program Product
SSP-ICF	System Support Program Product Interactive Communications Feature
SUB	session unit block
SVC	supervisor call (instruction) or X.25 switched virtual circuit
TB	task block
TDDM	target distributed data management
TDM	task distributed manager
TEB	termination exit block
TUB	terminal unit block
TWA	task work area
TWS	task work space
UHLI-8	user-specified tape header label 1 through 8
WSQS	work station queue space
X.21	communications type
X.25	communications type
XR1	index register 1
XR2	index register 2
XSUB	translated session unit block
ZPAM	sectorized data management access method

SYSTEM/36 PROGRAMMING

For System/36, your job consists of supporting the following IBM-supplied programming:

- System/36 System Support Program Product (SSP)
- System/36 SSP-Interactive Communications Feature (SSP-ICF)
- System/36 SSP data communications
- System/36 Multiple Session Remote Job Entry (MSRJE) feature
- System/36 3270 Device Emulation feature
- Communications and Systems Management feature
- Ideographic Generator and Sort Program Product
- OFFICE/36:
  - Personal Services/36 (PS/36)
  - DisplayWrite/36 (DW/36)
  - Query/36
- Utilities Program Product:
  - Data file utility
  - Source entry utility
  - Work station utility
  - Screen design aid
- System/36 languages/compilers:
  - BASIC
  - FORTRAN
  - Assembler
  - COBOL
  - RPG II

System/36 SSP

Section 2: System Function/Control Flow divides the SSP into the following major functions:

1. System start
2. Command processing
3. Job start
4. Job execution
5. Job termination
6. System services
7. Diagnostic aids
8. System utilities
9. Data communications
10. SSP-ICF

The major function numbers (keyed 1 through 10) are used in the control flow charts in Section 2 and in control flow chart references in *Section 4: Directory*. These major function numbers are also used in *Section 3: SSP Programming Interfaces* to refer to SSP function/control flow descriptions in Section 2.

Control flow charts are explained under *SSP Subfunction Descriptions* in Section 2.

Chart 0 shows the control flow between the major SSP functions:

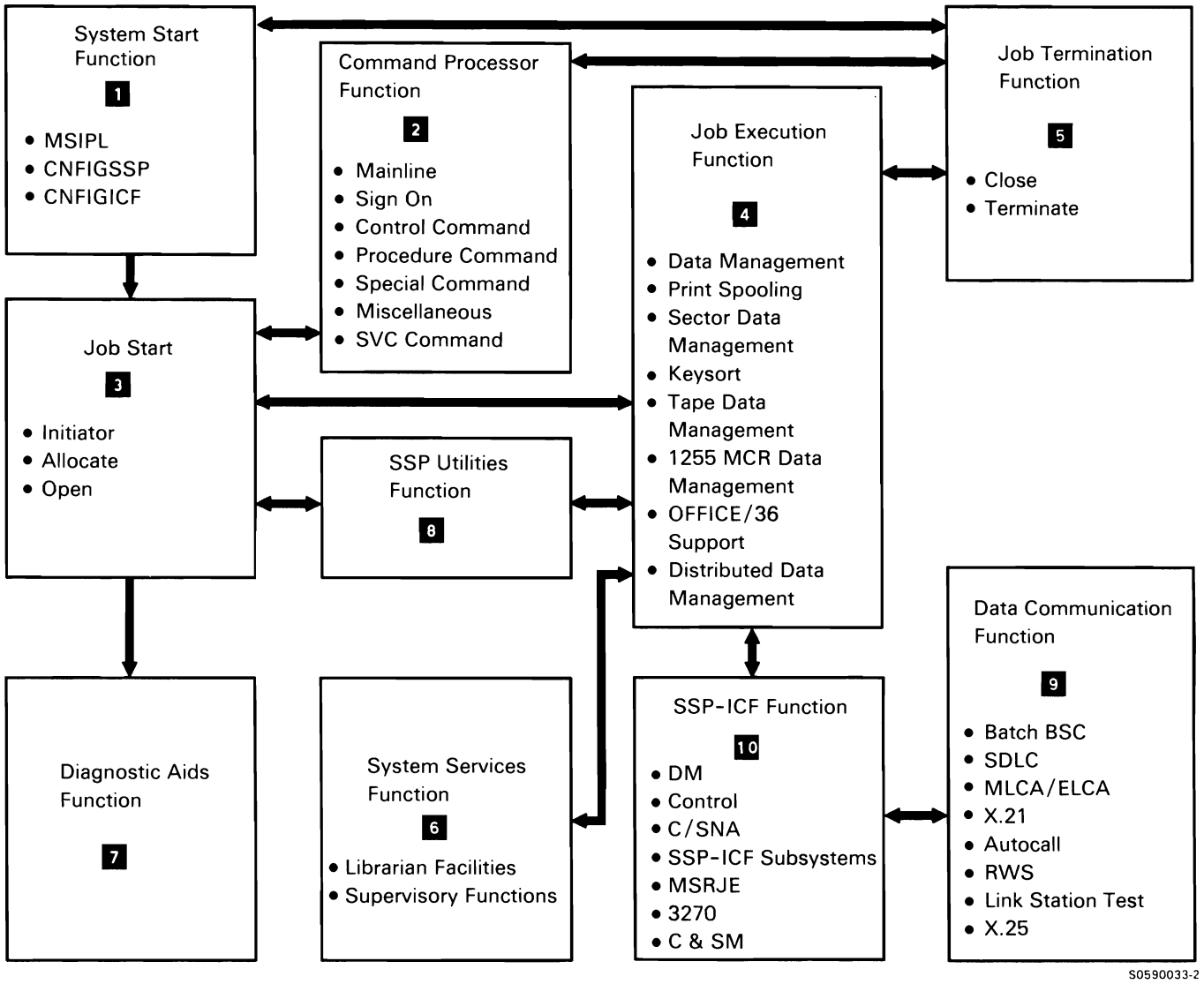


Chart 0. System Support Program Major Function Overview Control Flow

SSP SUBFUNCTION DESCRIPTIONS

In this section, each major function (1 through 10) is divided into *subfunctions*, which are described in control flow charts. Each control flow chart includes descriptive text and a diagram of the subfunction's control flow from one module or routine to another. The text includes a numbered list of the processes performed in a subfunction; each process statement describes why control flows to a module or routine at its number-identified point in the chart.

Whenever possible, the processes are listed in operational sequence; however, in more complex charts the processes are usually listed in the sequence that makes the chart easiest to use. When an understanding of the operational sequence is essential, the operational sequence is given in the description of the subfunction.

Example of a Subfunction Description

The following example (taken from major function 2, *Command Processing Function*) shows the parts of a subfunction description; the symbols used in the charts (arrows and boxes) are explained under *Chart Conventions*:

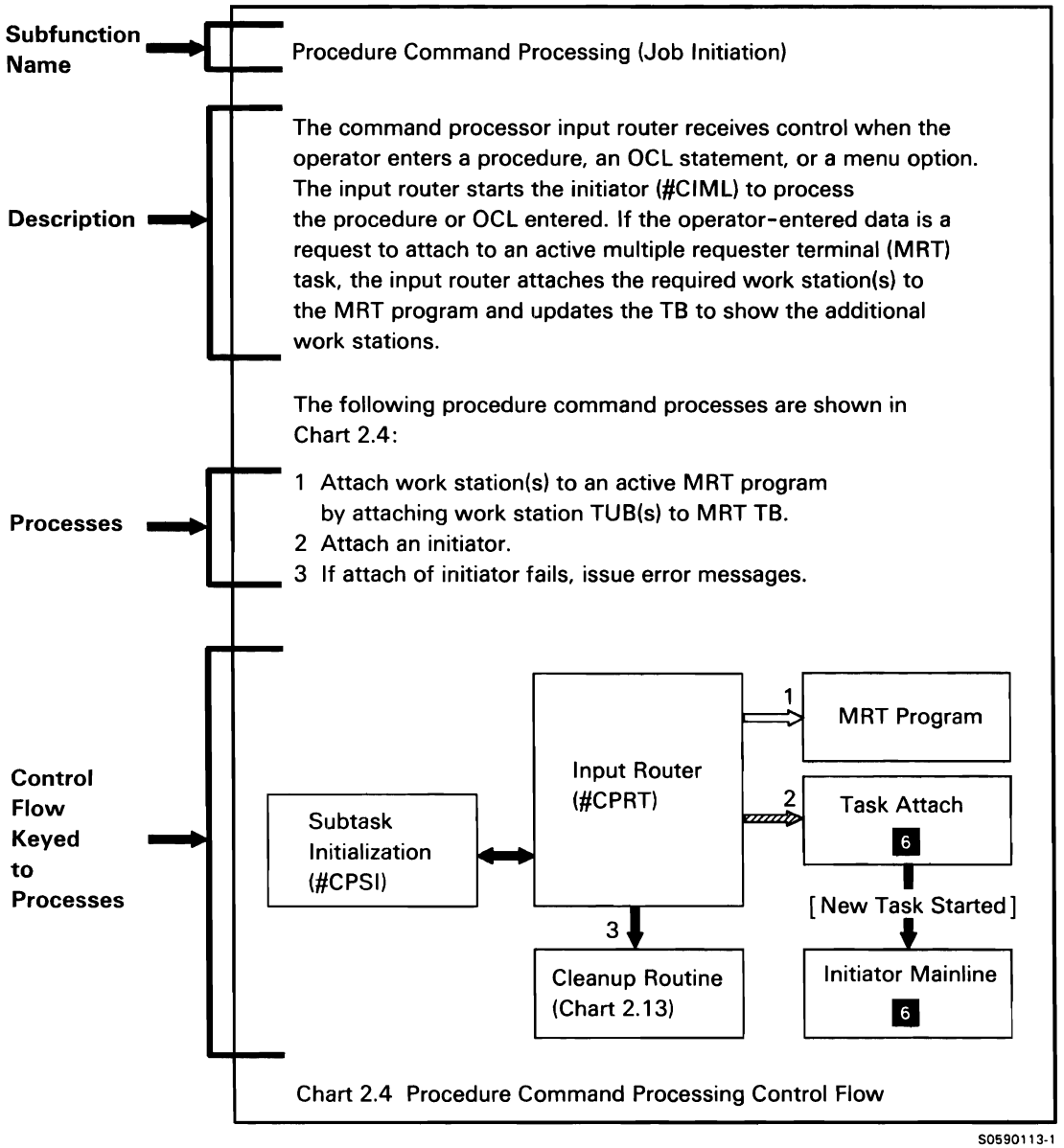


Chart 2.4 Procedure Command Processing Control Flow

S0590113-1

Notice that this example includes the subfunction name, a description of the subfunction, a list of the processes performed, and a diagram of the control flow from one module/routine to another. As shown in this example, the charts refer to modules/routines in three ways:

- *Module's descriptive name with chart number reference*: This type of reference indicates that the module is described in more detail in the referenced chart within this major function. In the example, a more detailed description of the command processor cleanup routines can be found in Chart 2.13, which is also a part of *Command Processing Function* 2 .

- *Module's descriptive name and a reverse number*: This type of reference indicates that the module is part of a different major function; a more detailed description can be found in the referenced major function. References in this example show that the supervisor task attach transient module is part of the *System Services Function* 6 , and the initiator is part of the *Job Start Function* 3 .

- *Module's descriptive name and module name*: This type of reference indicates that the module is not described in *more* detail anywhere else in this manual; this does not mean that this is the only chart in which the module name is used. If more than one control flow path to a module exists, the module name will appear in more than one control flow chart.

  For example, the module name #CPRT may appear in more than one control flow chart, but the description will not be any more detailed than what is shown in this chart. The directory entry (in Section 4) for each module lists all charts in which the module name is used.

The example also gives the following information about the control flow for this subfunction:

- For all three of the listed processes, control must be passed through command processor resident mainline.

- The command processor input router (#CPRT) performs the mainline (routing) functions for procedure command processing.

- If required, work stations are attached to an active MRT by command processor input router (#CPRT).

- Three modules, command processor resident mainline, command processor input router (#CPRT), and supervisor task attach, are involved in starting the initiator.

- Three command processor modules, command processor resident mainline, command processor input router (#CPRT), and the cleanup routine, are involved in processing initiation failure.

CONTROL FLOW CHARTS

The control flow charts in this section do not show *all* module-to-module control flow. Only modules that perform a distinct function in a particular process are shown. Hence, a module that performs a general function (such as SYSLOG or SYSLIST) within a major function may not be shown.

The control flow charts show the *principal* modules or routines involved in the execution of a given subfunction, and the processes they perform for that subfunction. The charts are not intended to show the mechanics of control flow. Therefore, control flow via post SVC instructions is shown with little or no distinction from control flow via transfer SVC instructions.

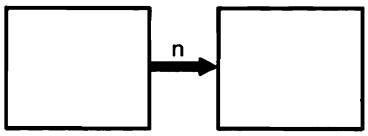
Chart Conventions

Generally, the charts show control flow from top to bottom, and from left to right. The following conventions are used to illustrate control flow in this manual ('n' represents a numbered process).

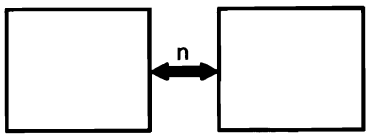
This block represents a module passing control for the purposes stated in process n.

This block shows a routine within a module receiving control to perform the process(s) stated in process n.

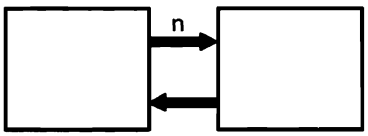
This block shows a procedure passing control for the purposes stated in process n.



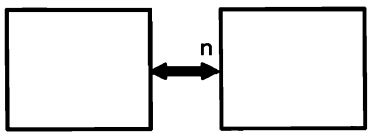
When process n is performed, control is passed unconditionally to the receiving module, as in a transfer SVC instruction with RETURN-NO, WAIT-NO.



Control is passed to the receiving module for the purposes stated in process n and then returned to the next sequential instruction (NSI) of the calling module, as in a transfer SVC instruction with RETURN-YES. (The called module performs a subroutine-like function.)



Control is passed to the receiving module for the purpose stated in process n and then returned to the caller at a point *that is not necessarily the NSI* of the calling module (as in subtasking, or event wait/post with WAIT-NO).

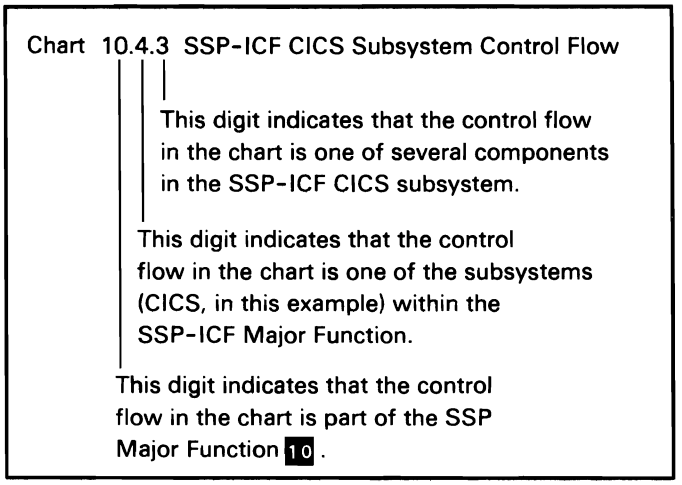


If there is no module performing an apparent mainline function, or if there may be doubt as to which module or routine (on either side of a double-headed arrow) is performing process n, the process number is located on the arrow, close to the executing module or routine.

Arrows that enter a module or subroutine block, with another arrow in the other side of the box, imply a through arrow. Although the module or subroutine may perform some process other than routing, the arrows on both sides of the block apply to the same numbered process (n).

Chart Numbering

The charts in this section are numbered by major function and subfunctions within major functions. The first digit (1 through 10) of the chart number shows the major SSP function to which the subfunction belongs. The second digit represents a breakdown of the major function into subfunctions. Subsequent digits, if present, represent further break down of the subfunction. The example below explains the chart numbering system:



S0590129-1

Any chart number ending with a 0 is an overview chart. Depending on the complexity of the component's control flow, the overview chart may, or may not, be accompanied by more detailed charts.

System Start Function 1

The system start function consists of the subfunctions used to initialize the system. Although subfunctions within this function can be independently evoked, the system start function generally receives control from control storage IPL (CSIPL). When completed, this function passes control to the command processor, which it previously started up as a permanent task. Configuration is included in this major function because its output is closely related to main storage IPL. The system start function includes the following subfunctions:

- Main storage initial program load (MSIPL) — Chart 1.1.0
- System configuration — Chart 1.2.1
- SSP-ICF configuration — Chart 1.3.0

MAIN STORAGE INITIAL PROGRAM LOAD (IPL) SUBFUNCTION

Main storage IPL is executed after control storage IPL (CSIPL) starts the main storage processor. Main storage IPL executes under the command processor task block and performs the following SSP initialization and housekeeping functions:

- Initializes nucleus control blocks
- Runs the cross-reference resolver
- Processes work station configuration data
- Allocates and initializes areas for display stations and printers
- Initializes communications configuration data
- Loads resident routines
- Initiates the IPL sign-on sequence
- Initiates the IPL start-up procedure
- Processes IPL overrides

Main storage initialization uses the following input during execution:

- System communications area (SCA)
- Configuration record
- VTOC
- Task work area (TWA)
- Main storage nucleus

Main storage IPL can be performed from either disk or diskette. When loaded from diskettes, the IPL program can be either from IBM-supplied diskettes, as on initial installation, or from user diskettes generated by the SAVELIBR procedure.

Chart 1.1.0 shows overview control flow for both disk and diskette IPL:

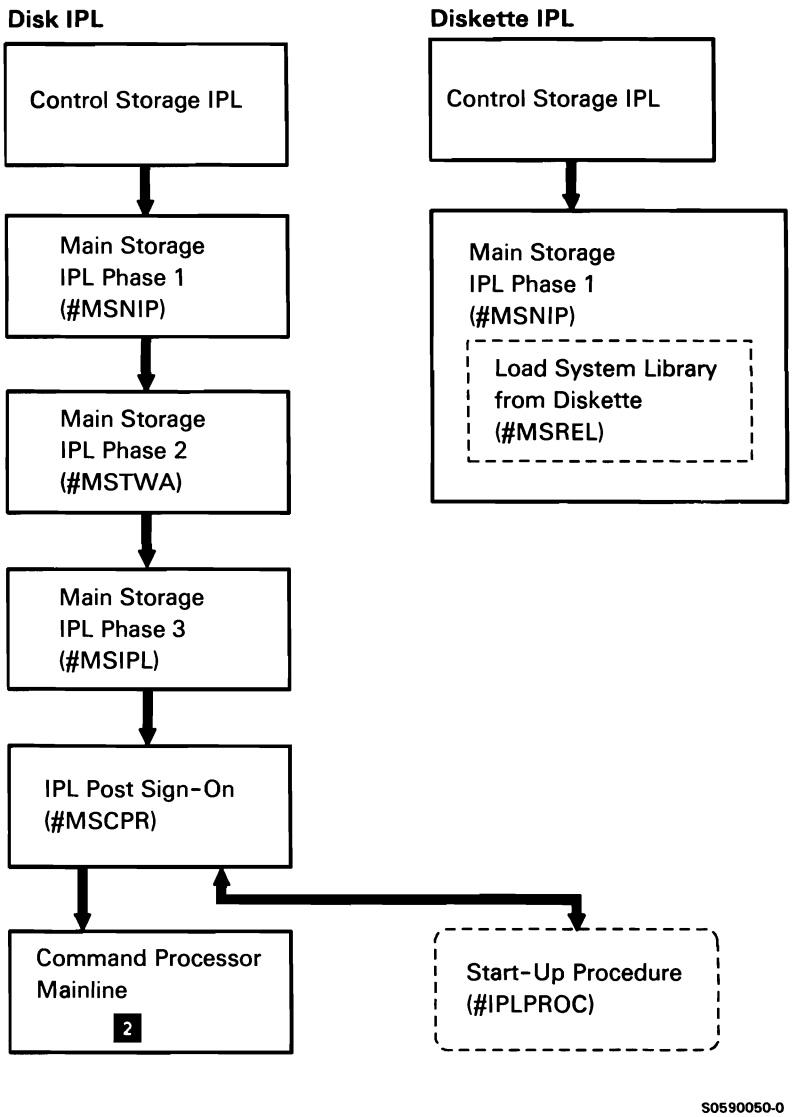


Chart 1.1.0 Main Storage IPL Overview Control Flow

Main Storage Initial Program Load Phase 1

The primary function of main storage initial program load (IPL) phase 1 is to initialize the nucleus to a point at which subsequent IPL phases can operate. The first 4K bytes of phase 1 is called by control storage as a transient, but it transfers to itself to run translated; this moves phase 1 out of real storage so it does not interfere with nucleus initialization.

If the IPL is from diskette, the remainder of the IPL code is read from the diskette; if the IPL is from disk, the remaining IPL code is read from the system library.

The following MSIPL phase 1 processes are shown in Chart 1.1.1:

1 Do the following:

- Build, assign, and queue disk and diskette error recovery blocks (ERBs).
- Initialize system communications area (SCA).
- Initialize transfer control table and #LIBRARY format 1.
- Validate SSP by checking for existence and validity of #LIBRARY, the VTOC, the system files, and the configuration record.

2 Read in remainder of IPL code from diskette or from disk.

3 If SSP is to be loaded, if system areas are to be relocated, or if microcode (DIAG24) is to be loaded, do the following:

- Perform additional SSP verification by checking system area VTOC pointers.
- Prompt for user ID and password if IPL is from diskette (MODE-F).
- Build, update and relocate system areas: #SYSHIST, #SYSTASK (in TWA), #SYSWORK (Configuration record and VTOC), and #LIBRARY.
- Load SSP from diskettes.
- If requested, load microcode from diskette.

4 Load nucleus-resident routines.

5 Set up where-to-go (WTG) table and transfer control tables.

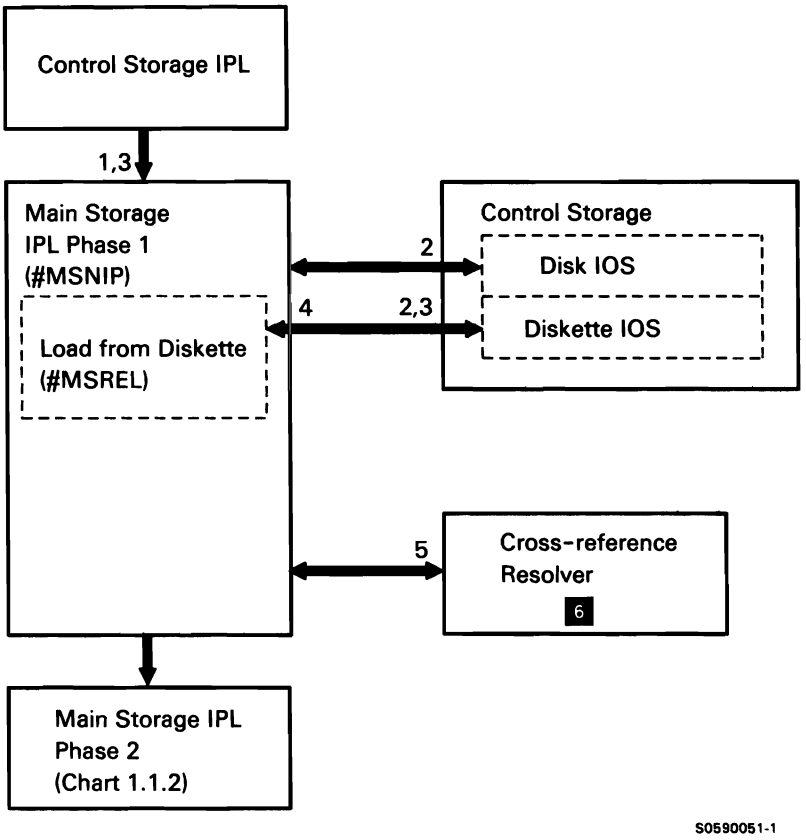


Chart 1.1.1 Main Storage IPL Phase 1 Control Flow

Main Storage Initial Program Load Phase 2

The primary function of main storage IPL phase 2 is to initialize the work stations and their associated control blocks, work areas and configuration records. All input/output operations for phase 2 are done using disk IOS (control storage). Chart 1.1.2 shows the following processes performed by main storage IPL phase 2:

1 Do the following:

- If applicable, build communications configuration record and place it in the system configuration record.
- Initialize the task work area (TWA).
- Perform auto configuration by checking for new configuration required, using the UDT (unit definition table), the native printer indicator in the SCA, and the work station controller configuration tables.

2 Allocate and initialize all configuration records for local and remote work stations.

3 Assign and build any additional device unit blocks required.

4 Build TUBs and PUBs for all local work stations.

5 Build device allocate table.

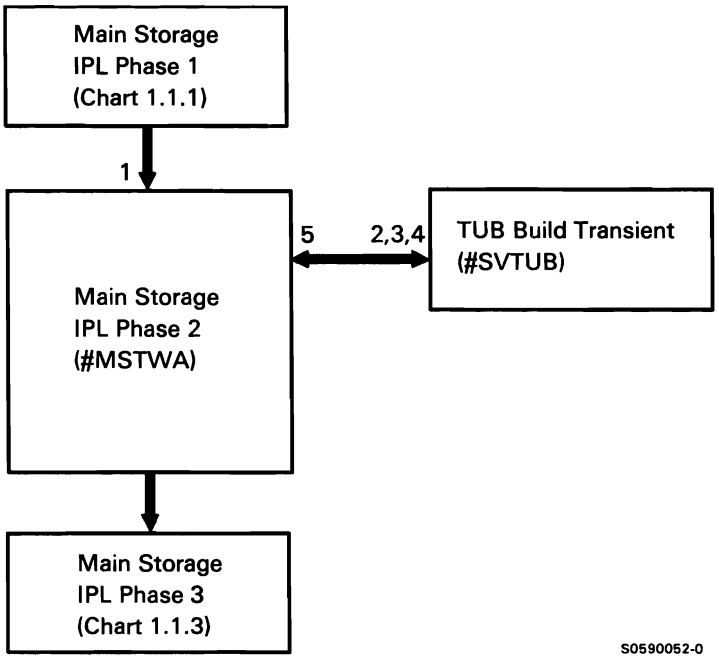


Chart 1.1.2 Main Storage IPL Phase 2 Control Flow

Main Storage Initial Program Load Phase 3

Phase 3 of main storage IPL (#MSIPL) performs sign-on initialization. The following MSIPL phase 3 functions are shown in Chart 1.1.3:

1 Perform mainline functions for the following:

2 Perform trace and security initialization.

3 Wait for an indication of a signed-on terminal unit block (TUB).

4 Assign and build subconsole TUB; clean up TUB chain console attributes for alternative console, if required.

5 Log any error data passed from the control storage IPL.

6 Write sign-on date to system configuration record.

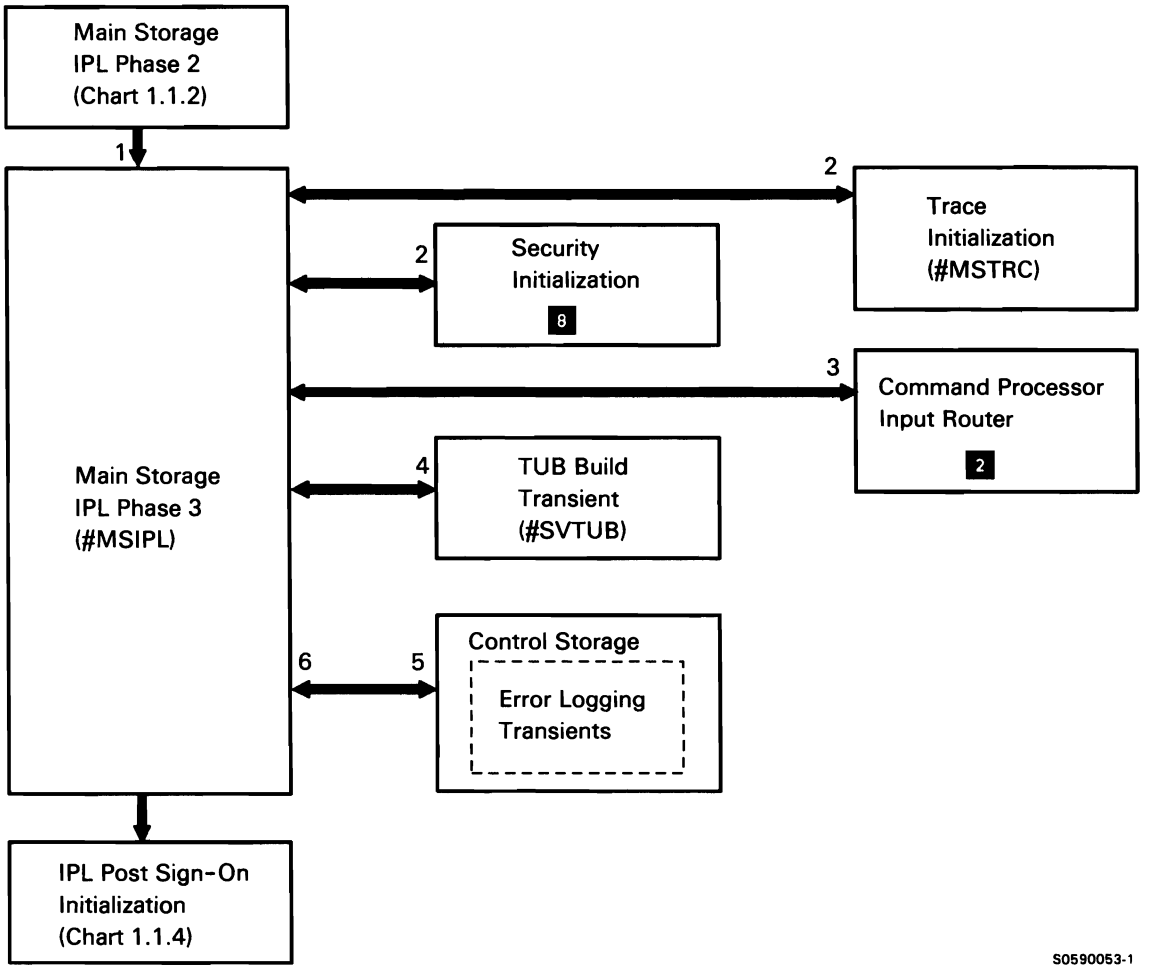


Chart 1.1.3 Main Storage IPL Phase 3 Control Flow

IPL Post Sign-On Initialization

After IPL phase 3 has executed, the system is functionally ready to process user data, but all SSP facilities have not been initialized and the command processor has not yet received control. IPL post sign-on completes the IPL initialization of SSP and passes control to the command processor.

The following post sign-on initialization functions are shown in Chart 1.1.4:

1 Set up override defaults and route for the following:

2 Perform text management services initialization.

3 Start IPL start-up procedure (#IPLPROC). #IPLPROC loads modules needed in subsequent processes.

4 Wait for start-up procedure completion and perform system file initialization.

5 Attach, as required, the autocall, X.21, or the EXTN task.

6 Attach remote work station tasks.

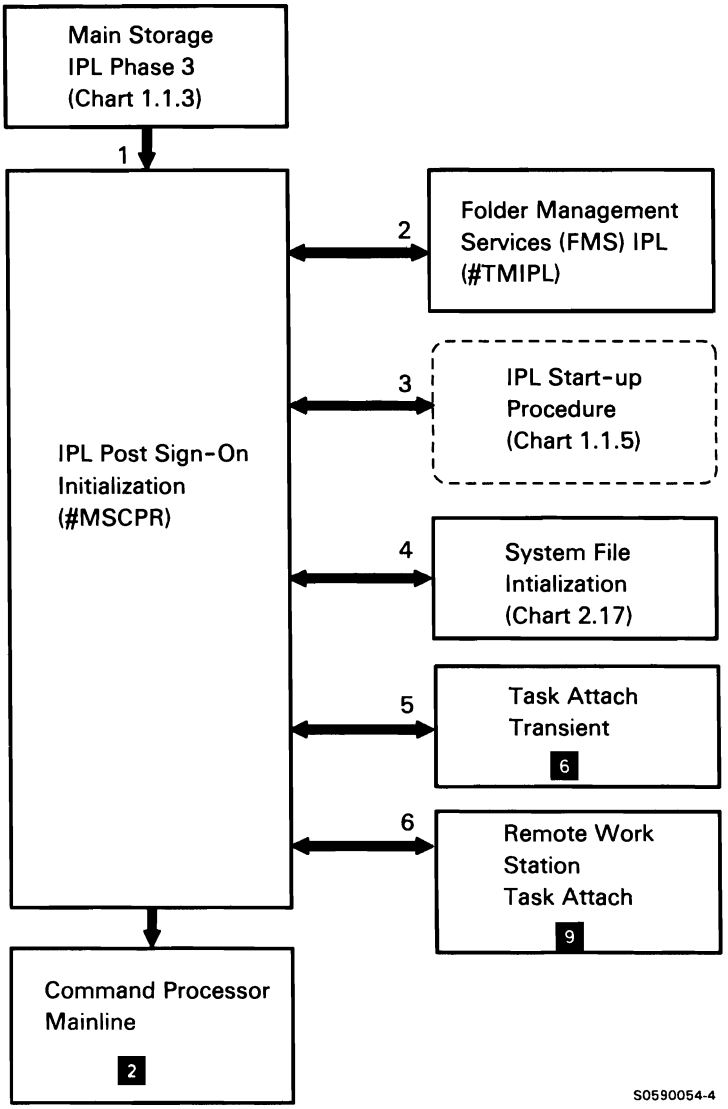


Chart 1.1.4 IPL Post Sign-On Control Flow

IPL Start-up Procedure

The IPL start-up procedure provides for the processing of user-selected start-up options. The start-up procedure allows the user to run procedures and programs that require a dedicated system and/or must be run each time the system is started up. IPL start-up can conditionally run APAR, drop optional SSP support, and apply PTF.

The following IPL start-up procedure processes are shown in Chart 1.1.5:

1 If selected, process IPL overrides.

2 If selected, rebuild VTOC format 1's.

3 If selected, rebuild FMS folders.

4 If selected, run APAR, drop optional SSP, or APPLYPTF.

5 Update where-to-go (WTG) tables.

6 If selected, run user IPL start-up procedure 1.

7 If selected, run user IPL start-up procedure 2.

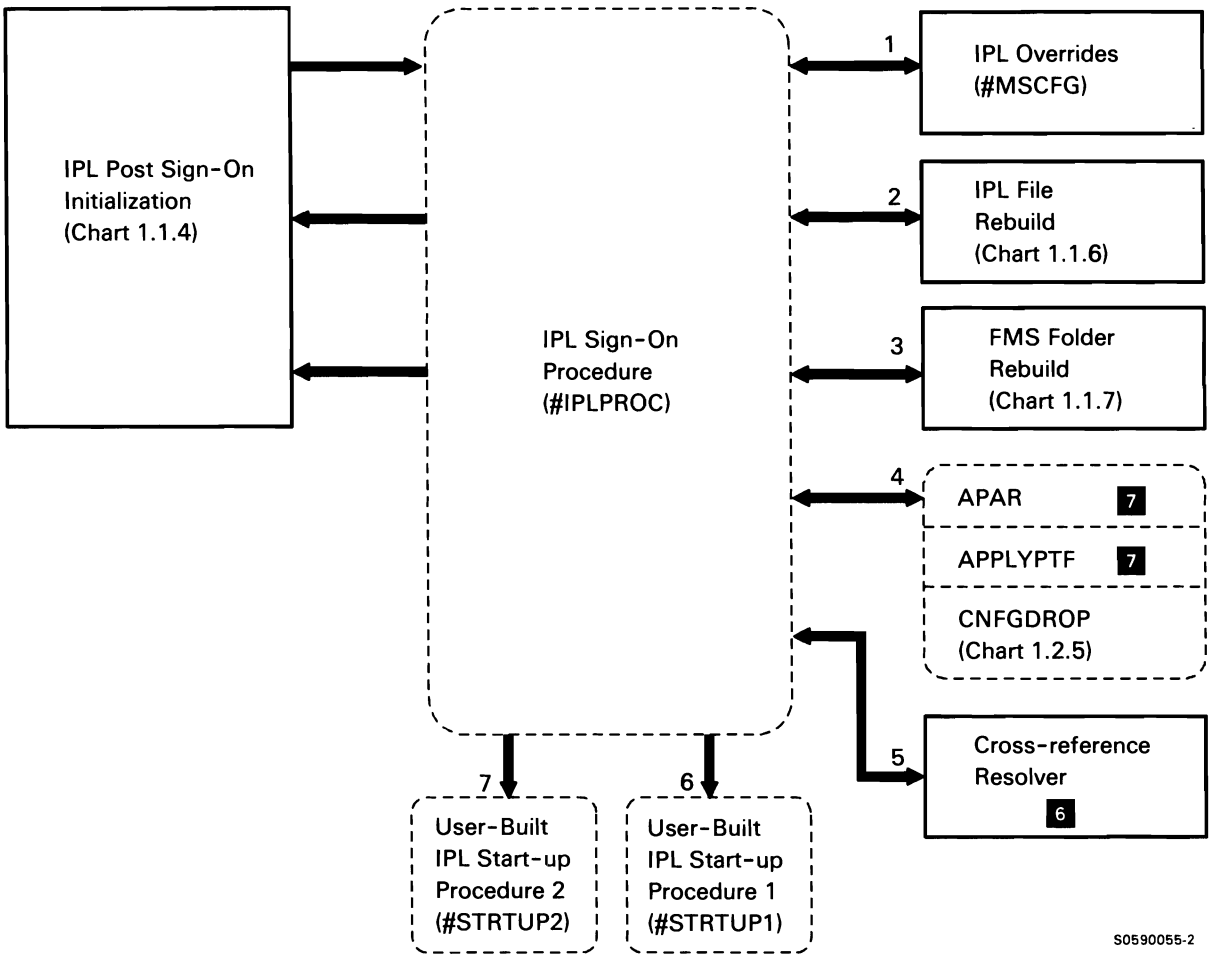


Chart 1.1.5 IPL Start-up Procedure Control Flow

IPL File Rebuild

The IPL file rebuild subfunction performs reasonability checks on the disk VTOC to ensure the integrity of the disk file format 1's. All phases of IPL file rebuild use control storage disk IOS to access the disk VTOC.

The following IPL file rebuild processes are shown in Chart 1.1.6:

1 Initialize for VTOC rebuild.

2 Initialize common area and verify correct format 1's for system and nonindexed user files.

3 Continue interrupted disk reorganizations.

4 Issue messages for any uncorrectable problems.

5 Verify and correct format 1's for user indexed files.

6 If necessary, reconstruct index.

7 Perform required keysorting.

8 For date chain files, ensure that all files with latest date have latest date indicator.

9 Compress the VTOC.

10 Reset the format 5.

11 Terminate this job step.

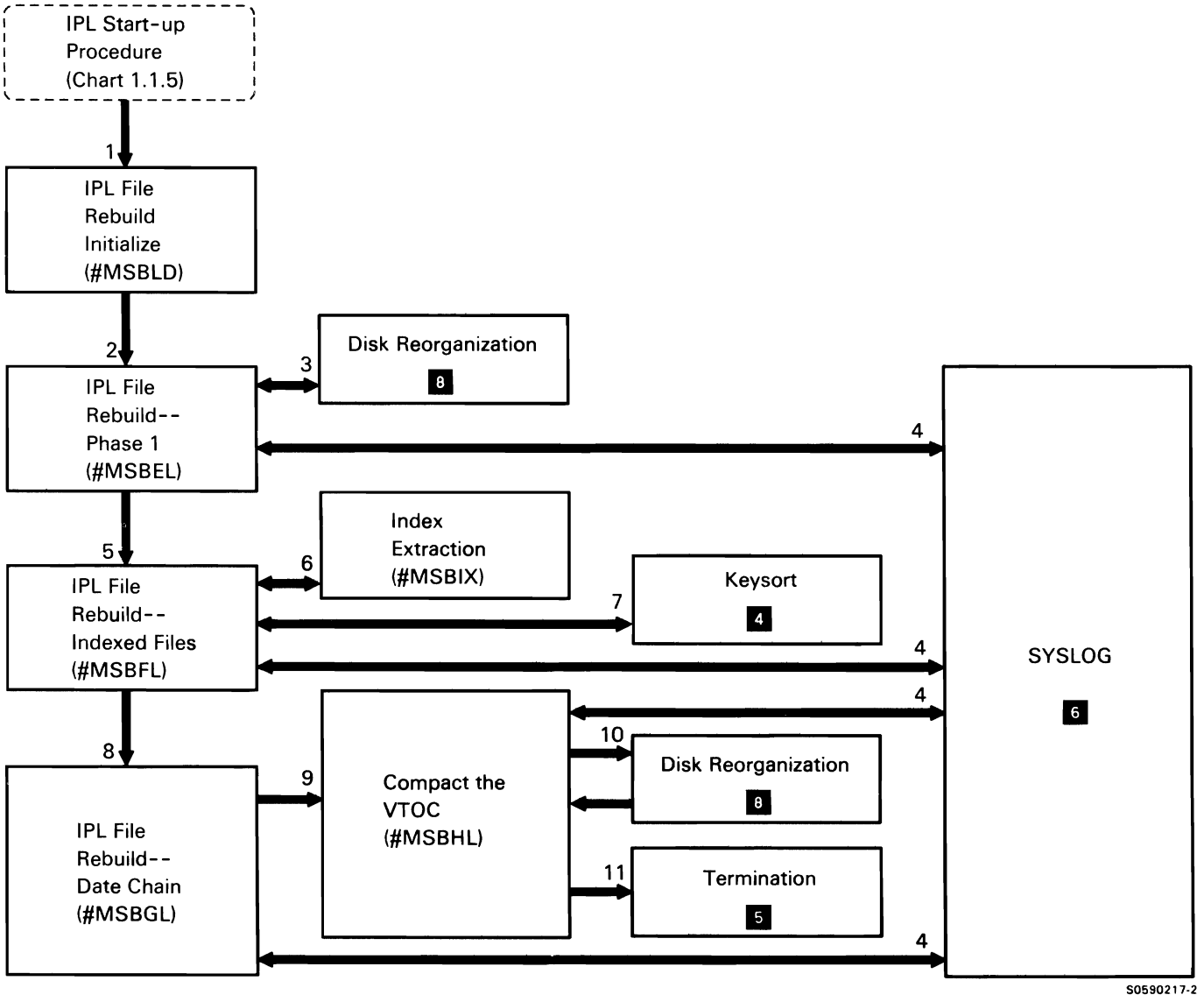


Chart 1.1.6 IPL File Rebuild Control Flow

IPL Folder Management Services (FMS) Folder Rebuild

IPL folder management services (FMS) folder rebuild checks the folders on the system to determine if any folders need to be rebuilt. The rebuild is performed if any of the following conditions exist:

- A folder's release level is incompatible with the current system release level.
- A folder's type header ID field does not contain the constant 'TMDT'.
- A folder is open when the system IPL is executed.

The following IPL FMS folder rebuild processes are shown in Chart 1.1.7:

1 Search disk VTOC for FMS folders.

2 Read folder header and type headers to check for the rebuild criteria listed above.

3 If required, rebuild the specified folder.

4 Issue any required error messages for #TMRBD.

5 Perform FMS cleanup processing.

6 Return to caller via EXIT SVC instruction.

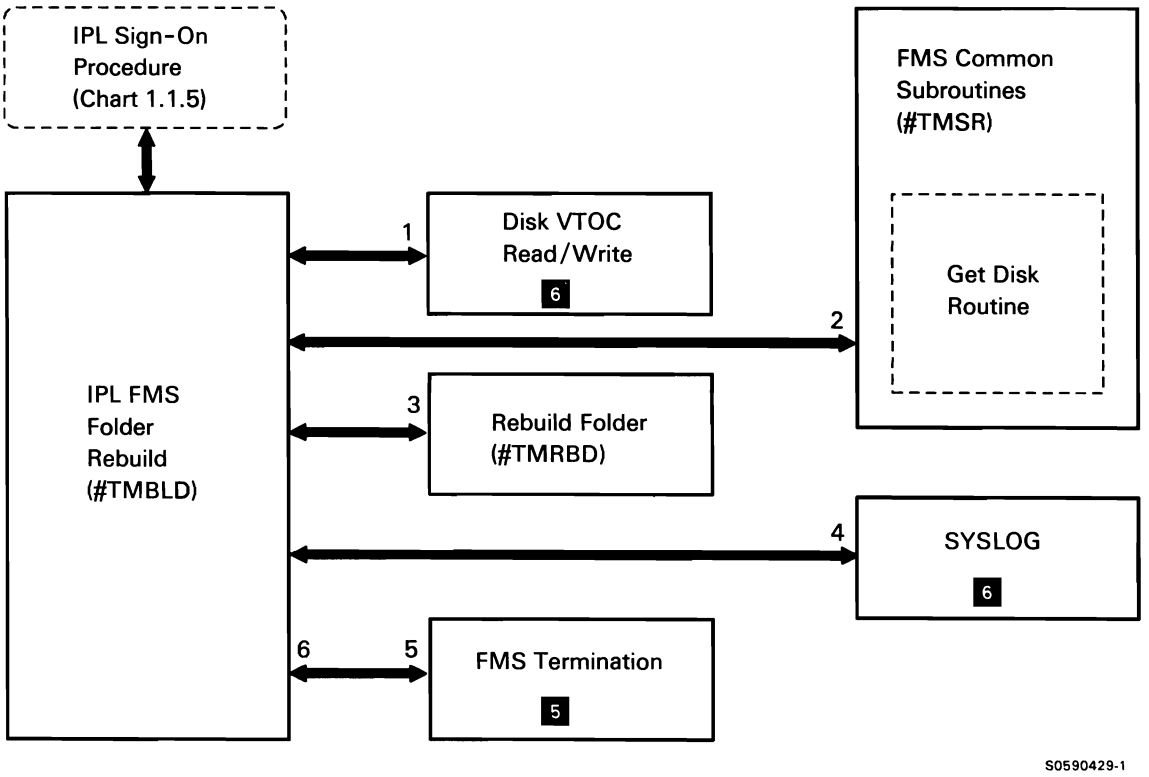


Chart 1.1.7 IPL Folder Management Services (FMS) Folder Rebuild Control Flow

SYSTEM CONFIGURATION SUBFUNCTION

System configuration is performed when the system is initially installed or any time a system or feature change requires configuration. The configuration information is saved in the configuration records. (See the *System Data Areas* manual for a description of the configuration records.)

The user can override certain configuration options at IPL or alter the configuration of each work station by using the set configuration utility ($SETCF) or alter the communications configuration record by using the communications personalization utility ($SETCP). The CNFDROP procedure, which prompts for the dropping of optional SSP support, can be invoked by the IPL startup procedure.

System configuration is initiated with the CNFIGSSP command following the initial loading of the base SSP from disk. (See the manual *Performing the First System Configuration For Your System* or the manual *Changing Your System Configuration.*)

Chart 1.2.1 shows control flow for the following system configuration processes:

1 Initialize common areas and open required files for system configuration.

2 Display main system configuration menu and route according to menu item(s) selected.

3 Process menu item 1, *How to Use CNFIGSSP*, by displaying menu for CNFIGSSP help text.

4 Process menu item 2, *Create, Change, or Delete a Configuration Member*.

5 Process menu item 3, *Review a Configuration*, by displaying a configuration member or the master configuration record.

6 Process menu item 4, *Print Configuration*, by formatting and printing a configuration member or the master configuration record.

7 Process menu item 10, *Configuration Support Aids*:

- Extend a library.
- Condense a library.
- Calculate #LIBRARY size and main storage user space available.
- Calculate the number of diskettes needed to save a library.

8 Read in configuration member.

9 Process menu item 12, *Apply Changes to the Master Configuration Record*, by applying changes from a configuration member.

10 Process menu item 13, *Rebuild the Master Configuration Record (Update to Next Release)*, using either a configuration member or the master configuration record. Refresh optional support.

11 Write master configuration record and return to CNFIGSSP procedure to copy required SSP support from software distribution diskettes.

12 Set flags in LDA based on configuration changes selected by operator.

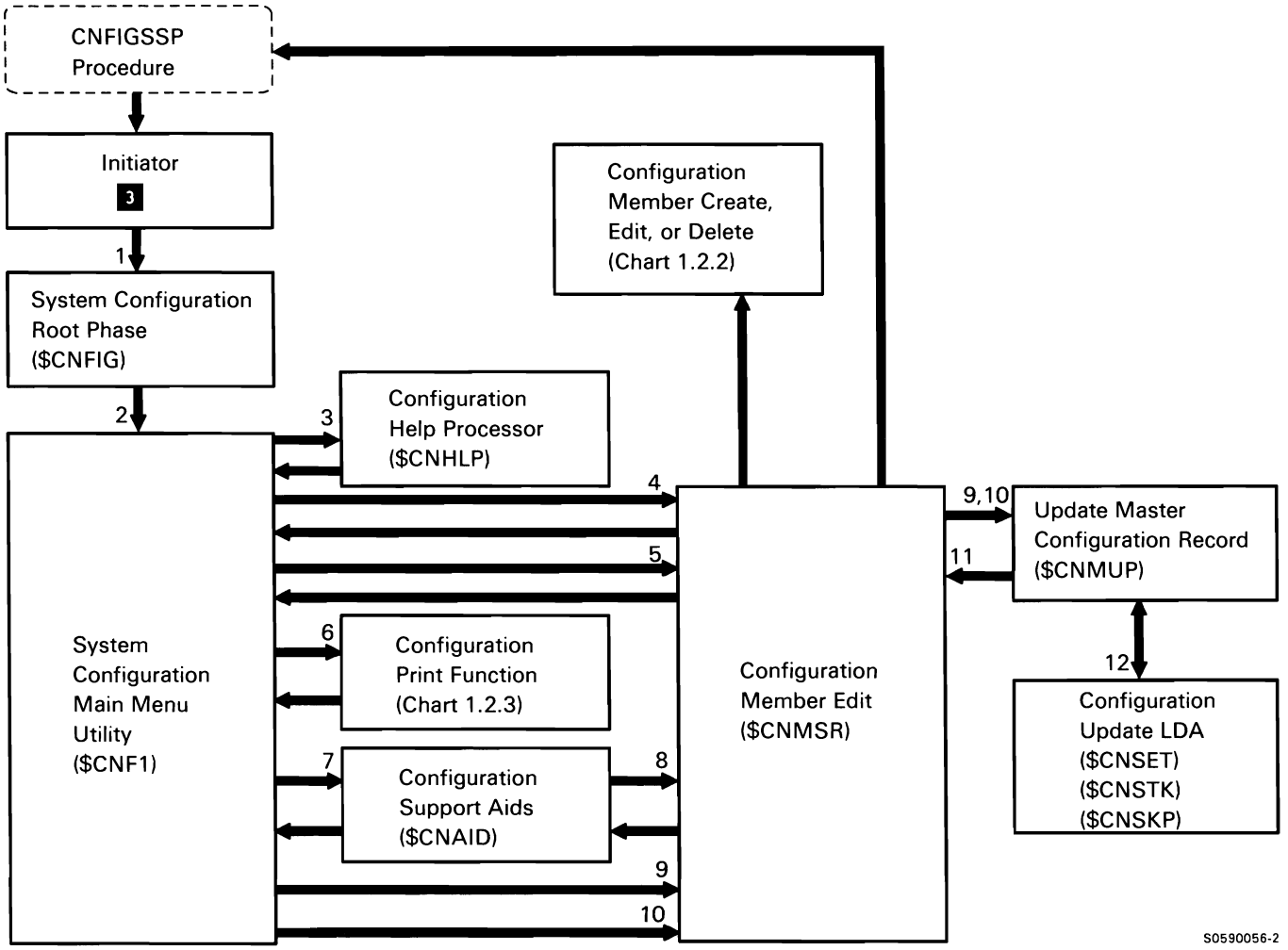


Chart 1.2.1 System Configuration Main Menu Routing Control Flow

Create, Edit, or Delete a Configuration Member

The following create, edit, or delete processes are shown in Chart 1.2.2:

1 Process configuration menu 5, option 5 by writing configuration member to library and returning to main menu.

2 Perform routing for menu 5 options.

3 Process option 1, by prompting for local and remote work station support.

4 Process option 2. Prompt for and set flags in configuration member for changes to program product, SSP features, and optional SSP.

5 Prompt for and set flags in configuration member for changes to SSP features and optional SSP.

6 Select specified translate tables for BSC 3270 and SNA 3270 device emulation, and 3278/3279 emulation via IBM Personal Computer.

7 Process option 3, setting flags and saving values for changes to base SSP.

8 Process option 4, saving values for TWA, VTOC, or history file size changes.

9 Read in member; then return to support aid processing.

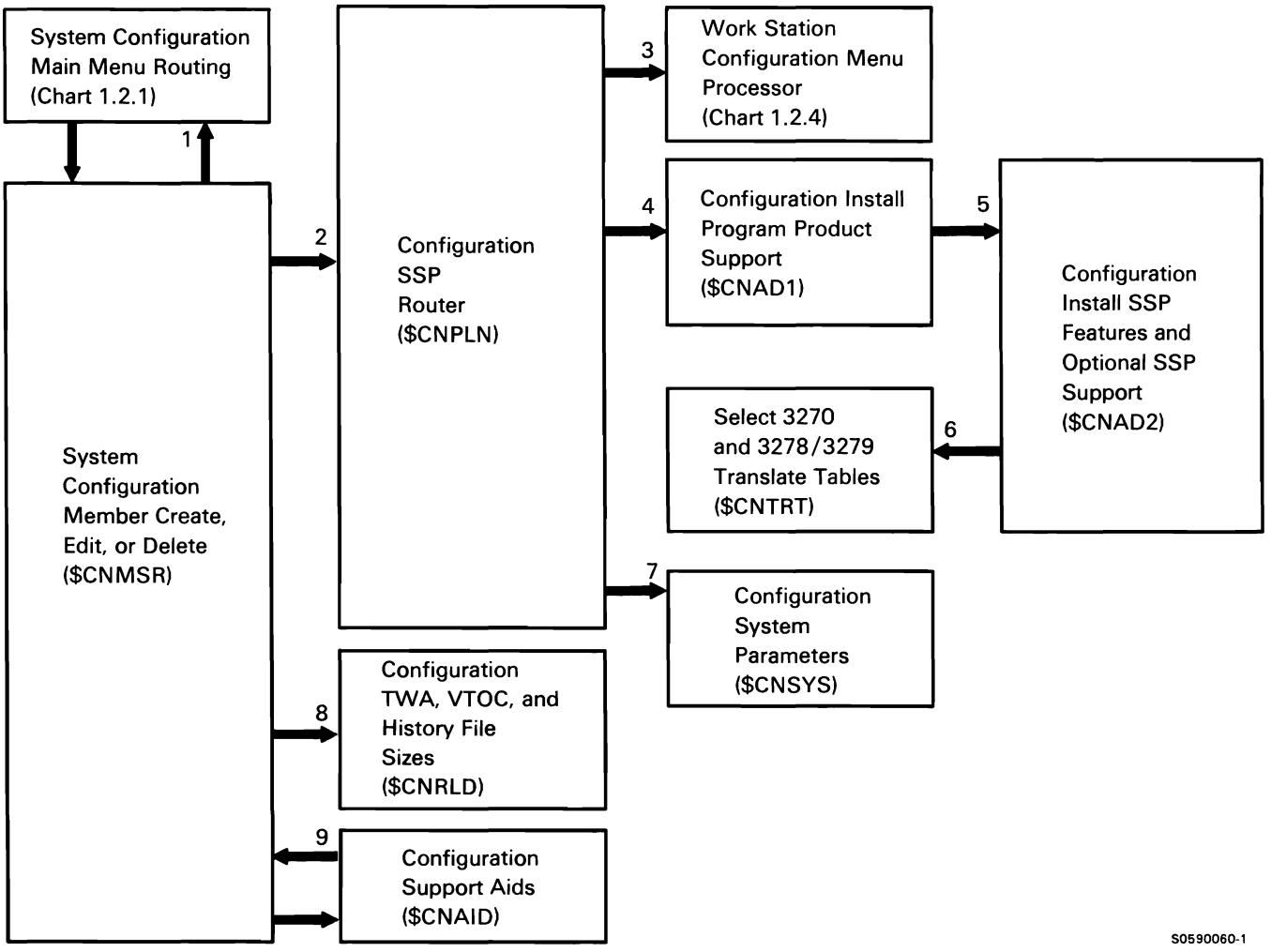


Chart 1.2.2 Create, Edit, or Delete Configuration Member Control Flow

CNFIGSSP Print

CNFIGSSP print provides options for printing information from configuration records.

The following print function processes are shown in Chart 1.2.3:

1 Display print option menu and process options selected:

- Process command key 3 by returning to display CNFIGSSP main menu.
- Display configuration member definition screen and process data entered.
- Process command key 19 by exiting CNFIGSSP.

2 Print requested documents.

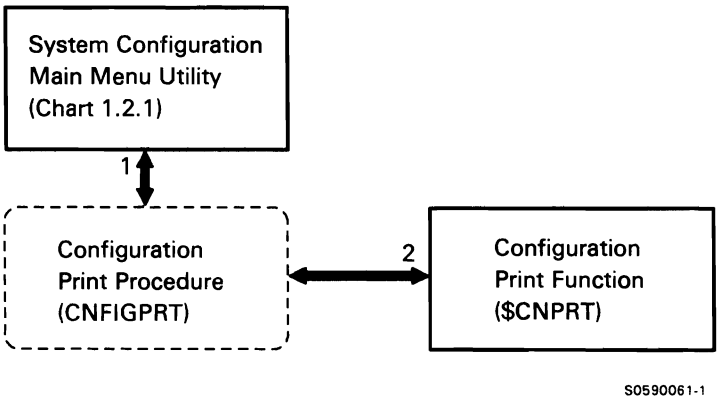


Chart 1.2.3 CNFIGSSP Print Control Flow

Work Station Configuration

Work station configuration provides prompting for all local and remote work station configuration. Based on operator responses, work station configuration modifies individual 64-byte work station table entries in a configuration member.

The following work station configuration processes are shown in Chart 1.2.4:

1 Process menu item 1, *Add or Delete Local Display Stations and Printers.*

2 Process menu item 4, *Assign Default Printers to Display Stations.*

3 Process menu item 5, *Assign Display Station Control (Subconsoles) for Printers.*

4 Process menu item 6, *Change Display Station or Printer Work Station IDs.* If print is active, pass control to print module, Chart 1.2.6.

5 Process menu item 7, *Change Display Station or Printer Characteristics.*

6 Process menu item 8, *Select the System Printer.*

7 Perform initial validity checks on work station assignments specified on screen 28.

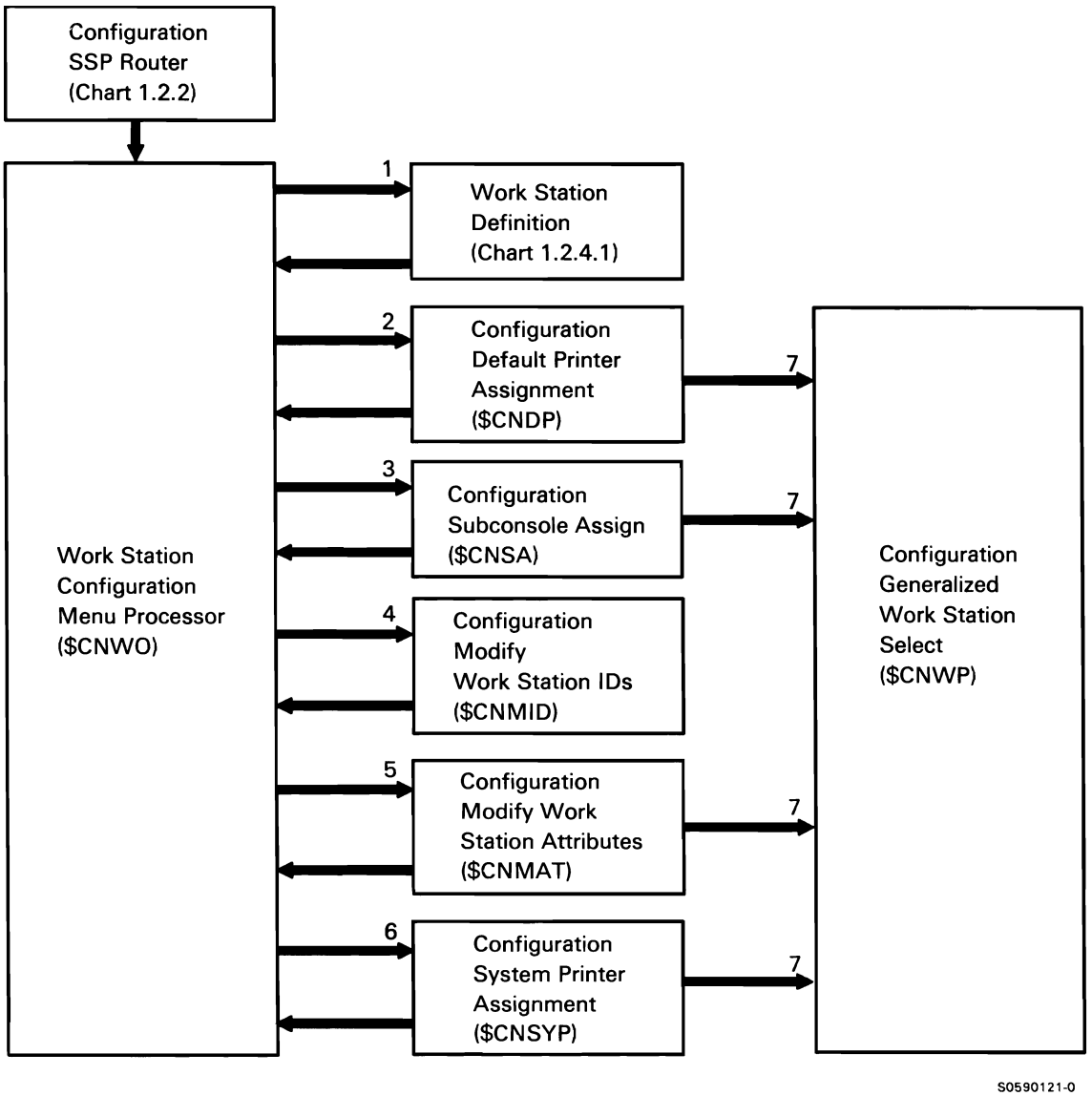


Chart 1.2.4 Work Station Configuration Control Flow

Work Station Definition

Work station definition modifies the configuration records for local and remote work stations. The following work station definition processes are shown in Chart 1.2.4.1:

1 Process menu item 2, *Add or Delete Remote Line Characteristics*, by updating remote work station line attributes in configuration member.

2 Process menu item 1, *Add or Delete Local Display Stations and Printers*, by altering local work station table entries in configuration member.

3 Process menu item 3, *Add or Delete Remote Controllers, Display Stations and Printers*, by altering remote work station table entries in configuration member.

4 Process menu item 9, *Add Remote Service Device Definition*, by building remote table entry, assigning controller ID of CFE, attaching printer and, if required, changing remote work station line 1 attributes.

5 Process menu item 10, *Delete Remote Service Device Definition*, by deleting entries made in process 4 above (except line attributes).

6 Assign local and remote work station addresses based on physical locations; assign work station and controller IDs based on default.

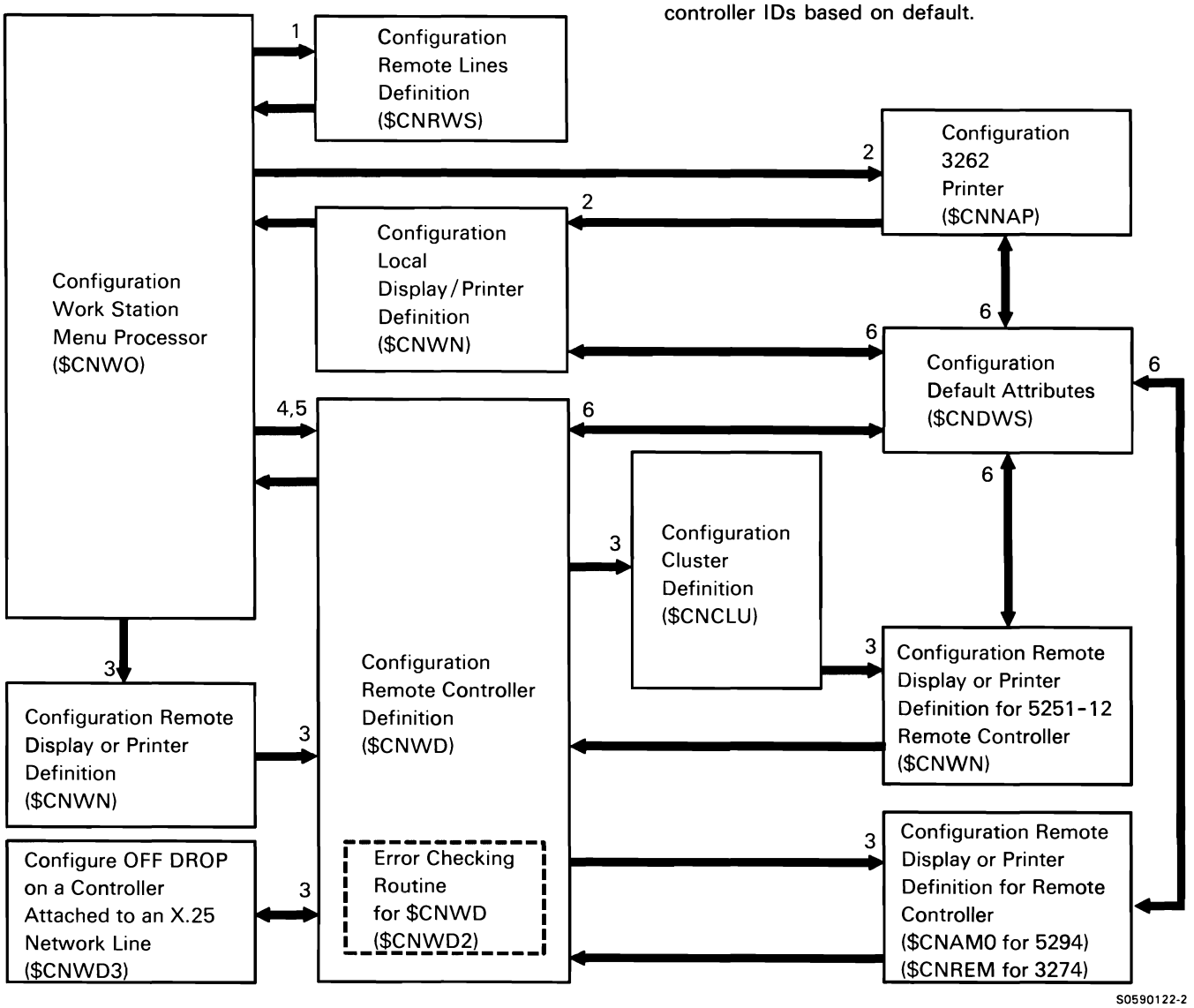


Chart 1.2.4.1 Work Station Definition Control Flow

Configuration Drop Support

Configuration drop support is invoked by the IPL start-up procedure. The procedure uses the local data area to prompt for and set optional SSP features to be dropped. The following drop support processes are shown in Chart 1.2.5:

1 Prompt for and set user requests for SSP and program product to be dropped by calling each of the procedures required.

2 Change system configuration record based on user responses.

3 Specify support to be dropped in LDA.

4 Remove specified library members.

5 Reset flags based on values set in LDA.

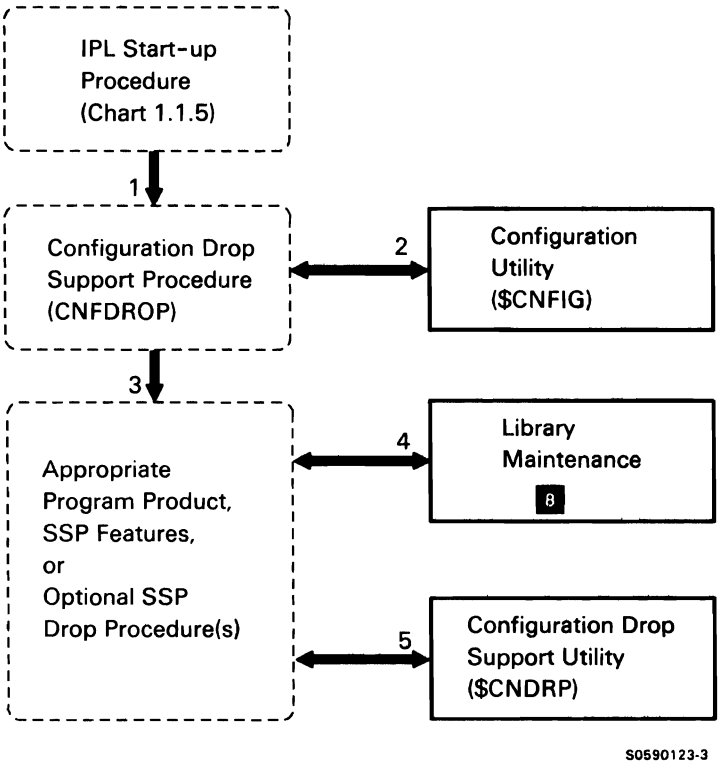


Chart 1.2.5 Drop Support Control Flow

Configuration Load Support

Configuration load support is invoked by the IPL start-up procedure. The procedure uses the local data area to prompt for and set optional SSP features to be dropped. The following drop support processes are shown in Chart 1.2.6:

1 Load programming support based on LDA settings. Load SSP features based on LDA settings.

2 Call individual load procedures for each program product, SSP feature, and optional SSP.

3 Create specified libraries.

4 Reset flags based on values set in LDA.

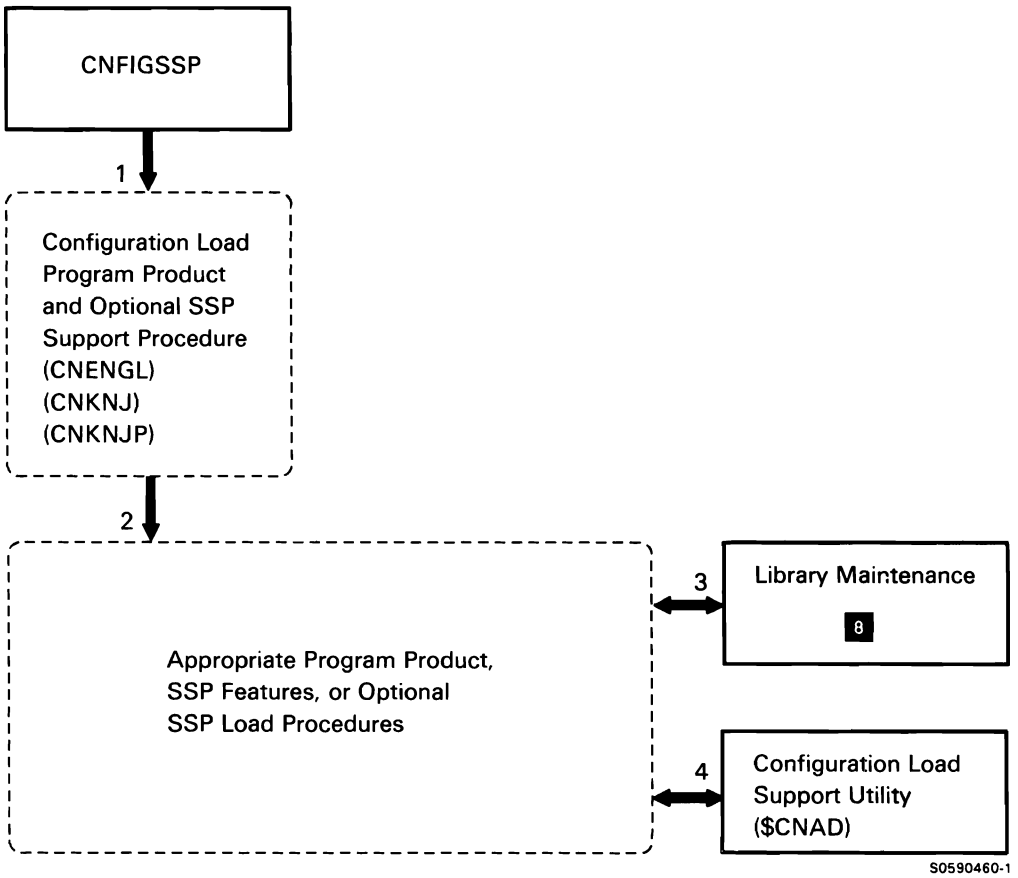


Chart 1.2.6 Load Support Control Flow

SSP-ICF CONFIGURATION SUBFUNCTION

SSP-ICF configuration provides for the creating, editing, deleting, and reviewing of SSP-ICF configuration members. SSP-ICF configuration processes two member types: line member (CNICL) and subsystem member (CNICS). These members are contained in the current library.

The SSP-ICF line configuration member defines the attributes of a communications line. The SSP-ICF subsystem configuration member defines the attributes of an SSP-ICF subsystem. The SSP-ICF line configuration member must exist before the subsystem configuration member for that subsystem is created.

SSP-ICF configuration is invoked via the CNFIGICF procedure command. After setting the configuration according to prompt responses from the operator, SSP-ICF configuration writes the configuration member back to disk. The CNFIGICF procedure defines the configuration member in display screens by individual SSP-ICF subsystem, allowing the user to modify all SSP-ICF configurations.

Note: For descriptions of the CNICL and CNICS (line and subsystem) members, refer to *SSP-ICF Configuration Record* in the *System Data Areas* manual.

This page is intentionally left blank.

The following SSP-ICF configuration overview processes are shown in Chart 1.3.0:

1 Initialize common areas for SSP-ICF configuration.

2 Display main menu for SSP-ICF line and subsystem configuration and route to next menu or perform menu-selected function.

3 Process menu item 1, *Create a member.*

4 Process menu item 2, *Edit a member.*

5 Process menu item 3, *Create a member from an existing member.*

6 Process menu item 4, *Remove a member.*

7 Process menu item 5, *Review a member.*

8 If SSP-ICF Intra subsystem configuration is requested, process the request.

9 If BSC, SNA, or asynchronous line configuration information is required, process the request.

10 If configuration information for a BSC subsystem is required, process the request.

11 If configuration information for an SNA subsystem is required, process the request.

12 If configuration information for an asynchronous subsystem is required, process the request.

13 Write the edited or new member to current library and return to main menu.

Note: For descriptions of the CNICL and CNICS (line and subsystem) members, refer to the *System Data Areas* manual.

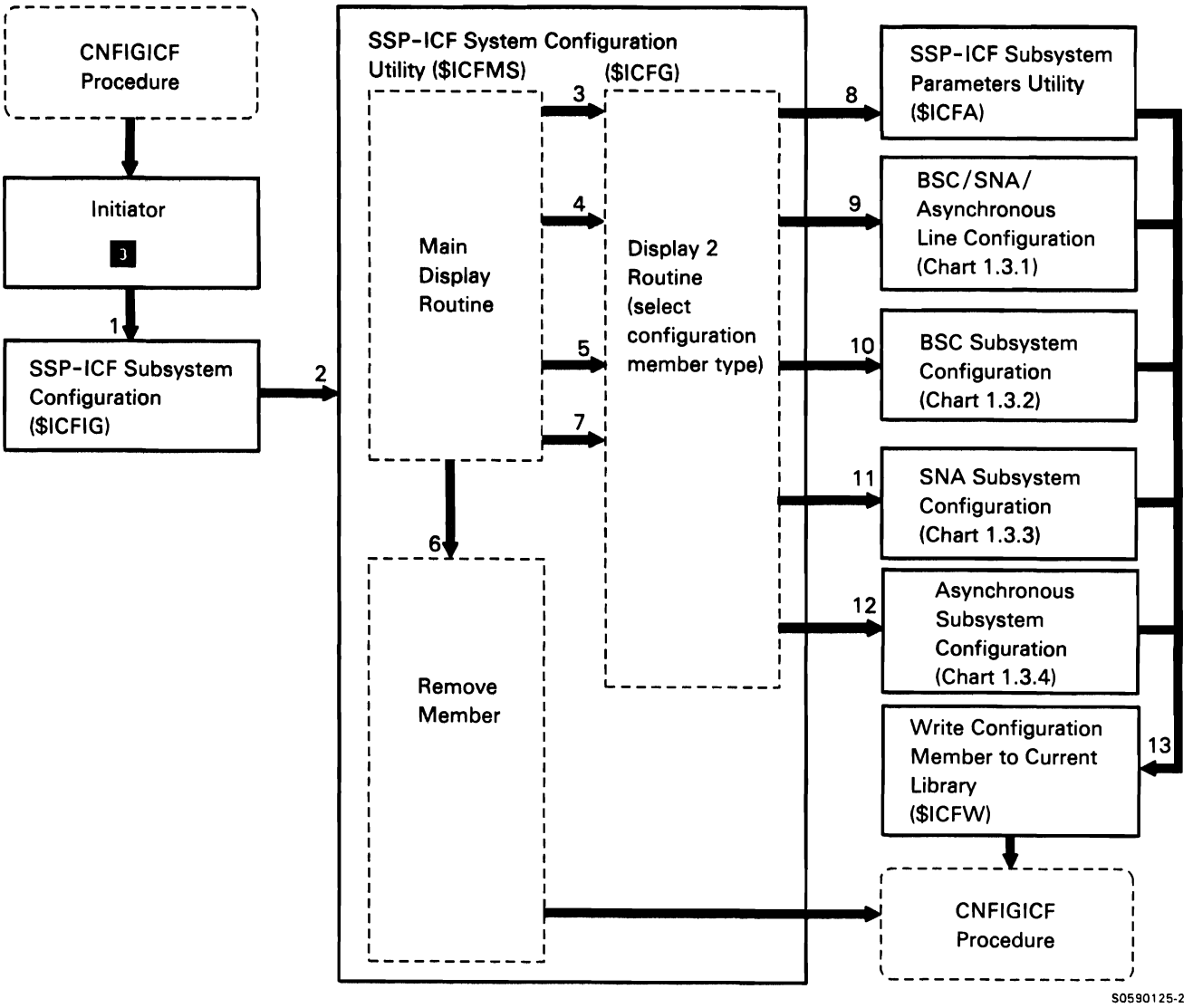


Chart 1.3.0 SSP-ICF Configuration Overview Control Flow

BSC/SNA/Asynchronous Line Configuration

BSC/SNA/asynchronous line configuration creates or edits line configuration members for BSC, SNA SSP-ICF, or asynchronous subsystems. The following BSC/SNA/asynchronous line configuration processes are shown in Chart 1.3.1:

1 If configuration is for BSC, display BSC screen 10.0 and store BSC line data in an SSP-ICF configuration member.

2 If configuration is for SNA, display SNA screen 12.0 and store SNA line data in an SSP-ICF configuration member.

3 If configuration is for asynchronous, display asynchronous screens 12.1 and 12.2 and store asynchronous line data in an SSP-ICF configuration member.

4 If configuration is for BSC-MSRJE, display BSC-MSRJE screens 10.5 and 11.0 and process the remote system parameters.

5 If configuration is for asynchronous, display asynchronous screen 12.5 to define remote systems.

6 If configuration is for SNA, display SNA screens 12.5 and 13.0 and process SNA remote system parameters.

7 If configuration is for SNA and LU information is needed, display screen 14.0, process responses, and update configuration member.

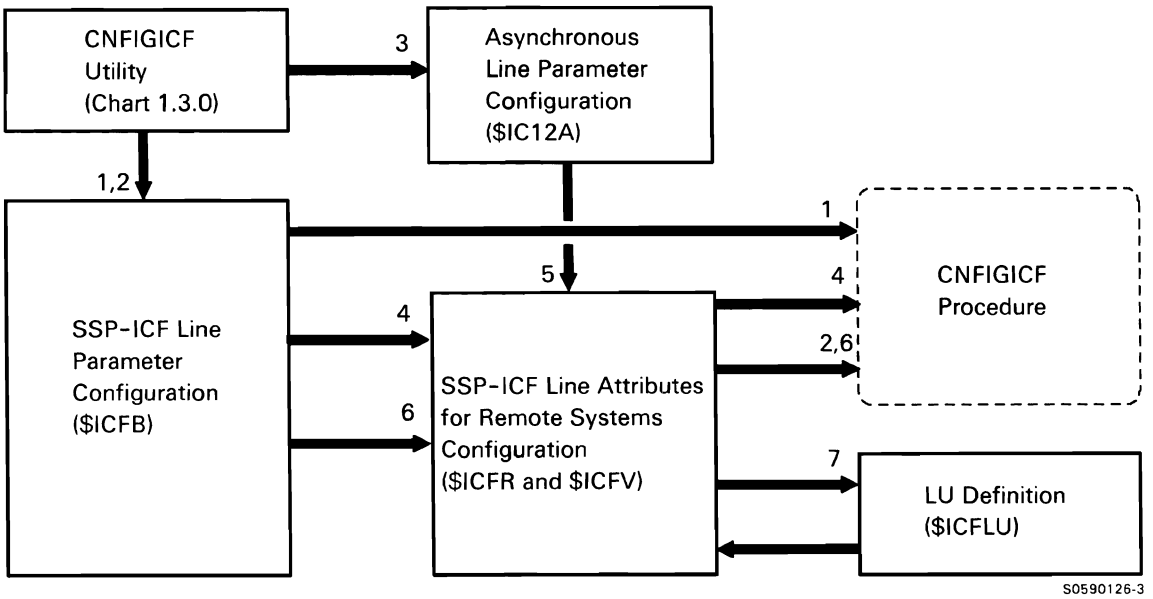


Chart 1.3.1 BSC/SNA Asynchronous Line Configuration Control Flow

BSC Subsystem Configuration

BSC subsystem configuration creates or edits subsystem configuration members for BSC SSP-ICF subsystems. The following BSC subsystem configuration processes are shown in Chart 1.3.2:

1 Display screen 22.0 and perform applicable subsystem configuration.

2 Perform BSC MSRJE subsystem configuration.

3 Perform BSC-CCP subsystem configuration.

4 Perform BSCEL subsystem configuration.

5 Perform BSC-CICS subsystem configuration.

6 Perform BSC-IMS/IRSS subsystem configuration.

7 Perform BSC 3270 subsystem configuration.

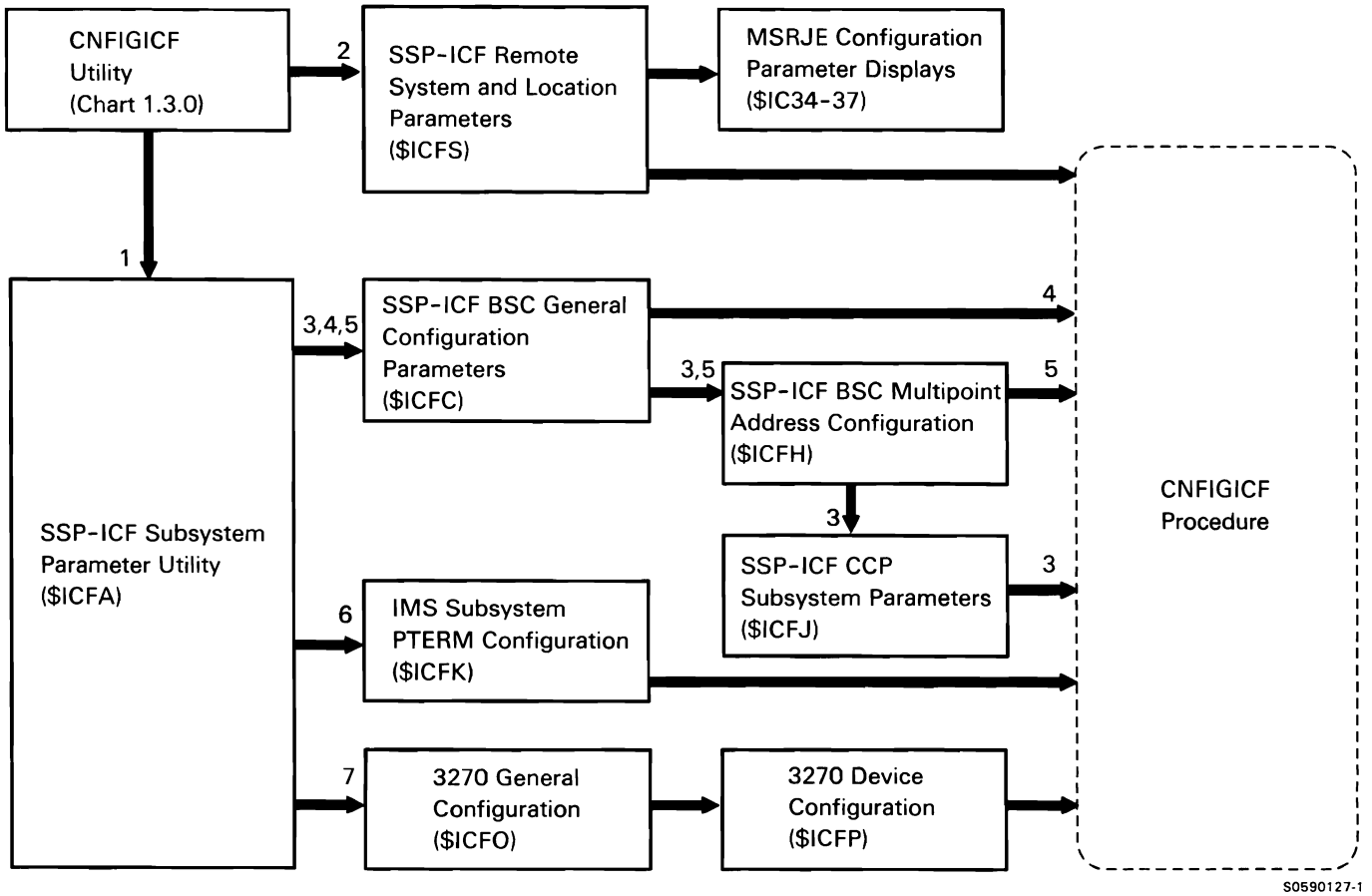


Chart 1.3.2 BSC Subsystem Configuration Control Flow

SNA Subsystem Configuration

SNA subsystem configuration creates or edits subsystem configuration members for SNA SSP-ICF subsystems. The following SNA subsystem configuration processes are shown in Chart 1.3.3:

1 Display screen 22.0 and perform applicable subsystem configuration.

2 Perform SNA MSRJE subsystem configuration.

3 Perform SNUF subsystem configuration.

4 Perform SNA 3270 subsystem configuration.

5 Perform Finance subsystem configuration.

6 Perform Peer subsystem configuration.

7 Perform APPC subsystem configuration.

8 Perform APPN subsystem configuration.

9 Remote location definition for APPN.

10 Session group definition for APPN.

11 Single session location definition for APPN.

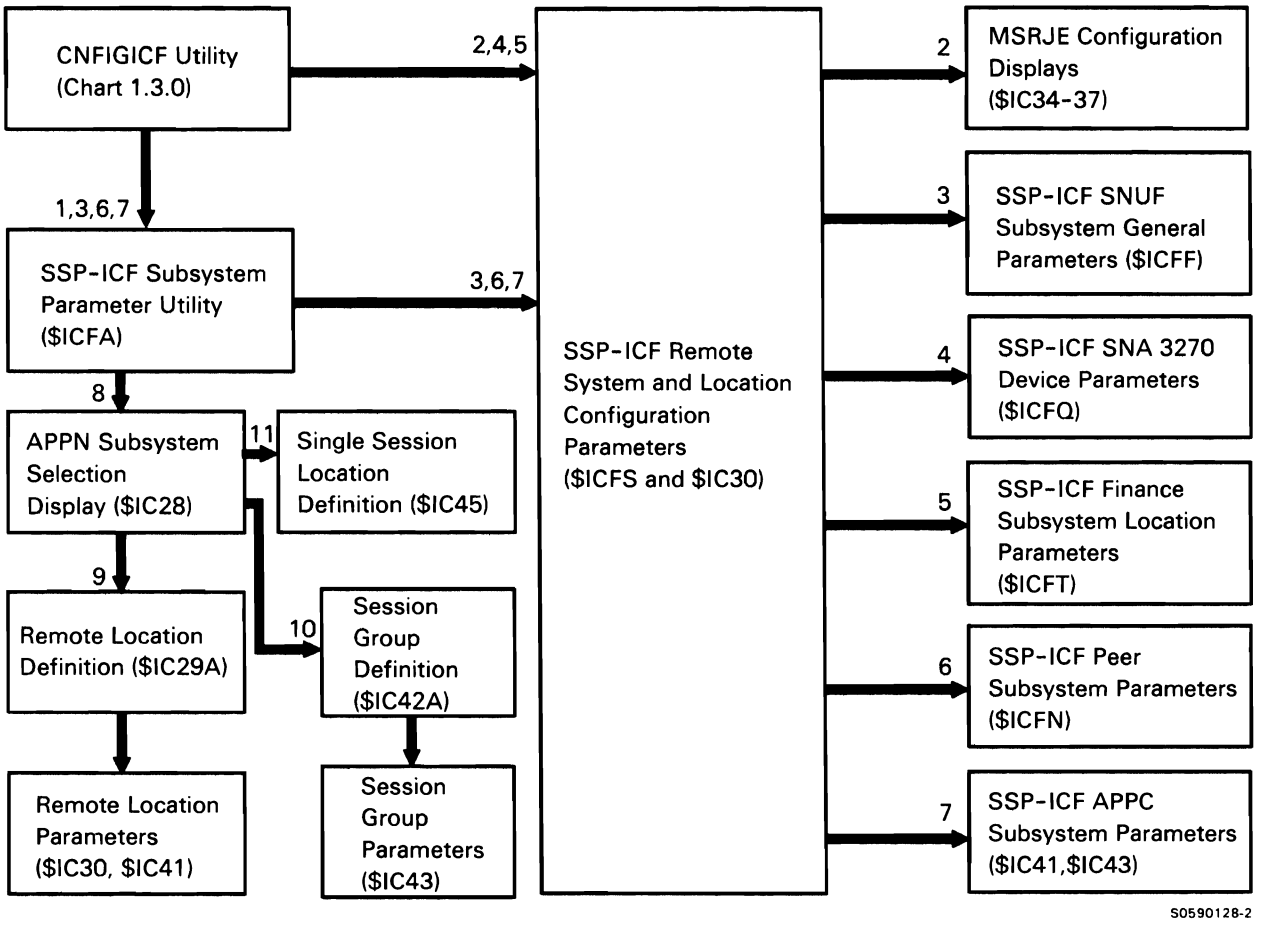


Chart 1.3.3 SNA Subsystem Configuration Control Flow

Asynchronous Subsystem Configuration

Asynchronous subsystem configuration creates or edits subsystem configuration members for asynchronous ICF subsystems. The following asynchronous subsystem configuration processes are shown in Chart 1.3.4:

1 Display screen 22.0 and perform applicable subsystem configuration.

2 Perform asynchronous subsystem configuration.

3 Perform asynchronous subsystem remote system and location configuration.

4 Perform asynchronous subsystem parameters configuration.

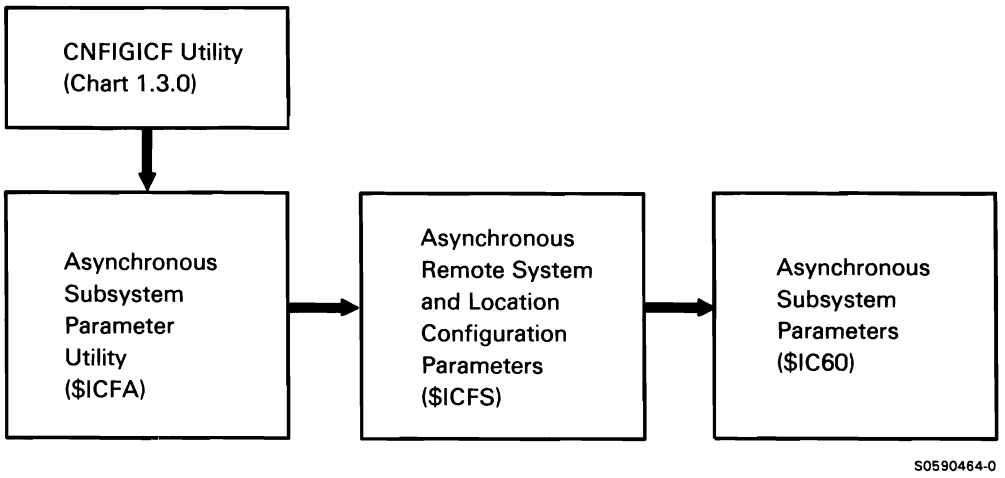


Chart 1.3.4 Asynchronous Subsystem Configuration Control Flow

Command Processing Function 2

The command processing function enables the system operator or the display station operator to direct System/36 in processing tasks and controlling the system.

After it initially receives control from main storage IPL (part of the system start function 1), the command processing function 'owns' all display stations that have no other jobs running at them (idle display stations). It controls the system console display, subconsole displays, and the display screens; command key processing; and function key processing of all display stations that it owns.

The command processing function receives control when an operator presses the Enter key or a command key, or invokes a high-level aid. If the input is a user menu option, the command processor retrieves the command text from disk and uses it as input. If the input is a control command or help menu option, control is passed to the appropriate command processor transient program. If the input is the name of an active MRT program, the command processor attaches the display station to the MRT program. If input is anything other than those mentioned above, the command processor starts an initiator 3 as a new task.

The command processing function also performs the following miscellaneous functions:

- Provides an error recovery interface.
- Processes SSP-ICF procedure start requests received from remote systems.
- Displays the sign-on display.
- Processes inquiry requests.
- Processes high-level aids.
- Performs display station release at end of job.
- Simulates certain control commands.

COMMAND PROCESSOR TASK STRUCTURE

The command processing function is divided among several tasks. Two are permanent tasks attached during IPL, and the others are subtasks that are attached as required. A description of each task follows.

Command Processor Mainline Task

All command processor posts are first received by the mainline task. This task must determine the initial action required and ensure that conflicting command processing functions are not processed concurrently.

I/O Error Processing Task

This task receives control whenever the system requires some type of I/O error processing. The main storage processing for all I/O errors is driven by this task.

Functional Subtasks

When the mainline task identifies a request for a specific command processor function, it attaches a subtask to perform all the processing required for that request. When the subtask is attached, the mainline task indicates that it is to receive a post when the subtask terminates. This is the way that the mainline task receives notification when all processing for the request has completed.

Task-to-Task Communication Subtask

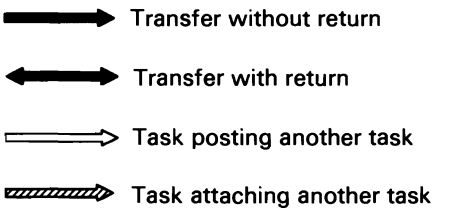
This subtask is attached when the mainline task receives a task-to-task communication (TTC) post. For more information, see Chart 2.5.

The command processing function includes the following subfunctions:

- Command processor mainline — Chart 2.1
- Sign-on — Chart 2.2
- Control command processing — Chart 2.3.n
- Procedure command processing (job initiation) — Chart 2.4
- High-level aids and task-to-task communications router — Chart 2.5
- Work station release — Chart 2.6
- Inquiry menu option processor — Chart 2.7
- Miscellaneous input — Chart 2.8
- Special keys — Chart 2.9
- Command processor/work station data management interface — Chart 2.10
- Console display — Chart 2.11
- System request/enter and power-on aid — Chart 2.12
- Command processor cleanup — Chart 2.13
- I/O error recovery — Chart 2.14
- Display station error recovery — Chart 2.15
- Initiator command interface — Chart 2.16
- System file initiation — Chart 2.17

COMMAND PROCESSING FUNCTION CHART CONVENTIONS

The following conventions are used to illustrate control flow in the command processing section of this manual:



S0590465-0

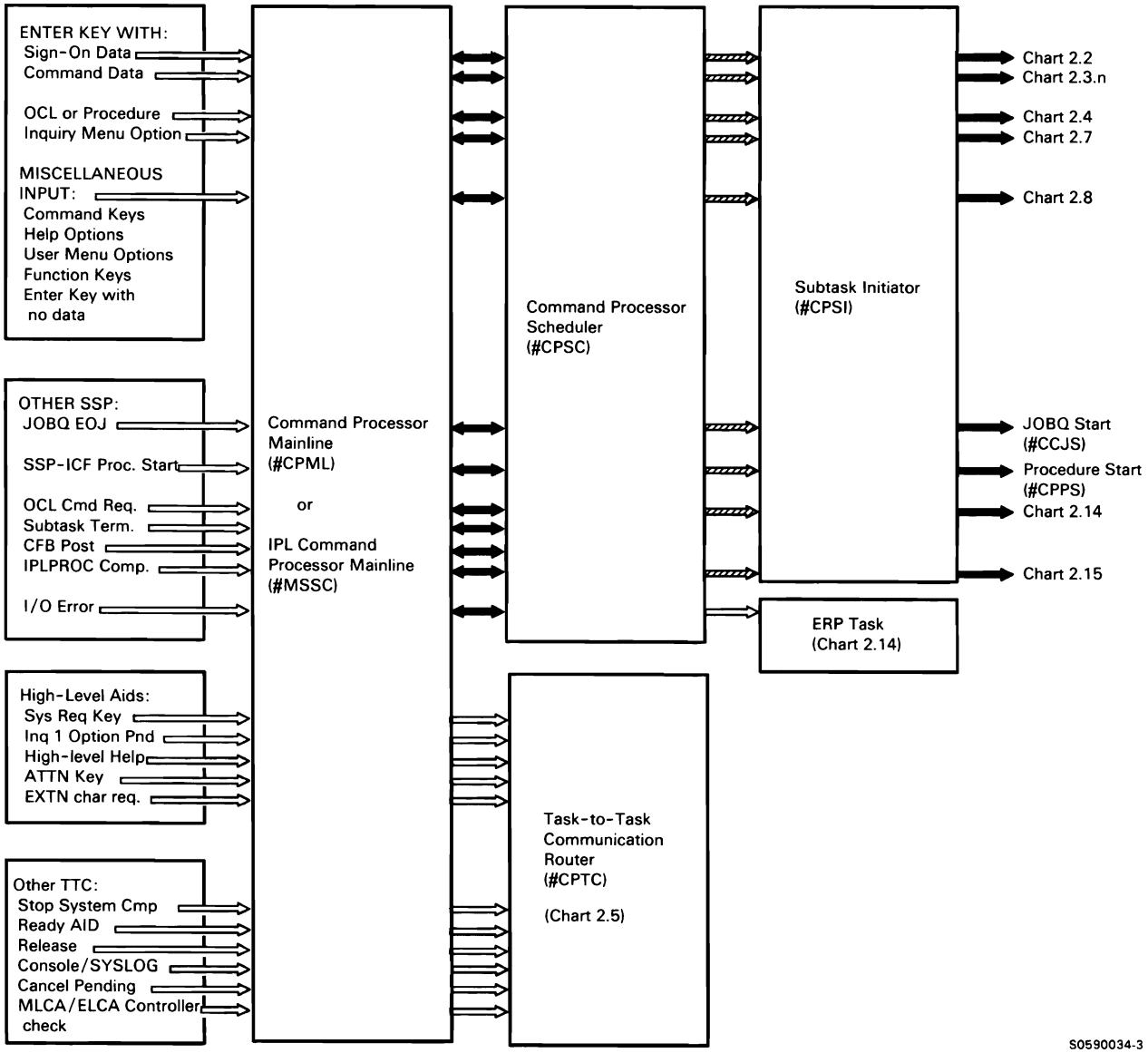
COMMAND PROCESSOR MAINLINE SUBFUNCTION

Command processor mainline waits for events to occur and attaches a command processor subtask to process those events.

Events that cause the mainline task to gain control are:

- Invite op-end: An operator has pressed a command key or entered a command, OCL statement, or menu option.
- Attn post: An operator has requested the inquiry function by pressing the Attn key.
- EXTN character post: An operator has entered a request for one ideographic character.
- Sys req post: An operator has pressed the System Request and Enter/Rec Adv key to get a sign-on display or to change interfaces at the system console or subconsole.
- Task-to-task communications post: A task has posted the command processor task via the post SVC instruction.
- I/O error: An I/O error has occurred.

- OCL command request: A control command issued from a system program is being processed.
- Power on post: An operator has powered on a display station.
- SSP-ICF procedure start: An SSP-ICF subsystem has posted the command processor to start a procedure.
- Subtask termination: A command processor subtask has completed its processing and has terminated.
- CFB post: A task has posted the command processor with a command/function block (CFB).



Note: The mainline subfunction does not directly route control to the command processor work station interface subfunction (Chart 2.9), and the command processor cleanup subfunction (Chart 2.12). These subfunctions are called directly by command processor transients.

Chart 2.1 Command Processor Mainline Control Flow

SIGN-ON SUBFUNCTION

Sign-on establishes the interface between the display station operator and the SSP. The following sign-on processes are shown in Chart 2.2:

1 Do the following and route for remaining control flow.

- Initialize display station for commands or OCL.
- Initialize display station for acquisition (standby mode).
- Create job control block (JCB).
- If IPL is in progress, set time/date.

2 If requested, create test request task.

3 Validate password security information.

4 If library information is specified, update JCB and build FSB.

5 If menu information is specified, set up menu information in JCB and TUB.

6 If applicable, display the system error display.

7 If applicable, display the initial screen or attach the sign-on procedure.

8 Issue any required messages and put records to history file.

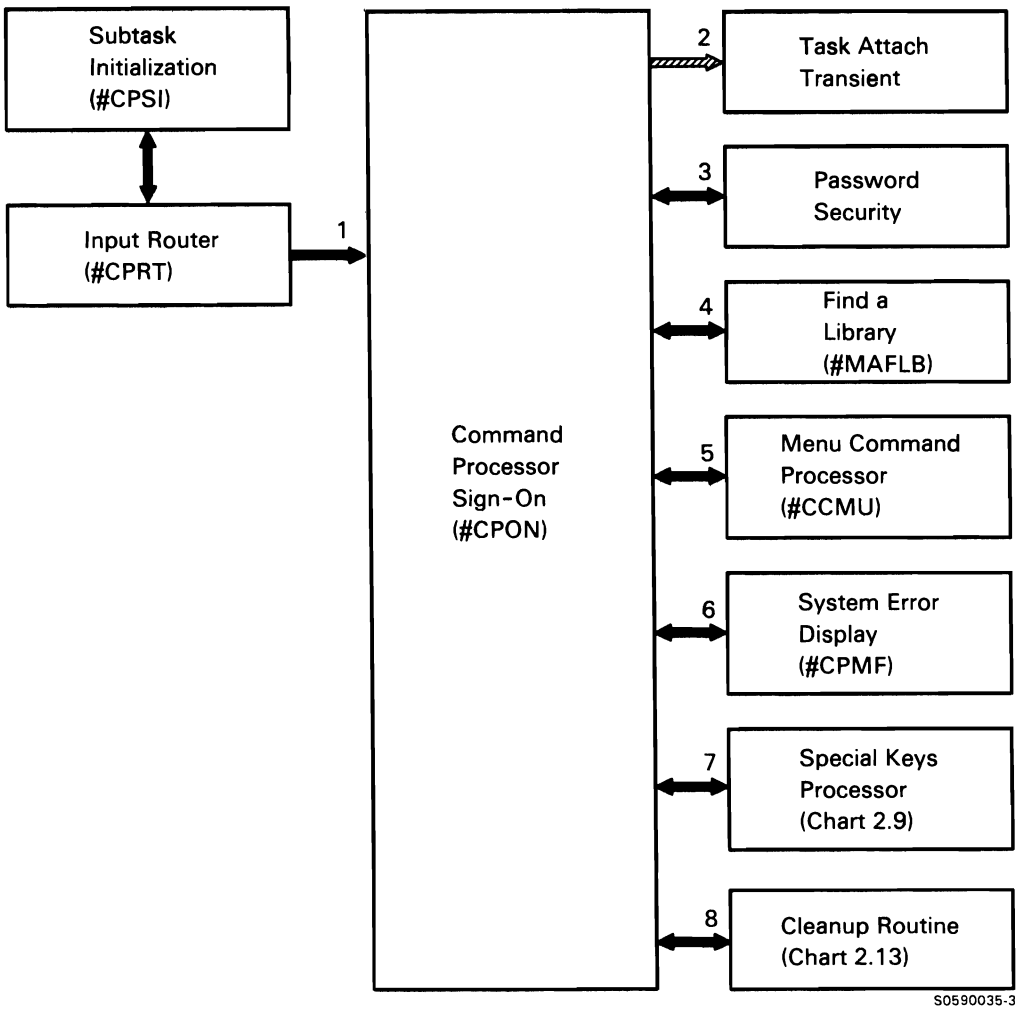


Chart 2.2 Sign-On Control Flow

CONTROL COMMAND PROCESSING SUBFUNCTION

Control command processing handles commands from display stations. Control commands do not require initiator functions for their processing. Charts 2.3.1 and 2.3.2 show the control flow for the control commands.

The chart below is a summary of the restrictions on control command usage from the various display station types and modes of operation:

Command Name	Display Station	Subconsole		System Console		System Service Device
		Command	Console	Command	Console	
ASSIGN				●	●	
CANCEL	■	■	■	●	●	●
CHANGE	■	■	■	●	●	●
CONSOLE	■	■		●	●	■
HELP	■	■	■	●	■	■
HOLD	■	■	■	●	●	●
INFOMSG	●	●	●	●	●	●
JOBQ	●	●		●		●
MENU	●	●		●		●
MODE	●	●		●		●
MSG	●	●	■	●	■	●
OFF	●	●		●		●
POWER	●	●	●	●	●	●
PRTY	■	■		●	■	●
RELEASE	■	■	■	●	●	●
REPLY			●	●	●	●
RESTART		■	■	●	●	●
START		■	■	●	●	■
STATUS	●	●	●	●	●	●
STATUSF	●	●	●	●	●	●
STOP		■	■	●	●	■
TIME	●	●	●	●	●	●
VARY				●	●	

● Indicates that the command can be used from the display station when it is in the specified mode.

■ Indicates that the full function of the command is not available to the given display station type and mode; for specific command function limitations, refer to the *System Reference* manual.

S0590292-0

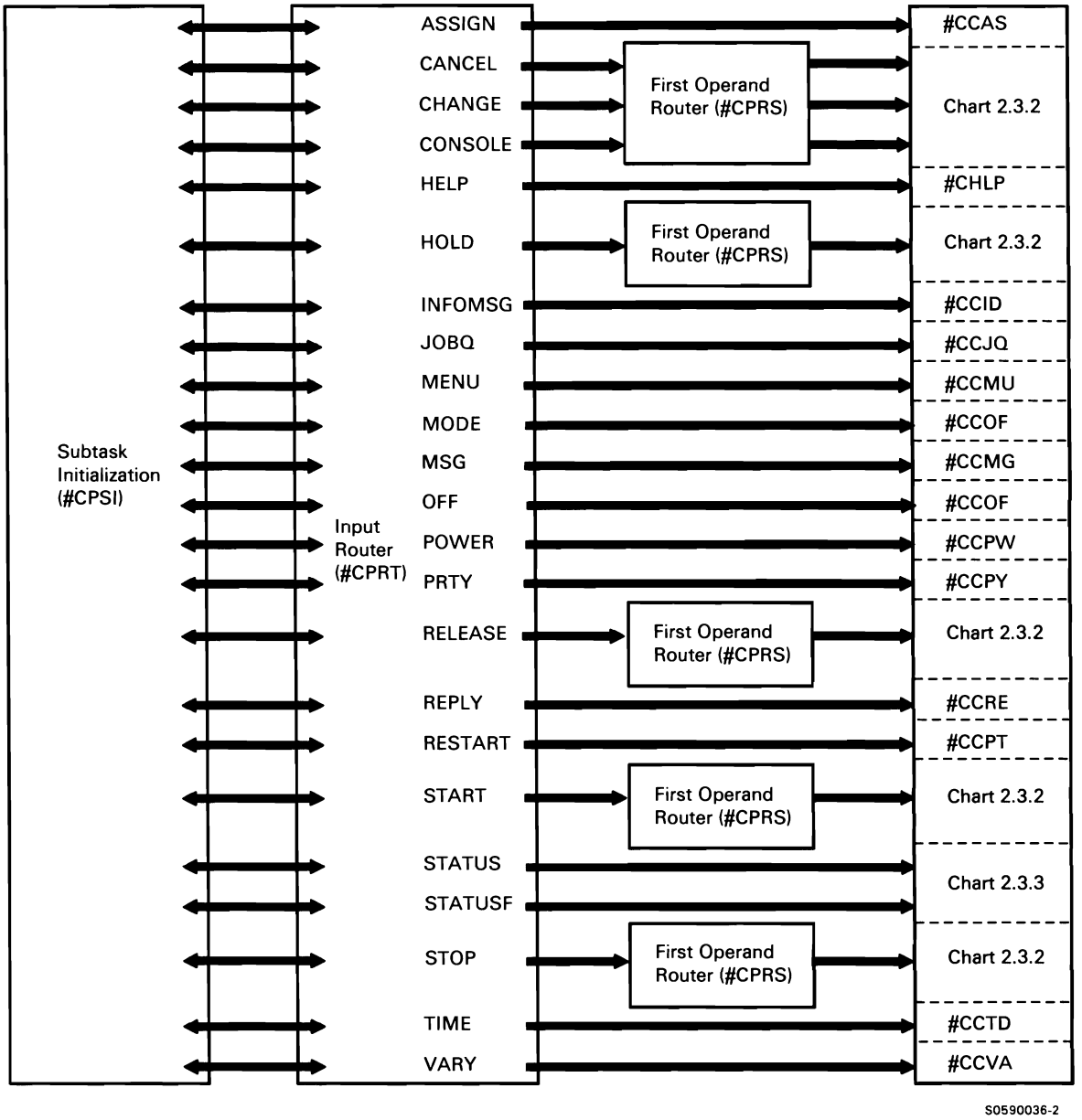


Chart 2.3.1 Control Command Processing Control Flow

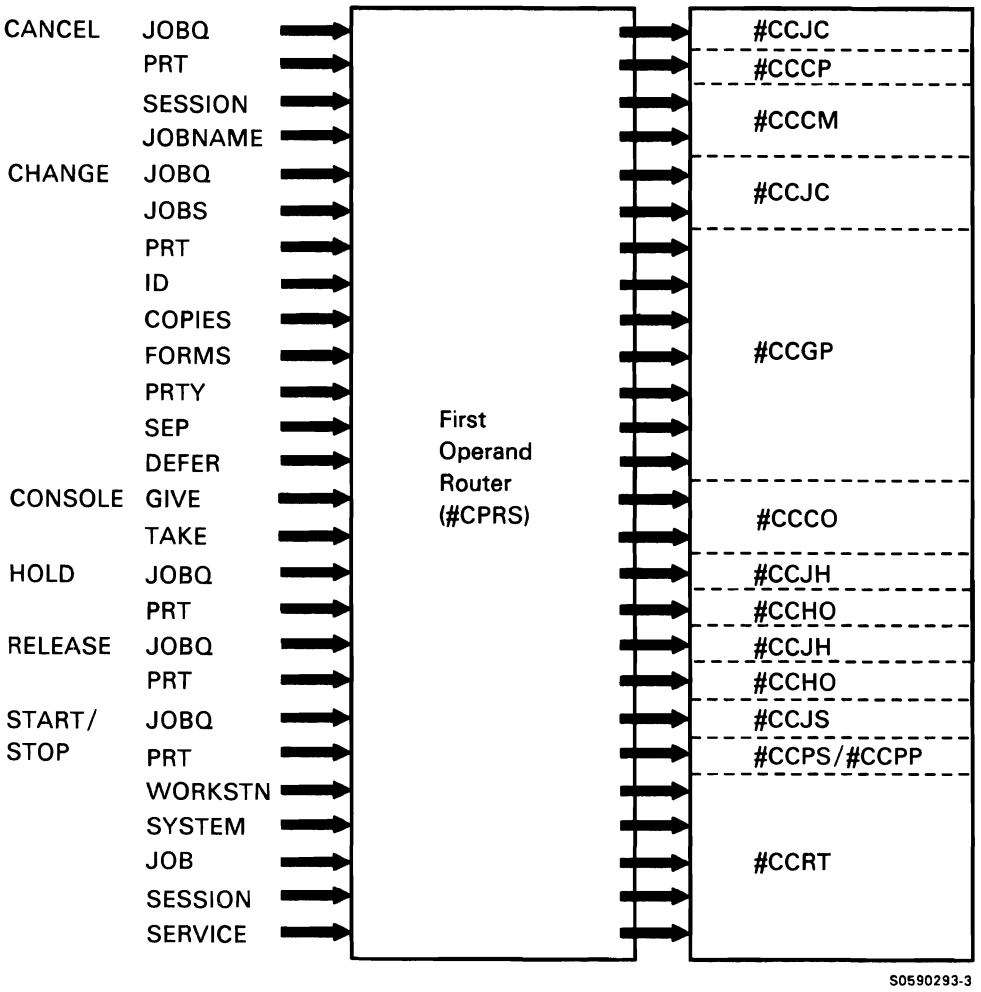
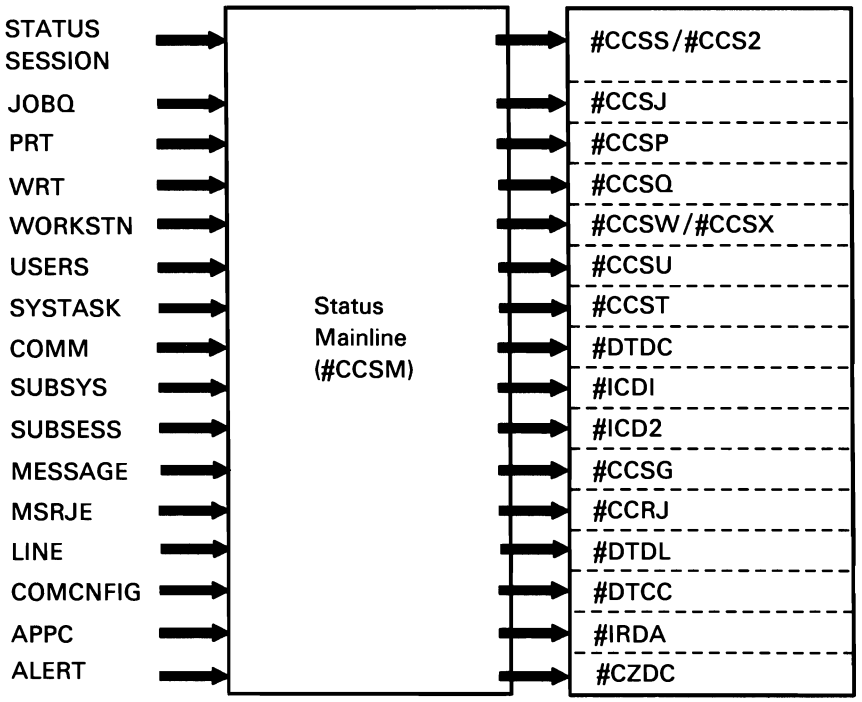


Chart 2.3.2 First Operand Router Control Flow



Note: Control flow shown for the JOBQ, PRT, WORKSTN, and USERS parameters also applies to the equivalent parameters used with the STATUSF command.

S0590363-1

Chart 2.3.3 Status Command Control Flow

PROCEDURE COMMAND PROCESSING (JOB INITIATION) SUBFUNCTION

The command processor input router receives control when the operator enters a procedure, an OCL statement, or a menu option. The input router starts the initiator (#CIML) to process the procedure or OCL entered. If the operator-entered data is a request to attach to an active multiple requester terminal (MRT) task, the input router attaches the required work station(s) to the MRT program and updates the TB to show the additional work stations.

The following procedure command processes are shown in Chart 2.4:

1 Attach work station(s) to an active MRT program by attaching work station TUB(s) to MRT TB.

2 Attach an initiator.

3 If attach of initiator fails, issue error messages.

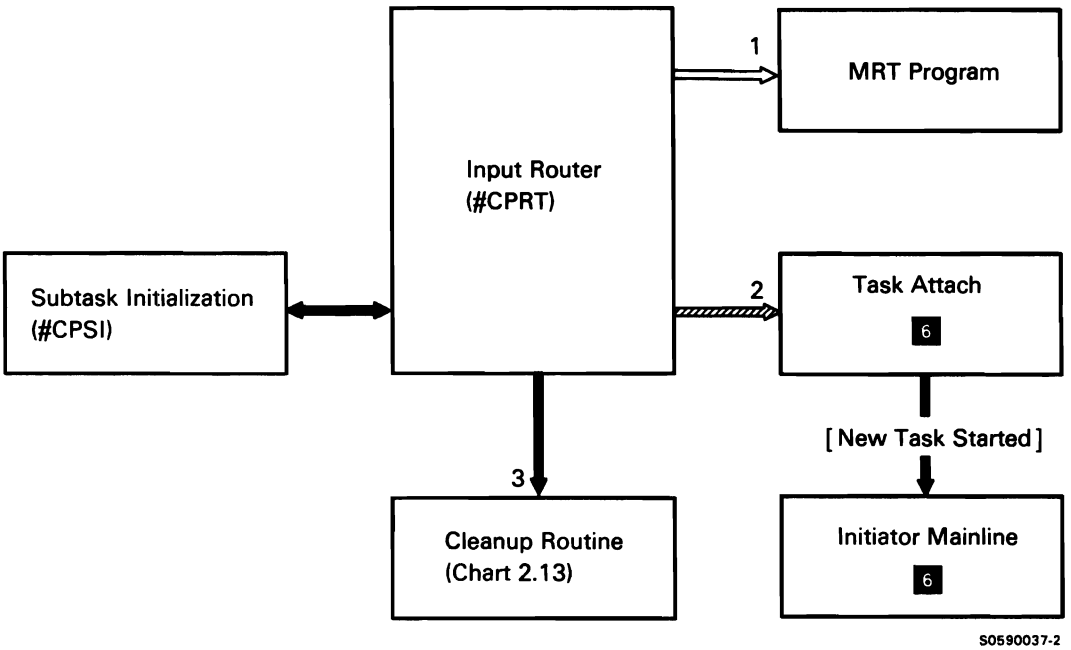


Chart 2.4 Procedure Command Processing Control Flow

This page is intentionally left blank.

HIGH-LEVEL AIDS AND TASK-TO-TASK COMMUNICATIONS ROUTER SUBFUNCTION

When a task-to-task communications (TTC) post is received, it could indicate one or more different types of requests. The TTC subtask is attached by the mainline command processor task to search for these requests by scanning the DUB chain, scanning the JCB chain, and checking indicator bits in the SCA. When any request is found, a CFB is built for the request and posted to the mainline task so functional subtasks can be attached to perform the processing. Once attached, the TTC subtask remains active until no more requests have been found and no more new posts have been received.

The following processes are shown in Chart 2.5:

1 Sound the alarm for new console messages and look for any pended console display requests.

2 Process any pended cancel job requests.

3 Start functional subtask for a request.

4 Sys Req key and power-on aid requests.

5 Attention key requests.

6 Help key (in operator error mode) requests.

7 Work station release requests.

8 Stop system complete.

9 Ready aid.

10 Pended console display requests.

11 Extended character requests.

12 Error recovery aid.

13 MLCA/ELCA controller check.

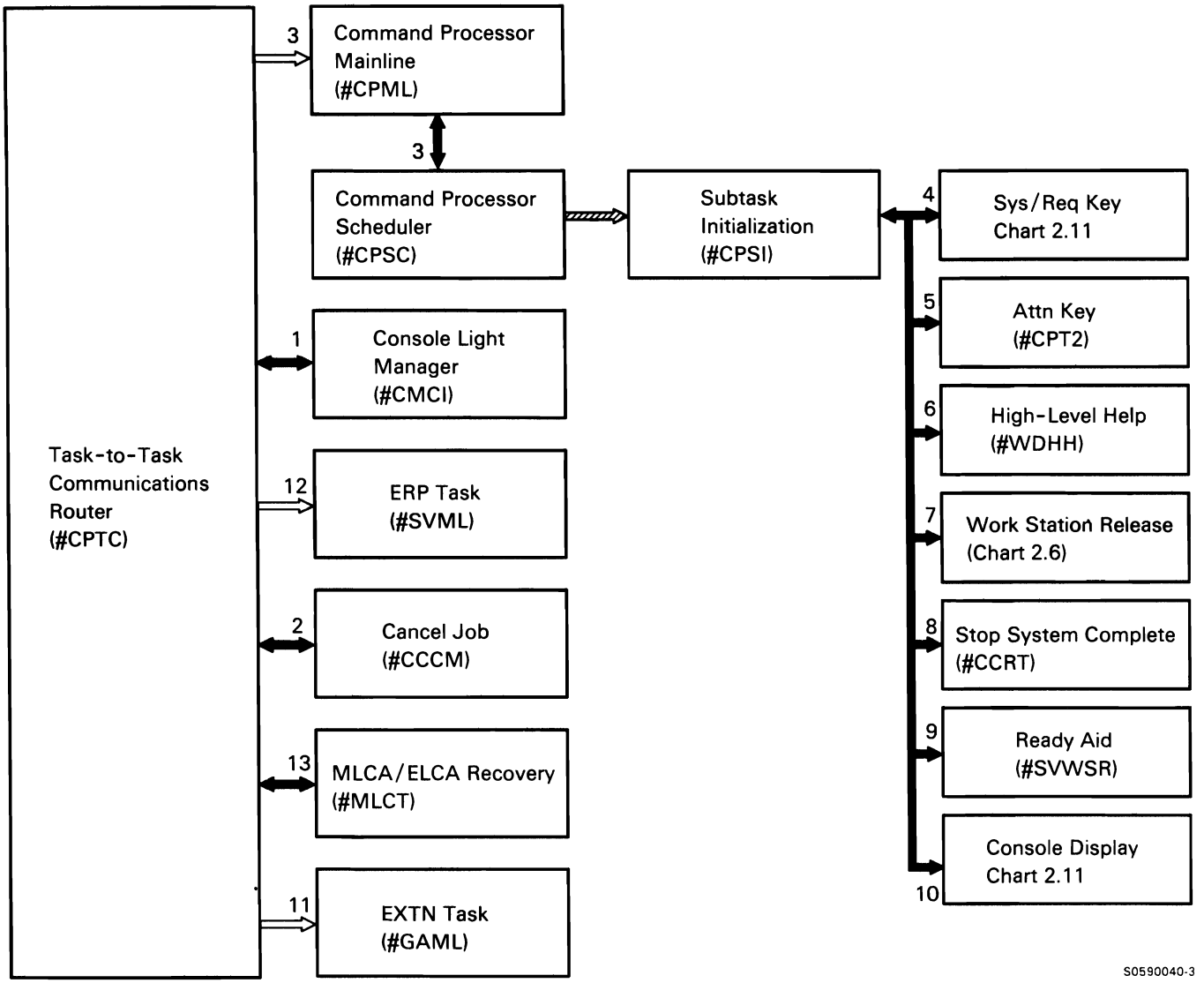


Chart 2.5 High-Level Aids and Task-to-Task Communications Router Control Flow

WORK STATION RELEASE SUBFUNCTION

Work station release returns a work station to its previous owner. If the previous owner was a job, work station release attaches the initiator to process the next job step; if the previous owner was the command processor, work station release reestablishes the appropriate interface, based on the work station mode.

The task-to-task communications router (#CPTC) determines that a release should be done for a work station. #CPTC then posts the mainline task to create a new command processor subtask to perform the release. The following release subfunction processes are shown in Chart 2.6:

1 Reestablish the command interface at end of job by displaying the appropriate display:

   - Command display
   - Standby display
   - Help menu
   - Status display
   - Any waiting message(s)

2 Attach the initiator to process the next job step when a requester is released from an MRT task or when the current job step releases its requester via the ATTR OCL statement.

3 Sign off a work station if OFF OCL statement was processed.

4 If RESTORE-NO was specified, clear all of the input fields on the display.

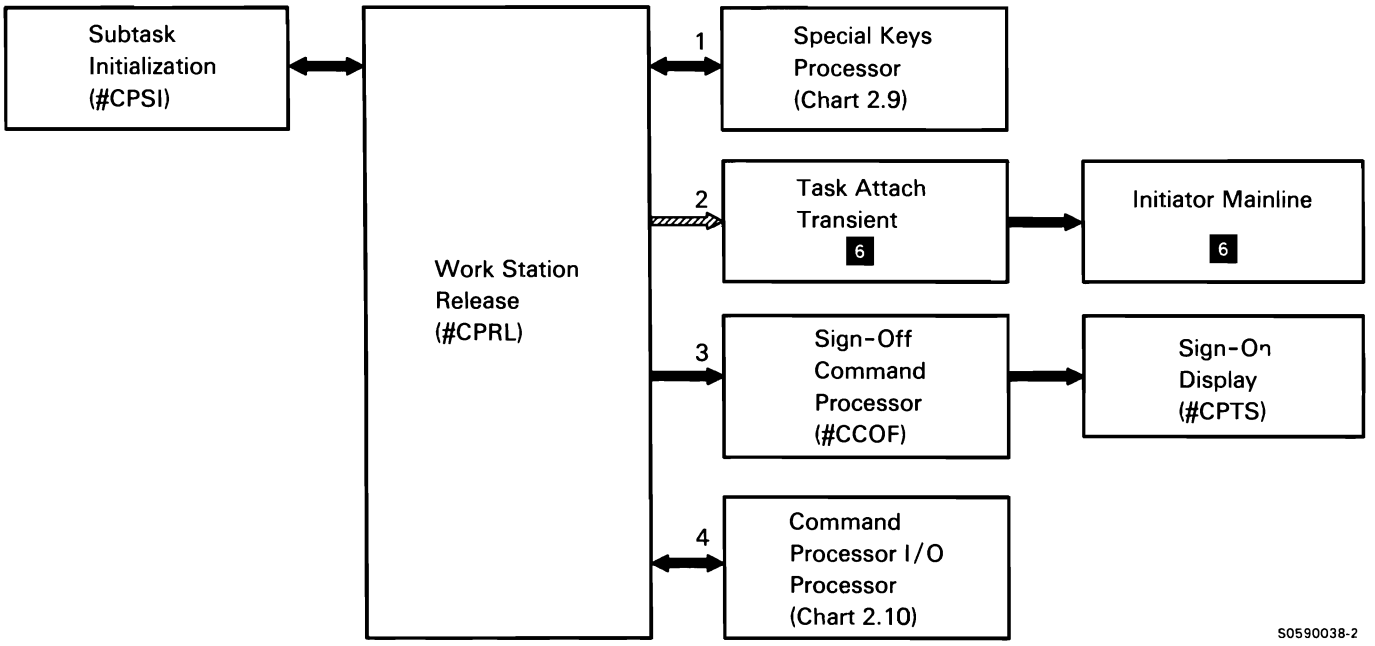


Chart 2.6 Work Station Release Control Flow

INQUIRY MENU OPTION PROCESSOR SUBFUNCTION

The inquiry menu option processor executes when the operator interrupts a currently executing program by pressing the Attn key. When the Attn key is pressed at a display station where an SRT job is running, the SRT job is suspended. A new TUB for inquiry is built and put on the vertical TUB chain in place of the TUB to which the SRT was attached.

When the Attn key is pressed at a display station that is attached to an MRT program, the MRT job is informed that it should not service the TUB. With the exception of those items listed below, processing for MRT and SRT inquiries is the same.

The following inquiry menu option processor processes are shown in Chart 2.7:

1 Process menu option 0: If the current interrupted job is an SRT, restore the previous user display and resume the current interrupted job. If the current interrupted job is an MRT, restore the previous user display and allow the TUB to be processed.

2 Process menu option 1: Establish command interface for inquiry.

3 Process menu option 2: If the current interrupted job is an SRT, perform an option 2 cancel of the SRT job. If the current interrupted job is an MRT, release the display station from the MRT job and continue with additional job steps.

4 Process menu option 3: If the current interrupted job is an SRT, perform an option 3 cancel of the SRT job. If the current interrupted job is an MRT, release the display station from the MRT job and cancel any remaining job steps.

5 Process menu option 4: Set the inquiry latch in the JCB and resume the current interrupted job.

6 Process menu option 5: Display session status.

7 Process menu option 6: Display waiting message(s).

8 Process menu option 7: If the alternate interrupted job is an SRT, restore the previous user display for the alternate interrupted job and resume the alternate interrupted job. If the alternate interrupted job is an MRT, restore the previous user display for the alternate interrupted job and allow the TUB to be processed.

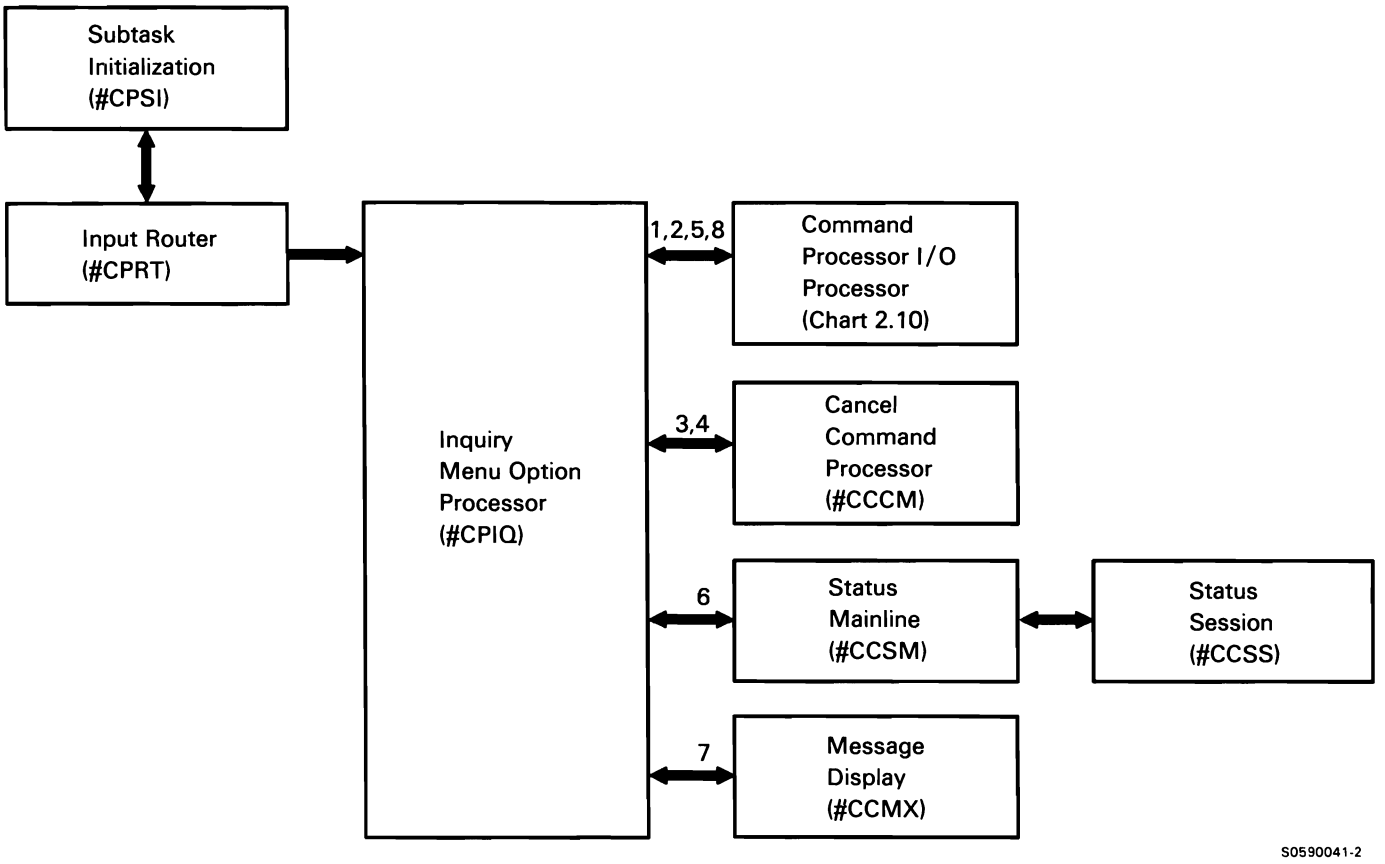


Chart 2.7 Inquiry Menu Option Processor Control Flow

MISCELLANEOUS INPUT SUBFUNCTION

The following input functions are shown in Chart 2.8:

1 Process the following input:

- Command keys.
- Home key.
- Roll keys.
- Test Request key.
- Enter key without data from a status display.
- I/O error.

2 Help key.

3 Option from SSP help menu.

4 Help for a control command.

5 Enter key with DUP characters.

6 Enter key without input data from console display.

7 Enter key to end help text for sign-on display.

8 Console command from IPL in progress display.

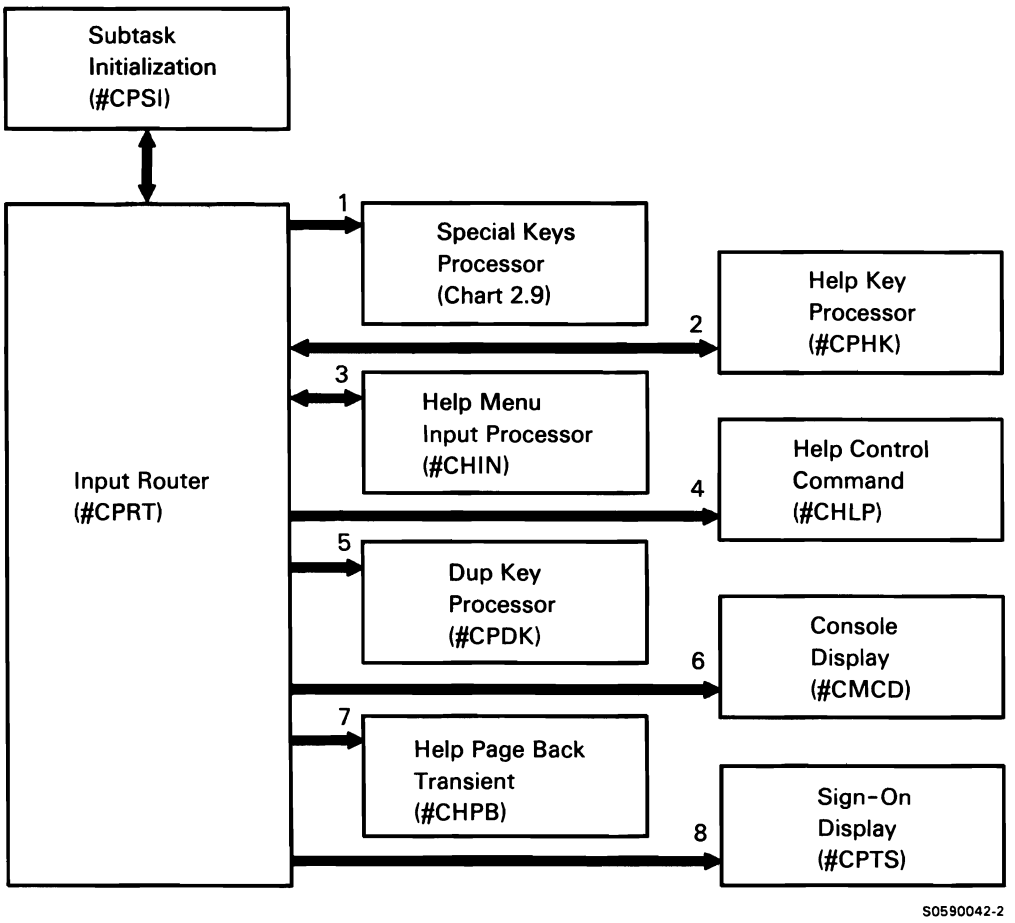


Chart 2.8 Miscellaneous Input Control Flow

This page is intentionally left blank.

SPECIAL KEYS SUBFUNCTION

The following functions and command keys are shown in Chart 2.9:

1 Put up system displays after a program has released a display station at end of job, but the OCL statement specified RESTORE-NO.

2 Display MSG command messages.

3 Restore console display after second-level text for a message was displayed at the system console or subconsole.

4 Attach sign-on procedure.

5 Display system error display.

Process the following keys:

6 Command key 1.

7 Command keys 3 and 7 from help prompt, help menu, or user menu.

8 Command keys 5, 6, 12, 23, 24, and the Home key.

9 Command key 8.

10 Command keys 9, 10, 15, and 16.

11 Command key 11.

12 Roll keys from status display.

13 Roll keys from help text for user menu.

14 Roll keys with SSP help active.

15 Roll keys with SSP help text active.

16 Enter key or command key 3 from help text for user menu.

17 Test Request key.

18 Issue error messages for invalid keys.

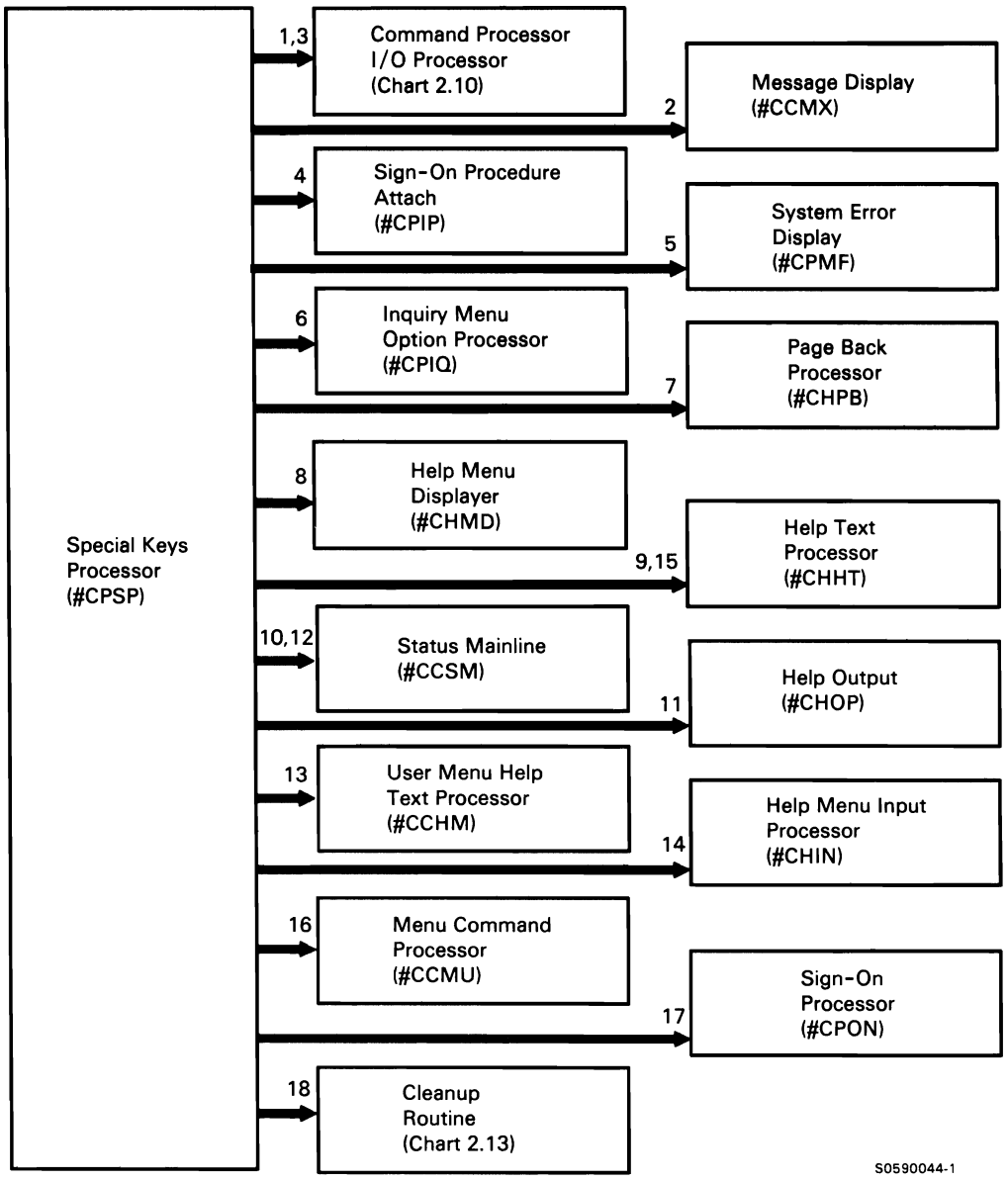


Chart 2.9 Special Keys Control Flow

COMMAND PROCESSOR/WORK STATION DATA MANAGEMENT (WSDM) INTERFACE SUBFUNCTION

The command processor/WSDM interface subfunction processes the requests from command processor routines for work station I/O operations. Callers invoke the command processor/WSDM interface via the transfer SVC instruction. The interface consists of two processing modules: #CPIO is the command processor work station I/O processor, and #CHOP is the help output transient. #CPIO handles save/restore requests and the displaying of command processor screens, user menus, and help text for user menus. #CHOP displays help menus and prompts; it also displays help text for help menus, prompts, and command processor screens. Some command processor programs call #CHOP directly, and others call #CPIO which determines if #CHOP should be called.

The following command processor/WSDM interface processes are shown in Chart 2.10:

1 Process caller request by building system parameter list for WSDM.

2 If help menu or help prompt is displayed, determine next format to be displayed.

3 Perform work station I/O operations requested by #CPIO or #CHOP.

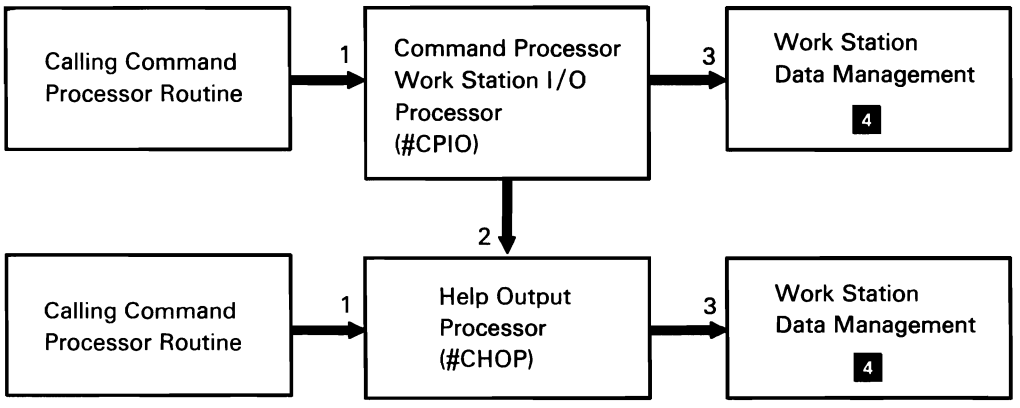


Chart 2.10 Command Processor I/O-WSDM Interface Control Flow

CONSOLE DISPLAY SUBFUNCTION

The console display subfunction retrieves messages to be displayed at a console or subconsole and performs the necessary operations to show them at the console display. These messages may have been issued by using the MSG command or // MSG OCL statement. They may have also been issued through SYSLOG, I/O error recovery, or any other function that uses a console queue element (CQE).

The following console display processes are shown in Chart 2.11:

1 Determine if a message can be issued and do the following for each message:

- Retrieve the message text and place it into a WSDM buffer.
- Determine which lines to roll off the screen to make room for the new message.
- Update the console screen image on disk.

2 Roll the screen and display the new messages.

3 Place asterisks next to messages that have been responded to.

4 Retrieve the text for messages sent via the MSG command.

5 Turn off the message light if no more messages are waiting to be displayed.

6 Issue message if some resource is not available.

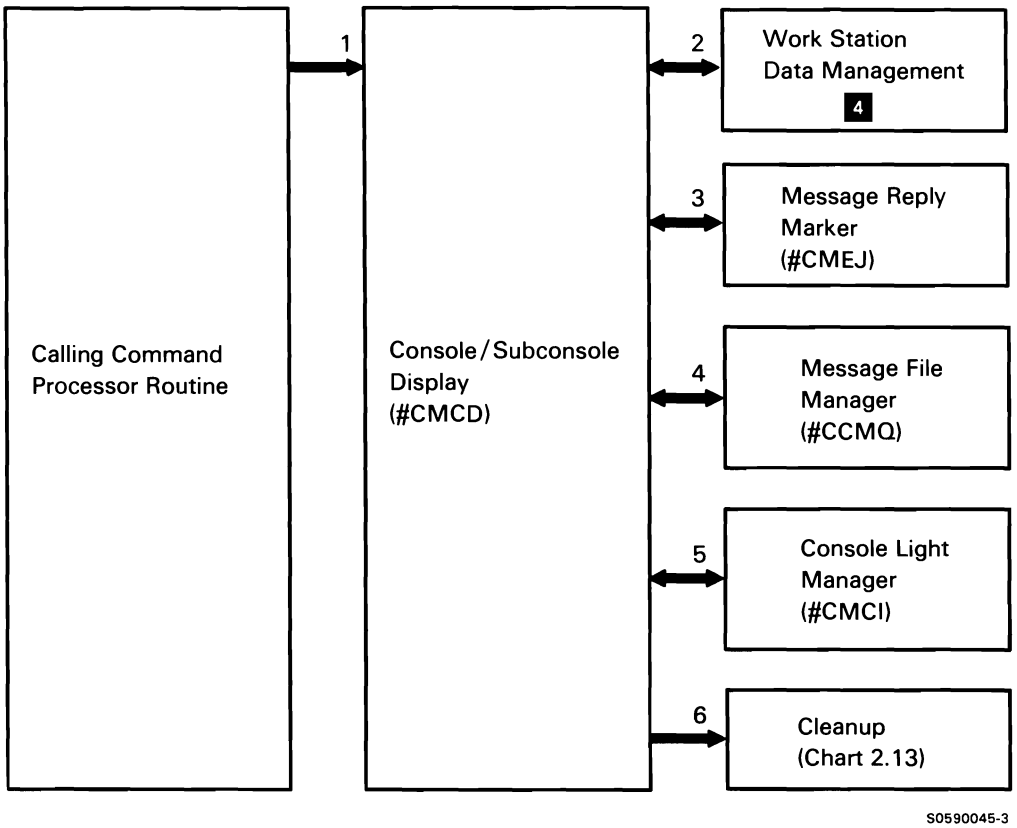


Chart 2.11 Consol Display Control Flow

SYSTEM REQUEST/ENTER AND POWER-ON AID SUBFUNCTION

The system request/enter and power-on aid subfunction controls display station mode switching and the displaying of the sign-on screen. When the command processor changes a display station from command mode to console mode, it saves the user screen, swaps the TUBs, and restores the console display. The command mode TUB is placed on the horizontal TUB chain and the console mode TUB is placed on the vertical TUB chain. This allows the device controller (work station IOCH, remote work station support, or distributed host control facility) to process the console mode TUB. When the display station is switched back to command mode, the TUBs are swapped back and the user screen is restored.

The following system request/enter and power-on aid processes are shown in Chart 2.12:

1 Display sign-on screen.

2 Save/Restore command mode screen and stop outstanding invites.

3 Rebuild console screen.

4 Issue error message if system request not allowed at display.

5 If swapping from command mode to console mode, display outstanding messages for the console.

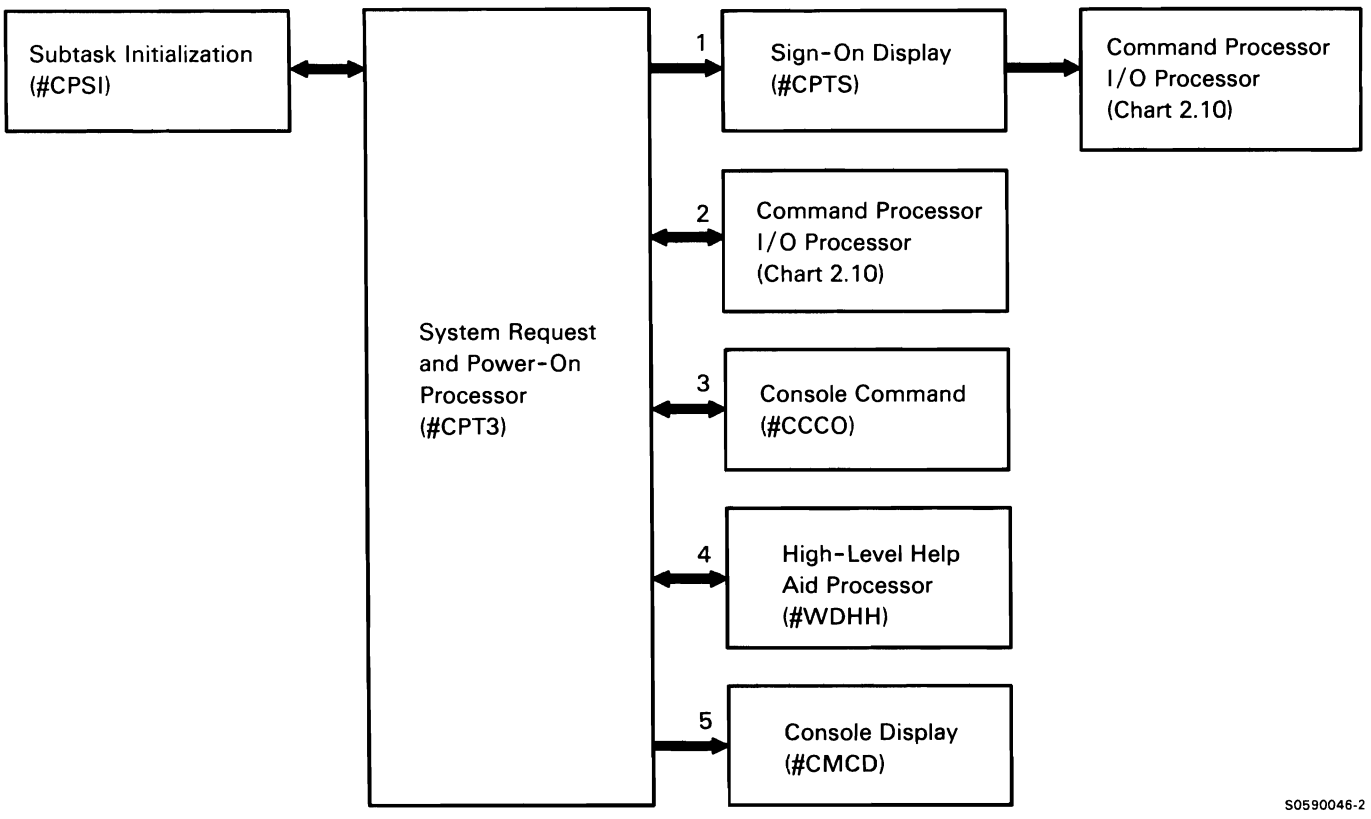


Chart 2.12 System Request/Enter and Power-On Aid Control Flow

COMMAND PROCESSOR CLEANUP SUBFUNCTION

Command processor cleanup provides miscellaneous command-processing-terminating activity and screen control for the other command processor transient modules.

The following command processor cleanup processes are shown in Chart 2.13:

1 Route for the following:

2 Retrieve messages by message identification code (MIC), or retrieve messages from main storage and display to specified display station.

3 Log input and/or messages to the history file.

4 Display message.

5 If active, refresh status screen and invite input in console or subconsole mode.

6 Display system console and subconsole messages (in console or subconsole modes).

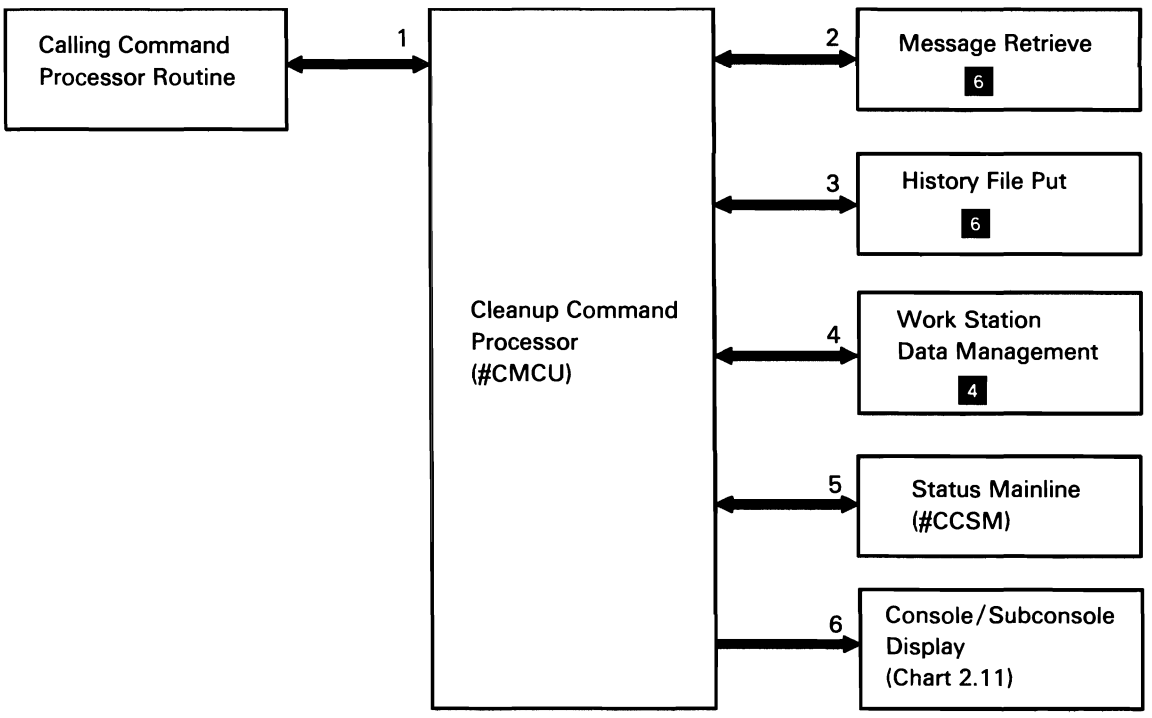


Chart 2.13 Command Processor Cleanup Control Flow

I/O ERROR RECOVERY SUBFUNCTION

I/O error recovery issues error messages, recognizes operator responses, and handles exception conditions. The error task and command processor subtasks serve as the interface between I/O error recovery (main storage) and the error recovery procedures in control and main storage. The following I/O error recovery processes, executed under the error task, are shown in Chart 2.14:

1 Route error message.

2 Perform printer error recovery.

3 Perform work station error recovery.

4 Perform error recovery for disk, diskette, tape, or MICR.

5 Handle work station configuration errors at IPL time.

6 Perform error recovery for MLCA/ELCA.

7 Purge error messages and turn off message waiting light.

8 Handle message response.

The command processor or error task initiates I/O error recovery when it is posted with an error event type by a control storage routine.

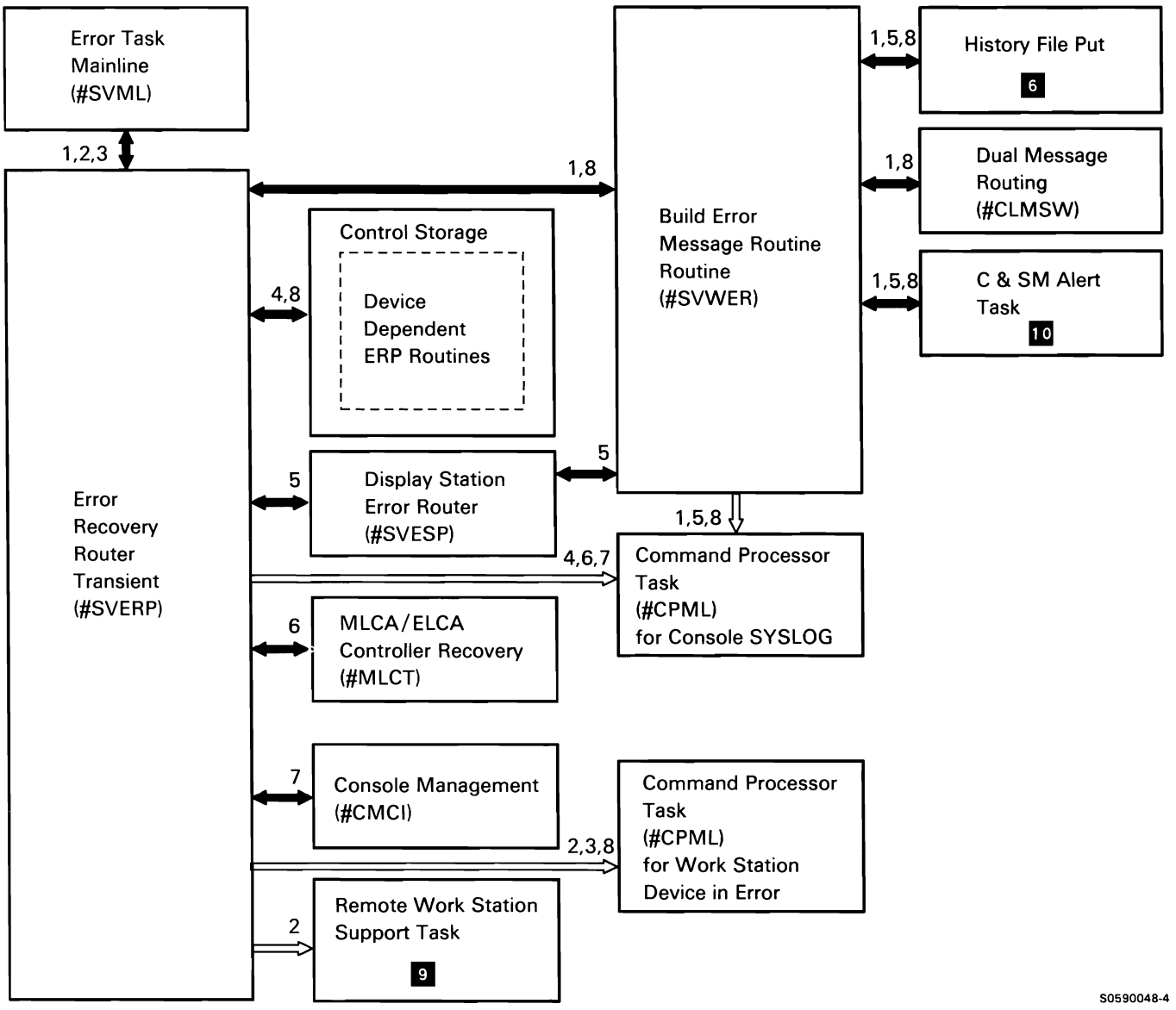


Chart 2.14 I/O Error Recovery Control Flow

DISPLAY STATION ERROR RECOVERY SUBFUNCTION

For display station error recovery, the processing is done under a command processor subtask as shown in Chart 2.15:

1 Route error message.

2 Handle message response.

3 Route for recovery from the following display station errors:

4 First command reject.

5 Subsequent command reject of command processor I/O operation.

6 Subsequent command reject of user program I/O operation.

7 Ready after command reject of command processor I/O operation (#SVWSR transfers to #SVRD).

8 Ready after command reject of user program I/O operation.

9 Retry previously unprocessed error.

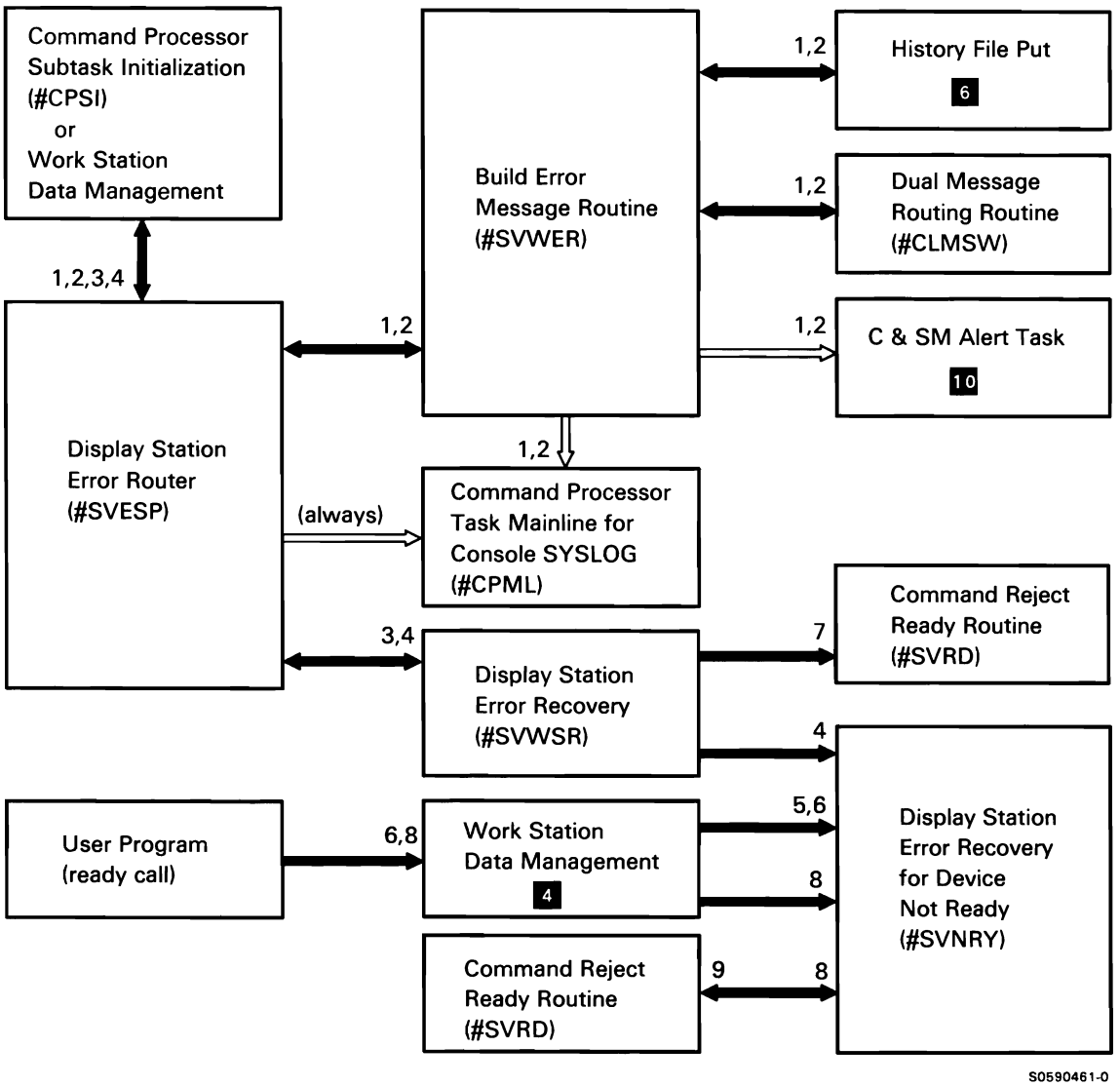


Chart 2.15 Display Station Error Recovery Control Flow

INITIATOR COMMAND INTERFACE SUBFUNCTION

The initiator command interface allows system programs to issue control commands as if the commands were OCL statements. The interface transients, #CPIN and #CPOC, build a parameter list, then pass control to the input router (#CPRT). The following initiator command interface processes are shown in Chart 2.16:

1 Establish an interface between the initiator and the command processor.

2 Set up command parameters and route the command.

3 Process command.

4 Return the results to the caller.

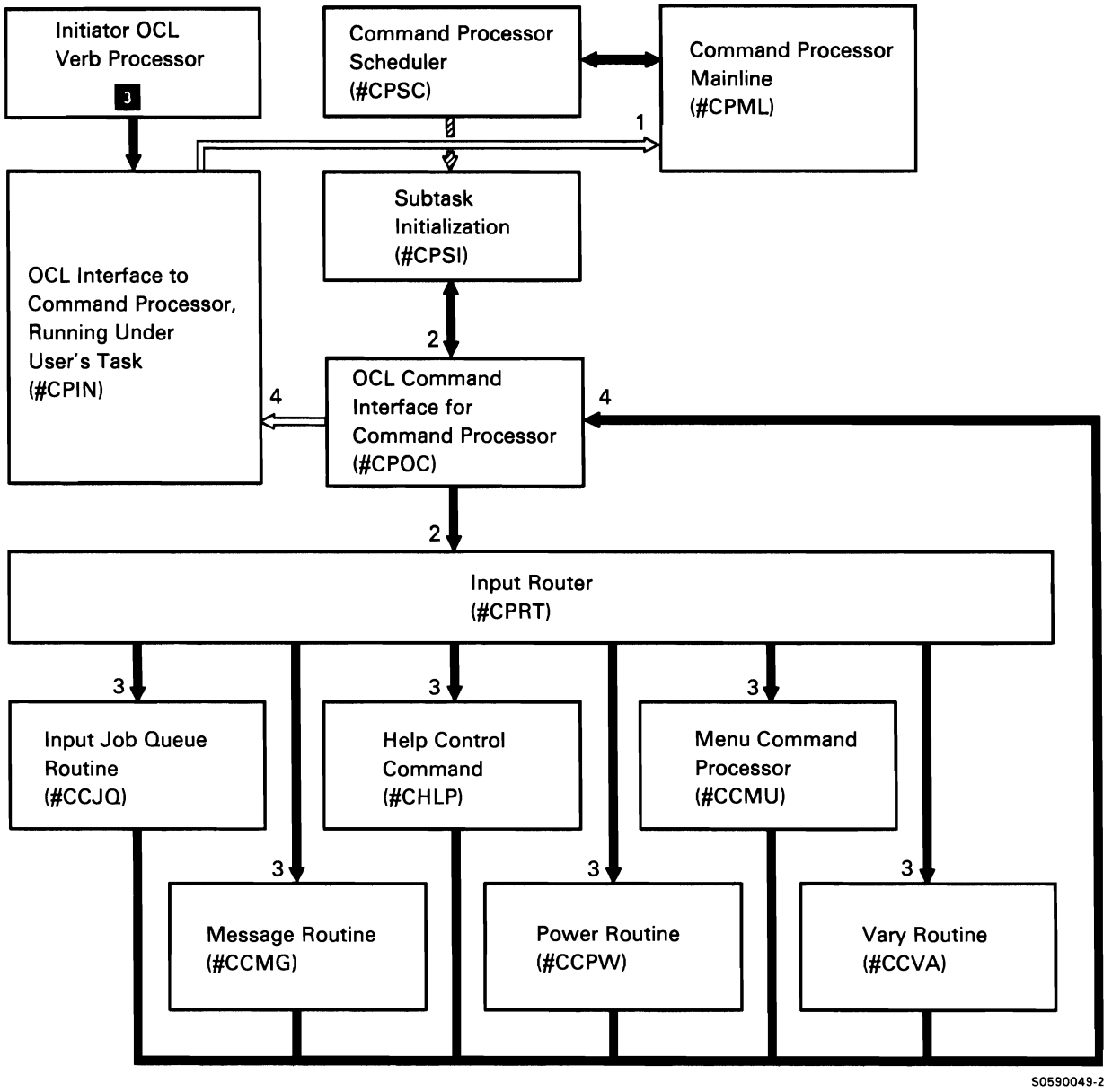


Chart 2.16 Initiator Command Interface Control Flow

SYSTEM FILE INITIALIZATION FILE SUBFUNCTION

This subfunction performs the initialization for system files required at IPL. The subfunction is started when the IPL Command Processor Mainline (#MSSC) receives a post indicating that #IPLPROC has completed.

The following processes are shown in Chart 2.17:

1 Initialize spool file.

2 Initialize job queue file.

3 Initialize message file.

4 Initialize extended character file.

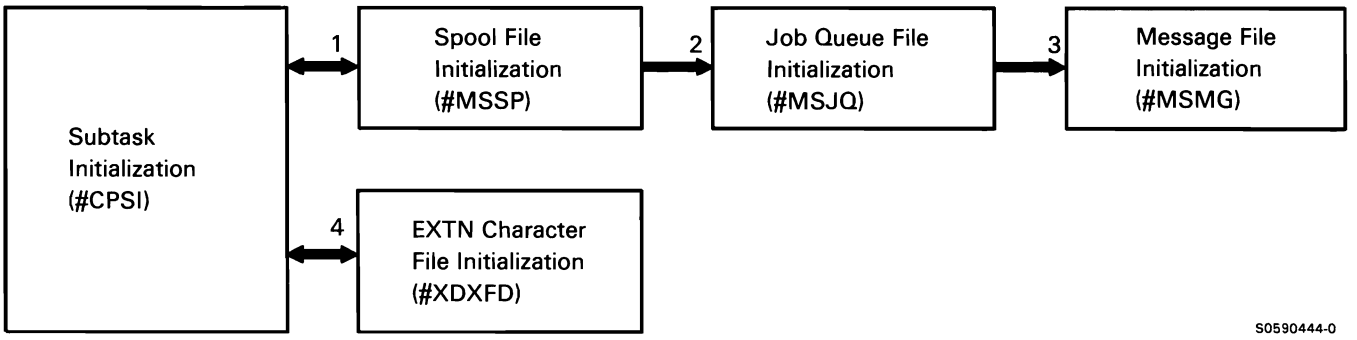


Chart 2.17 System File Initialization Control Flow

This page is intentionally left blank.

Job Start Function

The job start function contains all SSP programming required to start a job, except the command processor. It includes the following subfunctions:

- Initiator — Chart 3.1
- Allocate
  - Normal allocate — Chart 3.2.1
  - Special allocate — Chart 3.2.2
  - Deallocate — Chart 3.2.3
  - Data communications allocate — Chart 3.2.4.n
- Open — Chart 3.3.1

INITIATOR SUBFUNCTION

The initiator subfunction performs two main functions:

- OCL statement processing
- Program initialization

The OCL processor portion of the initiator reads, diagnoses, and interprets operational control language (OCL) statements supported by the system. The program initialization portion performs the steps required to load and pass control to the program specified on the LOAD OCL statement.

The initiator can be loaded by any of four programming functions:

- Command processor, when starting a new job
- Job step termination, when starting a new job step
- Release, when returning to a procedure from a multiple requester terminal (MRT) program or a released program
- Initiator procedure start setup, when evoking a startup procedure

The following table shows the locations of key data areas involved in the initiator subfunction:

Dump Address (in Hex)	Contents
1000	Mainline module
3000	Initiator work area
3800	Overflow initiator work area
4000	Procedure parameter save area (PPSA)
4800	SYSIN work area
5000	Active procedure table

The following initiator functions are shown in Chart 3.1:

1 Perform preliminary syntax checking of the statement and, if necessary, encode the parameters. Place the encoded parameters in the initiator work area (IWA), which is in task work space (TWS), for later use by the statement processor modules. If errors are detected, pass control to the initiator error handler; otherwise, route control to the appropriate verb processing module for each statement.

2 Check the statement for valid parameters and build or modify the appropriate data area (*Data Area Modified(M) or Built(B)*) for use by the system in processing the job.

3 Read the next statement. If not a RUN statement, pass control to verb processing module or initiator error handler, as before.

4 Issue error message based on message identification code (MIC) passed by verb processing module.

5 Issue error message based on MIC passed by initiator mainline.

6 Process RUN statement by performing the following program initialization processes:

- If the requested program is one of the program products requiring source, allocate and open the $SOURCE and $WORK work files. Also read the source statements and place them in the $SOURCE file.
- Enqueue required disk files, based on parameters in file statements, and acquire all required work stations.

7 If errors are detected, pass control to the initiator error handler; otherwise, load and pass control to the supervisor task attach transient.

8 Load and pass control to the requested program.

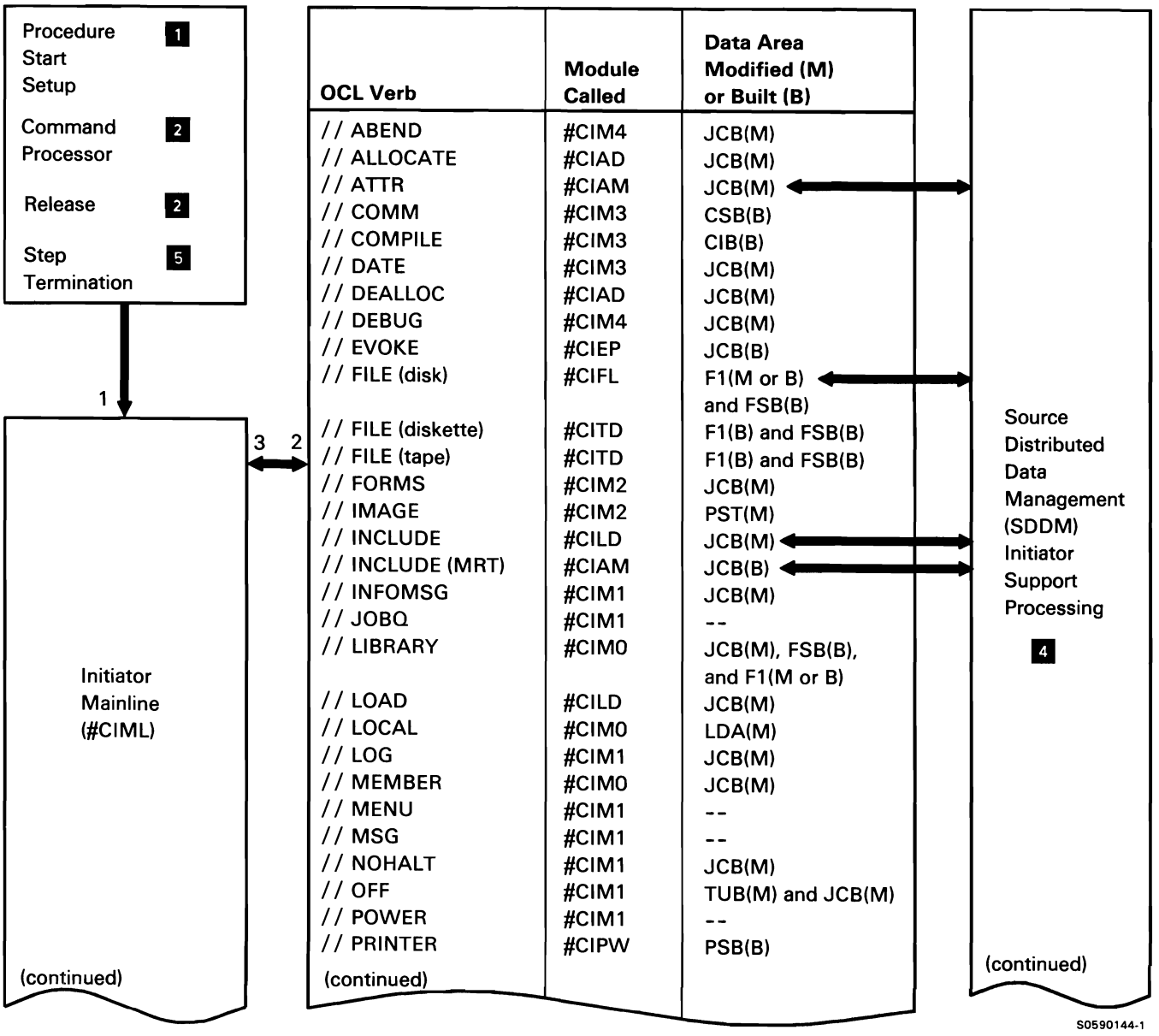


Chart 3.1 (Part 1 of 2) Initiator Control Flow

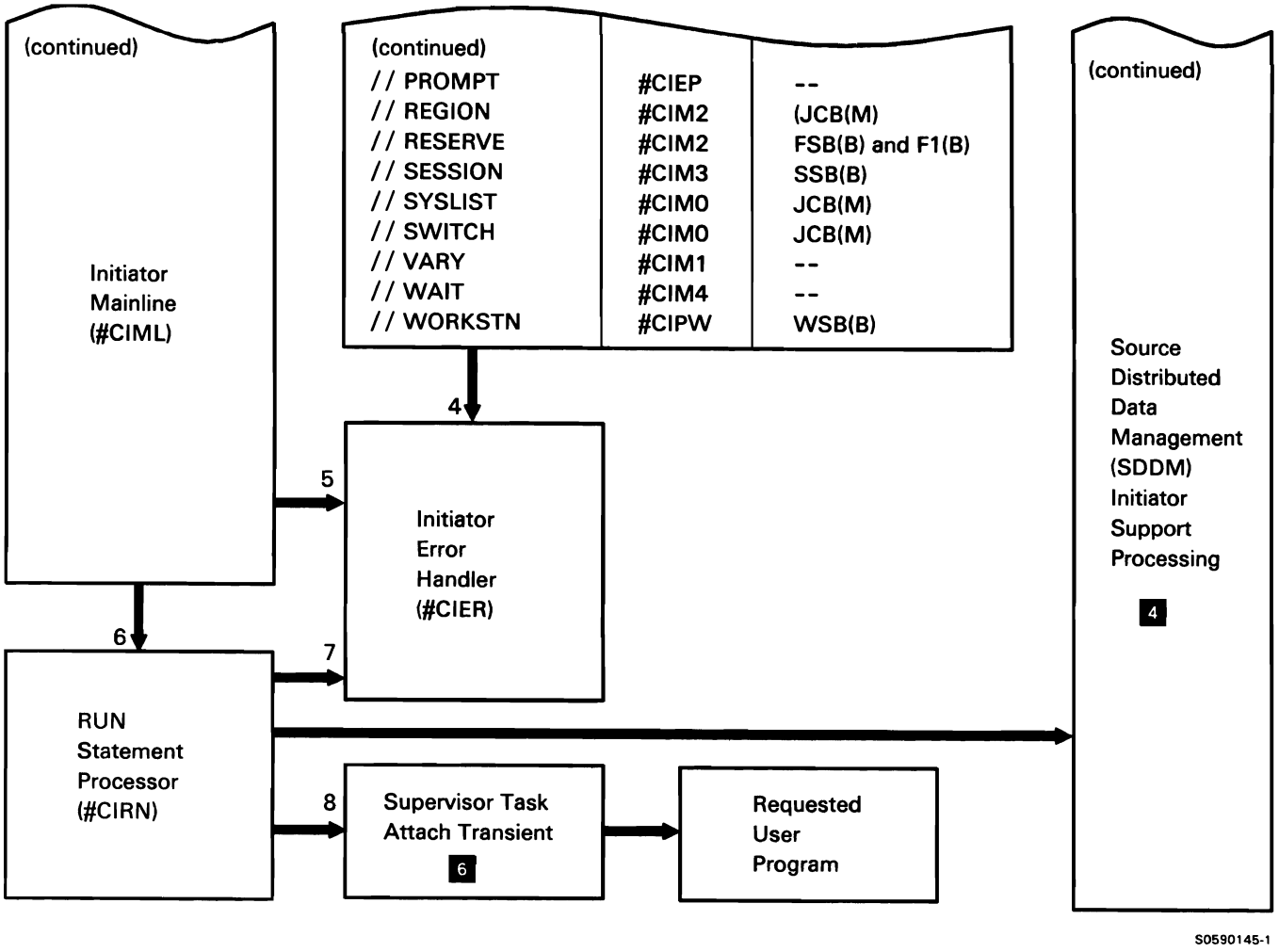


Chart 3.1 (Part 2 of 2) Initiator Control Flow

ALLOCATE SUBFUNCTION

The allocate subfunction controls the ownership of system devices and disk space for user (nonprivileged) or SSP (privileged) programs. It consists of four parts:

- Normal allocate: Allocates disk space and devices
- Special allocate: Allocates disk files, disk space, and printers
- Deallocate: Deallocates disk files, disk space, and devices
- Privileged communications allocate: Allocates communications lines for BSC and SDLC tasks

Allocate Region Map

The following table shows the locations of key data areas involved in the allocate subfunction:

Dump Address (in Hex)	Contents
1000	Mainline module
3800	Allocate work area
4000	Format-5 buffer
5000	Parameter list
6000	DTF

Normal Allocate

Normal allocate controls the ownership of devices and disk space for the user program. Normal allocate is the only allocate function that can be called by a nonprivileged user; user programs call allocate via a transfer SVC instruction with XR2 pointing to a chain of one or more DTFs. Normal allocate processes as logical transients.

Normal allocate is also called by privileged SSP programs to acquire ownership of system devices such as the diskette drive, the tape drive(s), and the 1255 MCR.

The following normal allocate processes are shown in Chart 3.2.1:

1 Do the following mainline functions:

- Acquire necessary task work space (TWS).
- Set indicators for other DTF processing required.
- Perform central router and mainline function for the processing of each applicable DTF on the chain.
- When processing for all allocation modules is completed, delete TWS and return to caller.

2 Perform the following disk file processing:

- If resource security is active, call security to check for user authorization.
- Allocate old files for input operations.
- Reset old-files-for-output attributes as required.
- Set attributes for new files according to FILE statement parameters and DTF.
- If requested file is remote, route control to source distributed data management allocate.

3 Perform the following for diskette files:

- Verify file format, check for file expiration, and check for file date differentiation.
- Display applicable errors to user via SYSLOG.

4 Do the following for BSC files:

- Acquire line and load necessary control storage communications programming.
- Call task attach to attach BSC interrupt handler.

5 Do the following for 1255 MCR files:

- Acquire 1255.
- Load necessary control storage MCR programming.

Special Allocate

Special allocate allows privileged SSP programs to perform initiator and allocate functions without user-specified OCL. The special allocate subfunction is evoked by a transfer SVC instruction with XR2 pointing to the special allocate parameter list; it processes only one request per call.

The privileged user has the option of processing errors, or allowing allocate to process them. If allocate does not process the errors, they are passed to the caller as MICs in the parameter list return code field. Special allocate processes as a logical transient.

The following special allocate processes are shown in Chart 3.2.2:

1 Do the following initiator and first-stage allocate functions:

- Acquire necessary task work space (TWS).
- Get request from parameter list.
- Ensure special allocate request is for printer file or disk file only.
- Perform central router and mainline functions for other required special allocate modules.
- If requested file is remote, route control to source distributed data management allocate.
- When all special allocate modules complete processing, delete TWS and return to caller.

2 Check for file existence when required.

3 Process multiple index files. Call VTOC access to get multiple index family from the VTOC.

4 Perform the following disk file processing:

- If resource security is active, call security to check for user authorization.
- Allocate old files for input operations.
- Reset old-files-for-output attributes as required.
- Set attributes for new files according to parameter list and DTF.
- If requested file is remote, route control to source distributed data management allocate.

5 Do the following for new disk file allocation:

- Calculate the required blocks.
- If request is for an S or J file and reserve area is available, get space from reserve area; otherwise, use system format-5 free space table to claim space.

6 Perform the following disk file processing for each output disk file being allocated:

- Calculate file capacity in number of records.
- Format files for file recovery on resident disk files Update or add a VTOC entry for each resident disk file.
- Establish disk sector addresses (index and data areas for indexed files).

7 Do the following for printer DTFs:

- Find or build the PSB and IOB. If possible, acquire the requested printer.
- If spooling is active for the printer, set up print file for print spooling.
- If request is for remote printer that is not available, call RWS vary on to check for printer condition.
- If printer file is direct and if printer type verification is requested, verify IGC printer type (IGC, IGC18, or IGC24).
- If printer file is direct, call image verify to confirm image, forms number, CPI, LPI, Ideographic CPI (IGCPI), and Ideographic shift in, shift out (SOSI).

8 Prepare data to be issued to SYSLOG (or for the return code to be set in the parameter list) for errors detected in any of the allocate modules.

Note: Although control flow is not shown, all allocate modules can call #CAMG directly.

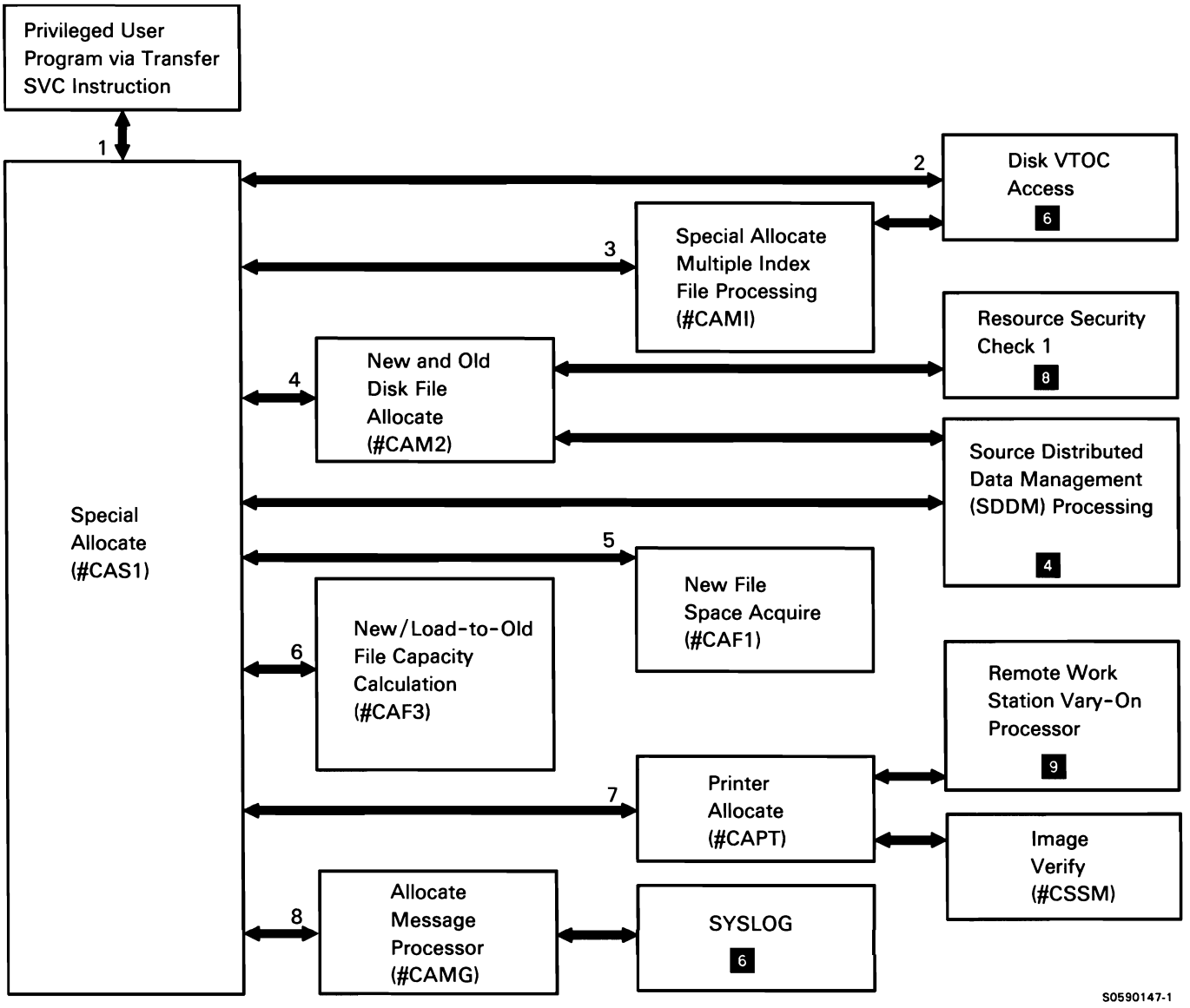


Chart 3.2.2 Special Allocate Control Flow

Deallocate

Deallocate allows a privileged SSP program to perform termination functions on disk files, printer files, and the tape or diskette drive without going through end-of-job processing. Privileged SSP programs call deallocate via a transfer SVC instruction with XR2 pointing to a special disk allocate parameter list, a special printer allocate parameter list, or a diskette, tape, MCR, or printer DTF. Deallocate processes only one request per call and processes as logical transients.

Privileged SSP programs have the option of processing errors, or allowing deallocate to process them. If deallocate does not process the errors, they are passed to the caller as MICs in the parameter list return code field.

The following deallocate processes are shown in Chart 3.2.3:

1 Do the following:

- Acquire necessary task work space (TWS).
- Initialize for deallocate.
- Check for valid deallocate call.
- Route for required processing.
- After all deallocate processing is complete, return to caller.

2 If resource security is active, ensure proper user authorization for deleting or changing a disk file.

3 If request is for remote file, perform distributed data management file deallocate.

4 If required, check for duplicate keys and perform file sort.

5 Test for file existence and perform latest date checking.

6 If multiple index file, deallocate it.

7 Perform deallocate function on printer files, diskette drive, 1255 MCR, and tape drive.

8 Free file space.

9 Prepare data to be issued to SYSLOG (or for the return code to be set in the parameter list) for errors detected in any of the allocate modules.

Note: Although control flow is not shown, all deallocate modules can call #CAMG directly.

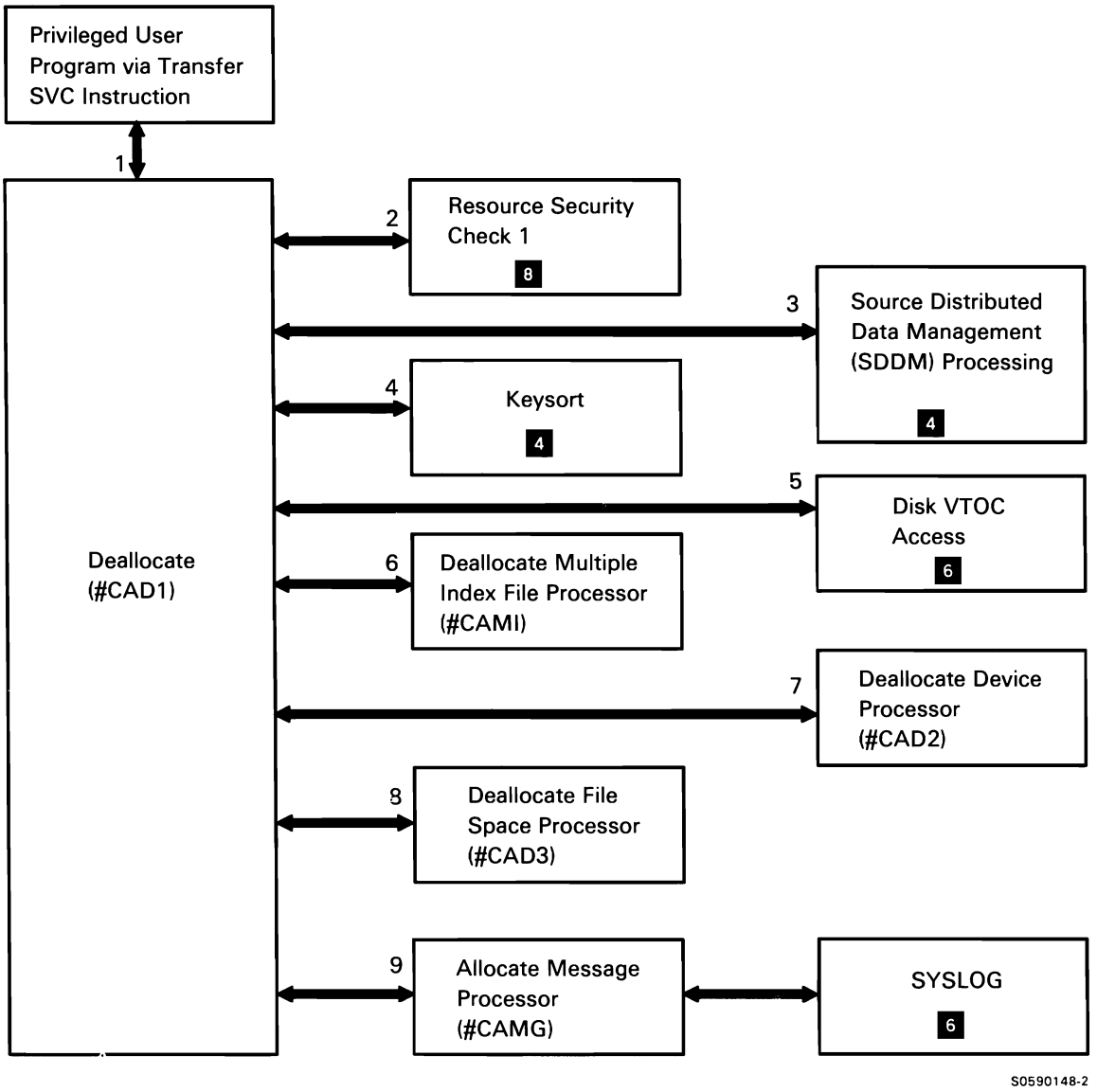


Chart 3.2.3 Deallocate Control Flow

Data Communications Allocate

Data communications allocate processes line allocate requests by doing the following, as required:

- Acquire communications line
- Load control storage microcode
- Build communications specification block (CSB)
- Build error recovery block (MLCA/ELCA only)

Data communications allocate is invoked by both privileged SSP programs and by user programs. When a privileged SSP program invokes allocate, XR2 points to the parameter list; when a user program invokes allocate, XR2 points to a chain of DTFs containing at least one data communications DTF.

The following user program data communications allocate processes are shown in Chart 3.2.4.1:

1 For each communications DTF in the chain, perform, or route for, the following:

   - Find or build a CSB for the line and acquire (resource enqueue) the line.
   - Load the appropriate communications microcode if not already loaded, and update the device allocate table.
   - If MLCA/ELCA, find or create an ERB.
   - Mark DTF and CSB as allocated.

2 Attach data communications task.

3 Issue required messages.

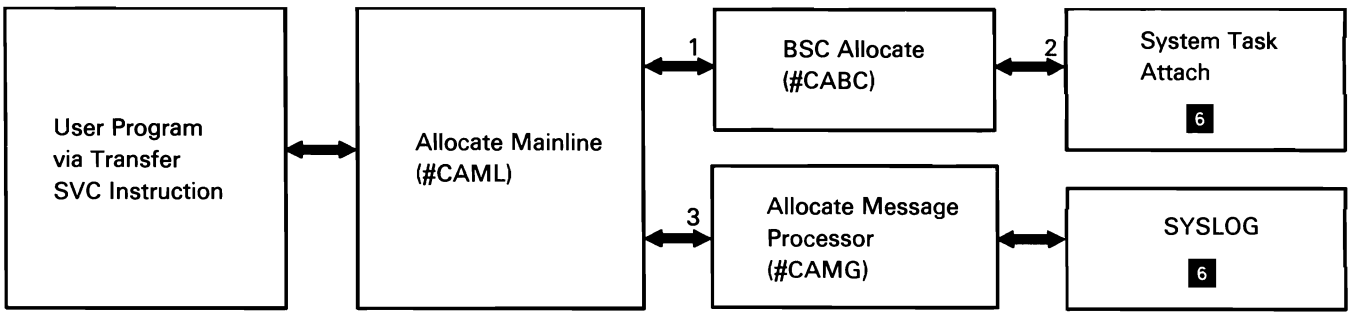


Chart 3.2.4.1 Data Communications Allocate Control Flow

The following *privileged* SSP program data communications allocate/deallocate processes are shown in Chart 3.2.4.2:

1 If SDLC parameter list, perform or route for the following:

- Find or build a CSB for the line and acquire (resource enqueue) the line.
- Load the appropriate data communications microcode (if not already loaded) and update the device allocate table.
- If MLCA/ELCA, find or create an ERB.
- Update the CSB and parameter list to show allocate status.

2 If BSC or asynchronous parameter list, perform or route for the following:

- Find or build a CSB for the line and acquire (resource enqueue) the line.
- Load the appropriate communications microcode (if not already loaded) and update the device allocate table.
- If MLCA/ELCA, find or create an ERB.
- Attach the appropriate data communications data management task.
- Update the CSB and parameter list to show allocate status.

3 Perform or route for deallocate requests:

- If there are no other line users, dequeue and free CSB and release (resource dequeue) the communications line.
- If there are no other users of the control storage microcode, unload the microcode and update the device allocate table.
- If MLCA/ELCA, and this is the last user of the ERB, dequeue and free ERB.
- Update parameter list to show successful completion.
- Return any error messages required in the caller's parameter list.

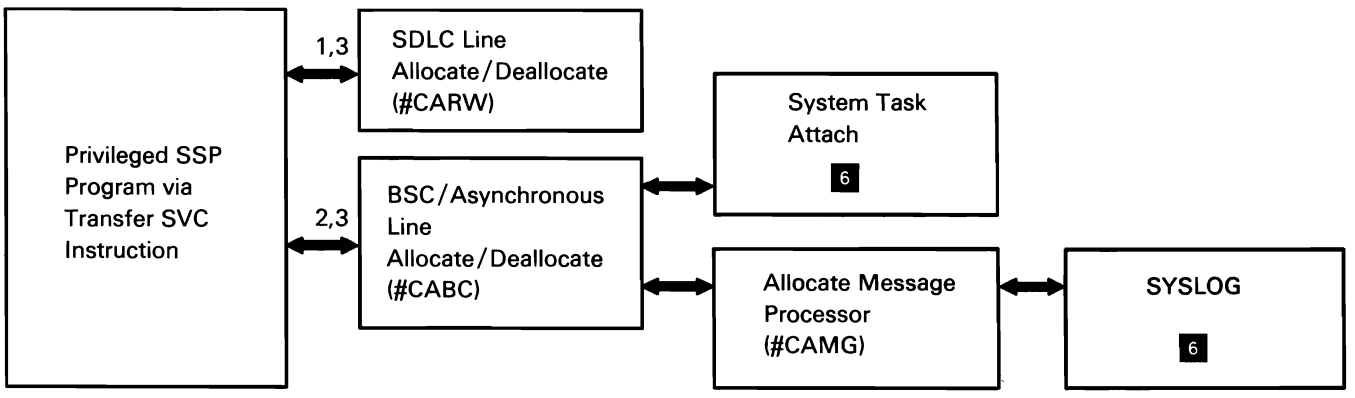


Chart 3.2.4.2 Data Communications Special Allocate Control Flow

OPEN SUBFUNCTION

The open subfunction prepares for the transfer of data to and from files and devices allocated to a program by:

- Initializing all necessary file DTF fields
- Assigning buffers, control blocks, and any required IOBs
- Preparing for I/O operations to/from data files and devices

Common open is contained within the first disk open transient and the printer open transient. Open consists of two common open transients and the following device-oriented open transients:

- Disk open
- Work station open
- Diskette open
- Printer open
- BSC open
- Tape open

The calling program invokes open with a transfer SVC instruction and XR2 pointing to the first DTF on the DTF chain. The control storage SVC processor loads the first common open transient (#DD10P).

The following open processes are shown in Chart 3.3.1:

1 Perform the following:

- Check all DTFs on the forward chain for valid device codes.
- Check that all DTFs to be opened have been allocated.
- If errors found, issue messages through SYSLOG.
- If no disk DTFs are on the chain, call other device opens.
- Perform diagnostic checking for all disk DTFs and issue any required messages through SYSLOG.
- If a privileged program requests it, perform disk pseudo open by moving the following information from format 1 to the DTF:
  - Record length
  - Block length
  - File type
- Complete processing for any sectorized data management access method (ZPAM) DTFs and route control to #DD2OP for other disk DTF processing.

2 Perform the following on disk (other than ZPAM) DTFs:

- Verify format 1, then copy format-1 data to DTF and FAB.
- Calculate file space needed for physical buffers and disk data management required control blocks.
- Determine if location for file space is within program (privileged programs only), appended to program, or in TWS.
- Map to file space.
- Format file space and set up SQS data areas for disk data management.
- Call keysort or quicksort if required.
- If DTF is for a remote file, call distributed data management open.
- If indexed file or multiple index file (and file is not remote), call #DD3OP.

3 Process disk DTF open requests from a remote system for local files or from a local user for remote files.

4 Perform the following indexed-file-DTF open functions:

- Set up high key data areas required for indexed files.
- Set up storage index for primary portion of file index.

5 Open any diskette DTFs on the forward chain.

6 Perform the following work station DTF open functions:

- Validate device type and build format index area (offsets from load member disk address) for format load members for the following:
  - Keyboard DTFs.
  - Console DTFs.
  - CRT DTFs.
  - Work station DTFs.
- Pass control to printer data management open.

7 Place all DTFs that are on the forward chain on the JCB backward chain, then process any printer DTFs on the forward chain:

- Check for valid entries in the user DTF.
- Initialize fields in the user DTF.
- Check that an IOB was assigned by allocate; initialize the IOB fields.
- Send initial SCS (printer prepare character string) commands to the printer.

8 Process any BSC DTFs on the forward chain, then return control to printer data management open.

9 Execute job with opened DTFs.

10 Open all tape DTFs:

- Build tape IOB.
- Set up physical buffer.
- Load appropriate tape data management module.

11 Issue any required error messages.

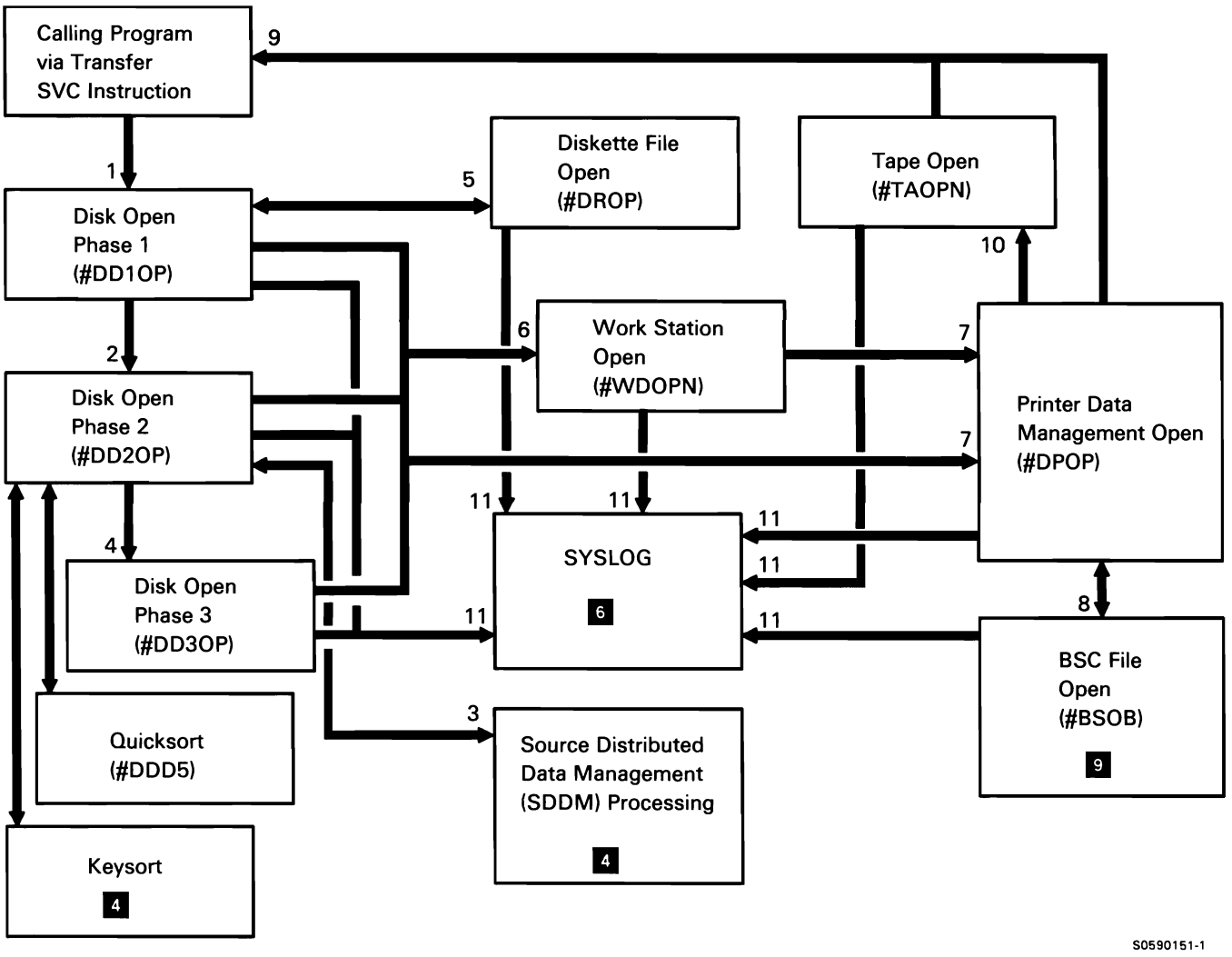


Chart 3.3.1 Open Control Flow

Job Execution Function 4

The job execution function consists of common data handling subfunctions that are used by application programs and/or privileged SSP programs. It includes the following subfunctions:

- Data management processing — Chart 4.0
- Disk data management processing — Chart 4.1.1
- Diskette data management — Chart 4.2.1
- Printer data management — Chart 4.3.1
- Work station data management — Chart 4.4.0
- Print spooling — Chart 4.5.0
- Sector data management to disk — Chart 4.6
- Keysort — Chart 4.7
- Tape data management — Chart 4.8
- 1255 MCR data management — Chart 4.9
- OFFICE/36 support — Chart 4.10.0
- Distributed data management (DDM) — Chart 4.11.0
- Query data management (QDM) — Chart 4.12.1

DATA MANAGEMENT PROCESSING SUBFUNCTIONS

SSP data management subfunctions provide an SVC/DTF interface to calling programs, and common error detection/recovery for user files. The disk, printer, and work station data management subfunctions are invoked via the fast transfer SVC instruction; front-end code in each of these subfunctions determines if the request can be responded to immediately or if it must be delayed due to processing requirement.

The data management for each device ensures that the DTF is open and then checks the DTF device code to determine the type of data management requested. Chart 4.0 shows the overview control flow for the data management subfunctions.

The following preliminary data management processes are shown in Chart 4.0:

1 Check fast transfer request to determine if I/O or other time-consuming operation is required to process request.

2 If requested data is in a buffer, if the requested operation requires minimal time, or if DTF is not open, fast exit to caller with completion code.

3 If detail processing is required, fast exit to process request and return to caller with completion code.

4 Process data management service requested in user's DTF.

5 Perform detail data management service requested.

6 Perform detail OFFICE/36 support service requested.

7 Perform detail distributed data management service requested.

8 Perform detail query processing requested.

Note: The return to user is not necessarily from the detail operation processor. In some cases, the applicable data management router receives control back and then passes control and the completion code back to the caller.

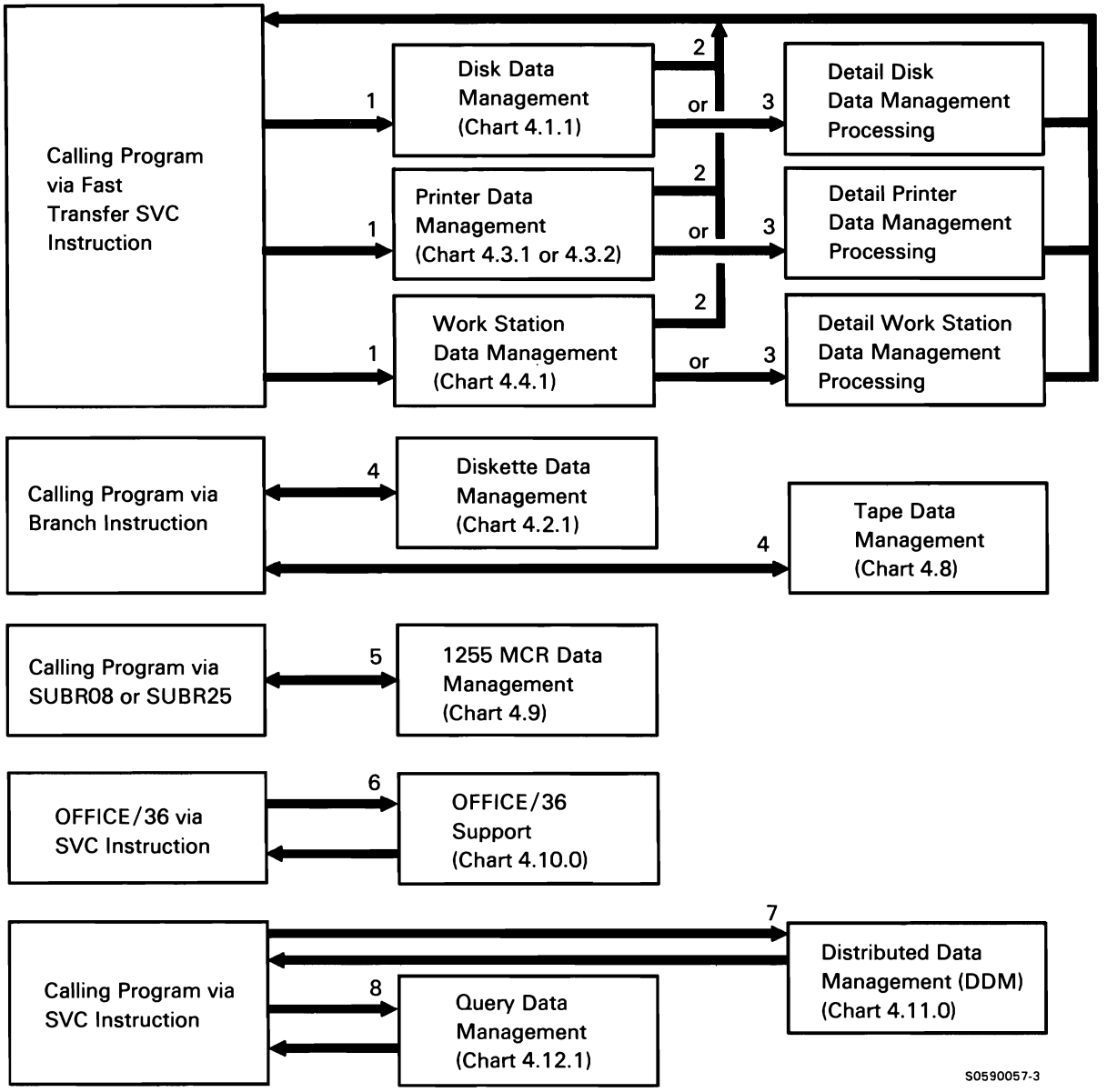


Chart 4.0 Data Management Overview Control Flow

Disk Data Management

Disk data management performs all main storage disk file processing. It consists of a resident portion that is loaded by main storage IPL and a set of system transients.

Disk data management provides support for the processing of the four System/36 disk file types:

- Sequential: Records are in the order that they were originally entered.
- Indexed: Records are in any order, with access by an index that contains a key or physical location entry for each record.
- Direct: Records are ordered by a programming algorithm (usually based on a field value in the record).
- Multiple Indexed: Records are in any order, with access by a conventional indexed-file index that contains a key or physical location entry for each record.

Disk data management provides support for the processing of the following user requests:

- Get
- Update
- Delete
- Add
- Release

Disk Data Management Region Map

The disk data management mainline module (#DDDM) resides in the fixed nucleus. Disk data management data areas can reside in one of two places, dependent on the size of the application program region.

If the user program region is large enough, DTFs and logical buffers reside where they were compiled in the user's program. The physical buffers are appended to the end of the user's program.

If the application program region is too small, the disk data management data areas reside in TWS at the locations shown in the following table:

Address (in Hex)	Contents
2000	DTFs
3000	Logical buffer
4800	Physical buffers

Generally, callers prepare for a call to disk data management by doing the following:

- Set up DTF (done by allocate, using FILE statement).
- Set up file access block (FAB), file buffer block (FBB), and disk buffer block (DBB) (done by open).
- For a put, place data in logical buffer; for a get, set up applicable record pointer in DTF.
- Set op code and file attributes in DTF and point XR2 at DTF.
- Issue fast transfer SVC instruction.

This page is intentionally left blank.

The following disk data management processes are shown in Chart 4.1.1:

1 Examine request and determine if it can be processed immediately or if it must be delayed.

- If request can be processed immediately, move data to or from caller; then return to caller via fast exit SVC instruction.
- If request requires delayed processing, go to mainline processing via fast exit SVC instruction.
- If request is for a remote file, route control to distributed data management get/put processing.

2 Process get/put request for remote file.

3 Do the following mainline processing:

- Verify that file is open and that access method is not sectorized data management access method (ZPAM).
- If necessary, map to file I/O area.
- Route to appropriate operation handler, based on request.
- If request was terminated because record was locked, retry.
- Return completion code to caller, based on completion code received from operation handler.

4 Do disk I/O operations:

- Get
- Update
- Delete
- Add

5 Process fix-end-of-data request:

- Verify that file is allocated exclusively, that file is sequential, that no indexes are defined, and that processing includes a delete.
- Reset last record address (in FAB) to current cursor plus 1, and reformat remainder of file to zeros.

6 Process set beginning-of-file (BOF)/end-of-file (EOF) request by resetting cursor state position (in FAB) to BOF or EOF.

7 Process exception condition:

- Initialization of variables for selection expression.
- Abnormal termination.
- Implicit update/delete read.
- File extension initialization.
- Keysort.

If operator selected SYSLOG option 2 in response to an error, return to caller; otherwise, return condition to #DDDM.

8 Process index entry insert request for each index to be updated:

- Ensure that buffer is large enough for in-storage insert or sort.
- If buffer is not large enough, determine exact point of insert, and determine whether an in-storage insert or sort is possible.
- If the buffer is not large enough, scan the index for a gap. If a gap is found, determine whether an in-storage insert or sort is possible.
- If an in-storage insert is possible, perform it; otherwise, shift overflow by 1 sector, starting at insert point and insert new entry.
- Update the index control block (XCB) queue.

9 Perform record selection based on selection expression passed by caller ($COPY or other utility):

- Check record against token in string with nested do-until loops; continue do-until loops until true, error, or end-of-file condition is detected.
- Return condition to #DDDM.

10 Process release request; unlock record by turning off lock bit in record queue block (RQB).

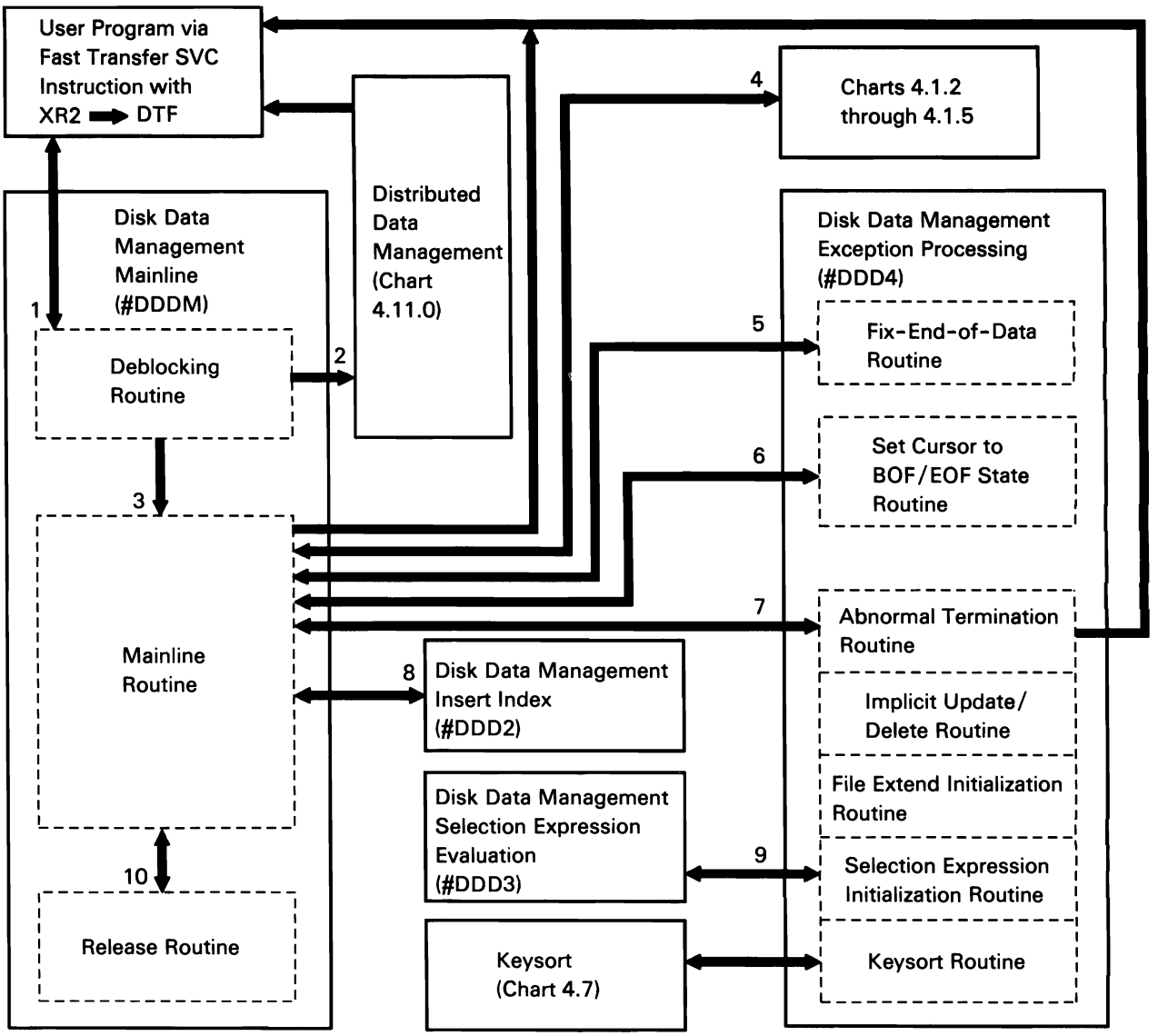


Chart 4.1.1 Disk Data Management Control Flow

Disk Data Management Get Request Processing

Disk data management get request processing proceeds in the following general sequence:

- If get by key, move key (pointed to by DTF) into index buffer.
- Insert scan op code and SSS@ into IOB.
- Move index buffer address from file buffer block (FBB) into IOB and point XR1 at IOB.
- Call control storage disk IOS to scan sectors and return the sector containing the requested key; move the RRN to the work area.
- If non-indexed-random get, move RRN from DTF to work area; if unkeyed sequential get, move RRN from FAB to work area.
- Convert work area RRN to SSS@ and insert read op code and SSS@ into IOB.
- Call control storage disk IOS to perform get.

The following disk data management get-request processes are shown in Chart 4.1.2:

1 Do the following:

- Check for request validity.
- Route for keyed or unkeyed get.
- Update cursor in FAB.
- Dequeue the file via format 1.
- Return DTF with completion code.

2 Perform unkeyed get:

- If shared file, enqueue file via format 1.
- Determine RRN of requested record.
- Route to get record routine.
- If requested record has been deleted or if record does not match a selection expression, read next record or terminate, based on request type.

3 Perform get:

- If operation and file sharing level require it, enqueue record.
- If record is locked, return to mainline for retry.
- If record is already in disk data management data area, move it to user's record buffer; otherwise, call control storage disk IOS to read sector(s) containing record from disk to disk data management data area.
- Copy record from disk data management data area to user's record buffer.

4 Perform keyed get:

- If shared file, enqueue data space via format 1.
- If next record is specified, determine prime and overflow entries relative to current key. The lesser of the two becomes the new current key. If no entry is found, terminate.
- If previous record is specified, determine prime and overflow entries relative to current key. The greater of the two becomes the new current key. If no is entry found, terminate.
- If first is specified, start at beginning of index; if last is specified, start at end of index.
- If get by key equal, high/equal, or high is specified, scan index for lowest key higher/equal to requested key. If no entry found, terminate.
- Determine RRN of requested record.
- Route to get record routine.
- If requested record has been deleted or if record does not match a selection expression, read next/previous record or terminate, based on request type.
- If limits were set and the record does not satisfy them, abnormally terminate the request.

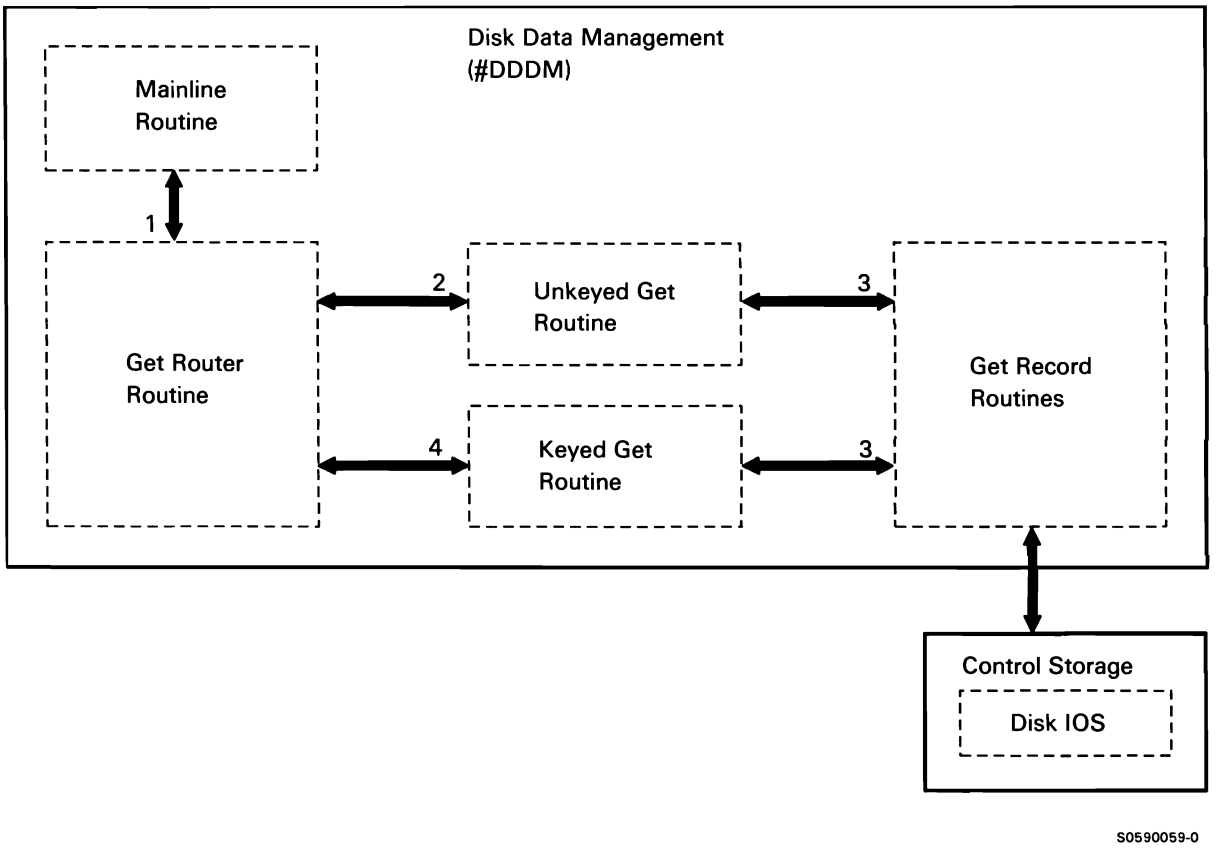


Chart 4.1.2 Disk Data Management Get Control Flow

Disk Data Management Update Request Processing

A get on the requested record must be performed before the update can be done. The requested record is contained in the user's buffer.

The following disk data management update-request processes are shown in Chart 4.1.3:

1 Do the following:

- Check FAB to ensure update is allowed.
- If file is delete-capable, check that first byte of record is not hex FF.
- If shared file, enqueue data space via format 1.
- Compare old key to new key.
- If key modification has been performed, ensure that index allows it.
- If index requires unique keys, check for duplicates.
- Move/write record.
- For each alternative index that required updating, read the entry corresponding to the updated record. Delete the old entry and add a new entry to each index.
- Dequeue data space via format 1.
- Return DTF with completion code.

2 Move/write record:

- Move record from user buffer to physical buffer.
- Call control storage disk IOS to write modified record to disk.

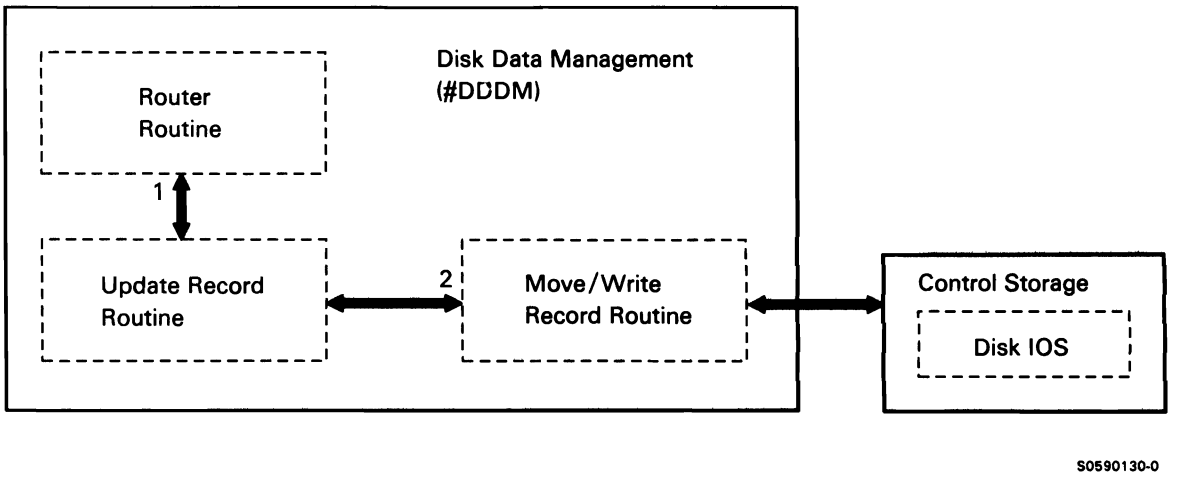


Chart 4.1.3 Disk Data Management Update Control Flow

Disk Data Management Delete Request Processing

A get operation on the requested record must be performed before the delete can be done. The requested record is contained in the user's buffer.

The following disk data management delete-request processes are shown in Chart 4.1.4:

1 Do the following:

- Check FAB to ensure that delete is allowed and that file is delete-capable.
- If shared file, enqueue data space via format 1.
- Mark record in physical buffer as deleted by filling record bytes with hex FF.
- If keyed, place hex 00 in key and hex FF in RRN.
- Call control storage disk IOS to write record to disk.
- For each alternative index defined over the file, read the entry corresponding to the deleted record. Mark the entry as deleted and call control storage disk IOS to write it back to disk.
- Dequeue data space via format 1.
- Return DTF with completion code.

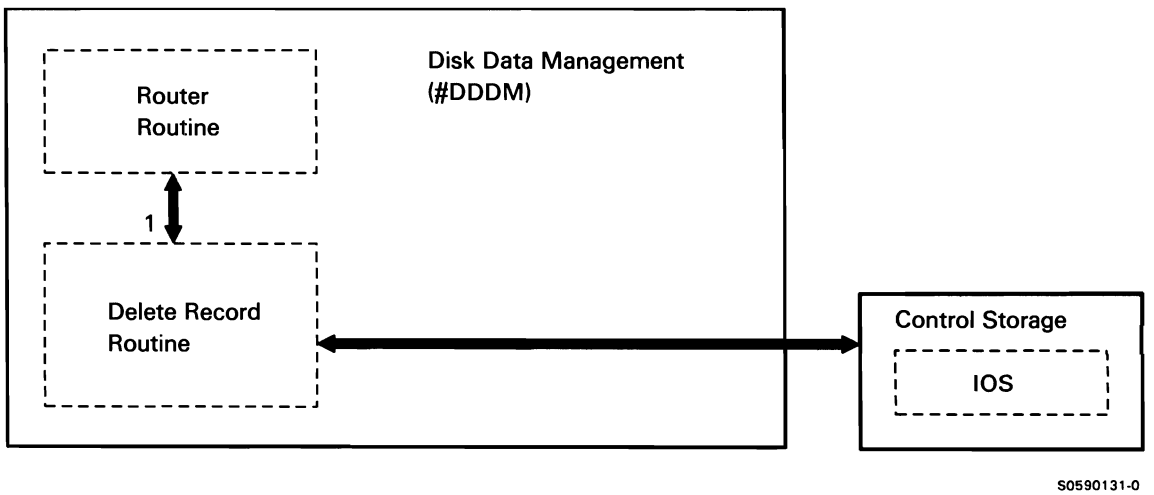


Chart 4.1.4 Disk Data Management Delete Control Flow

Disk Data Management Add Request Processing

Disk data management add request processing proceeds in the following general sequence:

- Move logical buffer to physical buffer.
- Determine if buffer must be written to disk by checking if buffer full condition or file sharing is applicable.
- If buffer must be written to disk, move physical buffer address from file buffer block (FBB) into IOB and point XR1 at IOB.
- Call control storage disk IOS to write buffer to disk.
- On return, get new key from user and add to or alter index buffer.

The following disk data management add request processes are shown in Chart 4.1.5:

1 Do the following:

- If file is delete-capable, ensure that first byte of record is not hex FF.
- If shared file, enqueue file via format 1.
- Route to subroutine to check indexes for invalid duplicate keys.
- If add (RRN) is specified:
  - Verify (using FAB) that the operation includes add (RRN).
  - Verify that file is delete-capable or new direct file.
  - Read the specified record into the physical buffer.
  - If the file is delete-capable and record slot does contain a record, abnormally terminate; otherwise, move record from user's buffer to physical buffer, and call control storage disk IOS to write it to disk.
- If add (EOD) is specified:
  - Verify (using FAB) that the operation includes add (EOD).
  - Verify that file is not a direct file.
  - Add record to end of file.
- If the last operation was an add (EOD) and output is deferred, move record from user's buffer to physical buffer without reading disk.
- If the last operation was not an add (EOD) or output is not deferred, read disk and move record space into physical buffer.
- For each index defined over the data space, add a new entry for the added record. Depending on sharing level, file attributes, and processing requested:
  - Mark format 1 to indicate file sort required.
  - Add key to end of overflow area or insert it, to preserve sequence.
- Dequeue data space via format 1.
- Return DTF with completion code.

2 Check each index defined over the data space. If duplicate keys are not allowed, ensure that added key is unique. If duplicate key found, abnormally terminate.

3 Insert key in index overflow area.

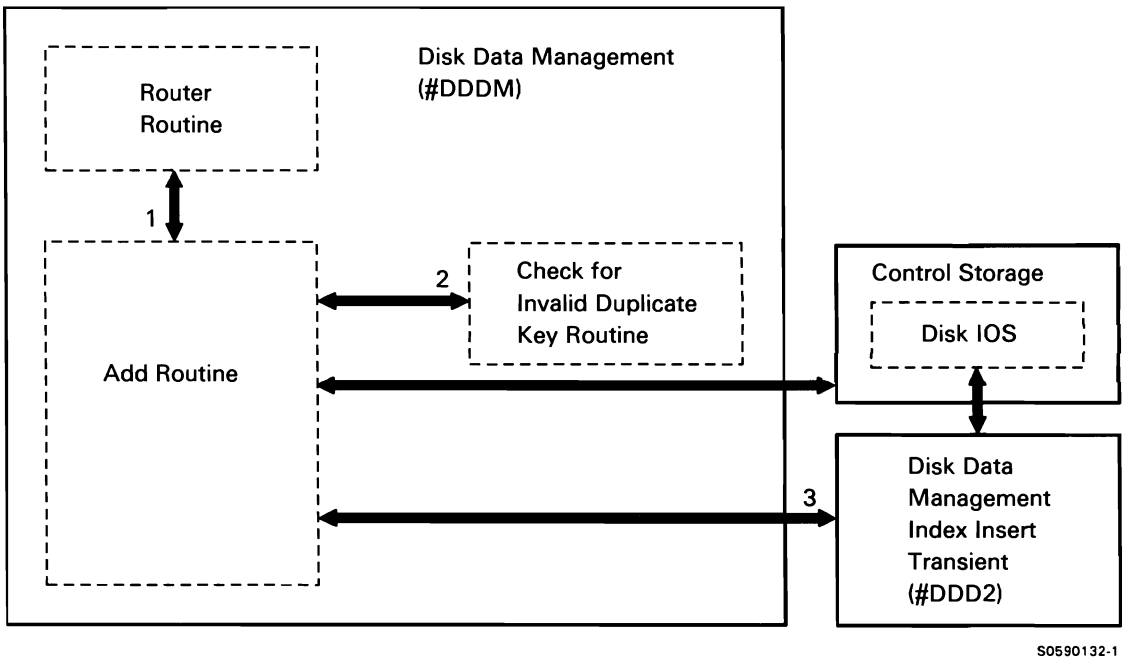


Chart 4.1.5 Disk Data Management Add Control Flow

DISKETTE DATA MANAGEMENT

Diskette data management is loaded by diskette open. It processes three file types:

- Basic data exchange files
- I-exchange files
- System files

Basic data exchange files are created by the System/36 $BICR utility, the System/36 $MAINT utility with BASIC-YES specified, or are copied by the System/36 $DUPRD utility. I-exchange files are created by the System/36 $BICR utility, or copied by the System/36 $DUPRD utility. System files are created by the System/36 $COPY or $MAINT utility or copied by the System/36 $DUPRD utility.

Diskette data management can process in either record mode or sector mode. The following *record mode* data management access types support the file types listed above:

- Put basic record (PBR)
- Get basic record (GBR)
- Add basic record (ABR)
- Put I-record (PIR)
- Get I-record (GIR)
- Add I-record (AIR)
- Put system record (PSR)
- Get system record (GSR)
- Add system record (ASR)

Sector mode data management provides users with the capability of moving considerable amounts of data to or from diskette. It uses a single input/output buffer that is filled by a single read operation, or is written to disk with a single write operation. Data management is loaded by diskette open.

The following diskette data management processes are shown in Chart 4.2.1:

1 Perform diskette data management services:

- Initialize for user request.
- Check for end of data.
- For move mode, block or deblock records between work buffer and I/O buffer.
- For locate mode, point DTF to location in I/O buffer containing requested input data or to location in I/O buffer where output data is to be placed.
- Read or write I/O buffer.
- Issue SVCs for diskette IOS.
- Restore pointers and data areas.
- If end-of-volume processing is required, start it; otherwise, set DTF completion code and return to user.

2 Process end of volume:

- If required, rewrite data set label.
- If required, set continuation indicator.
- If required, issue message (via SYSLOG 5) to operator.

3 If required, read/write diskette VTOC.

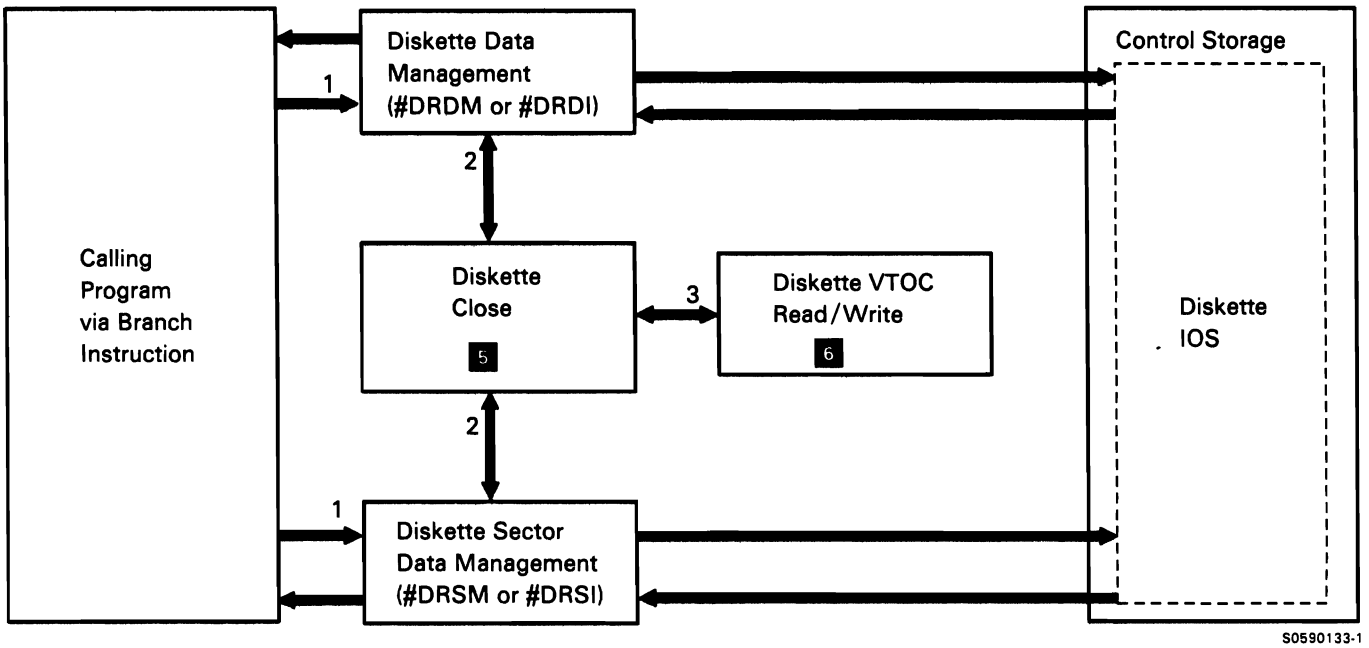


Chart 4.2.1 Diskette Data Management Control Flow

PRINTER DATA MANAGEMENT

Printer data management prepares and moves all print requests from user programs, and *most* print requests from system programs. (Some system programs use SYSLIST 6.) If spooling is active, printer data management prepares all print requests and routes the data to the spool file. Calling programs put the data into a logical buffer, insert skip and space values into the DTF, point XR2 at the DTF, then call the printer data management spool intercept router via the fast transfer SVC instruction.

The printer data management spool intercept router (#SPRT) is a nucleus resident program; all other printer data management modules are translated transients.

Printer data management processes for nonspooled requests are shown in Chart 4.3.1; printer data management processes for spooled requests are shown in Chart 4.3.2.

Printer Data Management Region Map

The following table shows the locations of key data areas involved in the printer data management subfunction:

Dump Address (in Hex)	Contents
0800	Nucleus
1000	Printer data management (#DPDM)
1800	Spool intercept (#SPINT)
2000	Printer DTF (mapped to when printer data management processes transparent-mode records and nonspooled requests)
3000	Logical buffer (mapped to when printer data management processes transparent mode records)
4000	Spool intercept buffer

The *active work block* (address contained in a queue header) contains pointers to the printer DTF, the printer IOB, the logical buffer and the physical buffer.

The following printer data management processes *for a nonspooled request* are shown in Chart 4.3.1:

1 Do the following:

- Set normal completion code in DTF.
- Move skip/space information from DTF to IOB.
- Set up physical buffer in SQS:
  - Move logical buffer to physical buffer.
  - Move data string length to IOB.
  - Set print indication in IOB control byte.

2 Unless a transparent-mode record or a get printer status request, prepare print buffer and check for EXTN characters.

3 Point XR1 at the IOB and call printer data management via the fast exit SVC instruction.

4 Check for the following:

- Data includes EXTN characters.
- Print request is for local printer.
- Print request is for remote printer.

5 If EXTN characters are in print request, convert them to RAM addresses.

6 If intelligent printer data stream (IPDS) printer, call intelligent printer facility (IPF).

7 If local print request, execute it.

8 If remote work station request, route request to remote printer.

9 Return DTF with completion code.

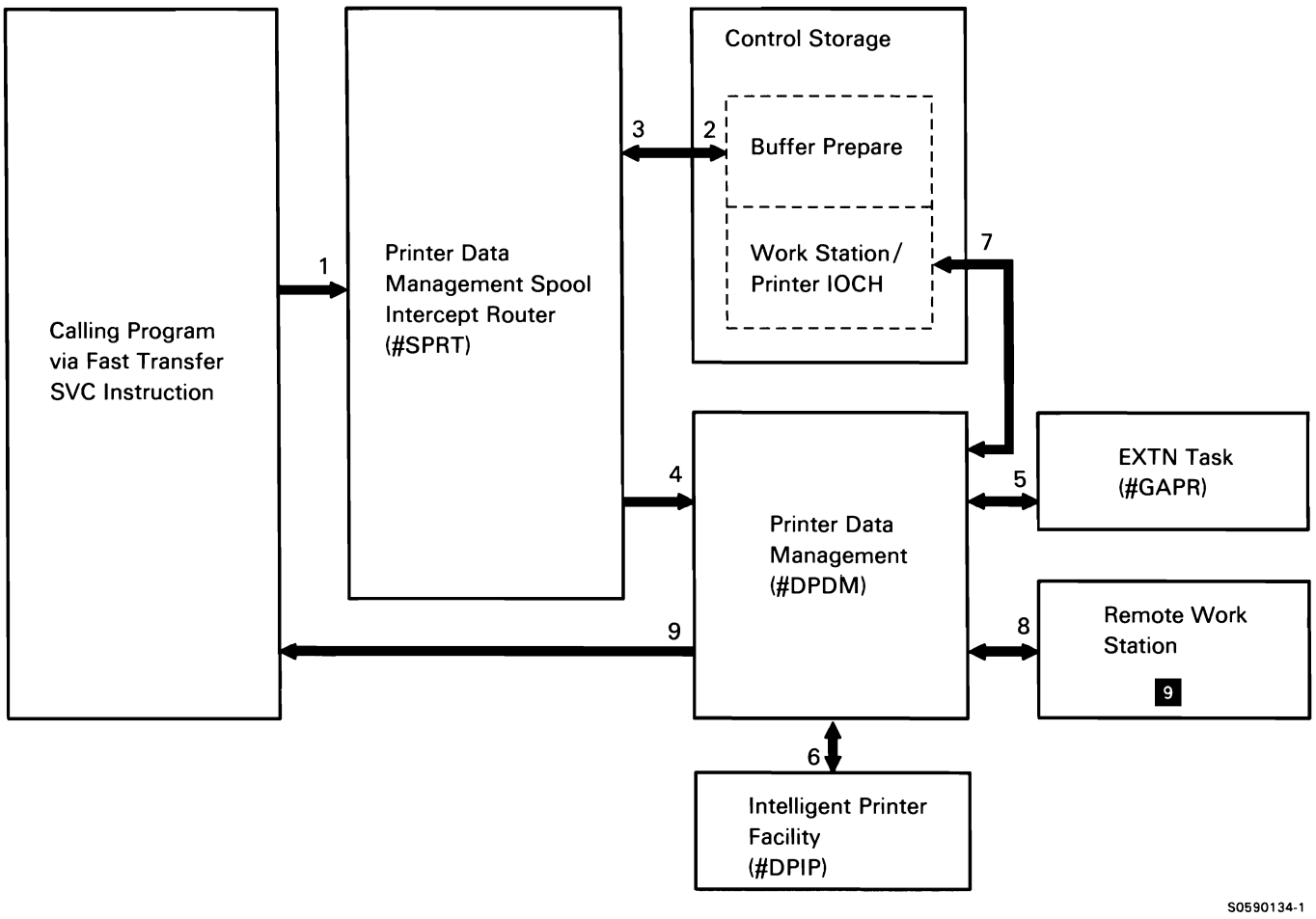


Chart 4.3.1 Nonspool Printer Data Management Control Flow

The following printer data management processes *for a spooled request, other than transparent-mode records and get printer status requests,* are shown in Chart 4.3.2:

1 Do the following:

- Set normal completion code in DTF.
- Move skip/space information from DTF to IOB.
- Move logical buffer to physical buffer.

2 Prepare print buffer and check for EXTN characters.

3 Move physical buffer to spool intercept buffer.

Note: If possible, the spool intercept buffer is appended to the user program; if not, it is built in SQS.

If intercept buffer is full, call printer data management via the fast exit SVC instruction; otherwise, return to caller via the fast exit SVC instruction.

4 Pass control to spool intercept to write buffer to disk.

5 Perform required disk operations.

6 Return DTF with completion code.

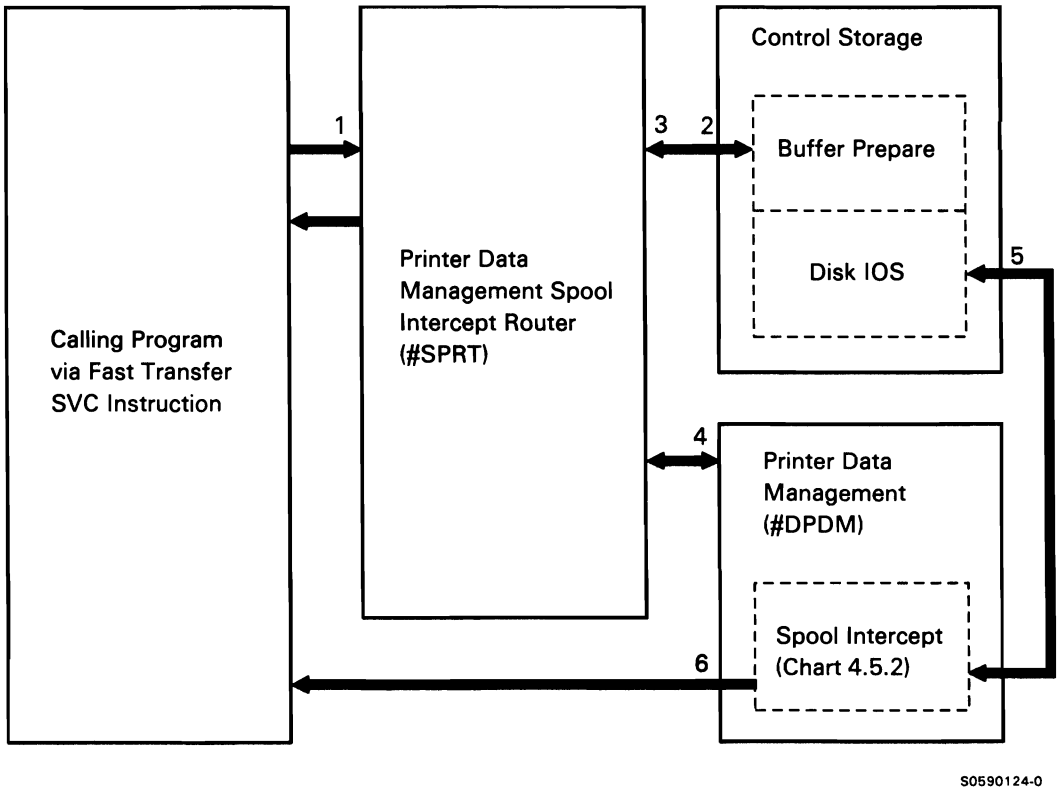


Chart 4.3.2 Spool Printer Data Management Control Flow

INTELLIGENT PRINTER FACILITY

The intelligent printer facility (IPF) is a data stream transformation routine and resource management routine for printers that support the intelligent printer data stream (IPDS).

IPF accepts SNA character string (SCS) commands, document control architecture-level 2 (DCA-L2) commands, or IPDS commands as input. The output from IPF is always on IPDS data stream.

Functions that normally interface with printers such as printer data management, syslist, and the spool writer will call IPF if the printer is an IPDS device. IPF will perform the data stream transformation and interface with the printer. The input to IPF is a printer IOB pointed to by XR1.

IPF Region Map:

0800	Nucleus
1000	IPF (#DPIP)
3800	Input buffers and work areas
3C00	Output buffer

The following IPF processes are shown in Chart 4.3.3:

1 Copy the input data to the input buffer.

2 Translate input data to IPDS and place it in the outside buffer.

3 If requested, retrieve graphics data from the graphics object file.

4 If this is the end of a page, send the print data:

- If the printer is locally attached, print data on the appropriate local printer.
- If the printer is remotely attached, send the print data to the appropriate remote printer.

5 Return.

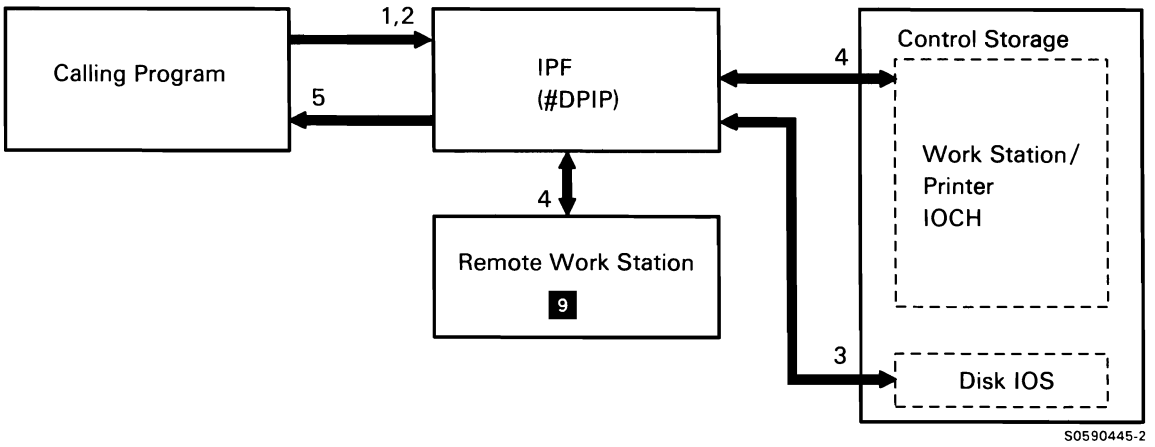


Chart 4.3.3 Intelligent Printer Facility Processes Control Flow

WORK STATION DATA MANAGEMENT

Work station data management (WSDM) runs as a subroutine under the user task and controls all I/O requests to display stations, including the routing of requests for SSP-ICF. *User program* requests to WSDM are passed via the fast transfer SVC instruction with XR2 pointing to the user's DTF; *system (privileged) program* requests are passed with XR2 pointing to a 48-byte work station parameter list (WSPL) in real storage. The WSDM router first checks to ensure that the user's DTF is opened; if it is not opened, WSDM fast exits back to the user with a completion code indicating the DTF is not opened. If the DTF is opened, the WSDM router fast exits to the appropriate module to handle the request.

WSDM converts any DTF request received to a work station parameter list (WSPL), assigns space in SQS, places the WSPL in the assigned SQS, then points XR2 at the WSPL.

For formatted output requests, WSDM merges the user's data with the SFGR-created screen format and writes the data to the display screen. For input requests, WSDM takes data entered at the display station and places it in the user's buffer. The WSDM router, #DWDM, resides in the main storage nucleus and calls translated routines to perform most detail operations.

Many work station data management processes are dependent on the work station being in certain states at the time the processes are performed. The following definitions explain these states.

ACQUIRE/ATTACH/OWN: Each work station is always *attached* to, or *owned* by, a system or user task. This attachment is indicated by a field in the work station's TUB. All work stations are owned by the command processor until they are *acquired* by a user task or a system task.

INVITE: If the system is to receive input from the display station keyboard, it must issue an *invite* to the display station. When a display station is invited, the keyboard is unlocked and the invite bit in the display station TUB is set on. Control storage work station IOCH posts work station data management when input is received from the keyboard.

The invite can be combined with a put operation. During a put with invite operation, the system puts up a display then allows for an operator response; during a stop invite then put operation, the TUB invite bit is set off (to prevent control storage work station IOCH from posting WSDM) then a put is performed.

RELEASE: When a system or user task no longer needs an attached work station, the task *releases* the work station to the command processor. On release, the ownership of the work station, as indicated by a field in the work station's TUB, is changed.

This page is intentionally left blank.

Work Station Data Management Region Map

The following table shows the locations of key data areas involved in the work station data management subfunction:

Dump Address (in Hex)	Contents
1000	Mainline module.
4800	Format index entries. or Mapped WSQS which includes format object S, H, and D specifications and the data stream.
A000	User's record area address.
C000	When display formats have been configured to reside in main storage, WSDM will map the format object in SWS at X'C000 and copy the format object to WSQS.

The following work station data management overview processes are shown in Chart 4.4.0:

1 Determine if input is for a work station or SSP-ICF:

- If first character of SYMID is numeric, route SSP-ICF request to SSP-ICF data management.
- If first character of SYMID is alphabetic, route request to work station data management put or get mainline.
- Build work station parameter list (WSPL) from information in DTF.

2 Perform diagnostic checks on put requests, perform initial request processing, and route for additional required processing.

3 Perform diagnostic checks on get requests and route for required processing.

4 If a severe error is detected, terminate.

5 If a work station error is detected, process error (command rejects or permanent errors) and set up return code.

6 Return work station parameter list to DTF; if input operation, return data, then return to caller.

7 If DTF was not opened, DTF completion code is only data changed; return DTF completion code to caller.

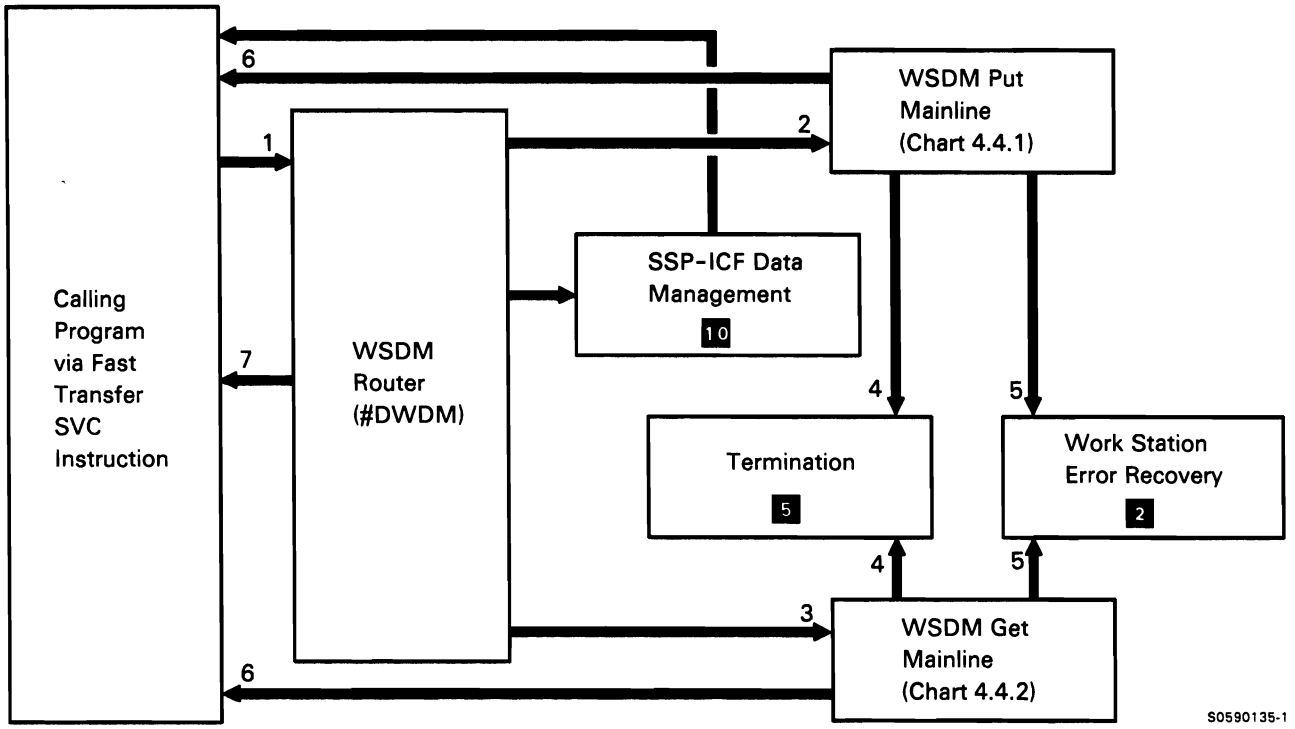


Chart 4.4.0 Work Station Data Management Overview Control Flow

Work Station Data Management Put Processing

A put operation sends a text stream to the work station display. If it is a formatted put, a screen format is read from disk or from SWS into the buffer area, and the task's variable data is inserted in the appropriate places. Work station data management put processing proceeds in the following general sequence:

- Stop outstanding invites.
- If formatted put, locate format via format index and merge the user's logical buffer with format.
- Set up put or put with invite op code in TUB, point XR1 at TUB, and issue request to work station IOCH.
- Return to caller.

If fields in the format were specifically defined as such with indicators, the application program can modify fields (display attributes) in the format using the put override operation.

The following work station data management put processes are shown in Chart 4.4.1:

1 Perform initial processing for put operations:
   - Set up for formatted or unformatted put operation.
   - Stop outstanding invites.
   - Route (within #WDDA) for required detail put processing.

2 Set up for the following, as required:
   - Logical I/O save/restore.
   - Save/restore RAM contents list (EXTN task only).
   - Write error output.
   - Roll output operation.
   - Clear output operation.
   - Erase operation.
   - Reset keyboard.
   - Print operation to assigned work station printer (in conjunction with a put operation).
   - Issue halt by calling #WDDQ.

3 Do the following, as required:
   - Issue halts.
   - Retrieve format MICs. (If EXTN characters, translate them to work station controller storage addresses.)

4 If requested, process get request in conjunction with put operation.

5 If pass-through display station request, process it.

6 If distributed host command facility request, process it.

7 If remote display station request, process it.

8 Write to local display station as required.

9 Print screen to specified printer:
   - Determine printer TUB to use.
   - Allocate and open DTF for printer.
   - Read screen into buffer.
   - Scan for EXTN characters and translate them to work station controller storage addresses.
   - Print screen to specified printer.
   - Return to WSDM.

10 Process H specifications defined by format, or update help list for clear screen.

11 Scan for EXTN characters and translate them to work station controller storage addresses.

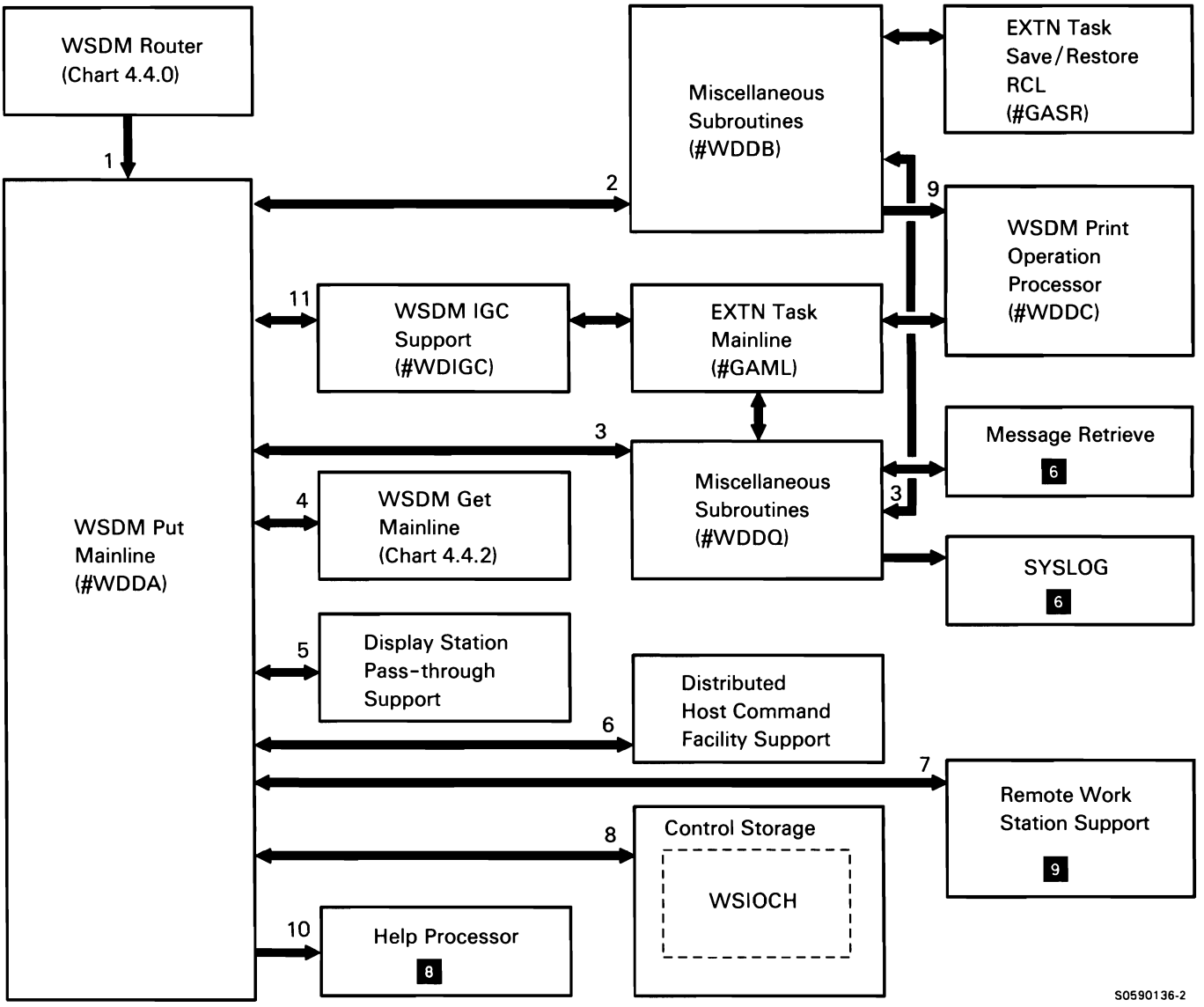


Chart 4.4.1 Work Station Data Management Put Control Flow

Work Station Data Management Get Processing

Work station data management get processing is normally started when the operator presses the Enter key or a valid command key in response to an invite issued by #WDDA. When an MRT program expects input from *any* of its attached display stations, it issues its equivalent of a get, the accept operation. Work station data management get processing proceeds in the following general sequence:

- If not already set, set the invite bit on.
- Set up READ op code in TUB, point XR1 at TUB, and issue request to work station IOCH.
- Move data that was placed in physical buffer by WSIOCH to user's logical buffer.
- Return to caller.

The following work station data management get processes are shown in Chart 4.4.2:

1 Perform the following processing:

- If requested, perform release operation.
- Route (normally within #WDDG) for required get processing.

Note: For an input operation, #WDDG waits for the operator to press a valid key (Enter key or an enabled command or function key). If an invalid key is pressed, #WDDG continues to wait for a valid key, with no return to the calling program.

2 Perform the following, as required:

- Logical I/O save/restore.
- Save/restore RAM contents list (EXTN task only).
- Reset keyboard.

3 Perform the following, as required:

- Acquire work station.
- Get work station attributes.
- Extended get work station attributes.
- Issue halts.
- Stop outstanding invites.

4 Process get request in conjunction with put operation.

5 Perform the following, as required:

- Process low-level request for the Help key.
- Process Print key requests by calling #WDDK.
- Perform function key masking.
- Issue message for Print key requests.

6 If pass-through display station request, process it.

7 If distributed host command facility request, process it.

8 If read request is for remote display station, process it.

9 Process read request for local display station.

10 Process Print key requests by doing the following:

- Determine TUB requested.
- Determine required formatting.
- Read screen.
- Write data to TWA.
- Attach the Print key task (#WDDP).

11 Print the screen image.

- Read data from TWA.
- Scan user record for the presence of ideographic characters (work station controller storage addresses) and translate them to EXTN characters.
- Allocate printer and open DTF.
- Print data.

12 If Help key is pressed and help areas are defined, do the following:

- If cursor is within help area, transfer TUB to help task.
- Return to #WDDH.

13 Display appropriate help formats for application help request.

14 Write to remote display as needed on write error.

15 Write to local display as needed on write error.

16 Scan user record for the presence of ideographic characters (work station controller storage addresses) and translate them to EXTN characters.

17 Browse through an online document.

18 Perform cleanup for help task abnormal termination.

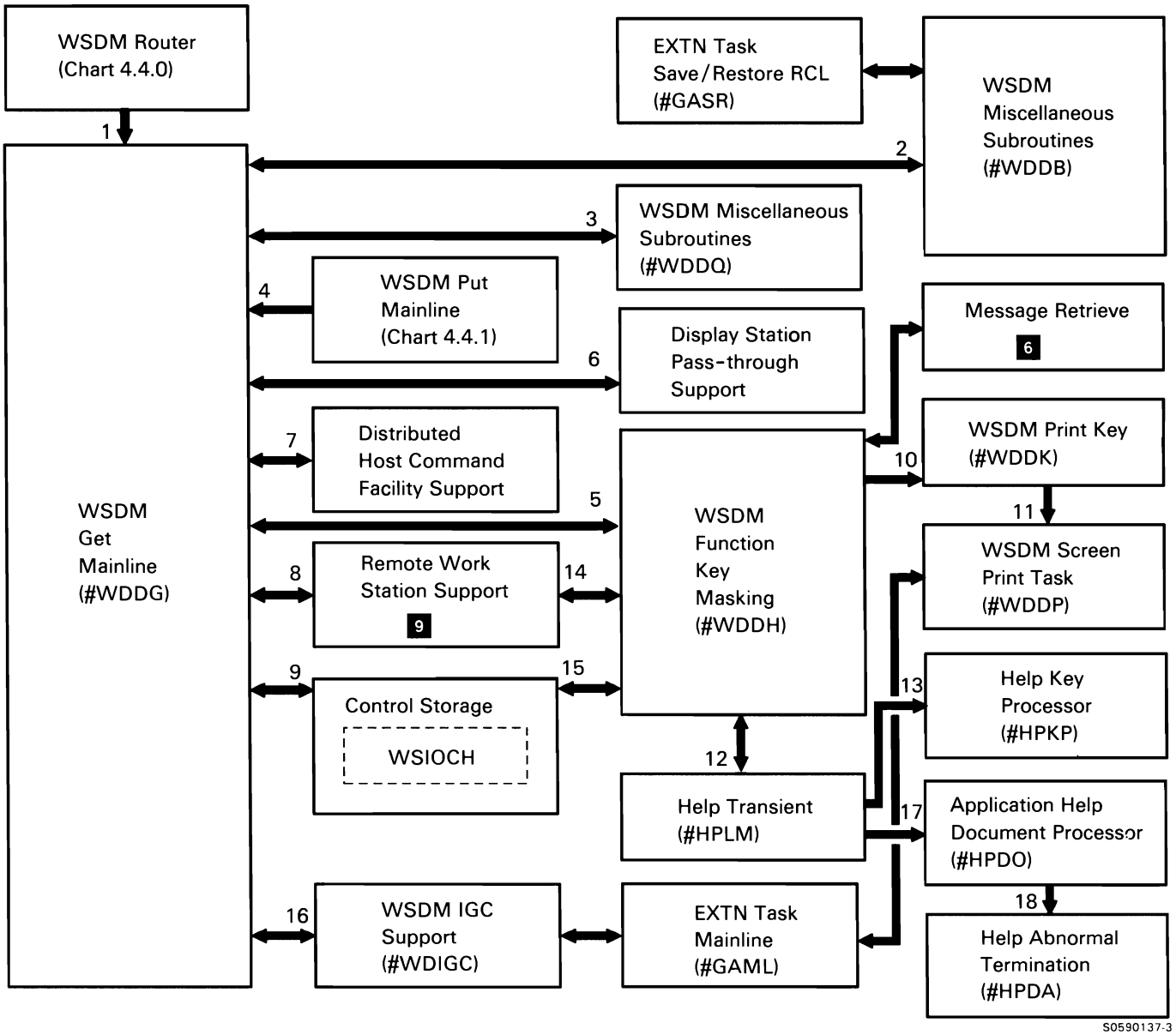


Chart 4.4.2 Work Station Data Management Get Control Flow

PRINT SPOOLING SUBFUNCTION

The print spooling feature intercepts printer requests and places them on disk, creating a spool file. When requested, the spool writer retrieves records from the spool file and sends them to the printer.

Chart 4.5.0 shows the overview control flow for print spooling:

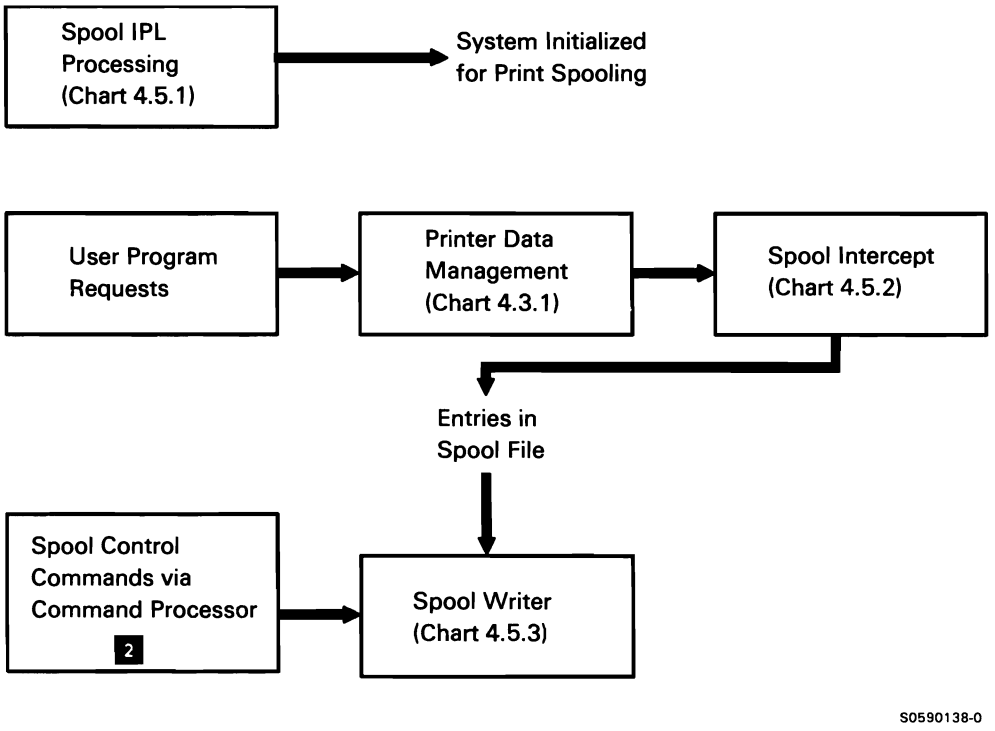


Chart 4.5.0 Print Spooling Overview Control Flow

Spool IPL Initialization

Spool IPL initialization processes IPL overrides for spooling. The overrides provide for deactivating print spooling, deleting spool files, formatting spool files, and attaching the spool writers if autowriter is selected. Spool IPL initialization runs as a command processor subtask.

The following spool IPL initialization processes are shown in Chart 4.5.1:

1 Activate or deactivate print spooling by routing for the following:

2 If print spooling is to be activated, build the writer descriptor blocks (WDBs).

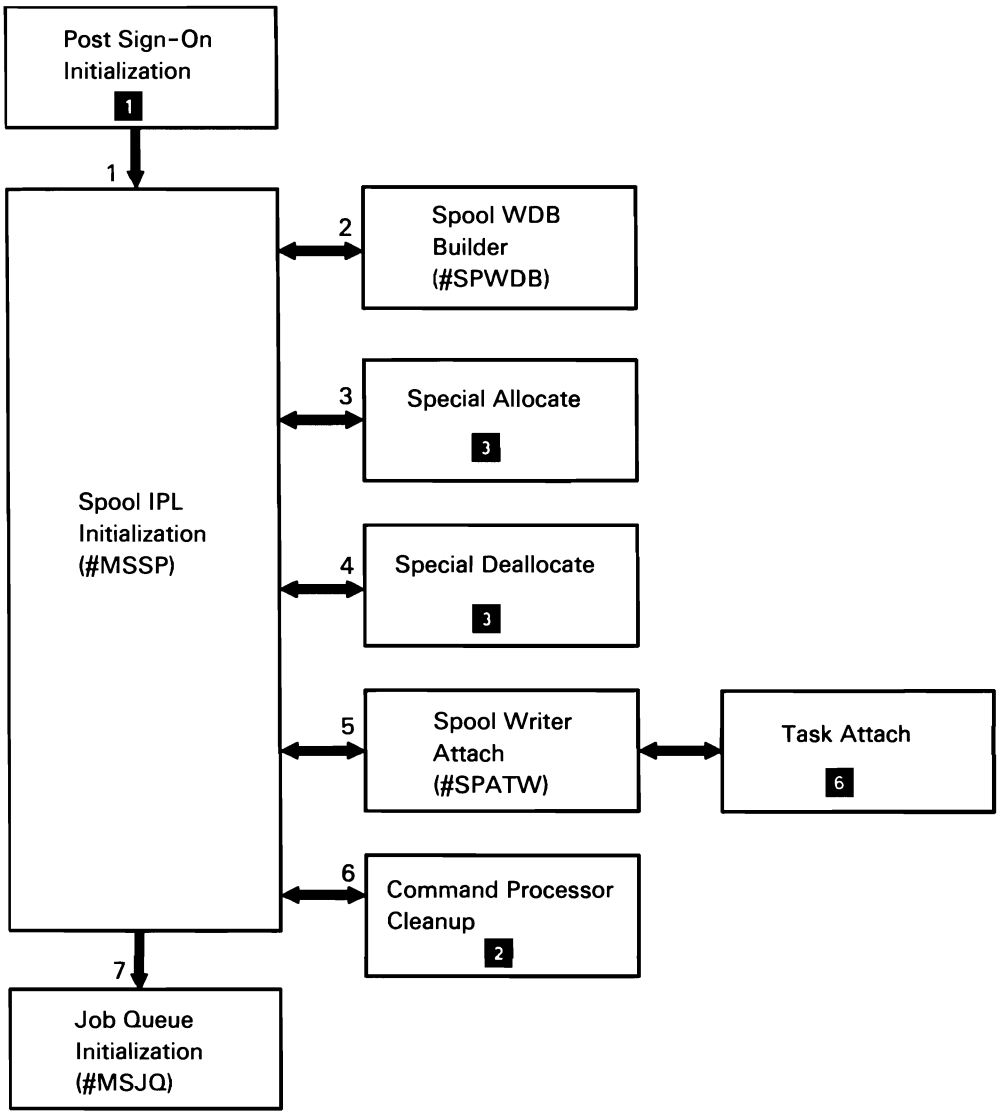
3 If first extent of spool file (##SPOOL1) must be allocated, allocate new disk file.

4 If any spool file extents must be deallocated, deallocate the disk files.

5 If autowriter was specified, start the spool writer; if necessary, attach the spool writer task.

6 Issue any required messages.

7 Exit to initialize job queue.



S0590139-0

Chart 4.5.1 Spool IPL Initialization Control Flow

Spool Intercept

Spool intercept saves data in the spool file when it is called by the printer data management spool intercept router (#SPRT). The call from the router comes via printer data management (#DPDM), to which spool intercept is linked. Spool intercept processes transparent mode records (records prepared in detail by a low-level-language user program) or nontransparent mode records (normal).

Printer data management (#DPDM) may preprocess records passed to spool intercept (#SPINT):

- If XR2 points to a DTF and it is not a transparent mode or a get printer status request, preprocessing was done by #SPRT. Printer data management branches directly to spool intercept.
- If XR2 contains hex 000000, no DTF is passed and no printer data management preprocessing is required. Printer data management branches directly to spool intercept.
- If neither of the above conditions is true, printer data management preprocesses the records before branching to spool intercept.

The following spool intercept processes are shown in Chart 4.5.2:

1 Save the spooled records in the spool intercept buffer; write the full spool intercept buffer to the spool file.

2 When necessary, allocate another spool file segment when the current segment is full.

3 Allocate another spool file extent when all segments of the current extents are full.

4 Allocate spool file space when a new print file is opened.

5 If DEFER-NO print data, attach the spool writer to start printing the data.

6 If the spool writer is started, create the spool writer task.

7 Allocate the first spool file segment for the new print file.

8 Issue any required messages.

9 Route message to appropriate display station.

10 If option 2 was selected in response to a message, close the spool file entry.

11 If an error was detected, close the spool file entry.

12 If DTF was passed to spool intercept, set completion code.

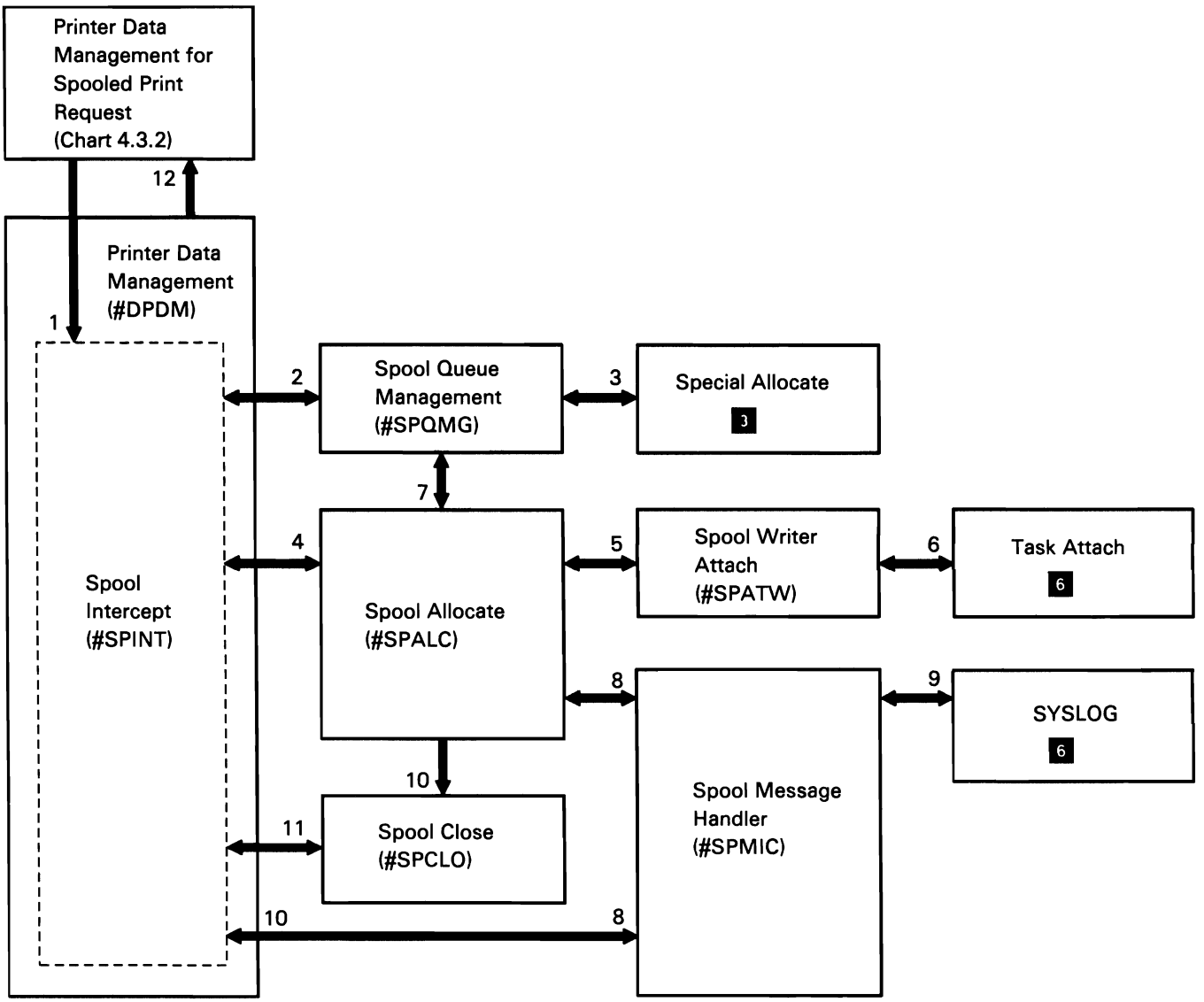


Chart 4.5.2 Spool Intercept Control Flow

Spool Writer

The spool writer reads print data from the spool file and routes it to a printer. The spool writer is controlled by the spool control commands. When active, there is a separate spool writer task for each printer configured on the system. Each spool writer can be started and stopped by spool commands or can be run under the autowriter (automatic start). When there are no more entries on the spool file for a given writer, the writer goes to end of job.

Spool Writer Region Map

The following table shows the locations of key data areas involved in the spool writer subfunction:

Dump Address (in Hex)	Contents
0800	Nucleus
1000	Spool writer (#SPWRT)
2800	TWS miscellaneous fields and save areas
2C00	TWS spool print file control sector save area
2D00	Print buffer
2E00	Print buffer overflow area
2F00	Disk buffer for spool file blocks
3900	Unused storage

The following spool writer processes are shown in Chart 4.5.3:

1 Create the spool writer tasks.

2 Find and update the print file being used by the spool writer.

3 If required, allocate the appropriate printer for the spool writer being activated.

4 Initialize the printer for the current print file.

5 Read a block of data to TWS buffer.

6 If ideographic characters are to be printed on a nonideographic printer, set the ideographic characters to blank.

7 If ideographic characters require extended character processing before being printed, process them.

8 Send the print data:

- If the printer is an IPDS printer, call the intelligent printer facility (IPF).
- If the printer is locally attached, print data on the appropriate local printer.
- If the printer is remotely attached, send the print data to the appropriate remote printer.

9 If required, put a time stamp in the history file.

10 Issue any required messages.

11 Route the message to the appropriate display station.

12 If required, purge any outstanding SYSLOG messages.

13 If required, purge any outstanding printer error messages.

14 When all data is printed, terminate task.

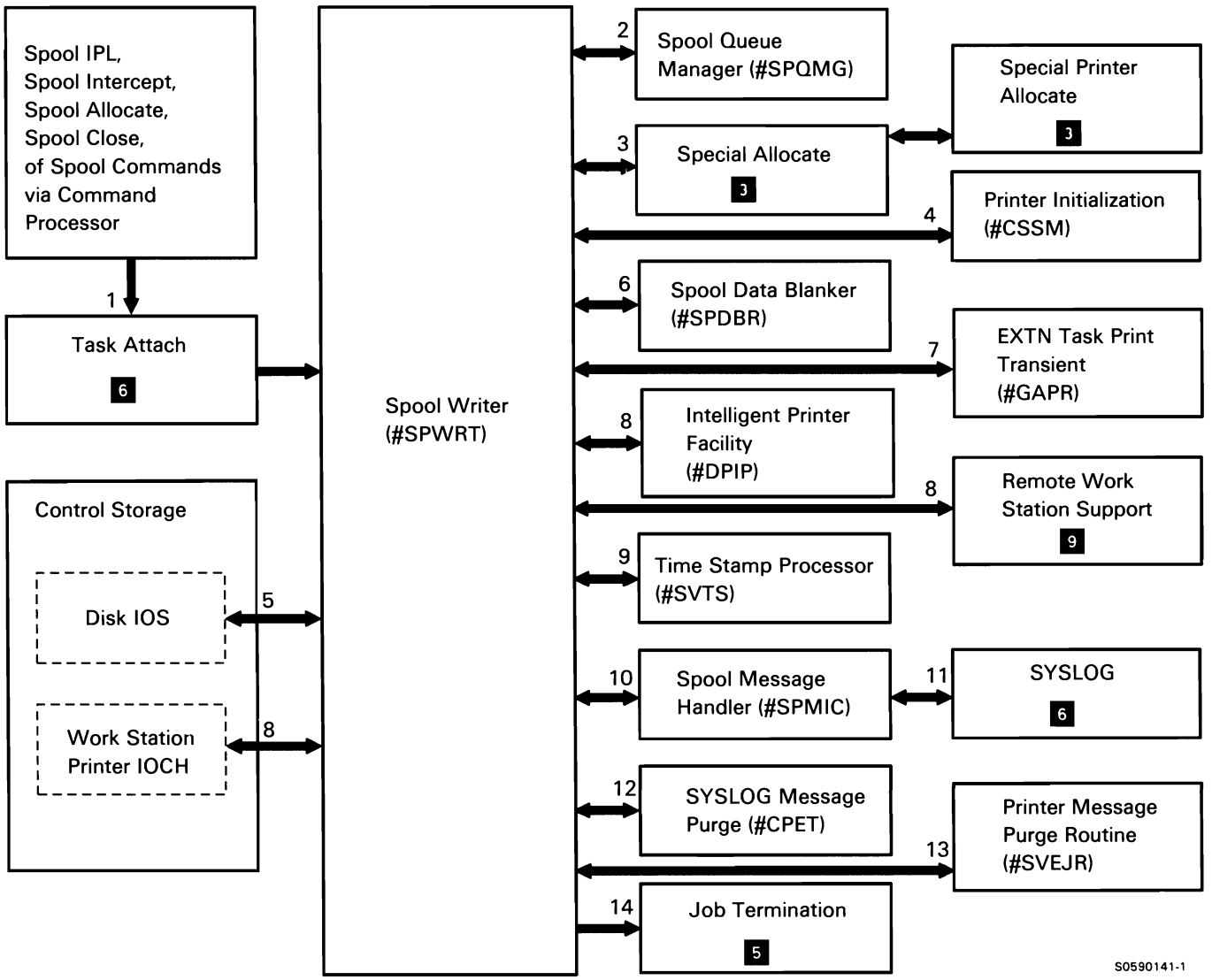


Chart 4.5.3 Spool Writer Control Flow

This page is intentionally left blank.

SECTOR DATA MANAGEMENT TO DISK SUBFUNCTION

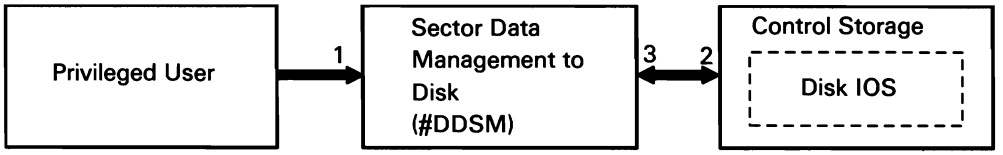
Sector data management provides privileged users with the capability of moving considerable amounts of data to or from disk. It uses a single input/output buffer that is filled by a single read operation, or is written to disk with a single write operation. Sector data management is loaded by disk open and is user-invoked by passing control to the address at which sector data management was loaded.

The following sector data management processes are shown in Chart 4.6:

1 Prepare to read or write disk data.

2 Process read or write operations for an entire block of data.

3 Set DTF completion code .



S0590142-0

Chart 4.6 Sector Data Management to Disk Control Flow

KEYSORT SUBFUNCTION

Keysort arranges indexed disk file index entries in ascending order based on the value of the index entries. Keysort is a logical transient invoked via the transfer SVC, with XR2 pointing to a 16-byte keysort parameter list in system queue space (SQS). The parameter list points to the file format 1 which describes the index to be sorted. All direct users of keysort are SSP privileged programs, but end users can evoke keysort through the keysort utility (described in *SSP Utilities* [redacted]).

The following keysort processes are shown in Chart 4.7:

1 Perform presort processing:

- Analyze parameter list and format 1 to determine level of keysort required.
- Initialize keysort common area.
- Design sort or merge, based on internal sorting area, buffer allocation, and disk work space.
- Allocate disk work space, if needed and if available.
- Return to keysort control.

2 Perform sort with work file:

- Perform sort phase 1: Build internal sort strings.
- Perform sort phase 2: Merge sort strings generated by phase 1 into larger and fewer strings.
- Perform sort phase 3: Final merge.

3 Perform merge only.

4 Perform sort without a work file.

5 Do the following post sort processing:

- Free resources.
- Set appropriate return information in parameter list.
- Adjust format-1 values according to keysort performed.
- Reinitialize data management control blocks as required.
- Return to keysort control.

6 If an error is detected in presort processing, issue MIC 0390 to terminate the task.

7 Report duplicate keys to user.

8 Return to caller.

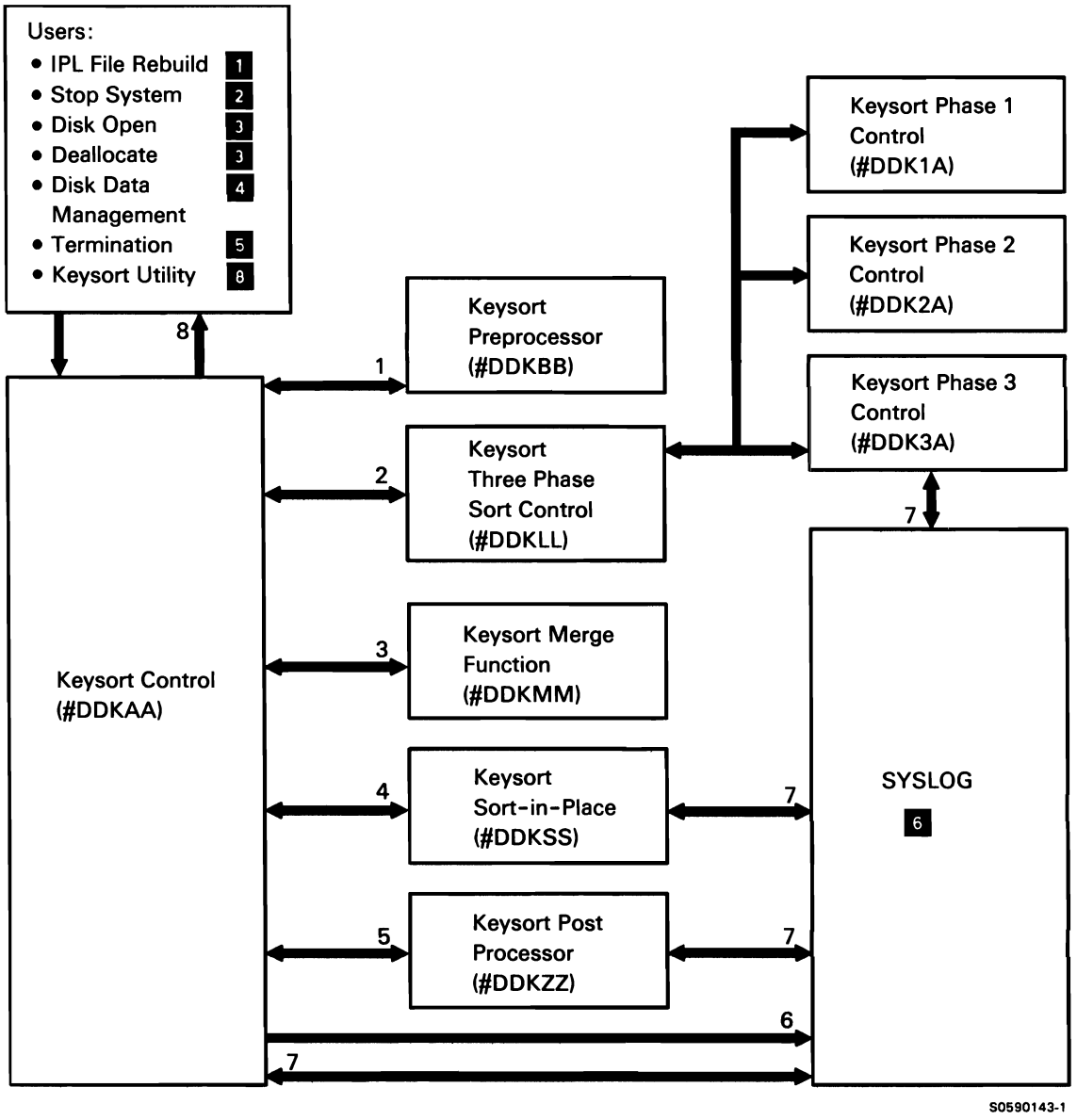


Chart 4.7 Keysort Control Flow

TAPE DATA MANAGEMENT

Tape data management performs record mode tape file processing. Device-to-device data transfers involving tape are performed using *Tape Save/Restore* 6.

Tape data management consists of the following functions:

- Tape find
- Tape get
- Tape put
- Tape add

Tape Find Data Management (#TAFND)

Tape find data management finds file labels using the name passed by the user. When it finds a label, tape data management moves the header label for the specified file from the physical buffer to the user's logical buffer and moves the block count from the trailer label into the tape DTF. When another request for the same label name is received, tape data management finds the header label for the next file of that name and returns its label in the logical buffer and its block length in the tape DTF.

Tape Get Data Management (#TAGET)

Tape get data management uses one of two #TAGET routines, depending on the user region size. If the user region is large enough, tape get data management reads the full block into the physical buffer, then moves the applicable record from the physical buffer to the user's logical buffer each time a get request is issued. If the user region is not large enough, tape get data management reads part of the block into the physical buffer, then moves the applicable record from the physical buffer to the user's logical buffer each time a get request is issued.

When an end-of-volume condition is returned on a read block, #TAGET loads the end-of-volume processing module to get the next tape that is mounted. When #TAGET receives control again, it continues reading blocks of data and processing get requests.

When a file is out of data, #TAGET sends an EOF completion code to the caller. Because the data storage controller has already read the trailer labels, no further processing is required of the caller.

Tape Put Data Management (#TAPUT)

Tape put data management uses one of two #TAPUT routines, depending on the user region size. If the user region is large enough, tape put data management builds a full block in the physical buffer, by moving data from the logical buffer into the physical buffer. When the physical buffer is full, #TAPUT writes the block to tape. If the user region is not large enough, #TAPUT moves records to the physical buffer until the next full record will not fit into the buffer. When a full record will not fit into the physical buffer, part of the record is moved to the physical buffer, then the physical buffer is moved to the data storage controller (hardware) buffer. After the move, the part of the record remaining in the logical buffer is moved to the physical buffer. When the last move from the physical buffer into the hardware buffer fills the hardware buffer to the size of the block length, the hardware buffer is written to tape.

When an end-of-volume condition is returned on a write block, the block and its EOV trailer labels have already been written to tape by the data storage controller. #TAPUT loads the EOV module to get the next tape that is mounted; on return from EOV, #TAPUT returns control to the user.

Tape Add Data Management (#TAADD)

Tape add data management uses one of two #TAADD routines, depending on the user region size. If the user region is large enough, tape add executes user add instructions by moving a record from the logical buffer to the physical buffer. When the physical buffer is full, #TAADD requests the data storage controller to write the block to tape.

If the user region is not large enough, #TAADD moves records to the physical buffer until the next full record will not fit into the buffer. When a full record will not fit into the physical buffer, part of the record is moved to the physical buffer, then the physical buffer is moved to the data storage controller (hardware) buffer. After the move, the part of the record remaining in the logical buffer is moved to the physical buffer. When the last move from the physical buffer into the hardware buffer fills the hardware buffer to the size of the block length, the hardware buffer is written to tape.

When an end-of-volume condition is returned on a write block, the block and its EOV trailer labels have already been written to tape by the data storage controller. #TAADD loads the EOV module to get the next tape that is mounted; on return from EOV, #TAADD returns control to the user.

Tape Read Block Data Management (#TARBK)

Tape read block data management uses one of two #TARBK routines, depending on the user region size. If the user region is large enough to contain it, tape read block data management reads the full block into the physical buffer each time a get request is issued. If the user region is not large enough, tape read block data management reads part of the block into the physical buffer each time a get request is processed.

When an end-of-volume condition is returned on a read block, #TARBK loads the end-of-volume processing module to get the next tape that is mounted. When #TARBK receives control again, it continues reading blocks of data and processing get requests.

When a file is out of data, #TARBK sends an EOF completion code to the caller. Because the data storage controller has already read the trailer labels, no further processing is required of the caller.

Tape Write Block Data Management (#TAWBK)

Tape write block data management uses one of two #TAWBK routines, depending on the user region size. If the user region is large enough, tape write block data management writes a full block from the physical buffer each time a put request is issued. If the user region is not large enough, tape write block data management writes part of the block into the physical buffer each time a put request is processed.

When an end-of-volume condition is returned on a write block, #TAWBK loads the end-of-volume processing module to get the next tape that is mounted; on return from EOV, #TAWBK returns control to its caller.

The following tape data management processes are shown in Chart 4.8:

1 Perform tape data management services:

- Initialize for user request.
- Check for end of data.
- Issue SVCs for tape data storage controller.
- If end-of-volume processing is required, start it; otherwise, set DTF completion code and return to user.

2 Perform end-of-volume processing:

- Unload tape drive that has an end-of-volume condition.
- If AUTO-YES was specified and another drive can be allocated, use other tape drive.
- Verify volume label and header labels; if incorrect, issue error message.

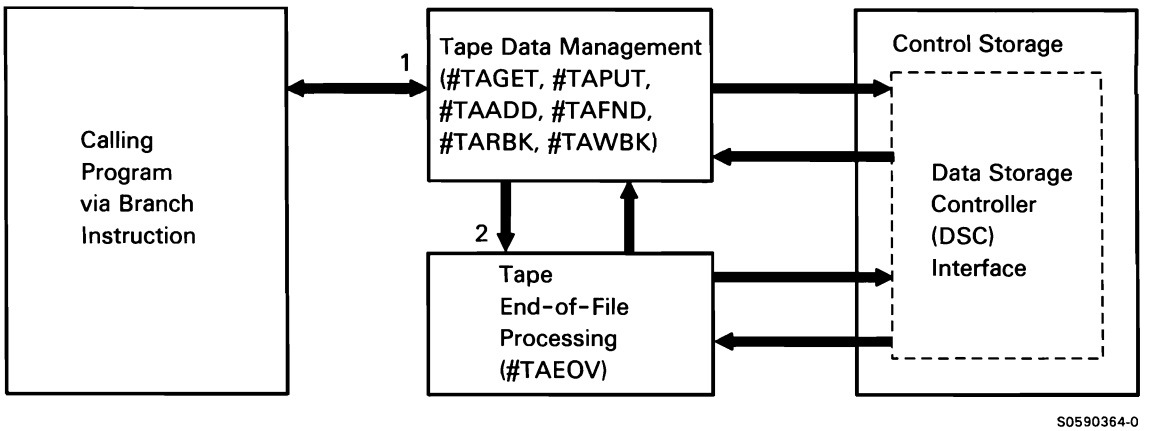


Chart 4.8 Tape Data Management Control Flow

1255 MCR DATA MANAGEMENT

User programs access 1255 MCR data management through one of two IBM-supplied user interface subroutines, SUBR08 or SUBR25. In earlier systems, the two subroutines were packaged separately; both subroutines are provided on System/36 to maintain compatibility among systems. The subroutines build 1255 MCR DTFs which they pass (via XR2 pointer) to the data management modules via transfer SVC instructions. In the control flow chart, two module names are given for each module's descriptive name. The name listed first is the module used with the SUBR08 subroutine; the second name is the module used with the SUBR25 subroutine.

The following 1255 MCR data management processes are shown in Chart 4.9:

1 Perform the following for 1255 MCR requests:

- Map to data area used in common with control storage MCR programs.
- Set up DTF data buffer.
- Determine request type and initialize IOBs, action control elements or error recovery blocks required.
- Route for the following, as required.

2 Perform open processing for 1255 MCR requests:

- Open DTF.
- Load 1255 controller.

3 Process 1255 MCR IOBs.

4 Perform 1255 MCR error recovery:

- Process error IOB pointed to by XR1, and parameter list pointed to by XR2.
- Issue any required messages via message retrieve and SYSLOG 6

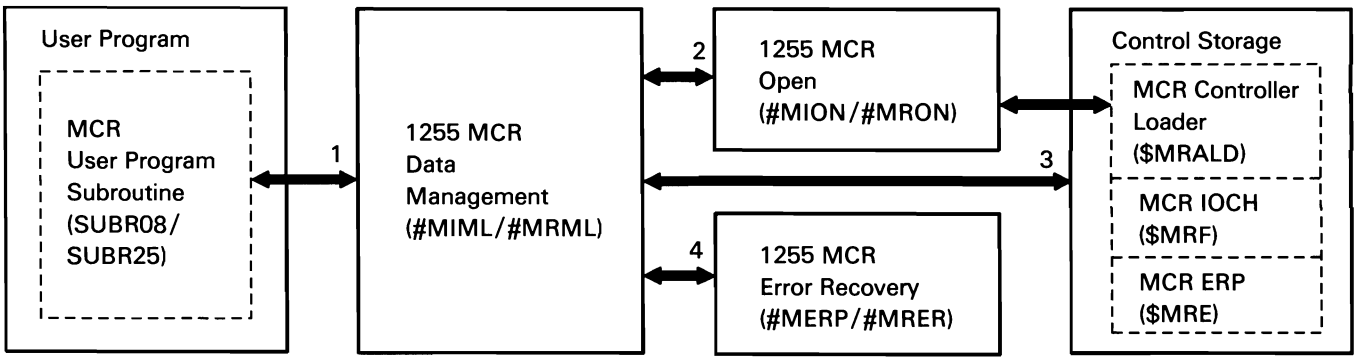


Chart 4.9 1255 MCR Data Management Control Flow

OFFICE/36 SUPPORT SUBFUNCTION

OFFICE/36 support provides data management folder support for the following products:

- Interactive data definition utility (IDDU)
- Query/36
- Personal Services/36 (PS/36)
- DisplayWrite/36 (DW/36)

OFFICE/36 support consists of the following:

- Office systems transforms and profiles processors
- Folder management services (FMS) I/O router and miscellaneous processing

The office systems transforms processor allows the office support utilities to communicate with each other and with FMS by translating between the DW/36 internal data format and the data formats used by other utilities; the office systems profiles processor provides for the maintenance of screen defaults and other user information that can be retrieved for use during an office system session. FMS provides support that allows the utilities and users to create, edit, delete, copy, and list the special libraries, members, and records (*folders*, *documents*, and *data items*) used by the OFFICE/36 program products.

Chart 4.10.0 shows the overview control flow for the OFFICE/36 support subfunction:

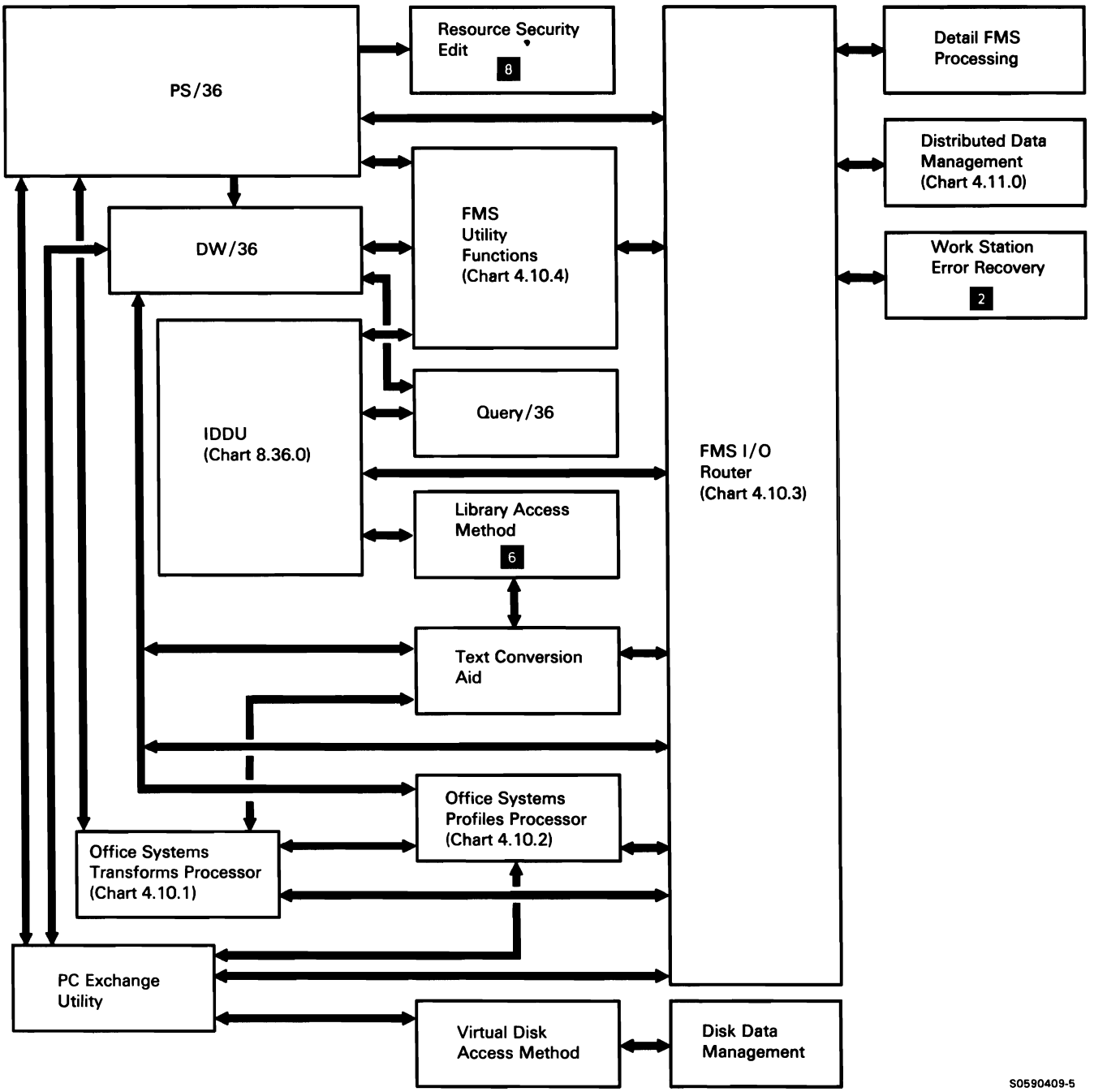


Chart 4.10.0 OFFICE/36 Support Overview Control Flow

Office Systems Transforms Processor

The office systems transforms processor converts an input data stream of one type to an output data stream of another type. The processor allows for the interchange of data among the OFFICE/36 program products and folder management services (FMS), and to and from I/O devices. The office systems transforms processor is called via a transfer SVC instruction and parameter list by the following:

- PS/36
- DW/36
- Text conversion aid
- PC exchange utility

The processor provides only the conversion capability and does not open or close the affected files, documents, or devices; in addition to being responsible for these, the caller must also interlock files or documents to prevent their modification during the transformation. The caller must position the input data stream pointer so the next read/get provides the information required for the conversion.

The transforms processor converts between internal DW/36 data stream format and several other external types of data streams. It can also convert between several external formats, but all conversions of this type require an intermediate conversion to/from the internal DW/36 data stream format. The DW/36 data stream format is compatible with System/36 SSP and the FMS librarian functions.

The following office systems transforms processor processes are shown in Chart 4.10.1:

1 Initialize for transforms processing:
   - Initialize transforms common area (TTCOMMON).
   - Map to parameter list and check it for errors.
   - Build any required FMS members.

2 Use parameter list to determine required routing; route control to appropriate conversion processor.

3 Convert for alternative character set.

4 Convert data in source or procedure to DW/36 internal format.

5 Convert text from System/36 Release 1 TMS format to DW/36 internal format.

6 Convert text from FFT-DCA to DW/36 internal format.

7 Convert text from DW/36 internal format to FFT-DCA.

8 Convert text from RFT-DCA to DW/36 internal format.

9 Convert text from DW/36 internal format to RFT-DCA.

10 Convert text from DW/36 internal format to SCS#1 format.

11 Convert text from DW/36 internal format to SCS#2 format.

12 Convert text from DW/36 internal format to WP/EBCDIC format.

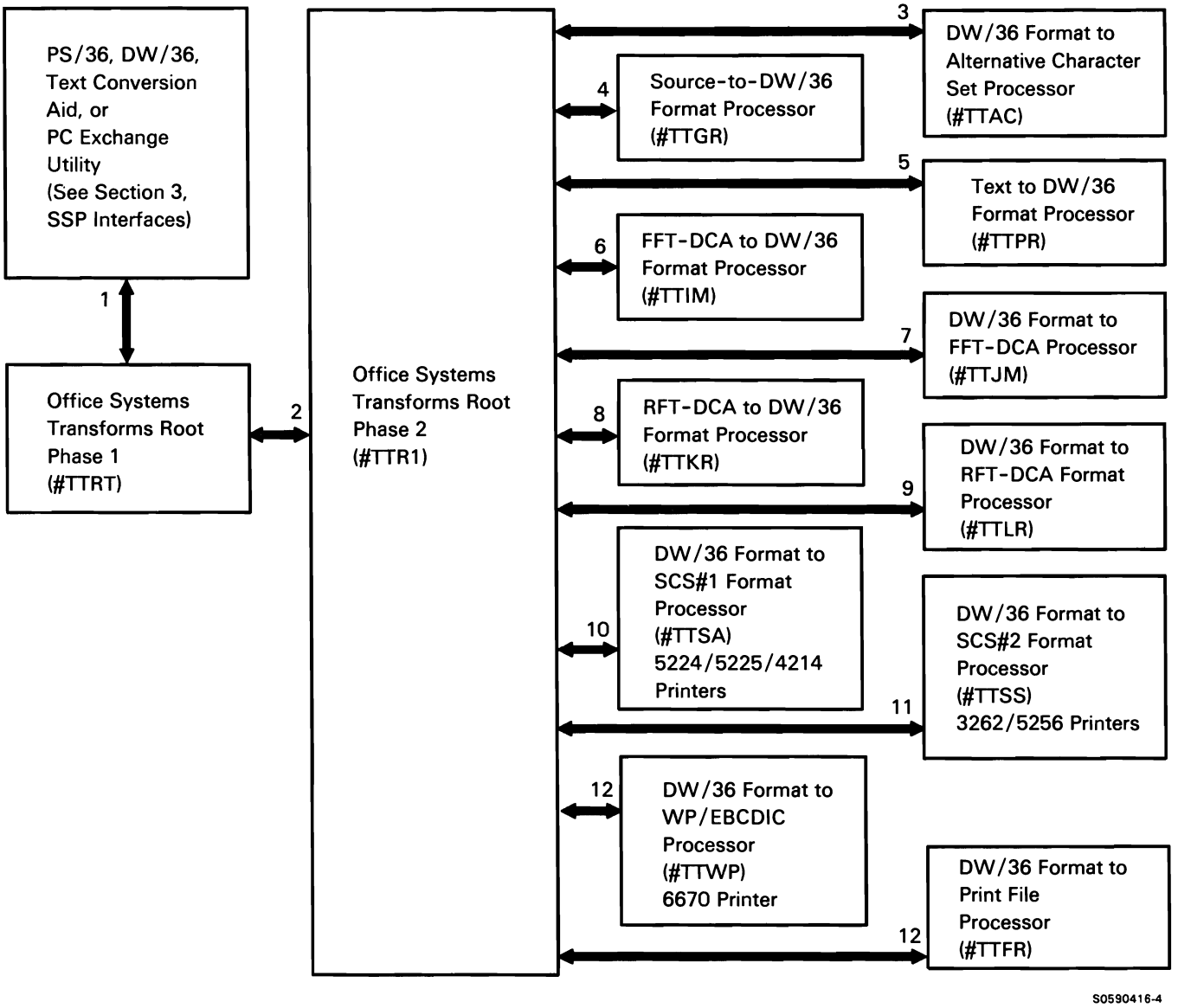


Chart 4.10.1 Office Systems Transforms Processor Control Flow

Office Systems Profiles Processor

The office systems profiles processor provides for the maintenance of profile folders and user profiles that contain screen defaults and other user information. The profile folders and user profiles are invoked at sign-on time to customize the operator's interface to office services.

The office systems profiles processor performs, or routes for, the following processes:

- Create profile folder.
- Create user profile.
- Get user profile.
- Put user profile.
- Delete user profile.
- Archive/retrieve user profile.
- Save/restore profile folder.
- Reorganize/extend profile folder.
- Delete profile folder.

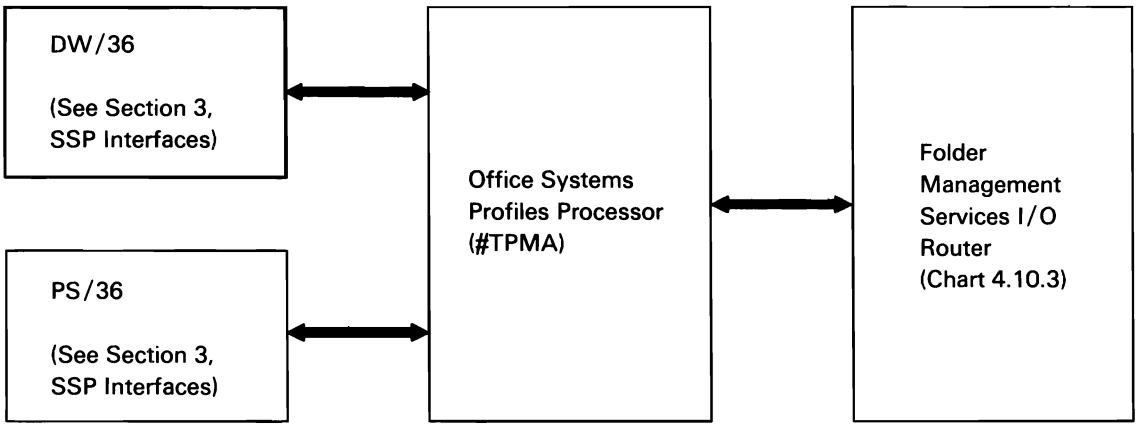


Chart 4.10.2 Office Systems Profiles Processor Control Flow

This page is intentionally left blank.

Folder Management Services (FMS)

Folder management services (FMS) provides the OFFICE/36 program products and associated functions with an interface to the local system document folder for the following purposes:

- Folder and member open and close
- Folder and member data processing support
- Data descriptor processing support
- Text data processing support
- Associate record processing support

Folder management services consists of the following:

- FMS I/O router
- FMS utility functions

FMS can be invoked via an SVC instruction interface (privileged) or through procedure calls. The procedure interface allows users to directly maintain information contained in the document folders. System procedures are executed by the FMS utility, described in *SSP Utilities,* Chart 8.35. The FMS utility uses the privileged (SVC) interface described below.

The SVC interface uses a parameter list to pass data and operation codes to FMS, and to pass return codes and data back to the caller.

Calling programs use the following data areas to interface to FMS processing:

I/O PARAMETER LIST: Each request for a FMS I/O operation must be accompanied by an I/O parameter list. The list includes an operation code and information on the attributes of the data to be used in conjunction with the requested operation.

DATA DESCRIPTOR AREA (DDA): The DDA is a portion of a folder that contains product-defined attributes of the data contained in the folder; the attributes describe the folder data down to the field level. The DDA also includes a summary description of the data for the benefit of operators scrolling folders.

DATA DESCRIPTOR AREA (DDA) LIBRARY HEADER: The DDA library header consists of the library description, the library validation data, and the DDA-type headers for the folder.

DATA TEXT AREA (DTA): The DTA stores the actual member text and any associated data. The text is broken into text record units that serve as page break units, format change units, and/or line units. The text record units are accessed via an allocation map located at the front of the DTA. The DTA also contains associate records (which contain IDDU fields, formats, and file information), DW/36 format descriptions, and other miscellaneous product-defined information.

FMS I/O Router

The folder management services I/O router performs all I/O associated with folders and documents. Callers invoke FMS I/O with a fast transfer SVC instruction and a TMIO parameter list.

Chart 4.10.3 shows the following FMS I/O control flow:

1 Set up parameter list and transfer to detail I/O function requested.

2 Open folder by protecting resources, allocating folder, and writing open folder flag to disk.

  Open member by locking folder and type, setting open member flag in physical directory, and checking member security and security fields in DDA.

3 Close folder:

  - Lock folder and type.
  - If folder was opened for CREATE and closed with UPDATE-NO, delete any member that was created.
  - Write type headers and allocation maps to disk.
  - Deallocate folder.

4 Get and lock DDA.

5 Retrieve information about a member's text data.

6 Get text lines or text units by page and line number.

7 Get associate record.

8 Read associate record into caller's buffer.

9 Put DDA.

10 Put text units to FMS member.

11 Write buffer contents as an associate record.

12 Delete one or more lines in a member.

13 Delete associate record or group of associate records.

14 Lock, list, and unlock one or more DDAs.

15 Lock, search (using search argument), and unlock one or more DDAs.

16 Continue search when #TMSDD successful search list is too long for user to process in one call.

17 Build replacement DDA with space for more data in all (or in specifically requested) areas.

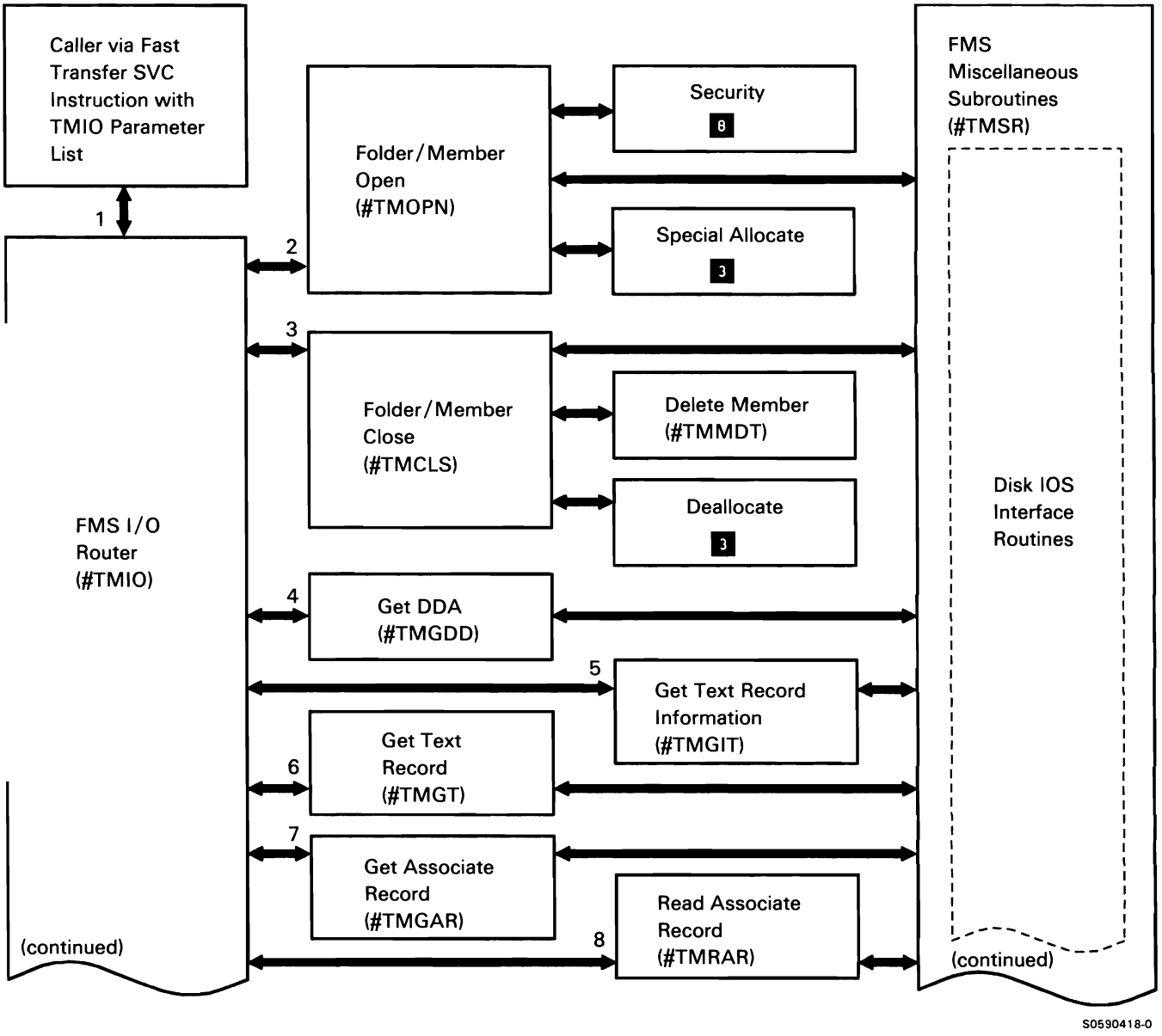


Chart 4.10.3 (Part 1 of 2) FMS I/O Router Control Flow

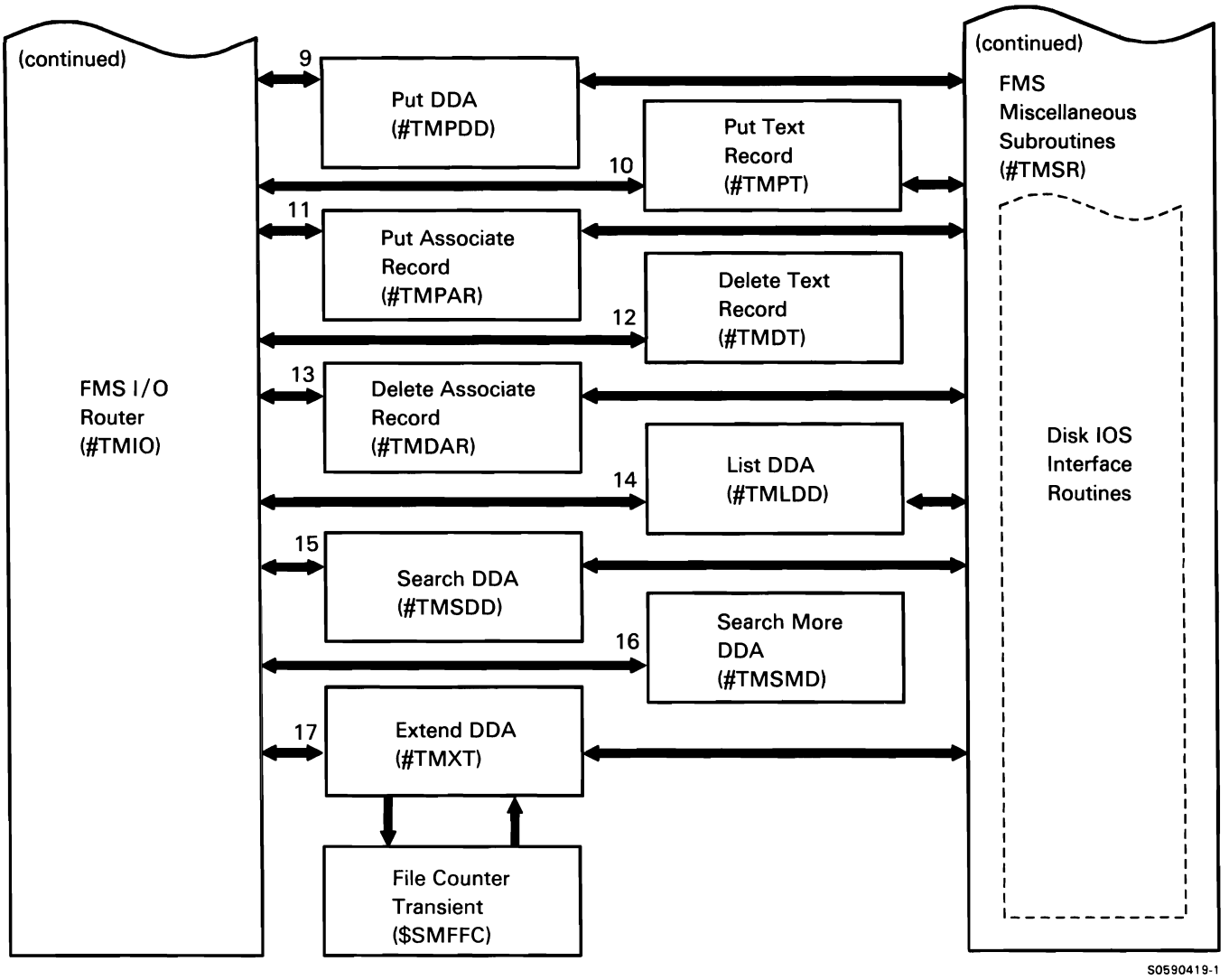


Chart 4.10.3 (Part 2 of 2) FMS I/O Router Control Flow

FMS Utility Functions Processing

Folder management services utility functions processing consists of individual modules that are directly called by the office systems utilities and related components via transfer SVC instructions and parameter lists.

The following FMS utility functions processing is shown in Chart 4.10.4:

1 Delete a member from a folder, delete the text portion (DTA) member of a folder, or delete a member without deleting the physical directory entries.

2 Rename a member within a folder.

3 Copy a member, creating a new member or replacing an existing member.

4 List information about a folder or group of folders.

5 Reorganize folder increasing or decreasing size or removing as much unused space as possible.

6 Save a folder to a diskette or tape file.

7 Restore a saved folder from a diskette or tape file.

8 Archive a member from a folder to a diskette or tape file.

9 Retrieve an archived member from a diskette or tape file.

10 Create a new folder on disk.

11 Create a DTA for a folder and set up allocation map for DTA.

12 List information about an archived member on the SYSLIST device.

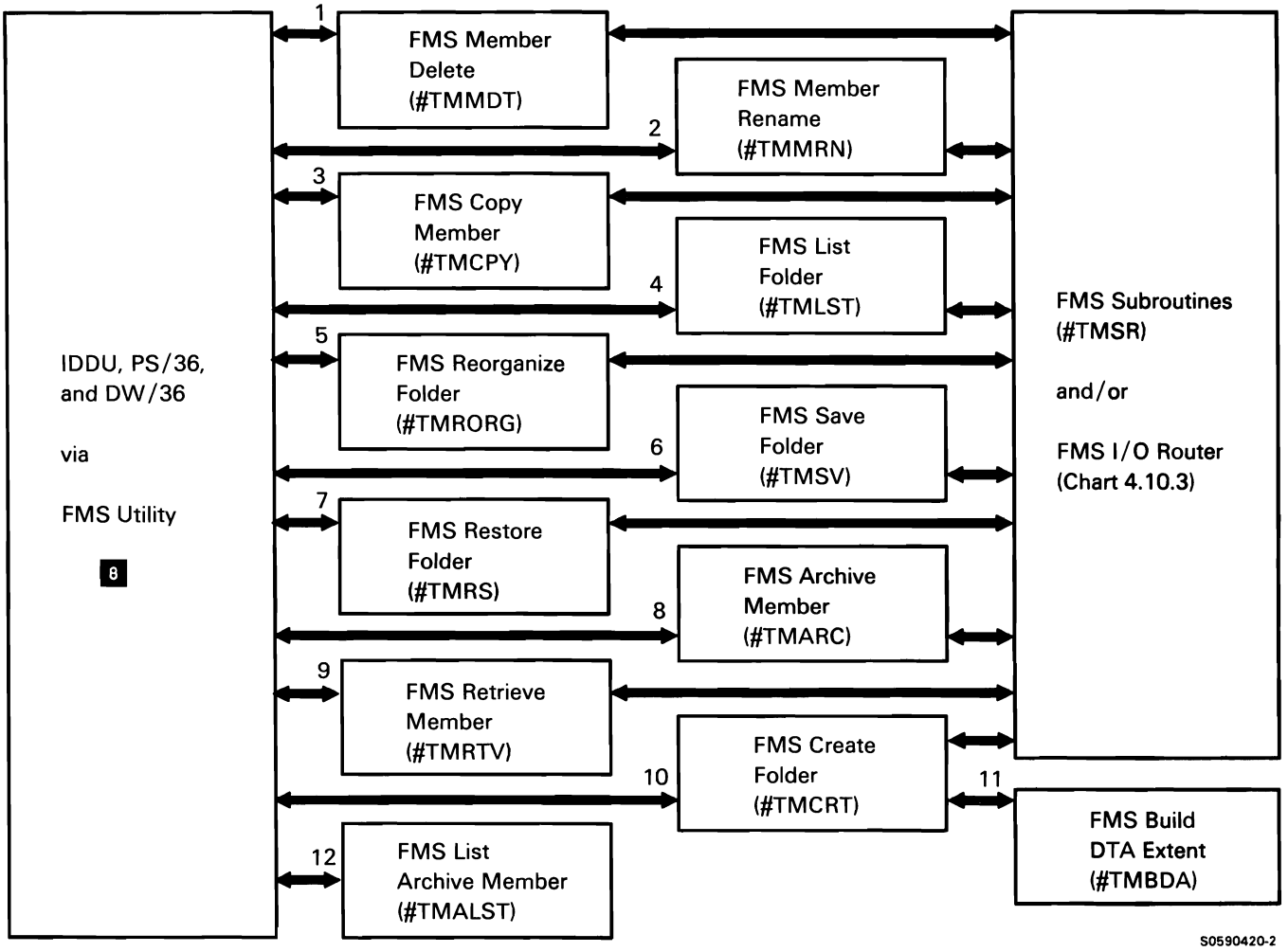


Chart 4.10.4 FMS Utility Functions Processing Control Flow

DISTRIBUTED DATA MANAGEMENT (DDM)

Distributed data management (DDM) allows the user to access remote files as if they were local files. DDM operates only within an APPC subsystem environment using LU6.2 and allows access only to data files.

For the purposes of this description, DDM is broken into two main components: source processing and target processing. Source DDM (SDDM) processing makes the complexity of accessing remote files transparent to the requesting procedure or program; target DDM (TDDM) processing makes file access requests received from other systems appear to SSP as local requests.

DDM uses the system's network resource directory (NRD) file to determine where requested files reside. Each system in a DDM network has its own NRD that contains entries for all remote files to which its local users have access. Each NRD entry contains the file name by which the file is known locally, the symbolic ID of the remote system on which the file resides, and the file name by which the file is identified to the remote system.

When DDM finds a NRD entry for a requested file (file is remote), DDM source processing communicates the file request to the remote system's DDM target processor, via the APPC subsystem. Requests for remote files can cascade through several systems before the ultimate target system processes the file request.

Note: The user can edit, save, restore, delete, and list the NRD using the network resource directory utilities. Refer to *SSP Utilities* 6 for more information on the NRD utilities.

This topic is organized according to the following:

Chart 4.11.0 shows the overview control flow for the distributed data management subfunction:

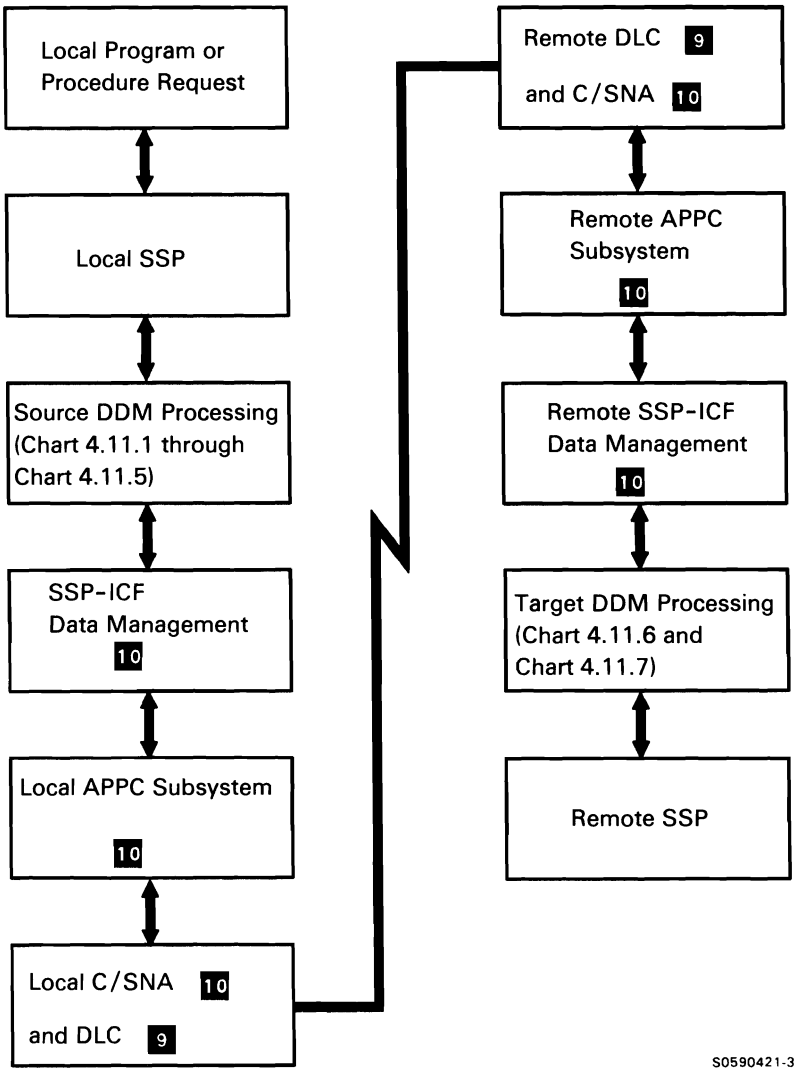


Chart 4.11.0 Distributed Data Management (DDM) Overview Control Flow

Source DDM (SDDM) VTOC Extract

SSP VTOC access 6 calls SDDM VTOC extract via a transfer SVC instruction when it cannot find a VTOC entry for the requested file and has determined that remote file access is allowed. SDDM VTOC extract finds the NRD entry in the network resource directory for the requested file; the NRD entry contains either the symbolic ID of the remote location on which the file resides or the symbolic ID of a remote location that contains an NRD entry for the file. SDDM VTOC extract returns all requested information to the caller of SSP VTOC access via the VTOC access parameter list and the user buffer.

The following SDDM VTOC extract processes are shown in Chart 4.11.1:

1 Process VTOC access parameter list file request:

- Route control to NRD access to find file label specified.
- If file label not found, consider local file and return to the caller of VTOC access.
- If file label found, queue an NRD entry lock (NEL) control block to ensure proper locking of the NRD entry.
- If request is test-for or read format 1:
  - Route control to SDDM communications initialization to attach the target distributed data management (TDDM) at the remote location.
  - If remote system is a Release 3 and Release 4 DDM system, route control to the Release 3 and Release 4 SDDM extract.
  - Send DDM LIST FILE ATTRIBUTES command to TDDM mainline.
- Alter VTOC parameter list with appropriate data and return code, place format 1 in user buffer (if requested), and return to caller of VTOC access.

2 Get network resource directory (NRD) entry for requested file from NRD.FLE file:

- Use control storage disk IOS to search NRD file for local label of remote file.
- If entry not found, set not-met return code and return to caller.
- If existence test specified, set normal return code and return to caller.
- If read function specified, return the remote label, length of remote label, and the remote location to the caller.

3 Process requests from SDDM extract and SDDM communications initialization:

- Acquire an APPC subsystem session.
- Evoke TDDM mainline.
- Process puts and gets between SDDM extract and remote system's TDDM mainline.

4 Attach TDDM at remote location:

- Queue a remote file session (RFS) control block to the JCB.
- Acquire an APPC subsystem session.
- Evoke TDDM mainline.
- Send DDM EXCHANGE SERVER ATTRIBUTES command to TDDM mainline to determine the level of DDM installed at the remote system.

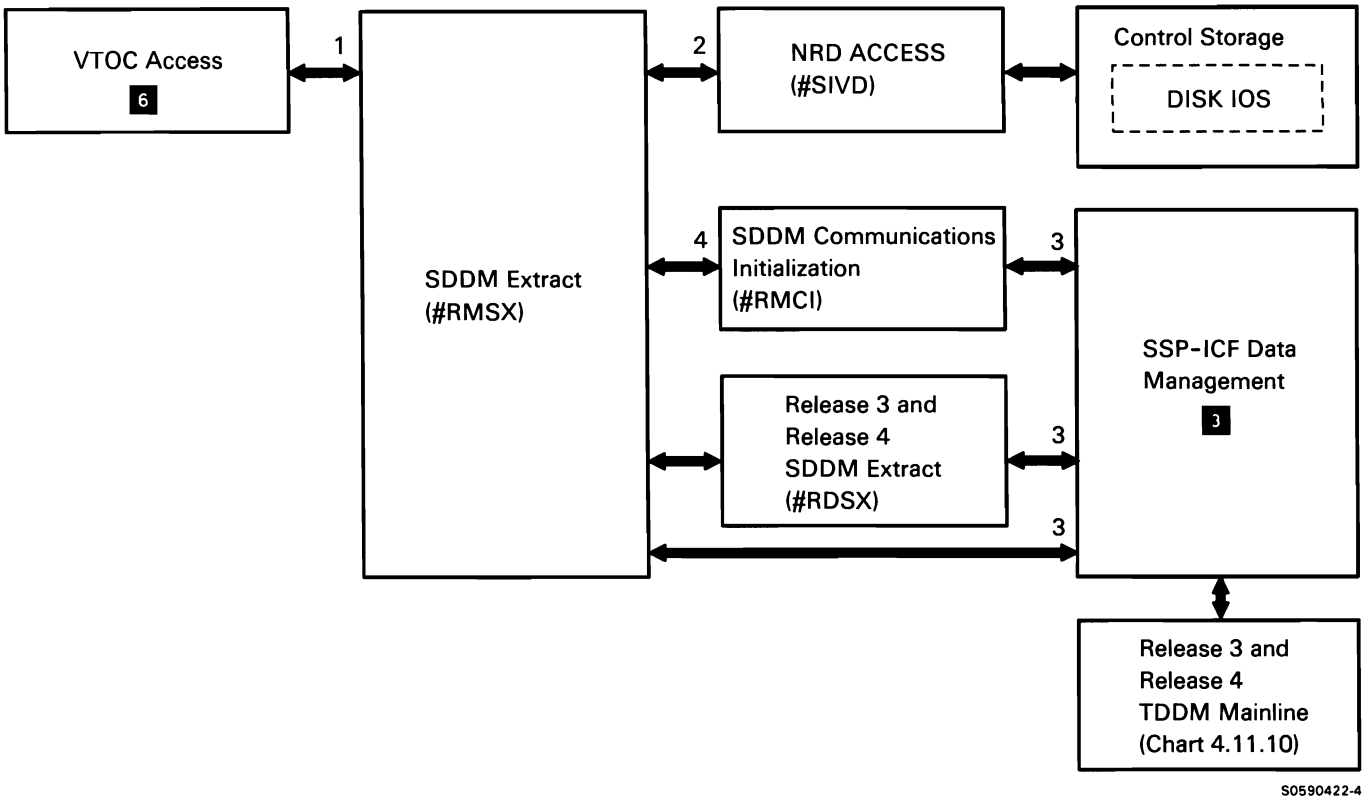


Chart 4.11.1 SDDM VTOC Extract Control Flow

Source DDM (SDDM) RENQ/RDEQ

Source DDM (SDDM) RENQ/RDEQ processes local user requests for the resource enqueueing or dequeueing of remote files. Resource enqueue (RENQ) establishes the level of sharing that is to exist between multiple users of a remote file. Resource dequeue (RDEQ) removes the specified share level for the remote file.

SDDM RENQ/RDEQ is called via an SVC instruction by the initiator 3, special allocate 3, deallocate 3, or termination 5.

The following SDDM RENQ/RDEQ processes are shown in Chart 4.11.2:

1 Process resource enqueue/dequeue request:

- Get remote file label and location by calling NRD access.
- If resource enqueue request:
  - If RETAIN-S file, force DISP-OLD.
  - If RETAIN-T or -P, get share level specified in FSB.
  - If requester is an NRT program and if the APPC subsystem session previously established by SDDM extract is not available, route control to SDDM communications initialization to attach the target distributed data management (TDDM) at the remote location. Otherwise, use APPC subsystem session established by SDDM extract.
  - If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM RENQ/RDEQ.
  - Send DDM LOCK FILE command to TDDM mainline.
  - If reply indicates a file was successfully locked and point FSB to remote file session (RFS), increment use count in RFS.
  - Process DDM reply sequence from TDDM mainline and pass return code to caller.
  - Queue a remote format-1 extension (RF1) to the format 1. For keyed and alternate index files, queue a second RF1 containing key definition information to the first RF1.
  - Place the address of the NRD entry lock (NEL) in the first RF1.
- If resource dequeue request:
  - If RETAIN-S file, force DISP-OLD.
  - If RETAIN-T or -P, get share level specified in FSB.
  - Get RFS for the remote location.
  - Use previously-established APPC subsystem session to send UNLOCK FILE command to TDDM mainline.
  - Process reply from TDDM mainline and pass return code to caller.
  - Decrement use count in RFS whether unlock was successful or unsuccessful; unbind FSB from RFS.
  - Decrement the enqueue count in the RF1.
  - If the enqueue count goes to zero, then dequeue and free any RF1s queued to the F1.

2 Process requests from SDDM RENQ/RDEQ and SDDM communications initialization:

- Acquire an APPC subsystem session.
- Evoke TDDM mainline.
- Process puts and gets between SDDM RENQ/RDEQ and remote system's TDDM mainline.

3 Attach TDDM at remote location:

- Queue an RFS control block to the JCB.
- Acquire an APPC subsystem session.
- Evoke TDDM mainline.
- Send DDM EXCHANGE SERVER ATTRIBUTES command to TDDM mainline to determine the level of DDM installed at the remote system.

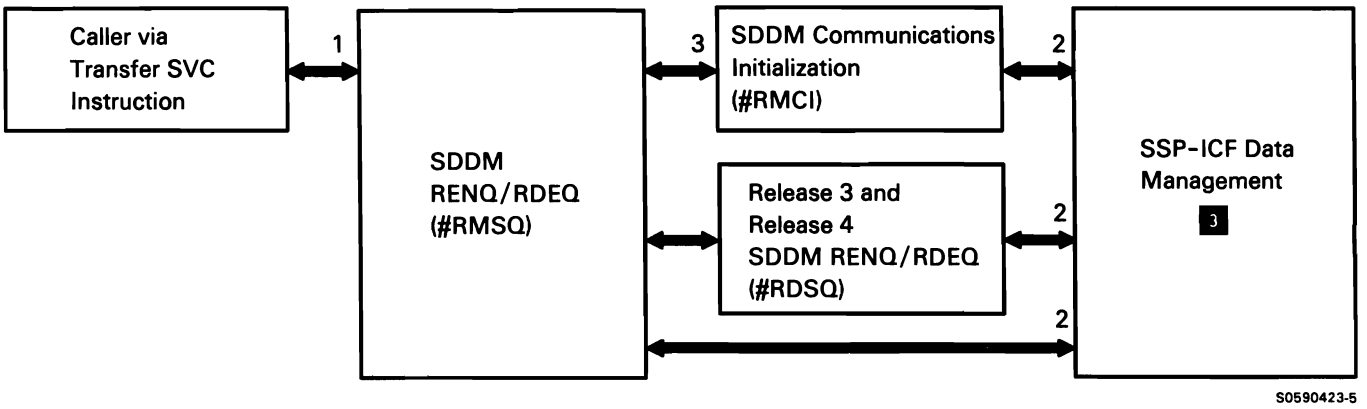


Chart 4.11.2 SDDM RENQ/RDEQ Processing Control Flow

Source DDM (SDDM) Initiator Support Processing

Note: Although the determination of whether a file is local or remote is done in some of the modules shown below, for the purposes of the following description it is assumed that the file has already been determined to be remote. For local file processing, refer to *Job Start Function* 3.

Source DDM (SDDM) initiator support processing is invoked by SSP OCL statement processing modules whenever the file named in the OCL statement is a remote file.

The following SDDM initiator support processes are shown in Chart 4.11.3:

1 Read the next OCL statement.

2 If SYSIN finds an IF DATAF1 PCE or a file size substitution expression, call SDDM VTOC extract to obtain the VTOC information about the remote file.

3 Perform syntax and diagnostics checks on the FILE statement, then do the following:

- Get the file's format 1 using SDDM VTOC extract.
- If no format 1 is returned and if DISP-NEW is specified, create a new format 1.
- Mark format 1 and FSB to indicate that file is remote.
- If the FILE statement occurred before the LOAD statement, call SDDM RENQ to send a DDM LOCK FILE command to the remote system.

4 Process RUN statement by doing the following:

- If RELEASE-YES was specified on the ATTR statement (a NRT program is being initiated), do the following:
  - Create a new JCB for the NRT and point the JCB to the TB.
  - Transfer any non-enqueued FSB(s) from the old JCB to the new NRT JCB.
  - Call DDM suspend to suspend DDM sessions being used by the old JCB and to transfer all unused DDM sessions from the old JCB to the new NRT JCB.
- After performing any required NRT processing, call DDM RENQ for each remote-file-invoking FILE statement that occurred after the LOAD statement.
- If a diagnostic condition occurs and a remote file's lock must be given up, call DDM RDEQ to send a DDM UNLOCK FILE command to the remote system.

5 Process INCLUDE statement for MRT procedures:

- If attaching to an active MRT, call DDM suspend to suspend DDM sessions acquired for this JCB.
- If attaching to an inactive MRT, create a new JCB for the MRT and call DDM SUSPEND to suspend DDM sessions acquired for the old JCB.

6 Process each DDM session owned by the old JCB by doing the following:

- If the use count in the RFS is non-zero or if the new JCB is for a MRT, suspend the DDM session by doing the following:
  - Set TB address and task ID in SUB to zero.
  - Set suspended bit in RFS.
- If SSP-ICF data management posted this task to indicate an asynchronous APPC subsystem condition, then:
  - Issue SSP-ICF INVITE operation to clear post (ACE) from this TB.
  - Set post in RFS.
- If the new JCB is for a NRT and the use count in the RFS is zero, move the RFS from the old JCB to the new JCB.

  Note: If a DDM session is suspended, it will be resumed when the command processor attaches a new task to the old JCB. When the new task calls an SDDM module, the SDDM module calls its internal subroutine to do the following:
  - Store the new TB address and task ID in the SUB.
  - Reset suspended bit in RFS.
  - If post is the needed bit and is set in RFS, post this TB.

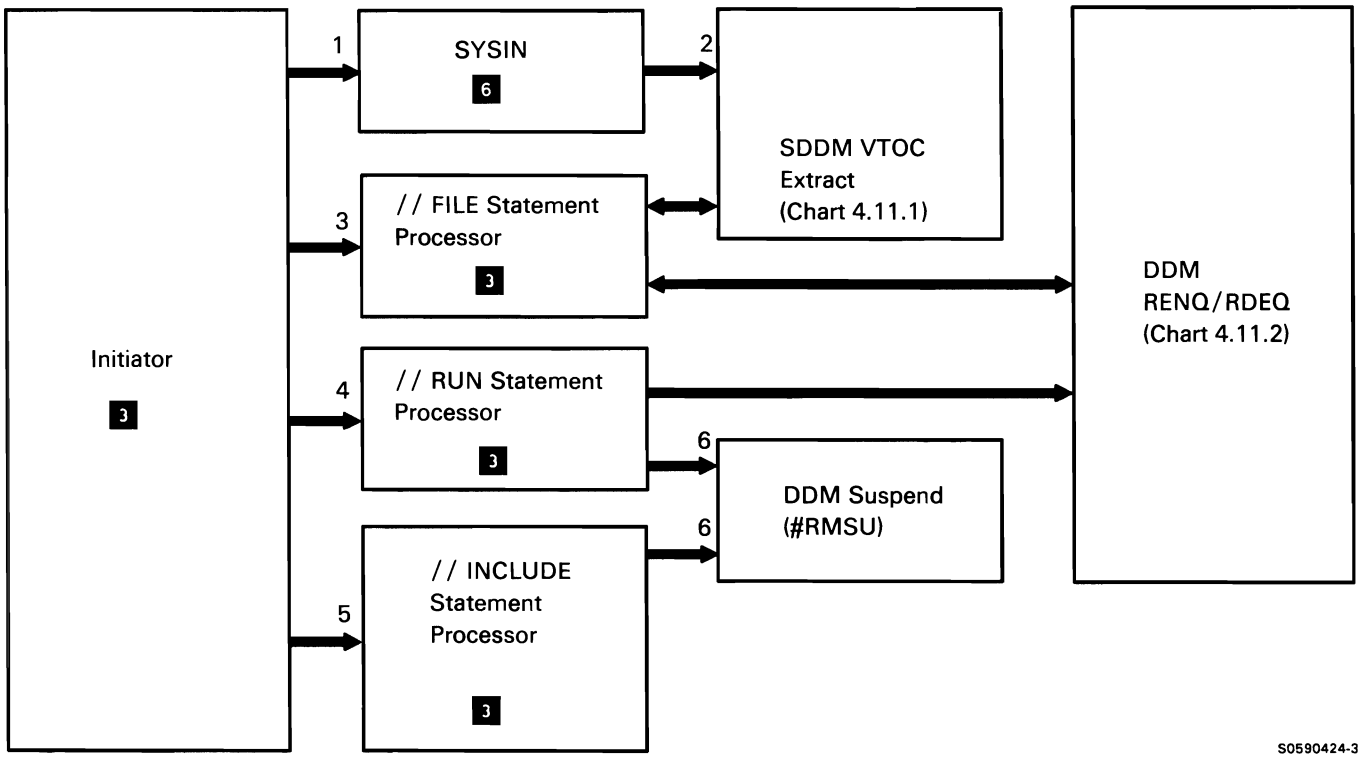


Chart 4.11.3 Source DDM (SDDM) Initiator Support Processing Control Flow

Source DDM (SDDM) Runtime Interfaces

During user program execution, other SSP programs invoke SDDM functions via SVC instructions accompanied by appropriate parameter lists. The following SDDM runtime interfaces are shown in Chart 4.11.4:

1 Process normal or special allocate request passed by #CAM2:

- Route control to network resource directory (NRD) access module to find entry for file requested by caller of allocate.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM runtime interfaces.
- If request is for the allocation of a new file, send DDM CREATE PHYSICAL FILE command to target distributed data management (TDDM).
- If request is a load-to-old-file allocate request, send DDM RECREATE PHYSICAL FILE command to TDDM.
- If request is to clear file, send DDM CLEAR PHYSICAL FILE command to TDDM.
- Process DDM reply sequence received from remote system's TDDM allocate processor.
- For all other allocate requests, the file will be allocated when the open command is processed. Indicate that allocate is successful.
- If the file is allocated successfully, turn on the remote indicator in DTF. The file is allocated in DTF.

2 Process open request from #DD2OP:

- Route control to network resource directory (NRD) access module to find entry for file requested by the caller of open.
- Convert from disk DTF request to DDM OPEN FILE command.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM runtime interfaces.
- Send OPEN FILE command to TDDM.
- Process DDM reply sequence received from remote system's TDDM open processor.
- If the file is opened successfully, initialize FAB and calculate the data in FAB for get and put use.

3 Process put or control function request from disk data management mainline:

- Convert disk data management put request into a DDM command.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM runtime interfaces.
- Send DDM command to TDDM.
- Process DDM reply sequence from TDDM and post results to caller.

4 Process get request from disk data management mainline:

- Convert disk data management get request into a DDM command.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM runtime interfaces.
- Send DDM command to TDDM.
- Process DDM reply sequence from TDDM and post results to caller.

5 Process close request from #DDCL:

- Convert from disk DTF request to DDM CLOSE command.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM runtime interfaces.
- If deferred output request, send the DDM PUT command to TDDM and process DDM reply sequence.
- Send DDM CLOSE command to TDDM.
- Process DDM reply sequence received from remote system's TDDM close processor.

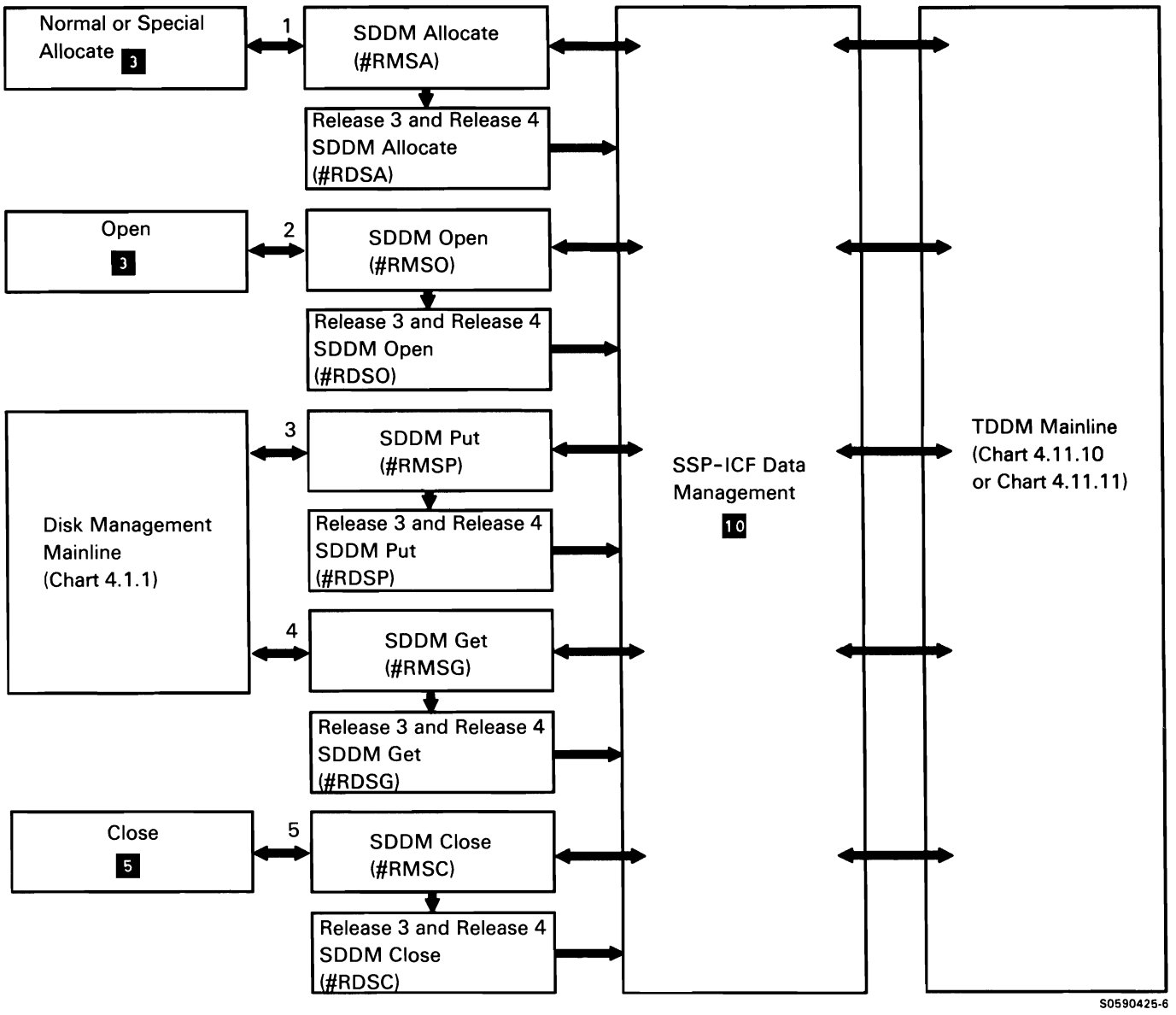


Chart 4.11.4 Source DDM (SDDM) Runtime Interfaces Control Flow

Source DDM (SDDM) Rename

Source DDM Rename processes a request from the calling module to rename a file that resides on a remote system.

The following SDDM Rename processes are shown in Chart 4.11.5:

1 Process rename request:

- Scan the NRD entry lock (NEL) queue to find the NRD entry for the old file label.
- Route control to NRD access to find the new file label.
- If file label not found, return an NRD entry not found message to the caller.
- If file label found, test the remote location. The new file location must be the same as the old one.
- If request is test-for or read format 1:
  - Route control to SDDM communications initialization to attach the target distributed data management (TDDM) at the remote location.
  - If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM extract.
  - Send DDM LIST FILE ATTRIBUTES command to TDDM mainline.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM rename.
- Send DDM RENAME FILE command to TDDM.
- Process DDM reply sequence from TDDM mainline and pass return code to the caller.

2 Get NRD entry for requested file from NRD file.

3 Process requests from SDDM Rename:

- Acquire an APPC subsystem session.
- Evoke TDDM.
- Process puts and gets between SDDM rename and remote system's TDDM mainline.

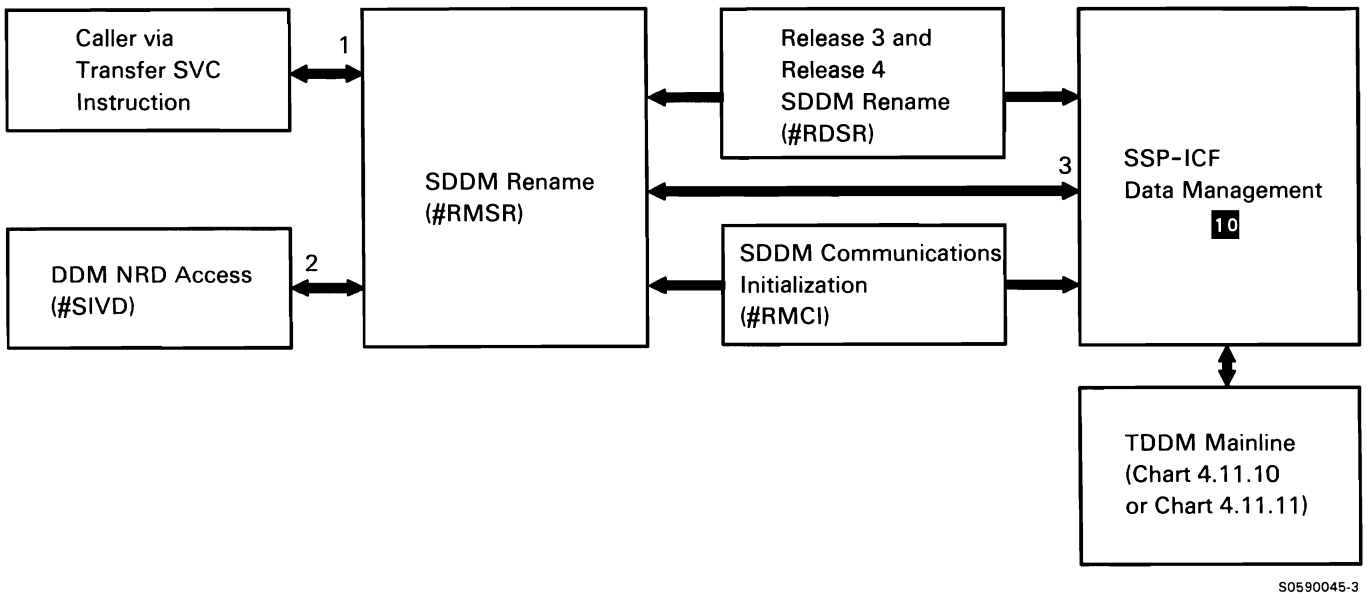


Chart 4.11.5 Source DDM (SDDM) Rename Process Control Flow

Source DDM (SDDM) BLDINDEX

Source DDM BLDINDEX processes a request from the calling module to build a remote alternate index file for a remote base file.

The following DDM BLDINDEX processes are shown in Chart 4.11.6:

1 Special allocate the base file.

2 Process build index request:

- Scan the NRD entry lock (NEL) queue to find the NRD entry for the base file label.
- Find RFS created when the base file is allocated.
- Route control to NRD access to find the alternate index file label.
- If file label not found, return an NRD entry not found message to the caller.
- If file label found, test the remote location. The logical file location must be the same as the physical file location.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM BLDINDEX.
- Send DDM CREATE ALTERNATE INDEX FILE command to TDDM.
- Process DDM reply sequence from TDDM mainline and pass return code to the caller.

3 Get NRD entry for a requested file from the NRD file.

4 Process requests from SDDM build index.

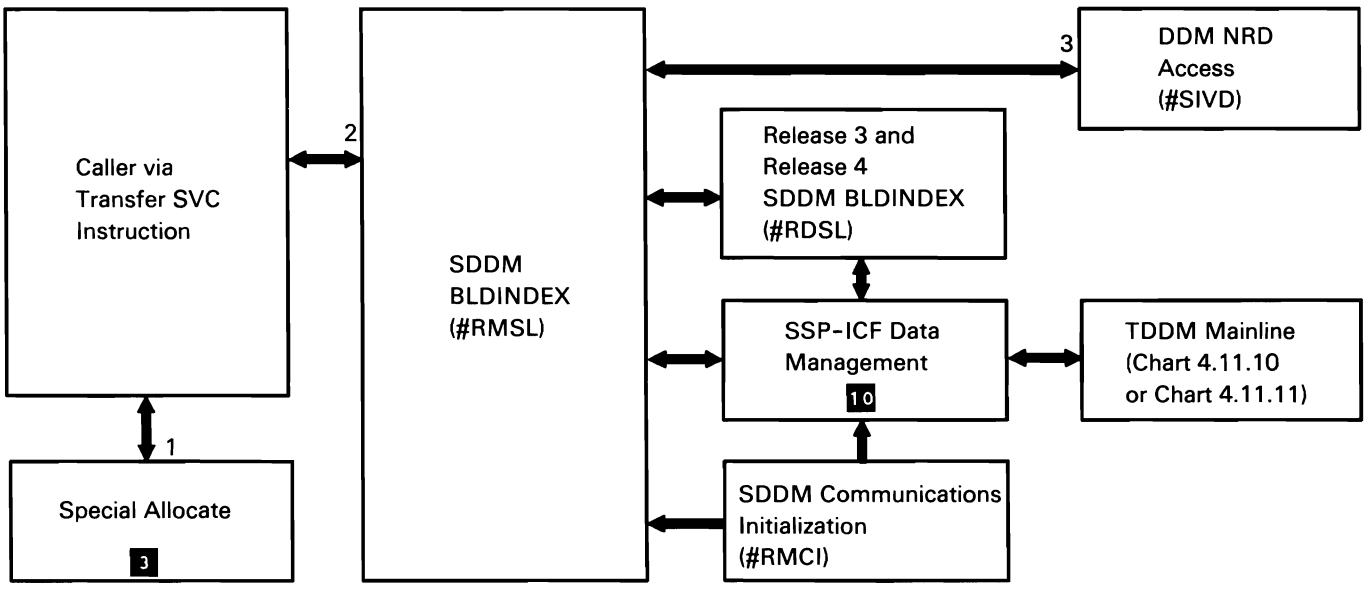


Chart 4.11.6 Source DDM (SDDM) BLDINDEX Process Control Flow

Source DDM (SDDM) Delete

Source DDM Delete processes a request from the calling module to delete the remote file that resides on a remote system.

The following SDDM Delete processes are shown in Chart 4.11.7:

1 $DELET calls DEALLOCATE (#CAD1) for a delete file request and the file is a remote file.

2 Normal termination is processing an existing resident remote file that should be deleted at the end of the step.

3 Abnormal termination is processing a new remote file that was allocated by the step.

4 Process DDM delete request:

- Build and send DDM DELETE FILE command to TDDM mainline.
- Process DDM reply sequence from TDDM mainline and pass return code to the caller.
- If remote system is a Release 3 and Release 4 DDM system, route control to Release 3 and Release 4 SDDM delete.

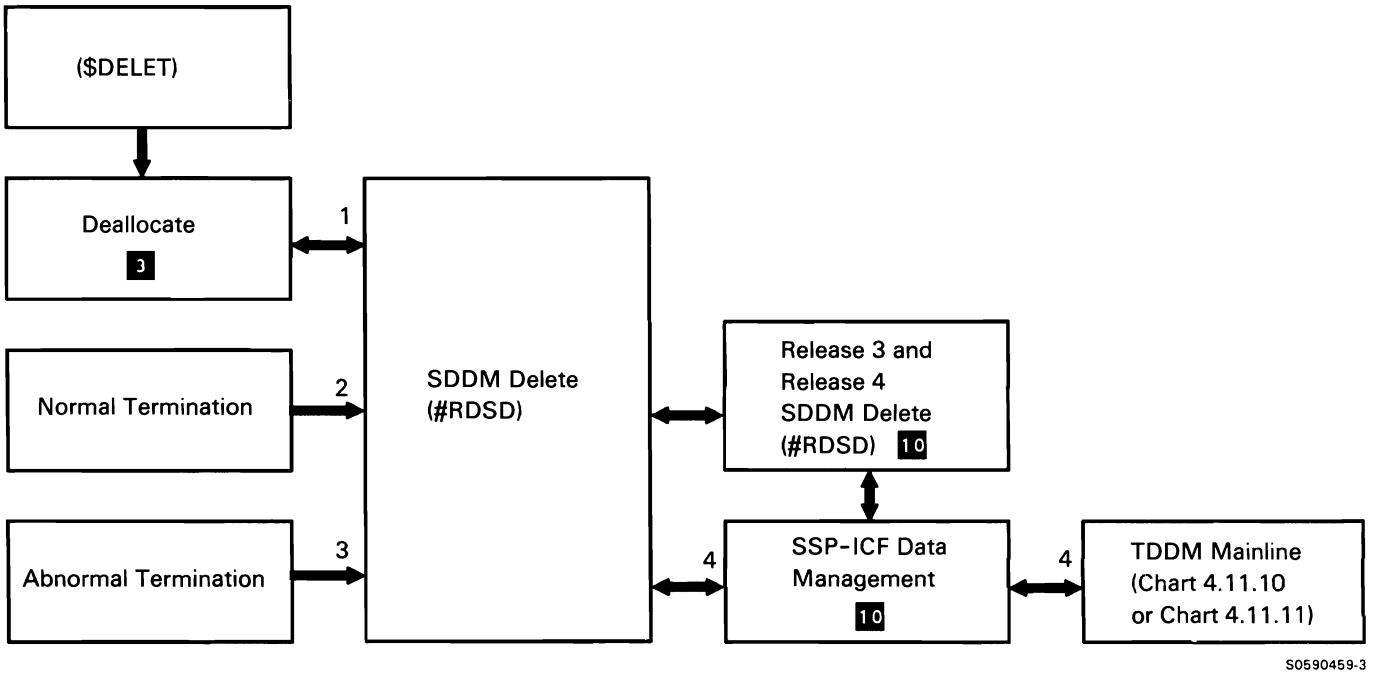


Chart 4.11.7 Source DDM (SDDM) Delete Process Control Flow

Source DDM (SDDM) File Transfer

Source DDM File Transfer processes a request from the calling module to copy an entire file to and/or from a remote system.

The following SDDM File Transfer processes are shown in Chart 4.11.8:

1 Process a LIST FILE SERVER ATTRIBUTES request. Return an indication to the caller whether or not the LOAD FILE, UNLOAD FILE, and COPY FILE commands are supported by the remote system.

2 Perform initialization for the load file and unload file requests and pass return code to caller. This includes loading put next loaded record or get next unloaded record into the caller's region.

3 Put the next record for a load file request and pass return code to caller.

4 Get the next record for an unload file request and pass record data and/or return code to caller.

5 Terminate the load file or unload file request and pass return code to caller.

6 Process the copy file request and pass return code to caller.

7 Process puts and gets between SDDM file transfer and remote system's Release 5 TDDM mainline.

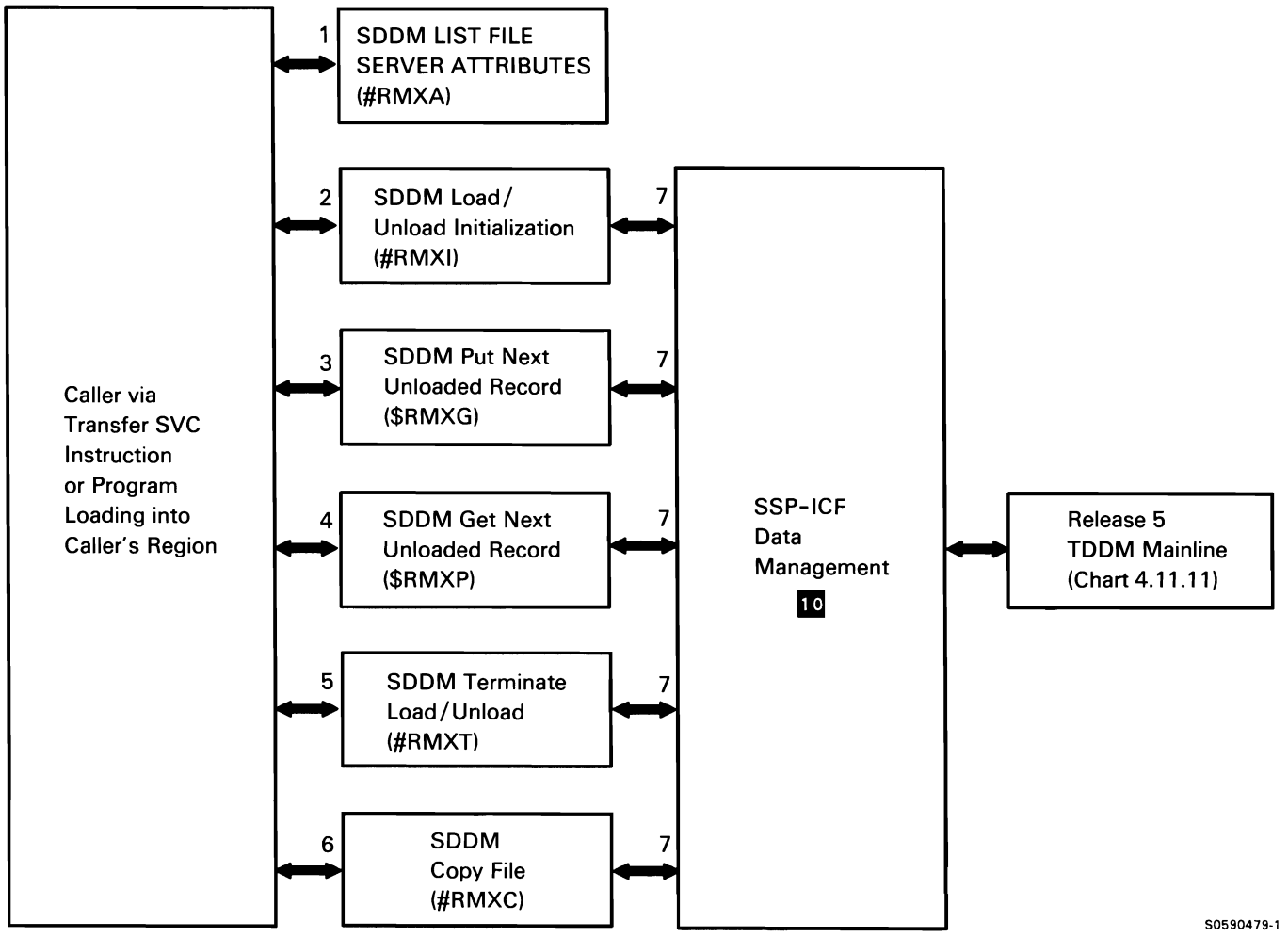


Chart 4.11.8 Source DDM (SDDM) File Transfer Process Control Flow

Source DDM (SDDM) Cleanup Processing

Before or during user program termination, other SSP programs invoke SDDM cleanup processing via SVC instructions accompanied by appropriate parameter lists. The following SDDM cleanup processes are shown in Chart 4.11.9:

1 Delete specified file and send equivalent command to TDDM under the following conditions:

- If deallocate request and file is a RETAIN-S file, or if the deallocate requester specified delete.
- If (normal) termination request and file is an old file to be deleted.
- If abnormal termination, delete all new files that were allocated.

2 Unlock (RDEQ) remote file by sending DDM UNLOCK FILE command to TDDM.

3 Do the following DDM termination cleanup:

- If end of job, release all DDM-acquired APPC subsystem sessions.
- If end of step, release any DDM-acquired APPC subsystem sessions not needed for next job step.
- Dequeue and free any session RFSs not needed by next job step.
- Dequeue and free any NELs that are no longer needed by any job in the system.

4 Process for the following errors:

- Errors detected by SDDM modules.
- Abnormal SSP-ICF return codes.

5 Use SYSLOG to issue message to history file and/or user's CRT or system console.

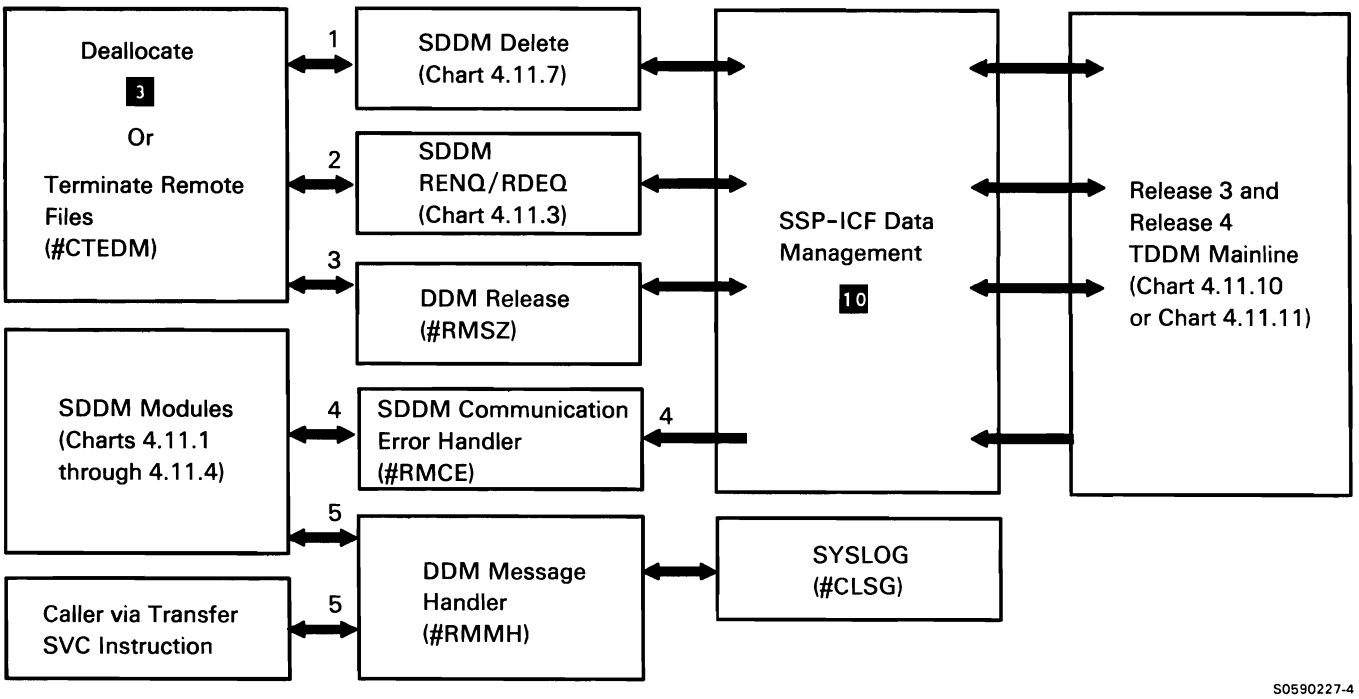


Chart 4.11.9 Source DDM (SDDM) Cleanup Processing Control Flow

This page is intentionally left blank.

Release 3 and Release 4 Target DDM (TDDM) Mainline

The Release 3 and Release 4 target DDM (TDDM) mainline is evoked by the remote system's source DDM (SDDM), to service file requests received from the remote system. Communications between the remote SDDM and the local Release 3 and Release 4 TDDM take place within an APPC subsystem session established by SDDM. Release 3 and Release 4 TDDM, simulating a local application program, makes the remote request appear as a local request to the SSP.

The following Release 3 and Release 4 TDDM mainline processes are shown in Chart 4.11.10:

1 Perform initial processing on all remote file access requests received from the remote system's SDDM. Route for appropriate processing of request and pass applicable completion status back to SDDM requester.

2 Process DDM LIST FILE ATTRIBUTES command received from SDDM extract.

3 Find specified file entry in VTOC or network resource directory (NRD) and pass results back to Release 3 and Release 4 TDDM extract.

4 Process response from VTOC extract:

- If requested file entry was found in VTOC, pass requested information back to Release 3 and Release 4 TDDM mainline in reply data element.
- If requested file entry was not found in VTOC or NRD, return error reply code to Release 3 and Release 4 TDDM mainline.

5 Process DDM LOCK FILE or UNLOCK FILE command received from SDDM RENQ/RDEQ.

6 Find specified file entry in VTOC or NRD and pass results back to Release 3 and Release 4 TDDM RENQ/RDEQ.

7 Process response from VTOC extract:

- If requested file entry was found in VTOC, process DDM LOCK FILE or UNLOCK FILE command:
  - If DDM LOCK FILE command, pass control to special allocate.
  - If DDM UNLOCK FILE command, pass control to deallocate.
- If requested file entry was not found in VTOC or NRD, return error reply code to Release 3 and Release 4 TDDM mainline.

8 Perform file allocate portion of DDM LOCK FILE command:

- Build and queue FSB and format 1.
- Resource enqueue format 1 (lock file).
- Return to Release 3 and Release 4 TDDM RENQ (allocation will be done by open).

9 Process DDM UNLOCK FILE command:

- Dequeue and free FSB (unlock file).
- Resource dequeue format 1.
- Return to Release 3 and Release 4 TDDM RDEQ.

10 Process to clear DDM CREATE or RECREATE PHYSICAL FILE command received from SDDM allocate:

- Convert DDM command parameters to special allocate parameter list.
- Call special allocate to clear create or recreate file.
- Call deallocate to deallocate file (file will be reallocated by open).

11 Find specified file entry in VTOC or NRD and pass results back to Release 3 and Release 4 TDDM allocate.

12 Process response from VTOC extract:

- If create physical file was found in VTOC or in the remote system's VTOC, return an error reply code to the Release 3 and Release 4 TDDM mainline. Otherwise, create the file.

- If recreate physical file was found in VTOC or in the remote system's VTOC, recreate the file. Otherwise, return error reply code to Release 3 and Release 4 TDDM mainline.

13 Perform allocation processing requested, then deallocate file (in anticipation of open request).

14 Process DDM OPEN FILE command passed by remote system's SDDM:

- Call special allocate with FSB.

- Place format-1 information in DTF.

- Convert DDM OPEN FILE command parameters into DTF parameters.

- Call open.

15 Find specified file entry in VTOC or NRD and pass results back to Release 3 and Release 4 TDDM open.

16 Process response from VTOC extract:

- If requested file entry was found in VTOC, process DDM OPEN command and route to open processor.

- If requested file entry was not found in VTOC or NRD, return error reply code to Release 3 and Release 4 TDDM mainline.

- Set local or remote status of requested file for later use by #RDTM's disk data management's interface.

17 Allocate specified file (pre-allocate processing was done at allocate time).

18 Open specified file and return to Release 3 and Release 4 TDDM open.

19 Process DDM record access commands received from remote system:

- If file status (set by Release 3 and Release 4 TDDM open) is remote, pass command to Release 3 and Release 4 TDDM relay.

- If file status is local:
  - Convert DDM command to disk data management op code.
  - Convert DDM command parameters to disk DTF parameters.
  - Issue DTF to disk data management.
  - Convert disk data management completion code to DDM completion status.
  - Send any data records and status back to SDDM requester.

20 Process DELETE FILE command received from SDDM file delete.

21 Process RENAME FILE command received from SDDM file rename:

- Find specified file entry in VTOC or NRD and pass results back to Release 3 and Release 4 TDDM file rename.

- Process response from VTOC extract:
  - If requested file entry was found in VTOC, process RENAME FILE command.
  - If requested file entry was not found in VTOC or NRD, return error reply code to Release 3 and Release 4 TDDM mainline.

- Rename specified file.

22 Process BLDINDEX FILE command received from SDDM file build index:

- Find specified file entry in VTOC or NRD and pass results back to Release 3 and Release 4 TDDM file rename.

- Process response from VTOC extract:
  - If requested file entry was found in VTOC, process BLDINDEX FILE command.
  - If requested file entry was not found in VTOC or NRD, return error reply code to Release 3 and Release 4 TDDM mainline.

- Special allocate physical file to perform security.
- Deallocate physical file to release security.
- Build index for specified file.

23 Find specified file entry in VTOC or NRD and pass results back to Release 3 and Release 4 TDDM file delete.

24 Process response from VTOC extract:

- If requested file entry was found in VTOC, process DELETE FILE command.
- If requested file entry was not found in VTOC or NRD, return error reply code to Release 3 and Release 4 TDDM mainline.

25 Delete specified file.

26 Process CLOSE FILE command received from SDDM file close:

- If file status (set by Release 3 and Release 4 TDDM open) is remote, pass request to Release 3 and Release 4 TDDM relay.
- If file status is local:
  - Convert DDM command to DTF op code.
  - Convert DDM command parameters to DTF parameters.
  - Issue DTF to close.
  - Convert close completion code to DDM completion status.

27 Close specified file.

28 Deallocate specified file.

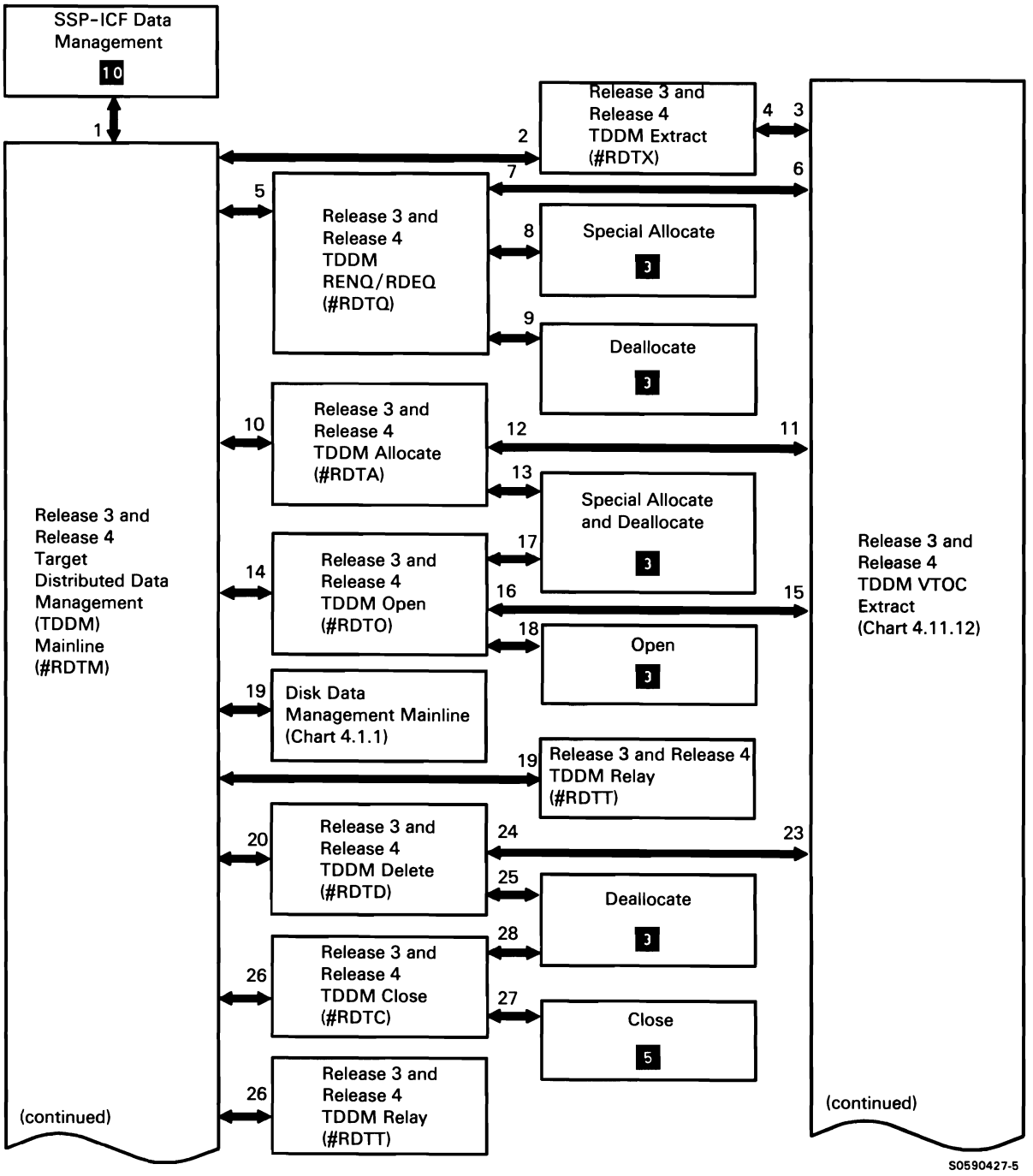


Chart 4.11.10 (Part 1 of 2) Release 3 and Release 4 Target DDM (TDDM) Mainline Control Flow

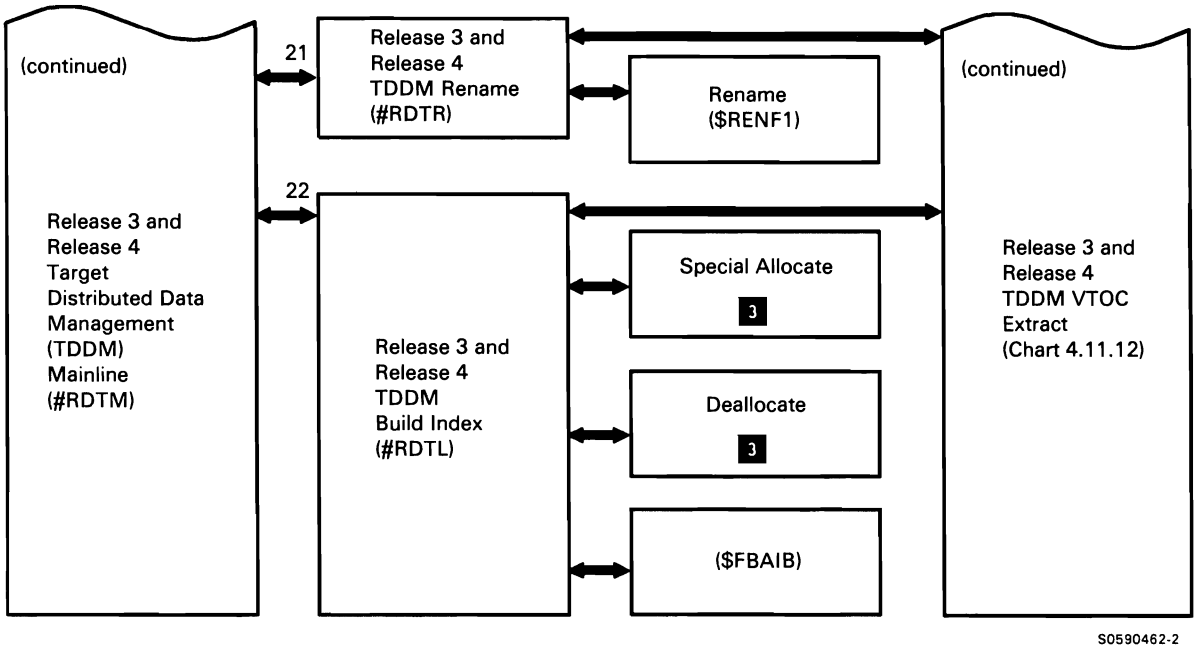


Chart 4.11.10 (Part 2 of 2) Release 3 and Release 4 Target DDM (TDDM) Mainline Control Flow

Release 5 Target DDM (TDDM) Mainline

The Release 5 target DDM (TDDM) mainline is evoked by the remote system's source DDM (SDDM), to service file requests received from the remote system. Communications between the remote SDDM and the local Release 5 TDDM take place within an APPC subsystem session established by SDDM. Release 5 TDDM, simulating a local application program, makes the remote request appear as a local request to the SSP.

The following Release 5 TDDM mainline processes are shown in Chart 4.11.11:

1 Perform initial processing on all remote file access requests received from the remote system's SDDM. Route for appropriate processing of request and pass applicable completion status back to SDDM requester.

2 Perform initialization for Release 5 TDDM mainline. Route control to TDDM exchange server attributes module.

3 Process EXCHANGE SERVER ATTRIBUTES command received from SDDM.

4 Process DDM LIST FILE ATTRIBUTES command received from SDDM extract.

5 Find specified file entry in VTOC or network resource directory (NRD) and pass results back to Release 5 TDDM extract.

6 Process response from VTOC extract:

- If requested file entry was found in VTOC, pass requested information back to Release 5 TDDM mainline in reply data element.
- If requested file entry was not found in VTOC or NRD, return error reply code to Release 5 TDDM mainline.

7 Process DDM LOCK FILE or UNLOCK FILE command received from SDDM RENQ/RDEQ.

8 Perform file allocate portion of DDM LOCK FILE command:

- Build and queue FSB and format 1.
- Resource enqueue format 1 (lock file).
- Return to Release 5 TDDM RENQ (allocation will be done by open).

9 Process DDM UNLOCK FILE command:

- Dequeue and free FSB (unlock file).
- Resource dequeue format 1.
- Return to Release 5 TDDM RDEQ.

10 Process to clear DDM CREATE or RECREATE PHYSICAL FILE command received from SDDM allocate:

- Convert DDM command parameters to special allocate parameter list.
- Call special allocate to clear create or recreate file.
- Call deallocate to deallocate file (file will be reallocated by open).

11 Perform allocation processing requested, then deallocate file (in anticipation of open request).

12 Process DDM OPEN FILE command passed by remote system's SDDM:

- Call special allocate with FSB.
- Place format-1 information in DTF.
- Convert DDM OPEN FILE command parameters into DTF parameters.
- Call open.

13 Allocate specified file (pre-allocate processing was done at allocate time).

14 Open specified file and return to Release 5 TDDM open.

15 Process DELETE FILE command received from SDDM file delete.

16 Delete specified file.

17 Process CLOSE FILE command received from SDDM file close:

- If file status (set by Release 5 TDDM open) is remote, pass request to Release 5 TDDM relay.
- If file status is local:
  - Convert DDM command to DTF op code.
  - Convert DDM command parameters to DTF parameters.
  - Issue DTF to close.
  - Convert close completion code to DDM completion status.

18 Deallocate specified file.

19 Close specified file.

20 Process input record operation requests received from SDDM get.

21 Process DDM record access commands received from remote system:

- Convert DDM command to disk data management op code.
- Convert DDM command parameters to disk DTF parameters.
- Issue DTF to disk data management.
- Convert disk data management completion code to DDM completion status.
- Send any data records and status back to SDDM requester.

22 Process output record operation requests received from SDDM put.

23 Process RENAME FILE command received from SDDM file rename.

- Find specified file entry in VTOC or NRD and pass results back to Release 5 TDDM file rename.
- Process response from VTOC extract:
  - If requested file entry was found in VTOC, process RENAME FILE command.
  - If requested file entry was not found in VTOC or NRD, return error reply code to Release 5 TDDM mainline.
- Rename specified file.

24 Process BLDINDEX FILE command received from SDDM file build index.

- Find specified file entry in VTOC or NRD and pass results back to Release 5 TDDM file rename.
- Process response from VTOC extract:
  - If requested file entry was found in VTOC, process BLDINDEX FILE command.
  - If requested file entry was not found in VTOC or NRD, return error reply code to Release 5 TDDM mainline.
- Special allocate physical file to perform security.
- Deallocate physical file to release security.
- Build index for specified file.

25 Process UNLOAD FILE command received from SDDM.

26 Process LOAD FILE command received from SDDM.

27 Process COPY FILE command received from SDDM.

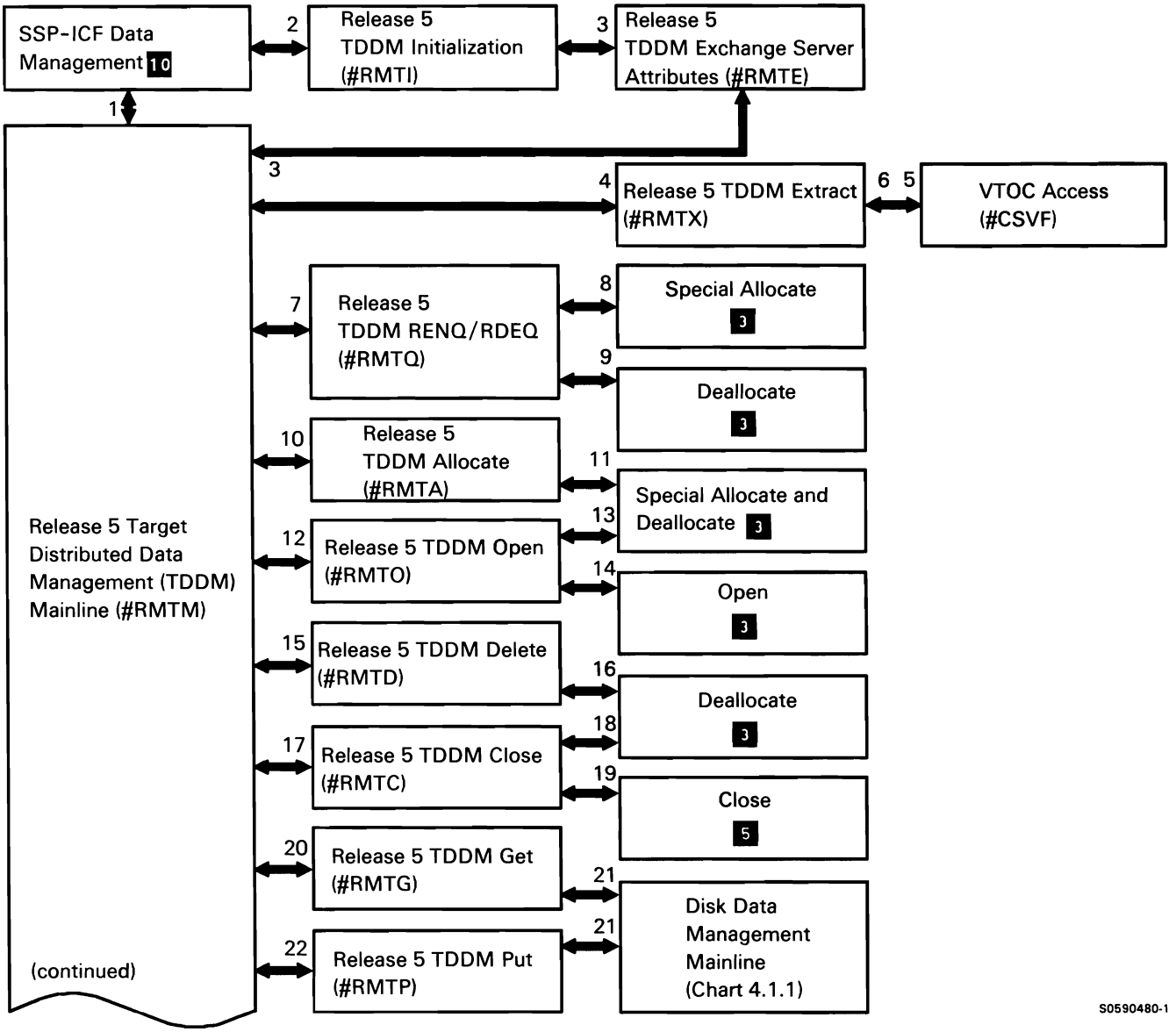


Chart 4.11.11 (Part 1 of 2) Release 5 Target DDM (TDDM) Mainline Control Flow

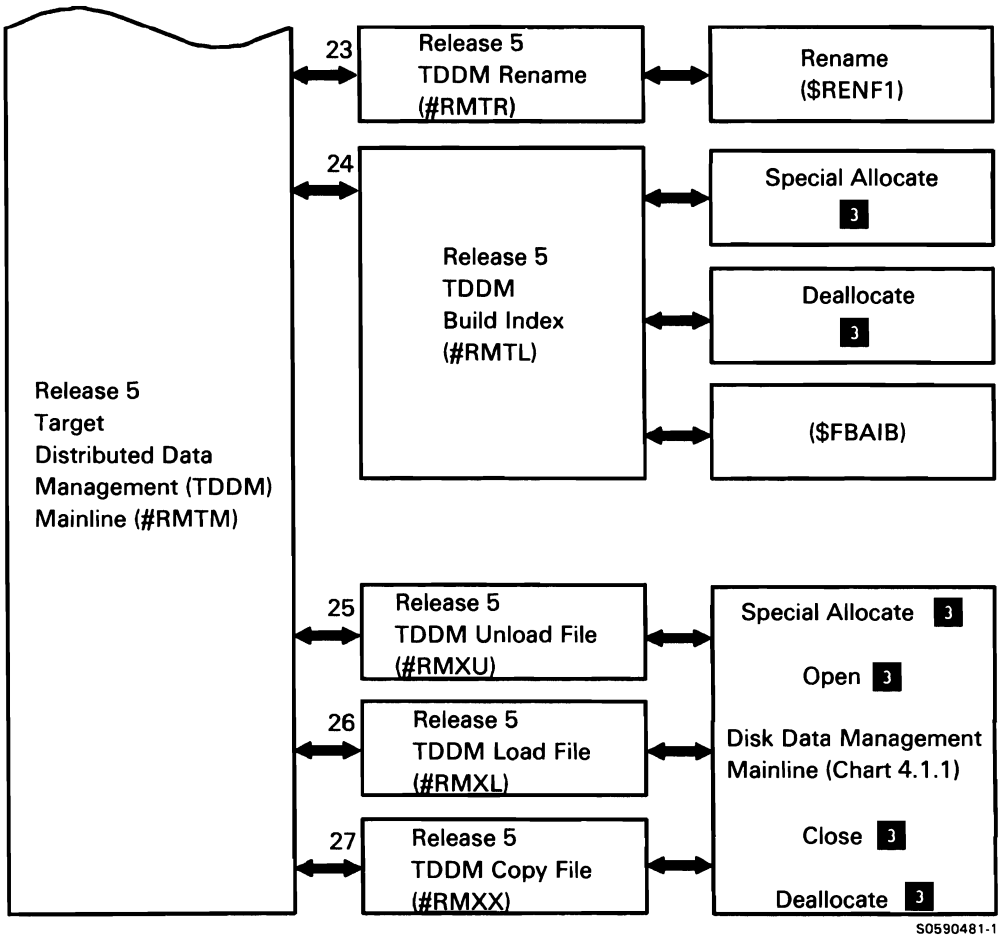


Chart 4.11.11 (Part 2 of 2) Release 5 Target DDM (TDDM) Mainline Control Flow

Release 3 and Release 4 Target DDM (TDDM) VTOC Extract

The Release 3 and Release 4 target DDM (TDDM) VTOC extract is called by other Release 3 and Release 4 TDDM modules to find the requested file label. If the file label is not found in the target system's VTOC, Release 3 and Release 4 TDDM VTOC extract searches the network resource directory (NRD). If the file label is found in the NRD, the request is passed to Release 3 and Release 4 TDDM relay for routing to the next Release 3 and Release 4 TDDM system.

The following Release 3 and Release 4 TDDM VTOC extract processes are shown in Chart 4.11.12:

1 Set *remote files not allowed* and call VTOC access.

2 Find specified file entry by checking VTOC; pass return code back to caller.

3 Process response from VTOC access:

- If requested file entry was found in VTOC, return format 1 to caller to continue processing requested by source DDM (SDDM) requester.
- If requested file entry was not found in VTOC, pass request to SDDM VTOC extract to search for the NRD entry.
- If requested file entry was found in NRD, pass request to Release 3 and Release 4 TDDM relay for routing to appropriate system's Release 3 and Release 4 TDDM.
- If requested file entry was not found in VTOC or NRD, return file-not-found reply code to caller to be passed to Release 3 and Release 4 TDDM mainline for transmission to SDDM requester.

4 If NRD entry found:

- Lock entry.
- Set up APPC subsystem session.

If requested file entry was not found in NRD, return file-not-found reply code to caller.

5 Relay request to system specified in NRD entry for requested file.

6 Send request passed by Release 3 and Release 4 TDDM relay to APPC subsystem for transmission to next Release 3 and Release 4 TDDM system.

7 Pass next target system's reply to caller.

Relay reply received from other Release 3 and Release 4 TDDM back to the SDDM system.

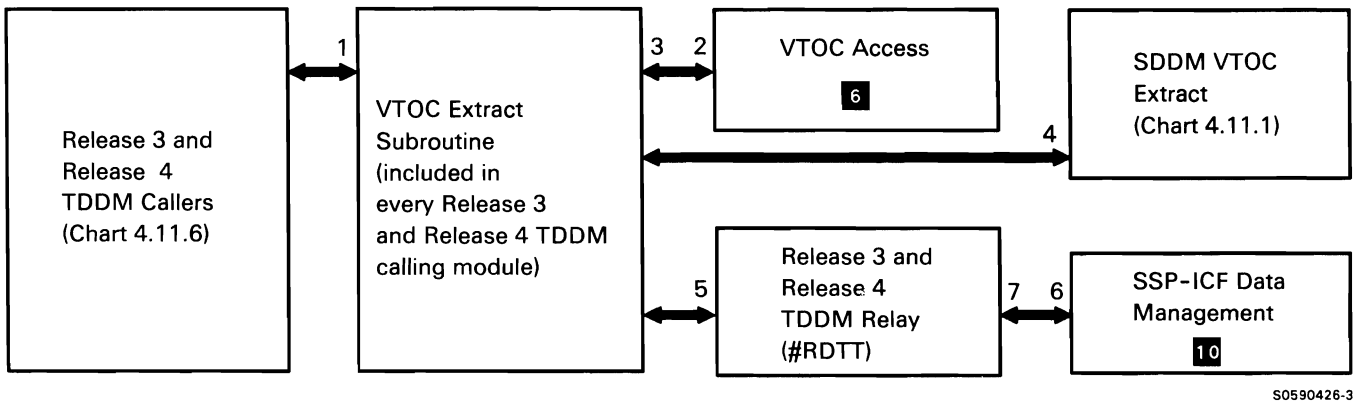


Chart 4.11.12 Release 3 and Release 4 Target DDM (TDDM) VTOC Extract Control Flow

DDM Trace (SDDM and TDDM)

DDM trace processes a request from an SDDM or TDDM module to log a DDM data stream to an active trace file. This processing is performed only when trace is active for DDM.

The following DDM trace processes are shown in Chart 4.11.13:

1 Call DDM communications trace to log data stream data to trace file.

2 Perform the trace logging.

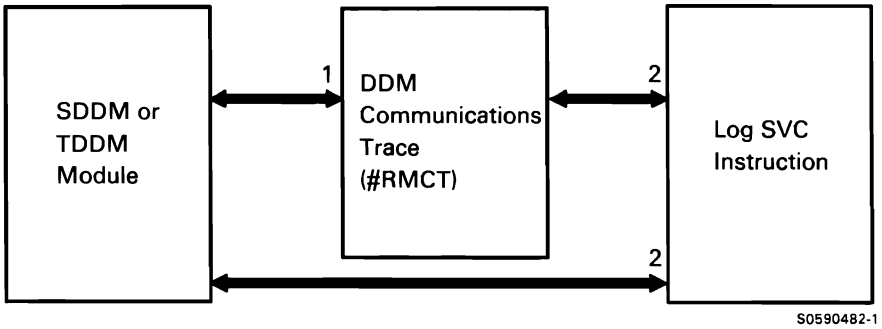


Chart 4.11.13 DDM Trace (SDDM and TDDM) Processing Control Flow

This page is intentionally left blank.

QUERY DATA MANAGEMENT

Query data management (QDM) is called by the Query/36 program product for the following purposes:

- Generate code for specified selection fields.
- Build sort control fields.
- Process expressions and record select criteria.
- Process joins.
- Execute user query requests.

The calling modules call query data management via a transfer SVC instruction with a query user parameter list. Query data management generates code customized for the user's request and stores it in the query common area of the query data management task work space. During the execution phase, query data management drivers call the generated code as subroutines.

The following table shows the locations of key data areas involved in query data management during the generation phase:

Dump Address (in Hex)	Contents
0000	Query common - program variables, generated code, RRN buffer (if used)
3800	Query code generator modules
C800	Query open (#DQQOP)
E800	Query parameter list DTF

The following table shows the locations of key data areas involved in QDM during the execution phase:

Dump Address (in Hex)	Contents
0000	Query common - program variables, generated code, RRN buffer (if used), and I/O buffers
D800	User output buffer (4K or 6K)
F000	Query driver modules

Chart 4.12.1 shows an overview of query generation and execution functions:

1 Perform initial processing based on user parameter list:

   - Map generator list.
   - Map to query common.
   - Dissect the query parameter list and build a new parameter list for each input file.
   - Route control to query code generators for each dissected parameter list built.

2 Use initialized information and dissected parameter list passed by query open to generate code to execute user request.

3 On return from code generators, do the following:

   - Call query index optimizer (#DQXOP), if used.
   - Initialize control path, based on user request. Specify driver in parameter list.
   - Pass control to applicable sort driver or return to caller.

4 Use task work space (TWS) perform the following:

- If joining files, call query driver to fill work area buffer with records selected from previously processed input file(s).
- Fill work area input buffer with records from input file.
- If required, build scratch file for sort/merge.
- Branch to generated code to compare each record in work area buffer for record select criteria and join records (if joining files).
- Call query sort (#DQQS) to sort work area output buffer.
- Write sorted work area buffer to scratch file.

5 Sort records in work area output buffer passed with request when sort complete, add end-of-buffer mark.

6 Use TWS to perform the following:

- If joining files, call query driver to fill work area buffer with records selected from previously processed input file(s).
- Use relative record number (RRN) buffer to select records from input file and fill work area input buffer.
- If required, build scratch file for sort/merge.
- Branch to generated code to compare each record in work area buffer for record select criteria and join records (if joining files).
- Call query sort (#DQQS) to sort work area output buffer.
- Write sorted work area buffer to scratch file.

7 If more than four sorted buffers are written to scratch file, use TWS to perform the following:

- Merge sorted buffers until number of buffers is four or less.
- Return control to caller with parameters set to call merge phase 2 (#DQM2).

8 Use TWS to perform the following:

- If joining files, call query driver to fill work area buffer with records selected from previously processed input file(s).
- Fill work area input buffer with records from input file.
- Branch to generated code to compare each record in work area buffer for record select criteria and join records (if joining files).
- Recursive calls between driver and caller (could be another driver if joining files) for each filled output buffer, until end-of-file.

9 Use TWS to perform the following:

- If joining files, call query driver to fill work area buffer with records selected from previously processed input file(s).
- Use RRN buffer to select records from input file and fill work area input buffer.
- Branch to generated code to compare each record in work area buffer for record select criteria and join records (if joining files).
- Recursive calls between driver and caller (could be another driver if joining files) for each filled output buffer, until end-of-file.

10 Use TWS to perform the following:

- If joining files, call query driver to fill work area buffer with records selected from previously processed input file(s).
- Fill work area input buffer with records from input file.
- Branch to generated code to compare each record in work area buffer for record select criteria and join records (if joining files).
- Using file specification block (FSB) pointed to by the query parameter list, write buffer(s) of selected records to caller specified file.
- On end of input file, return control to caller.

11 Use TWS to perform the following:

- Merge sorted buffer(s) in scratch file until end of data is reached.
- Write selected records from scratch file into user's output buffer.
- Recursive calls between merge driver and caller until end of data is reached.

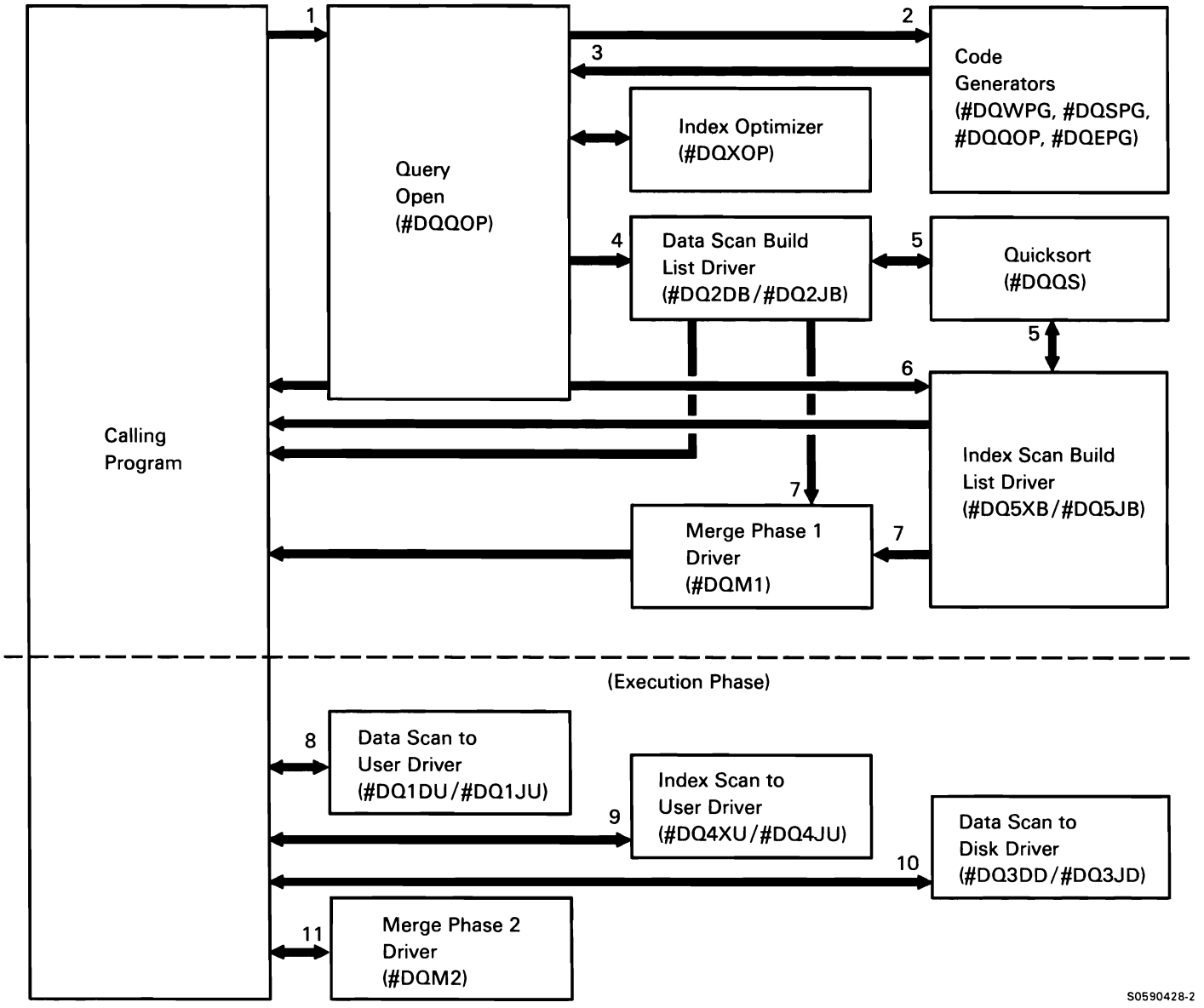


Chart 4.12.1 Query Data Management (QDM) Control Flow

Job Termination Function 5

The job termination function contains all SSP programming required for end-of-step, end-of-job, or abnormal termination processing. It includes the following subfunctions:

- Close Chart 5.1.1
- Termination Chart 5.2.0

CLOSE SUBFUNCTION

The close subfunction ensures that files used by a step or job are properly closed prior to step or job termination by:

- Completing the processing of any data in output buffers
- Freeing SQS data areas that were obtained when the file was opened
- Restoring all opened DTFs to a preopened status

Close also provides for the reallocation and reopening of disk files.

The close subfunction is performed by a common close transient and the necessary device-oriented close transients. Close is invoked by the transfer SVC instruction with XR2 pointing to the postopen DTF chain. The transfer SVC instruction causes the loading and execution of the common open transient, which loads and executes the necessary device-oriented close transients. Each device-oriented module searches the DTF chain to ensure that it has restored all applicable DTFs of its device type to preopen status before it terminates.

The following DTF close processes are shown in Chart 5.1.1:

1 Perform common close processing:

- Examine DTFs on backward DTF chain.
- If there are printer DTFs to be closed:
  - Complete processing of print buffer data.
  - Free assign/free space used for printer IOB and buffer (includes remote print buffer, if any).
  - Restore DTF to preopened status.
- If there are work station DTFs to be closed, alter the DTF attribute bytes to restore the DTFs to preopened status.
- If there are BSC DTFs to be closed, set up DTF and communications specification block (CSB) and transfer to the BSC task.
- If there are other device DTFs, call appropriate device-dependent close transient (call disk close last).

2 Find all diskette DTFs on the chain to be closed and close them:

- Update active format-1 image and call end-of-volume processing if required.
- If required, write data set label to VTOC.
- If an output error caused end of volume, issue error message and perform end-of-volume processing.
- Restore DTF to preopened status.

3 Find all tape DTFs on the chain to be closed and close them:

- If put or add with fixed block, write remaining records in physical buffer and trailer labels to tape.
- Process appropriate end condition: rewind, unload, or leave.

4 Find all disk DTFs on the chain to be closed and close them:

Note: If disk close is called by termination or abnormal termination, FSBs are used to close disk files.

- Complete the processing of data in the output buffers.
- Free SQS control blocks obtained by disk open.
- If sectorized data management access method (ZPAM) DTF, free the IOB (in SQS) that was obtained by disk open; if ZPAM output or add access method, update main storage data set label.
- If request is for a remote file, perform distributed data management file close.
- Restore DTF to preopened status.

5 Perform diskette end-of-volume processing.

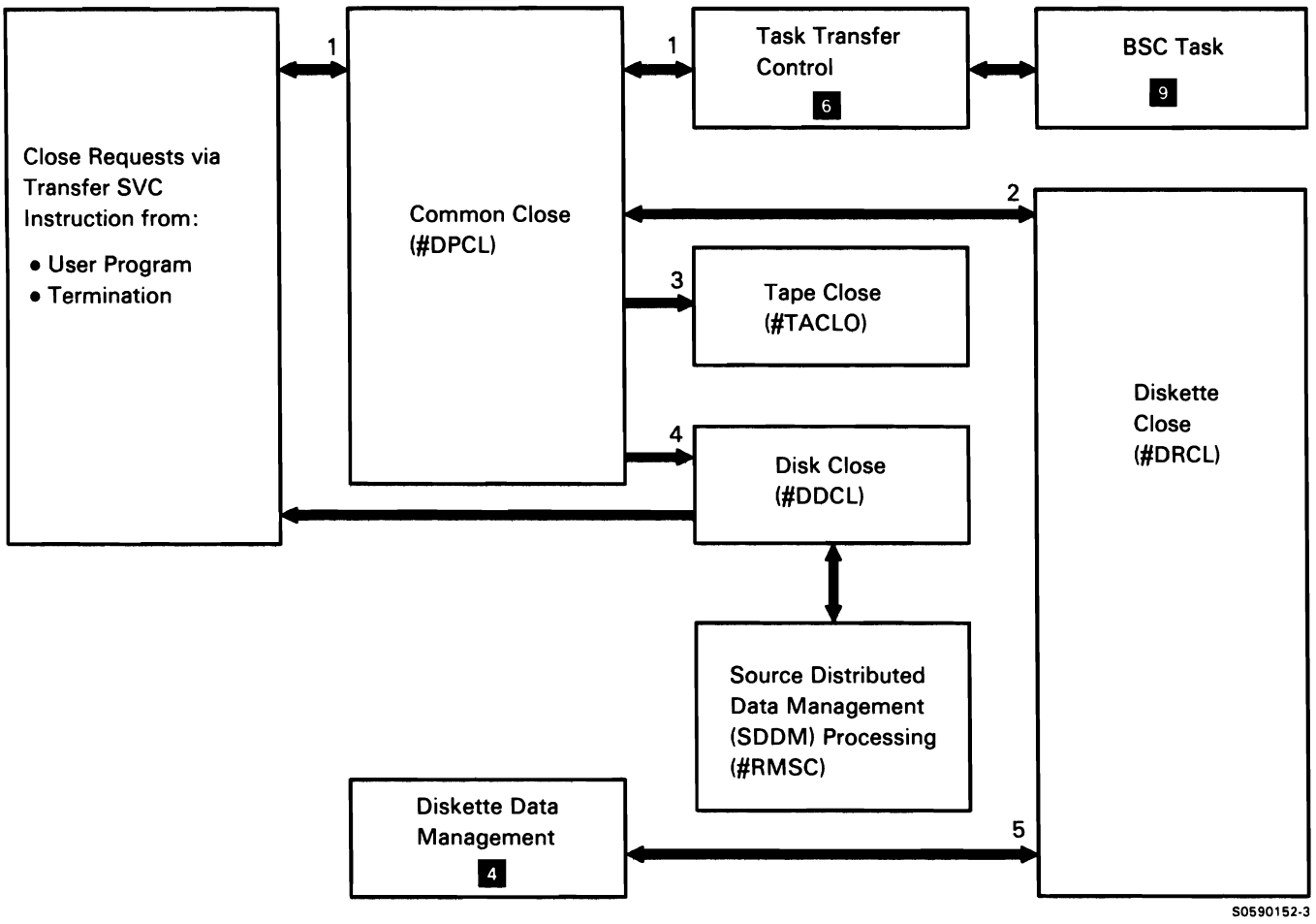


Chart 5.1.1 Close Control Flow

TERMINATION SUBFUNCTION

Note: Initial termination processing is performed by control storage nucleus task termination. This topic describes only termination processing performed in main storage.

The termination subfunction performs cleanup activity required to start a new job step or end the current job. Termination is called via the termination interface transient. It can be called in one of four ways:

- At step or job termination, the calling program invokes the termination interface transient via the transfer SVC instruction with an RIB.
- On an inquiry 2 or 3 option, or the CANCEL command, the command processor calls the termination interface transient.
- On an option 3 in response to a SYSLOG halt, SYSLOG calls the termination interface transient.
- On abnormal termination with a dump request, the control storage end-of-job transient ($EJ1) calls the termination interface transient.

The following termination overview processes are shown in Chart 5.2.0:

1 Perform initial termination processing:

   - If task has no JCB, route control to task detach.
   - If task has a JCB, route control to termination mainline for further processing.

2 Detach task.

3 Perform mainline processing, routing control where necessary.

4 Process for JCB termination, routing control where necessary.

5 At end-of-task cleanup, detach task.

6 At end of step, start next job step.

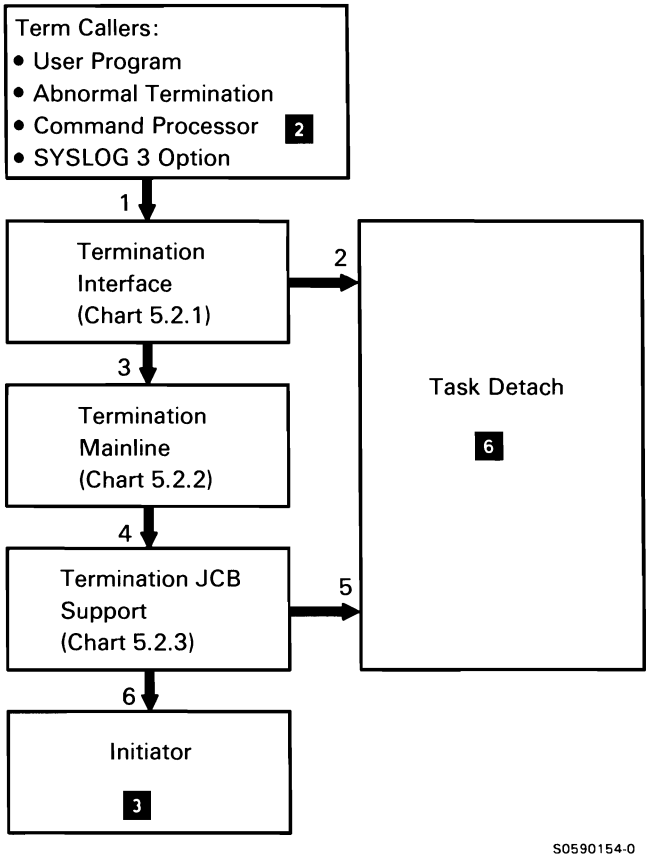


Chart 5.2.0 Termination Overview Control Flow

Termination Interface

The following termination interface processes are shown in Chart 5.2.1:

1 Perform initial termination interface processing:

   - If task has no JCB:
     - Route control to console cleanup to handle outstanding console messages.
     - Route control to error message handler to clean up outstanding error messages.
     - Route control to system measurement facility (SMF) to clean up SMF control blocks.
     - Route control to task detach.

   - If task has a JCB, perform other interface mainline processing, routing control where necessary.

2 Resolve cross-references between library modules.

3 Issue any required messages for control storage termination processing.

4 If applicable, perform cleanup processing on folder management services (FMS) files.

5 Close any remaining disk files.

6 Close any remaining files.

7 If applicable, perform cleanup on communications specifications block.

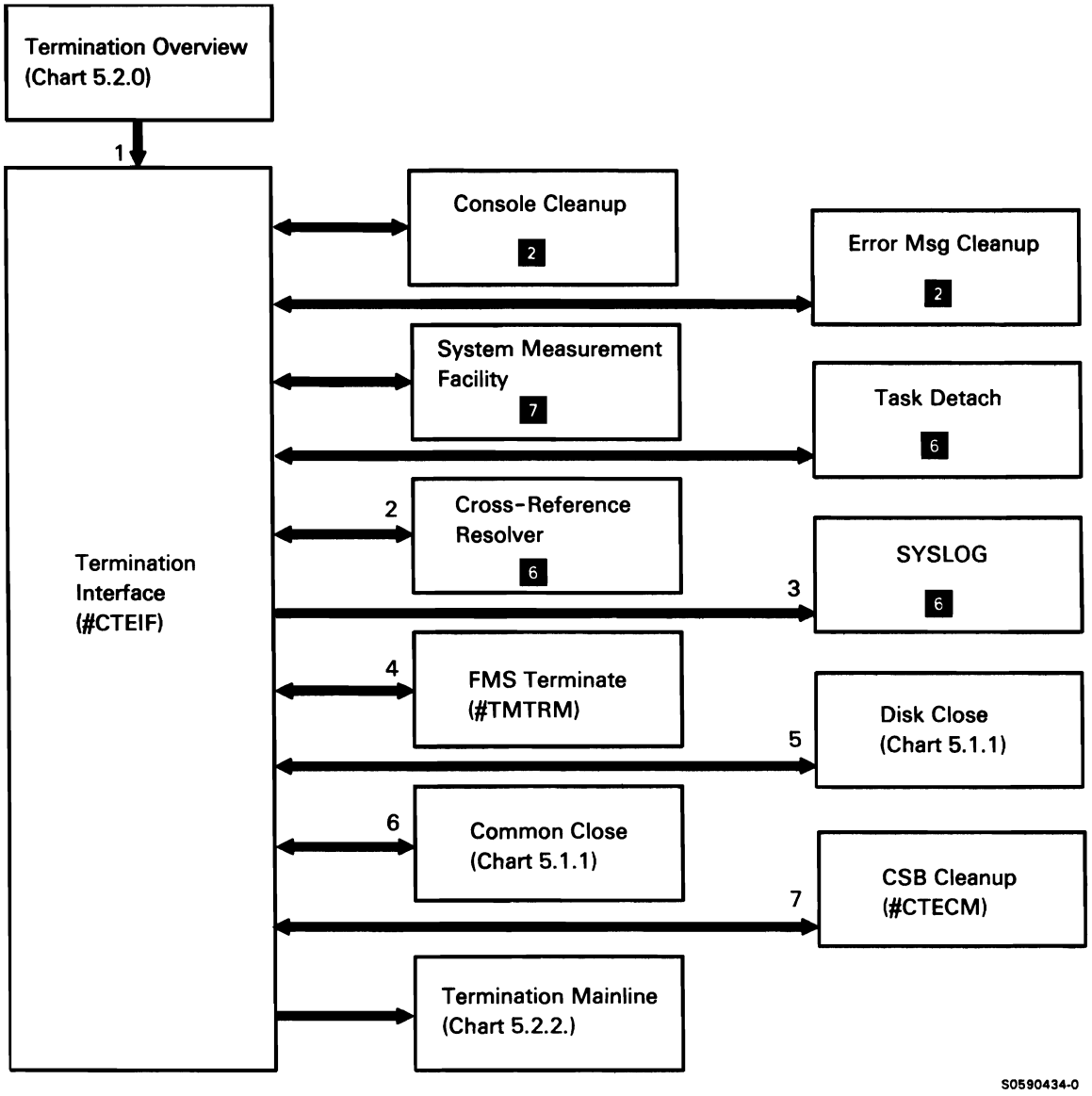


Chart 5.2.1 Termination Interface Control Flow

Termination Mainline

The following termination mainline processes are shown in Chart 5.2.2:

1 Perform termination mainline functions; and if necessary, keysort any disk for the following processes.

2 Perform cleanup on any outstanding console messages.

3 If applicable, perform cleanup processing on distributed data management (DDM) files, routing control for remote file delete, resource dequeue, and release (to release the session to the remote system).

4 Release remote files.

5 Perform cleanup on local files, routing control to disk VTOC access and SMF where applicable.

6 Perform any required processing of alternative indexed and text management services files, routing control to disk VTOC access and SMF where applicable.

7 Perform cleanup on procedure parameter save area (PPSA).

8 Perform any required library cleanup.

9 Perform cleanup for any outstanding error messages.

10 Perform cleanup on any associated spool files.

11 Perform any required cleanup on diskette file.

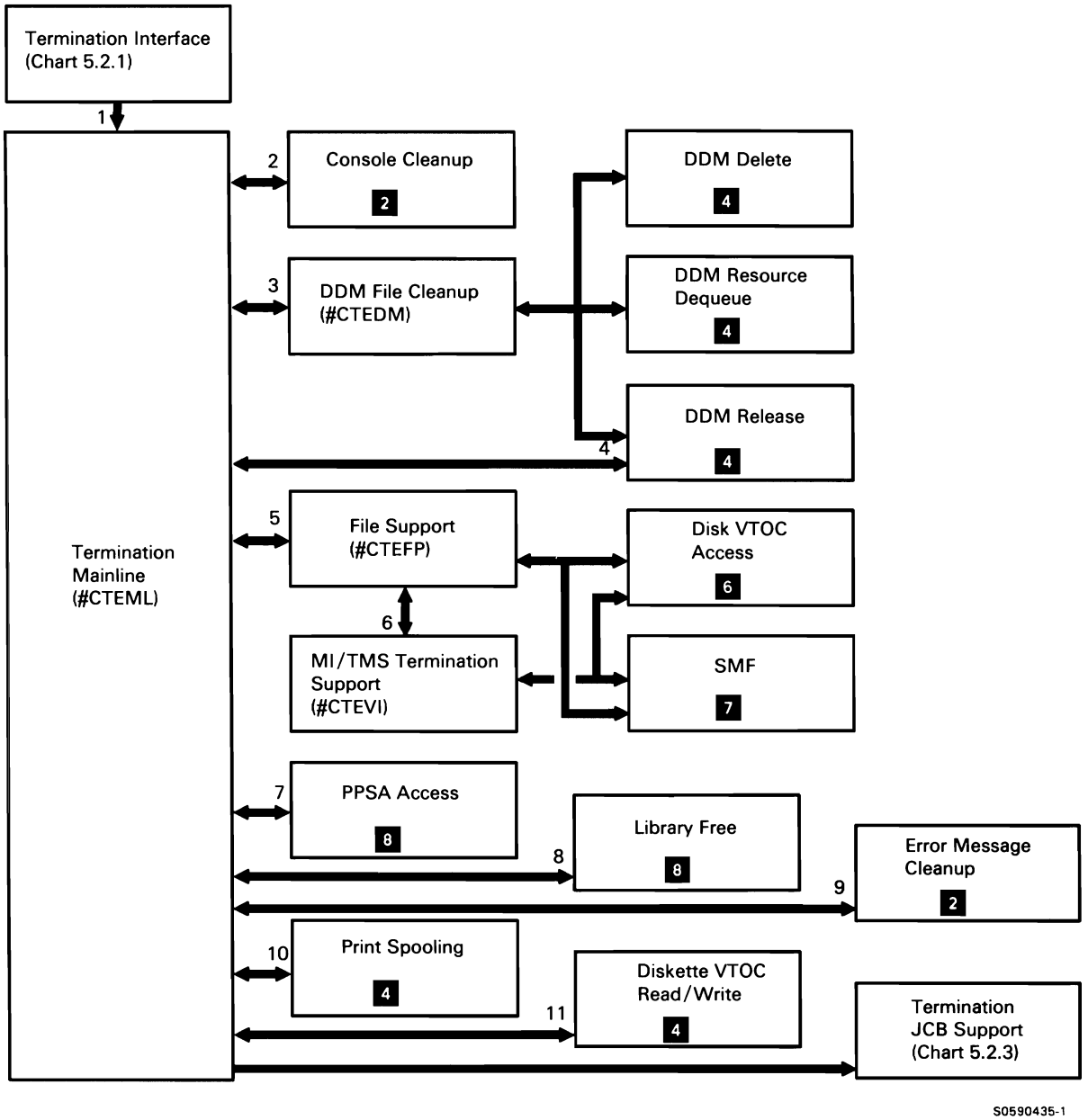


Chart 5.2.2 Termination Mainline Control Flow

Termination JCB Support

The following termination JCB support processes are shown in Chart 5.2.3:

1 Perform mainline functions for the following JCB termination processes.

2 Issue any required halts, messages, and return codes.

3 Issue end-of-SYSLIST-data message.

4 If applicable, clean up any outstanding session unit blocks (SUBs).

5 Process for display station command rejects.

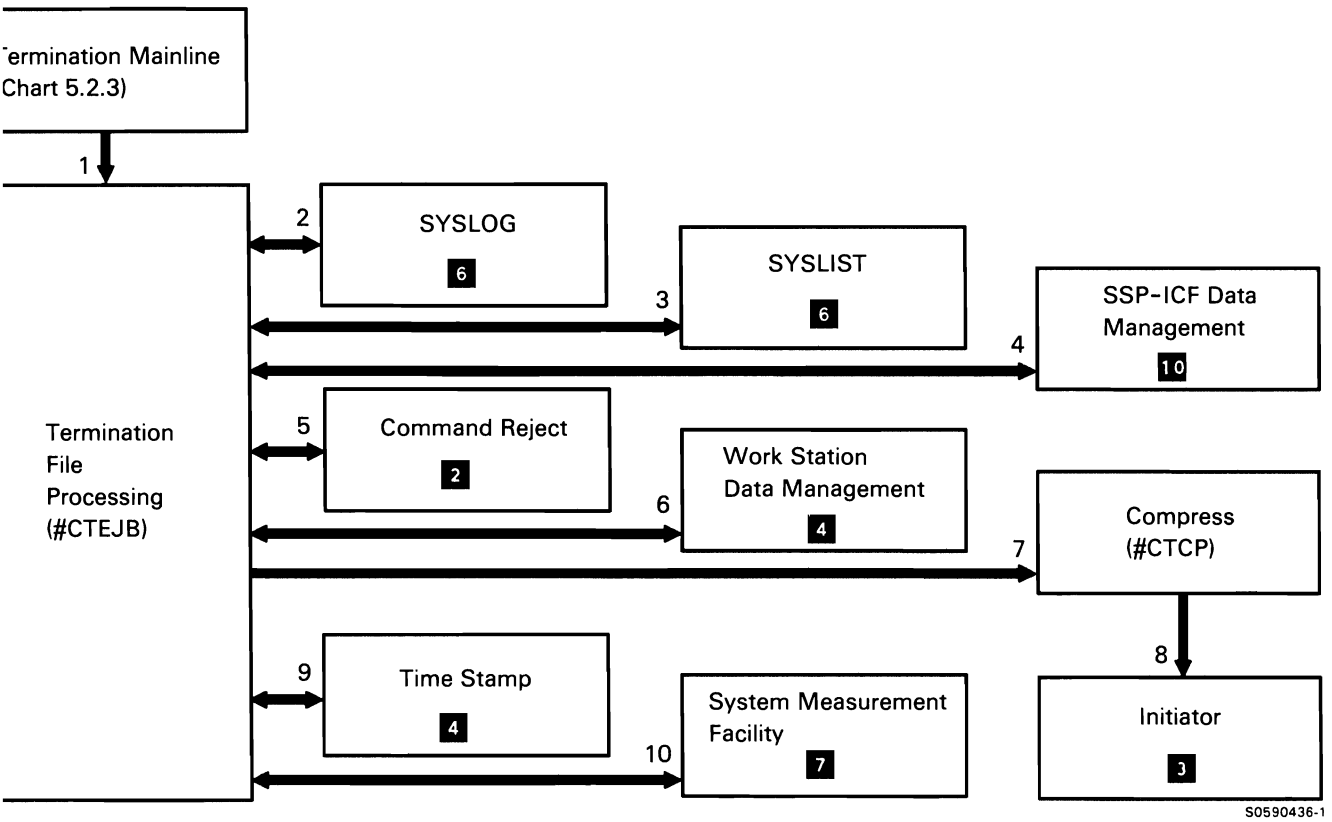
6 Release display stations associated with terminating task.

7 If required, compress disk.

8 Start the next job step.

9 Time stamp the job in the history file.

10 Perform system measurement facility cleanup.



art 5.2.3 Termination JCB Support Control Flow

System Services Function 6

The system services function consists of all SSP librarian facilities and miscellaneous services functions. It contains the following subfunctions:

- Librarian facilities — Chart 6.1.0
  - Find a library — Chart 6.1.1
  - Single name find — Chart 6.1.2
  - Librarian find — Chart 6.1.3
  - Source library get — Chart 6.1.4
  - Library record put — Chart 6.1.5
  - Library sector get/put — Chart 6.1.6
  - Library member delete — Chart 6.1.7
  - Library member protect — Chart 6.1.8
  - Library member change — Chart 6.1.9
  - Library access method — Chart 6.1.10
  - Library reallocate — Chart 6.1.11

- Miscellaneous services functions
  - Active format-1 area access — Chart 6.2.1
  - Cross-reference resolver — Chart 6.2.2
  - Build membership table — Chart 6.2.3
  - Disk VTOC access — Chart 6.2.4
  - Diskette VTOC read/write — Chart 6.2.5
  - Message retrieve — Chart 6.2.6
  - SYSIN — Chart 6.2.7
  - SYSLIST — Chart 6.2.8
  - SYSLOG — Chart 6.2.9
  - History file put — Chart 6.2.10
  - Supervisor task attach — Chart 6.2.11
  - Supervisor task detach/change origin point — Chart 6.2.12
  - Syntax checker — Chart 6.2.13
  - Information retrieval — Chart 6.2.14
  - Snap dump — Chart 6.2.15
  - Diskette data save/restore — Chart 6.2.16
  - Diskette magazine drive search — Chart 6.2.17

LIBRARIAN FACILITIES

The librarian facilities subfunction provides the user with accessability to members contained in the system library or in user libraries. The librarian routines are mainly used by SSP programming or by compiler-built programming, but they can also be invoked by the user. The librarian facilities subfunction consists of the following:

- *Find a library routine* locates a requested library.

- *Single name find routine* locates a library member in either the system library or the specified user library and optionally returns a loader parameter list.

- *Librarian find routine* locates library directory entries by type and full or partial name.

- *Source library get routine* gets records from either source or procedure members.

- *Library record put routine* places source or procedure records into a library.

- *Library sector get/put routine* gets members in sector mode from a library and puts members into a library in sector mode.

- *Library member delete routine* deletes a library member from a specified library.

- *Library member protect routine* prevents the simultaneous use of a library member by more than one task.

- *Library member change routine* changes a library member's name, subtype, and/or reference number.

- *Library access method routine* provides a single interface to many commonly-used librarian facilities:
  - Find a library
  - Member find
  - Member delete
  - Member rename
  - Member protection
  - Member get
  - Member put
  - Member copy

- Library reallocate routine reallocates a specified library that allows members located in an extent to be placed in the library.

Chart 6.1.0 shows the overview control flow for the librarian facilities subfunction:

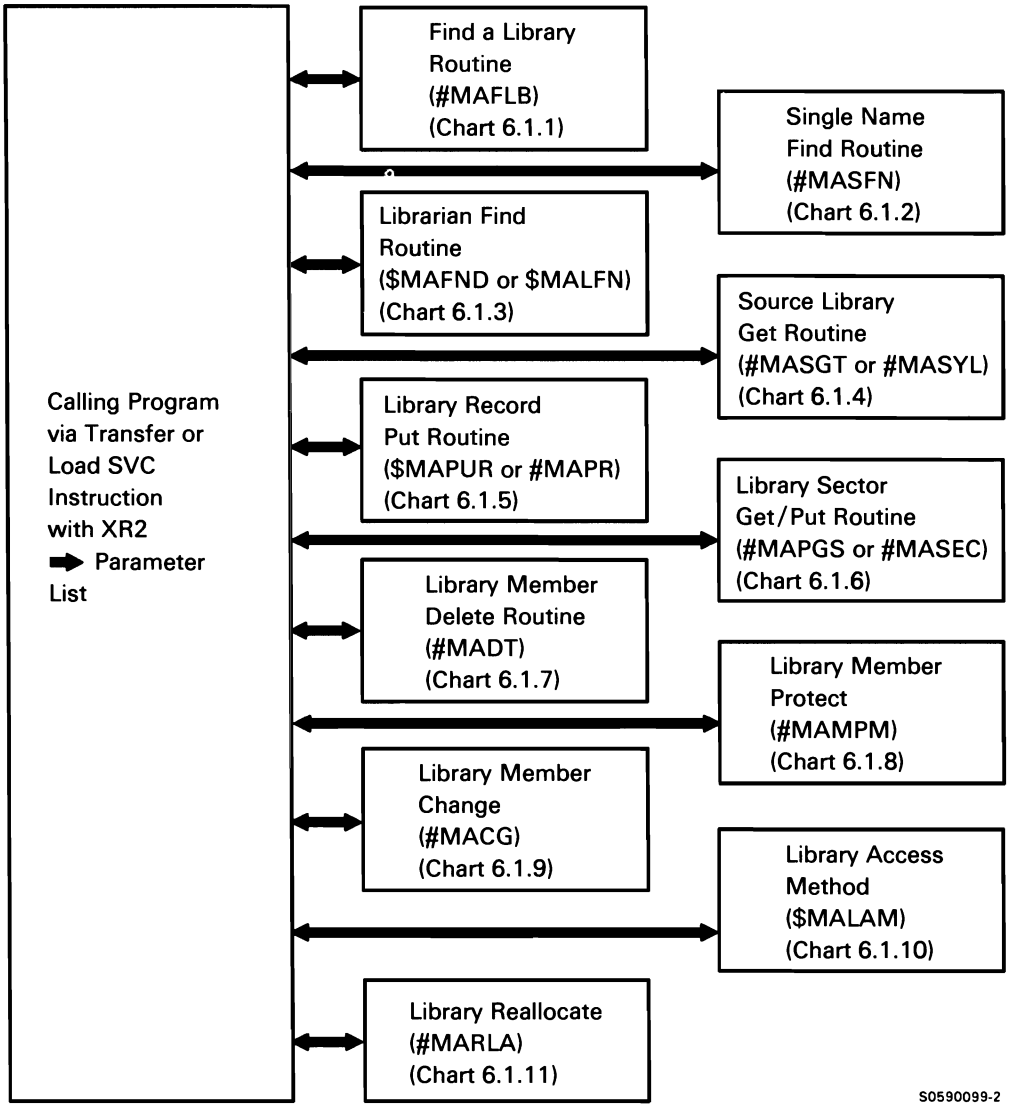


Chart 6.1.0 Librarian Facilities Overview Control Flow

Find a Library Routine

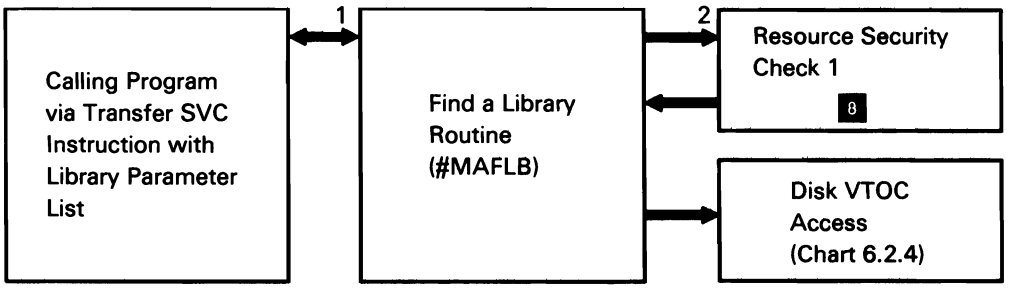
The find a library routine returns the address of the requested library format 1, when given an 8-byte library name. If the library cannot be located or is flagged as invalid, the routine returns an address of 0's. This routine is used when access is needed to members in libraries other than the designated user library. The format-1 address of a library is used by other librarian access routines.

The following find a library processes are shown in Chart 6.1.1:

1 Do the following:

- If requested library is #LIBRARY, get format-1 address from SCA.
- Find or build requested library format 1 in active format-1 area (AFA).
- Chain requested library format 1 to caller's JCB with a file specification block (FSB).

2 If security is active and the requested library is resource-protected, verify requester's authorization.



S0590294-0

Chart 6.1.1 Find a Library Routine Control Flow

Single Name Find Routine

The single name find routine finds a specified library member by searching first in the user library if there is one, then in the system library. Unless otherwise requested, the user can override the designated user library and specify any user library. The user can specify to search only the system library or only the user library. Single name find returns 17 bytes of loader information and a portion of the directory entry or a transfer parameter list and indicates if the member was found in the user library or the system library.

The following single name find processes are shown in Chart 6.1.2:

1 Enqueue library for duration of search.

2 Search user library directory if specified.

3 If necessary, search system library directory.

4 Process returned altered parameter list.

5 If requested member not found, process returned unaltered parameter list.

6 If requested member not found and build loader parameter list was specified, display name of member not found, and issue message.

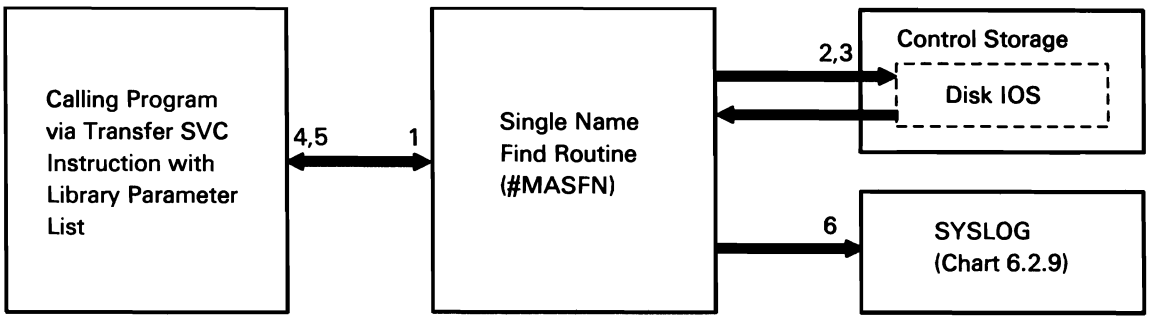


Chart 6.1.2 Single Name Find Routine Control Flow

Librarian Find Routine

The librarian find routine locates directory entries by type and full or partial name. The caller provides a buffer, specifies the library to search, gives the member types, and gives the member name or partial name and length. The librarian find routine returns the address of the member in the buffer of the next directory entry meeting the criteria or indicates that no more members meet the specified criteria.

Note: This routine exists in both a transient version ($MAFND) and a loadable version ($MALFN). Calling programs invoke $MAFND via the transfer SVC instruction or $MALFN via the load SVC instruction. Their functions are identical except that the transient version needs a 25-byte work area following the parameter list, and the partial name find is not supported.

The following librarian find processes are shown in Chart 6.1.3:

1 Determine library to search.

2 Enqueue library directory and scan for request; read directory into user's buffer.

3 Search for match on type and name.

4 Update parameter list for result of search.

5 Perform cleanup.

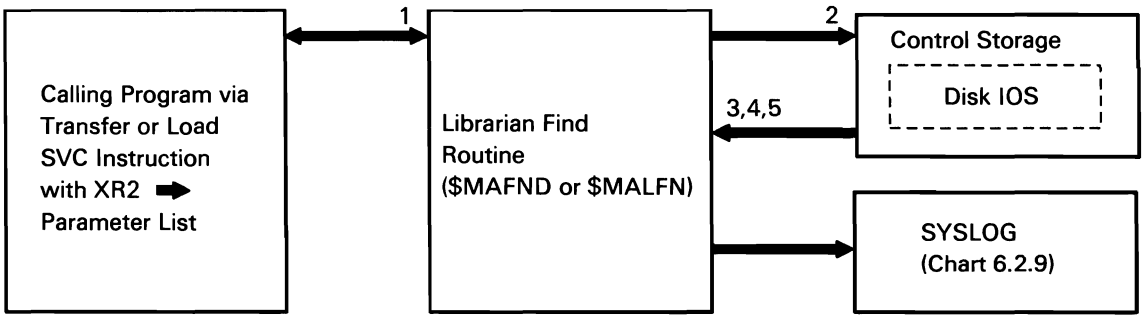


Chart 6.1.3 Librarian Find Routine Control Flow

Source Library Get Routine

The source library get routine finds either a source or procedure member, and moves the records, one at a time, from the member into the user's record buffer. When it transfers the records, any deleted blanks are padded back into the record.

Note: This routine exists in both a transient version (#MASGT) and an include version (#MASYL). Using programs invoke #MASGT via the transfer SVC instruction and #MASYL via the load SVC instruction or a branch instruction with XR1 pointing to a 70-byte work area. Their functions are identical except that the transient version requires a 16-byte work area following the parameter list, and the include version does not perform the find function.

The source library get processes are shown in Chart 6.1.4:

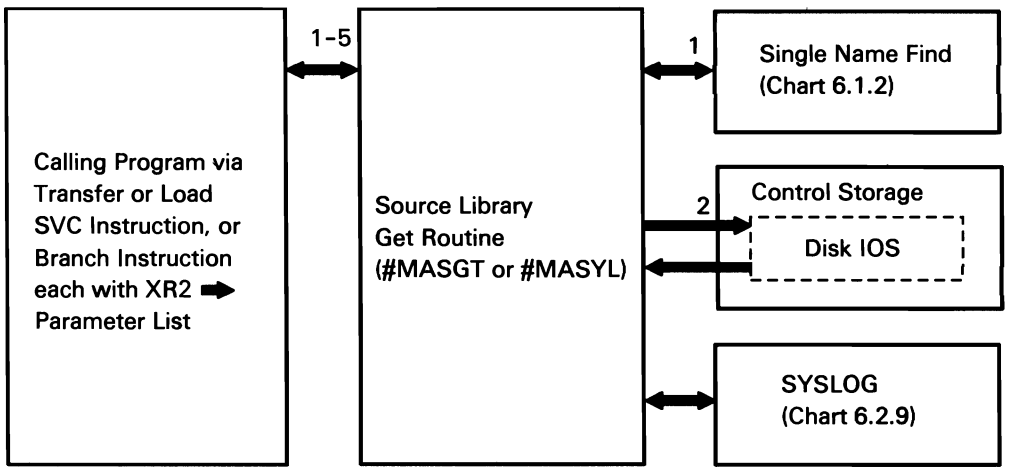
1 If find request, find the member (#MASYL).

2 If first request, read library member.

3 Expand blanks into record and place in user record buffer.

4 Update the parameter list.

5 If last record, set EOF indicator.



S0590297-0

Chart 6.1.4 Source Library Get Routine Control Flow

Library Record Put Routine

The library record put routine places records into a source or procedure library member. The routine compresses contiguous blanks using a hexadecimal counter in place of the blanks. Normal character strings are moved into the member intact, but are prefixed with a count byte.

Note: This routine exists in both a transient version (#MAPR) and a loadable version ($MAPUR). Calling programs invoke #MAPR via the transfer SVC instruction or $MAPUR via the load SVC instruction. Their functions are identical.

The following library record put processes are shown in Chart 6.1.5:

1 If open request, open the library.

2 Put source or procedure records, in compressed format, into the buffer.

3 Write the buffer to the library and update the number of available sectors in the LCS.

4 If close request, close the library.

5 Return to caller.

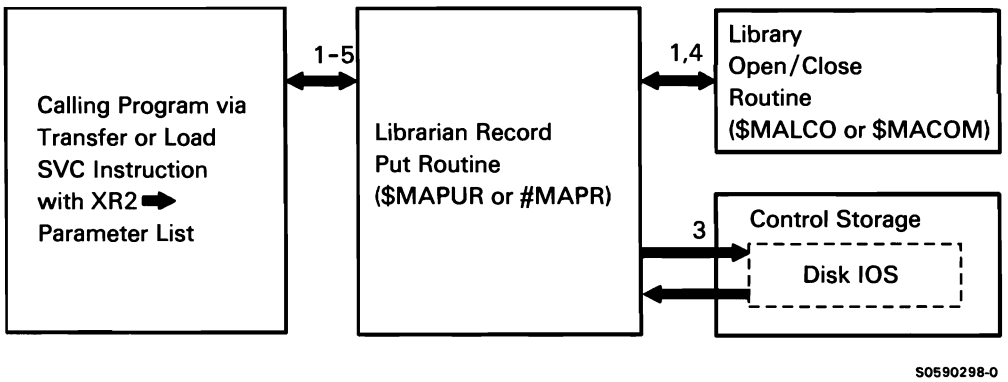


Chart 6.1.5 Library Record Put Routine Control Flow

Library Sector Get/Put Routine

The library sector get/put routine can get members, in sector mode, from a library and pass them via a buffer to the requester or put members into a library via a caller-supplied buffer. It also handles library open requests and performs a forced close of the library if a get or put is done on a complete library member.

Note: This routine exists in both a transient version (#MASEC) and a loadable version ($MAPGS). Calling programs invoke #MASEC via the transfer SVC instruction or $MAPGS via the load SVC instruction. Their functions are identical.

The library sector get/put processes are shown in Chart 6.1.6:

1 If find/get request, find the member.

2 If get request, retrieve member from the library.

3 If get request and there are PTFs applied, get PTF information.

4 If put request, open the library.

5 Put the member into the library.

6 Close the library if necessary.

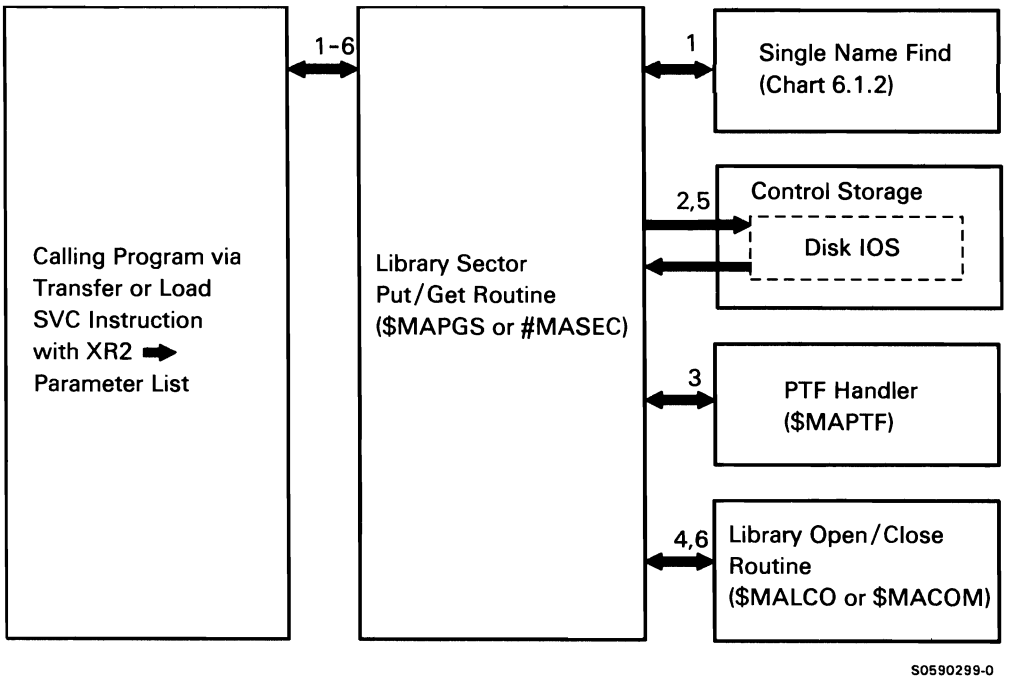


Chart 6.1.6 Library Sector Get/Put Routine Control Flow

Library Member Delete Routine

The library member delete routine deletes non-IBM-supplied members by name and specified library. It performs the delete by setting the total number of sectors field to 0, then compacting the directory. The library member delete routine is invoked via a transfer SVC instruction with a delete parameter list that specifies the member name, the library member type, and the format-1 address of the library.

The following library member delete processes are shown in Chart 6.1.7:

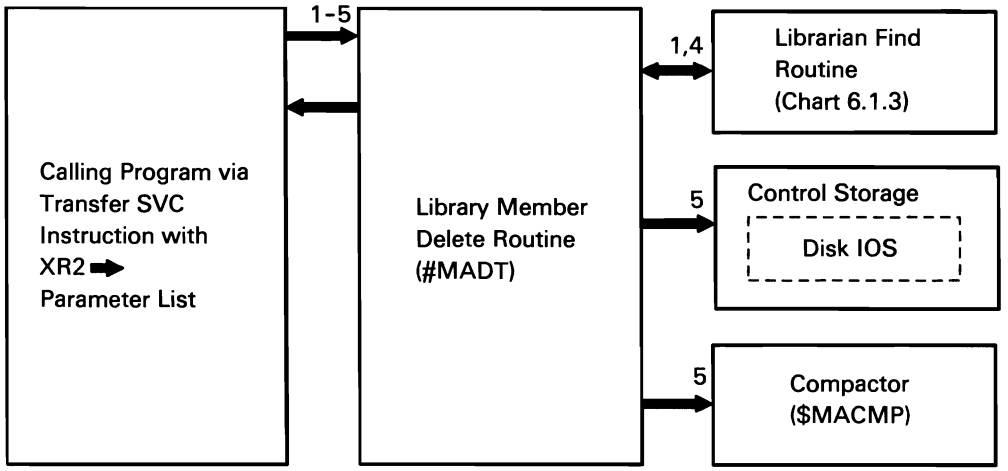
1 Find the specified member.

3 Update the member directory entry(s) to show zero total sectors.

4 Write the directory back.

5 Set switch in library control sector (LCS) to compact directory.

Note: The compactor is shown here to illustrate the entire member delete process; control flows to the compactor only as a result of the LCS switch that was set by #MADT.



S0590300-0

Chart 6.1.7 Library Member Delete Routine Control Flow

Library Member Protect Routine

The library member protect routine can build a new member chain element and add it to the chain, or it can release an element from the chain. The library member protect routine is invoked via a transfer SVC instruction with a member protect parameter list that includes member type, member name, library format-1 address, and an operation byte.

The following library member protect processes are shown in Chart 6.1.8:

1 Search the member protect chain for a matching element.

2 If match not found and protect was requested, add the member to the chain.

3 If match found and release was requested, release the member from the chain.

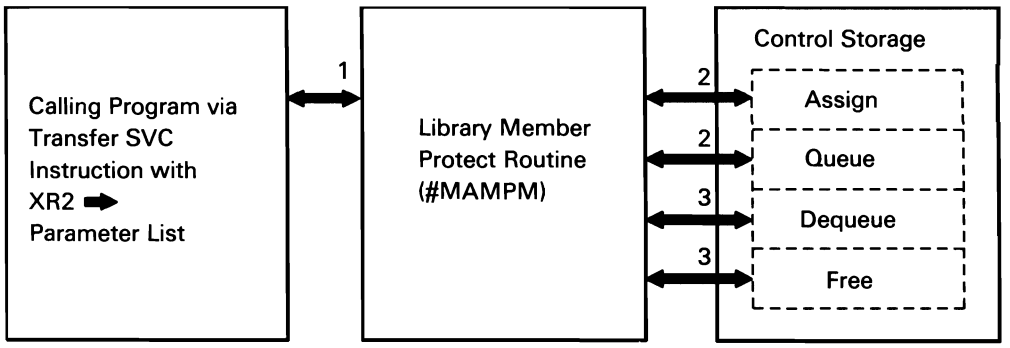


Chart 6.1.8 Library Member Protect Routine Control Flow

Library Member Change Routine

The library member change routine can be used to change the name, subtype, and/or reference number of a library member. The library member change routine is invoked via a transfer SVC instruction with a member change parameter list that includes the library format-1 address, the member type, the member name, the member new name, the member subtype, the member reference number, and an operation byte.

The following library member change processes are shown in Chart 6.1.9:

1 If change of name is not requested, find member(s) to be changed.

2 Ensure this library member is not owned by another task.

3 If requested, change the name, subtype and/or reference number in the directory entry buffer.

4 Write the updated directory entry back to disk.

5 If a change of name was requested, make any previously-deleted directory entries available.

6 Find member to be changed.

7 Ensure this member is not owned by another task.

8 Write the directory entry back to disk (the entry was previously marked as a deleted entry).

9 Write the updated directory entry from the buffer to disk.

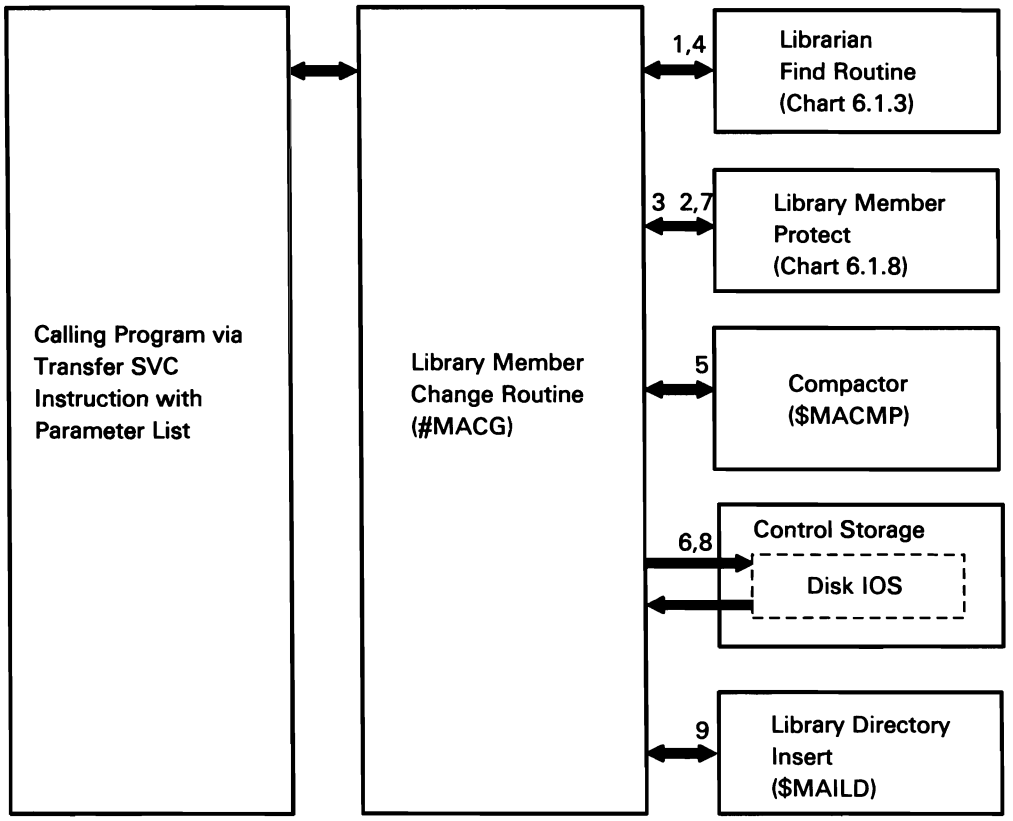


Chart 6.1.9 Library Member Change Routine Control Flow

Library Access Method Routine

The library access method routine provides one user interface to the following librarian facilities:

- Find a library
- Member find
- Member delete
- Member rename
- Member protect
- Member get (by records/sectors)
- Member put (by records/sectors)
- Member copy
- Directory list
- Library reallocate

With one call to the library access method routine, the requester can invoke combinations of the functions provided by these facilities. The library access method routine is invoked via a transfer SVC instruction with a library access parameter list. The parameter list includes an operation byte, library name, member type, member name, and new member name.

The following library access method processes are shown in Chart 6.1.10:

1 Find the library.

2 If operation is member find, member get, or member copy, find the member.

3 If operation is delete, delete the member.

4 If operation is rename, change the member name.

5 If operation is protect, get, or copy, protect the member.

6 If operation is get on source or procedure member, do source get.

7 If operation is get on load or subroutine member, do sector get.

8 If operation is put on source or procedure member, do record put.

9 If operation is put on load or subroutine member, do sector put.

10 If operation is copy, do sector get, then sector put.

11 If operation is directory list, call 105 to scan and read the directory.

12 If operation is library reallocate, reallocate the library.

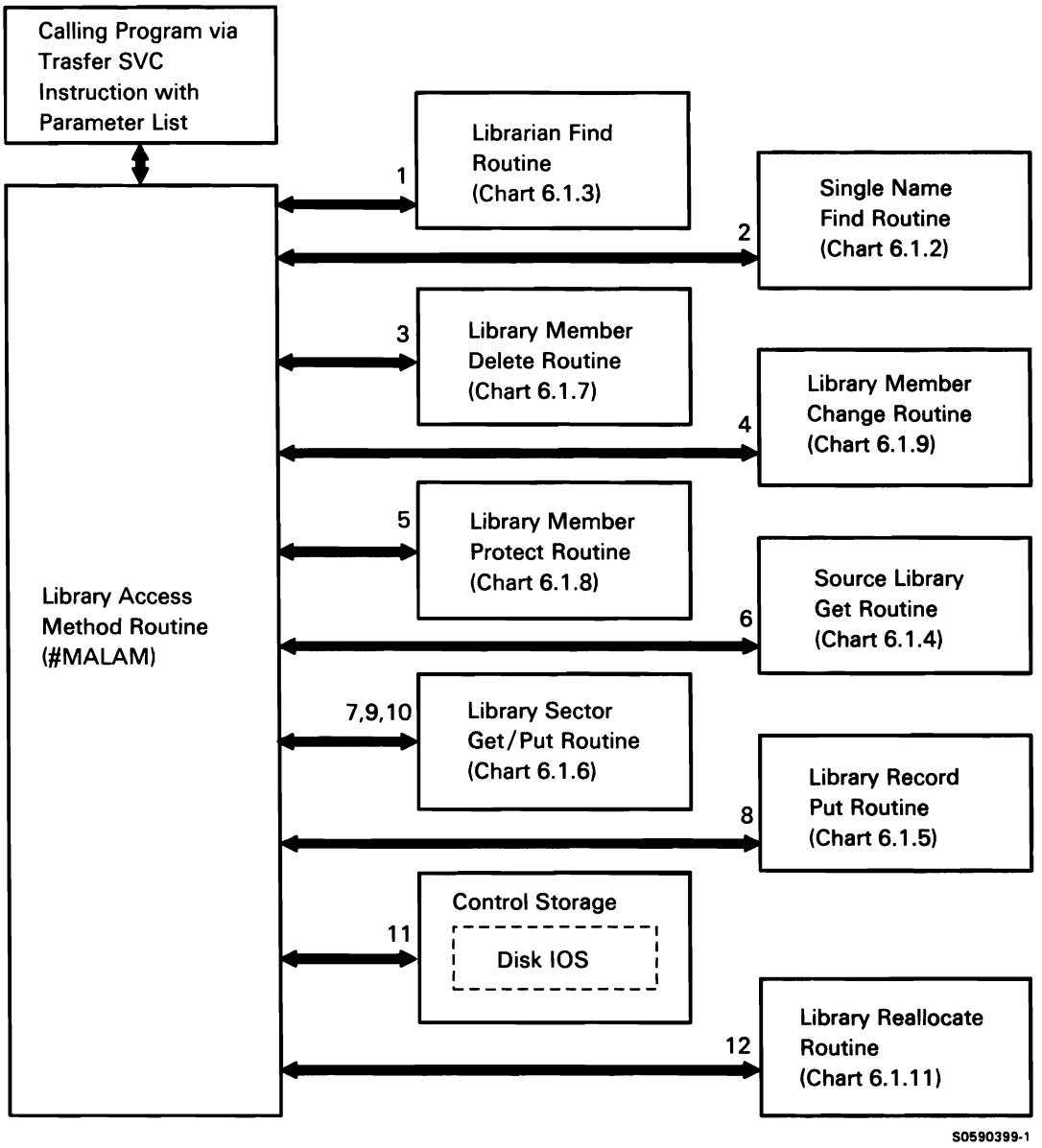


Chart 6.1.10 Library Access Method Routine Control Flow

Library Reallocate Routine

The library reallocate routine reallocates a specified library to allow members located in an extent to be placed in the library. The user can specify new library and directory sizes for the library to be reallocated, even if an extent does not exist. The library reallocate routine is evoked by the ALOCLIBR procedure command via library maintenance mainline.

The following reader to library copy processes are shown in Chart 6.1.11:

1 Resource enqueue the library.

2 Read the library control sector (LCS).

3 Allocate disk space to move the library.

4 Copy the directory, the members and, if it exists, the file extent to the allocated disk space.

5 Update the LCS.

6 Fill any unused directory sectors with hex FF.

7 Update the directory for any members moved from the file extent.

8 Update the library format 1 and the extent format 1 and write them back to the VTOC.

9 Zero the old library and its extent.

10 Deallocate the old library and its extent.

11 Return to mainline.

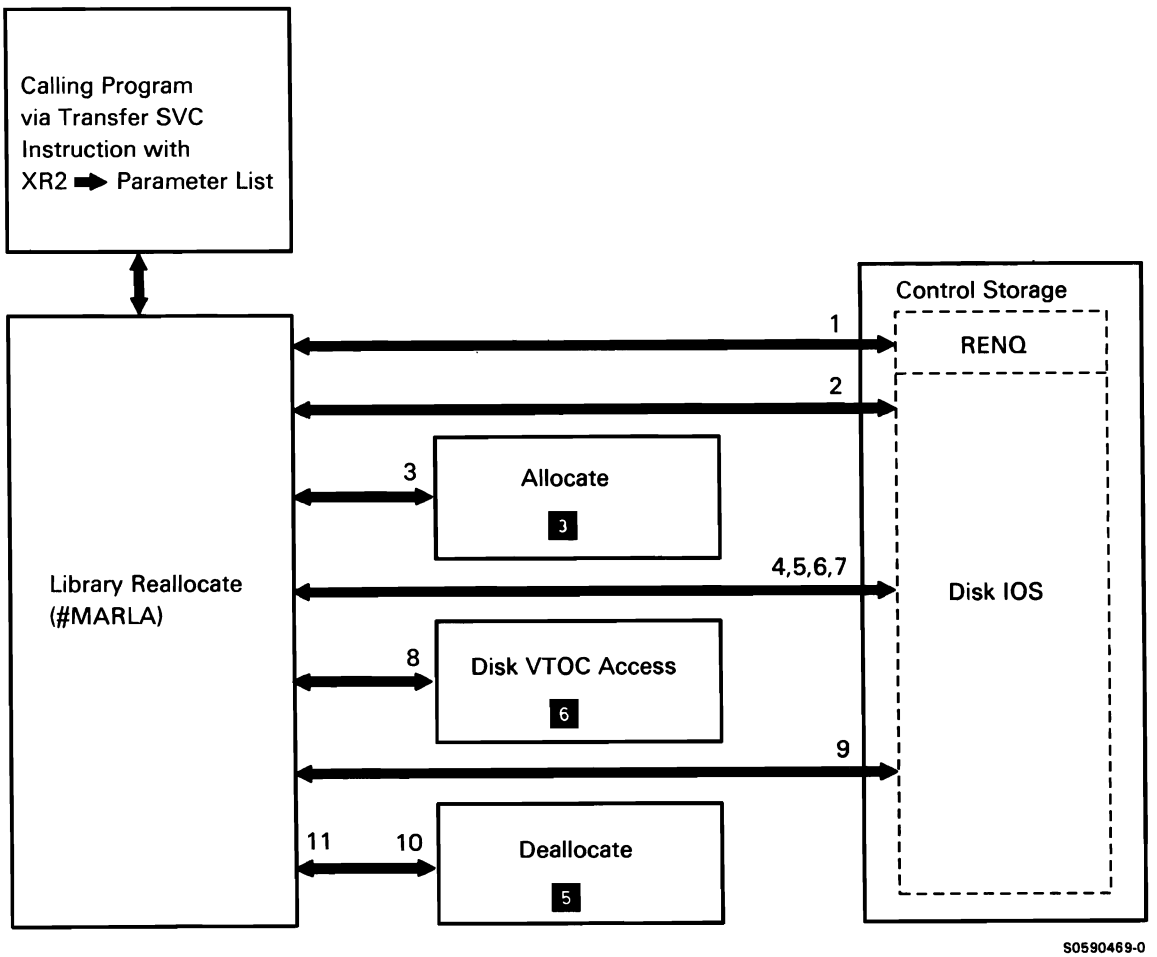


Chart 6.1.11 Library Reallocate Routine Control Flow

This page is intentionally left blank.

MISCELLANEOUS SERVICE FUNCTIONS

The miscellaneous service functions are for the most part, subroutine-type functions provided by SSP to privileged and nonprivileged users. Except where otherwise noted, privileged and nonprivileged users invoke their supervisory functions via transfer SVC instructions accompanied by RIBs.

This function includes the following utilities:

- Active format-1 area access — Chart 6.2.1
- Cross-reference resolver — Chart 6.2.2
- Build membership table — Chart 6.2.3
- Disk VTOC access — Chart 6.2.4
- Diskette VTOC read/write — Chart 6.2.5
- Message retrieve — Chart 6.2.6
- SYSIN — Chart 6.2.7
- SYSLIST — Chart 6.2.8
- SYSLOG — Chart 6.2.9
- History file put — Chart 6.2.10
- Supervisor task attach — Chart 6.2.11
- Supervisor task detach — Chart 6.2.12
- Syntax checker — Chart 6.2.13
- Information retrieval — Chart 6.2.14
- Snap dump — Chart 6.2.15
- Diskette data save/restore — Chart 6.2.16
- Diskette magazine drive search — Chart 6.2.17

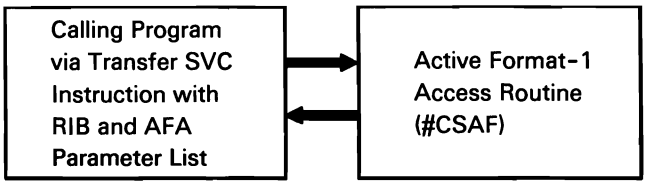
Active Format-1 Area Access

The active format-1 area access routine processes requests for gets and puts of format-1 blocks to the active format-1 area (AFA). The user can submit the following requests via the AFA access parameter list:

- Get by label
- Get by name (optionally returns file specification block (FSB) and its address)
- Get by address
- Put by address

Active format-1 area access processes are listed below:

- Determine the function requested:
  - If get by label for disk file, find format 1 in AFA chain and move it to caller's I/O area.
  - If get by label for diskette file, scan file specification block (FSB) chain pointed to by JCB; if format 1 contains the correct unit, label, and (optional) date and ID, place format-1 address in caller's parameter list and move format 1 to the caller's I/O area.
  - If get by name, search the FSB chain for format 1; if format 1 contains the correct unit, place the format-1 address in the caller's parameter list and move the format 1 to the caller's I/O area.
  - If get by address, move format 1 at specified address to the caller's I/O area.
  - If put, move format 1 in caller's I/O area to the AFA specified in the parameter list.
  - Update the parameter list return code and return to caller.



S0590302-01

Chart 6.2.1 Active Format-1 Area Access Control Flow

Cross-Reference Resolver

The cross-reference resolver modifies where-to-go (WTG) tables that are referred to by privileged IBM modules when they call another module. The WTG tables allow a calling module to omit the process of calling the system find routine, before calling the required module. The cross-reference resolver is run any time IBM load members in the system library are moved. This may be after an IBM load, library condense, member delete, IPL, or an IBM load member copy to the library.

In addition to modifying the WTG tables, the cross-reference resolver also fills in display format index tables in IBM modules and updates the transfer table in main storage.

The following cross-reference resolver processes are shown in Chart 6.2.2:

1 Do the following, routing control where necessary:

- Read library format 1's.
- Build a library table of all IBM load modules including characters 2 through 5 of the name, the disk address, the number of text sectors, the module attributes, the RLD displacement, and WTG and format index table (FIT) indicators.
- Build a main storage format index table (FIT) from the index sectors of the command processor screen format modules.
- Read 6 sectors of the WTG table for each module with a WTG or FIT using the library table.
- For each module with a WTG table and/or a FIT, either search the library table for a matching entry (WTG processing) or use the FIT displacement to get information from the main storage format index for the index table being updated (FIT processing).
- If an entry is not found in the library table (WTG) or the format is not found (FIT), insert the no-op screen and issue an error.
- Write the updated 6 sectors of the applicable modules back to their original locations on disk.

2 Get library format 1's.

3 Read WTG table sectors as required.

4 Issue no-op errors as required.

5 Issue messages as required.

6 Write updated WTG table sectors back to their original locations.

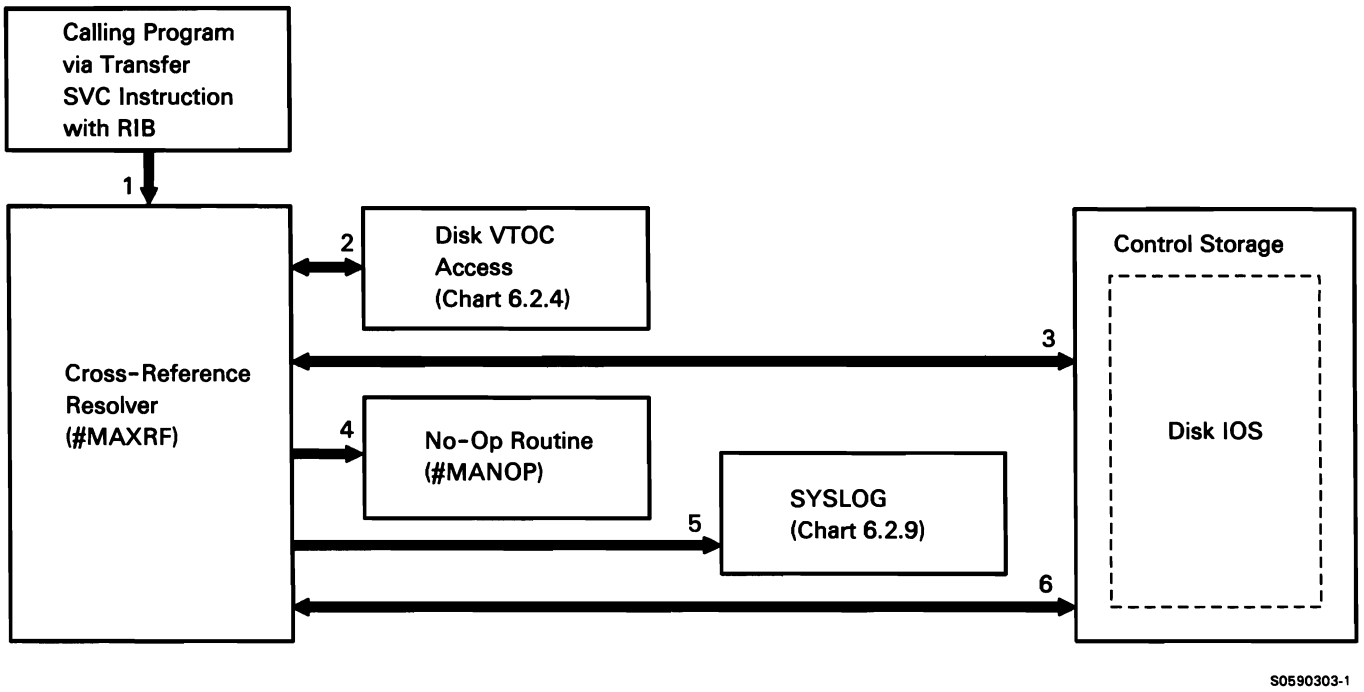


Chart 6.2.2 Cross-Reference Resolver Control Flow

Build Membership Table

The build membership table transient passes a table containing the printable characters on a particular printer to the caller. It is invoked via a transfer SVC instruction by the following:

- The command processor when it builds a STATUS SESSION display.
- $MAINT when printing a load member.
- $COPY when printing the contents of a file.

Build membership table processes are listed below:

- Determine the type of printer and its image/character set:
  - If called by command processor, point to work station WSWA and read printer specification table.
  - Find TUB; if TUB is not found and remote work stations are active, process remote work station printer.
  - If printer is not a 3262, process image for nonimage printer.
  - Read printer specification table (PST).
  - If the PST indicates the image is inline, process it.
- Process image source member.
- Return to caller.

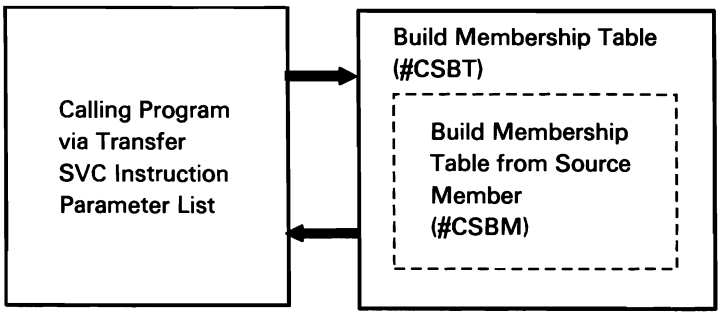


Chart 6.2.3 Build Membership Table Control Flow

Disk VTOC Access

The disk VTOC access routine performs three main functions:

- Format-1 read
- Format-1 write
- Existence test

Disk VTOC access is called via a transfer SVC instruction with XR2 pointing to a parameter list.

Disk VTOC access processes are listed below:

- Determine the function requested.
- Process format-1 read request:
  - Scan VTOC for requested format 1.
  - If found, put format-1 sector displacement in parameter list and move format 1 to caller's I/O area.
  - If not found and if caller allows remote file access, call distributed data management (DDM) extract to search network resources directory (NRD).
- Process format-1 write request:
  - Build IOB for disk IOS to read sector specified in parameter list.
  - Move format 1 from caller I/O area into sector just read and direct disk IOS to write sector back to disk.
  - Return to caller.
- Process format-1 existence test:
  - Set date indicator if requested and direct disk IOS to scan.
  - If format 1 found, update the parameter list return code and return to caller.
  - If format 1 not found and if caller allows remote file access, call distributed data management (DDM) extract to search network resources directory (NRD).

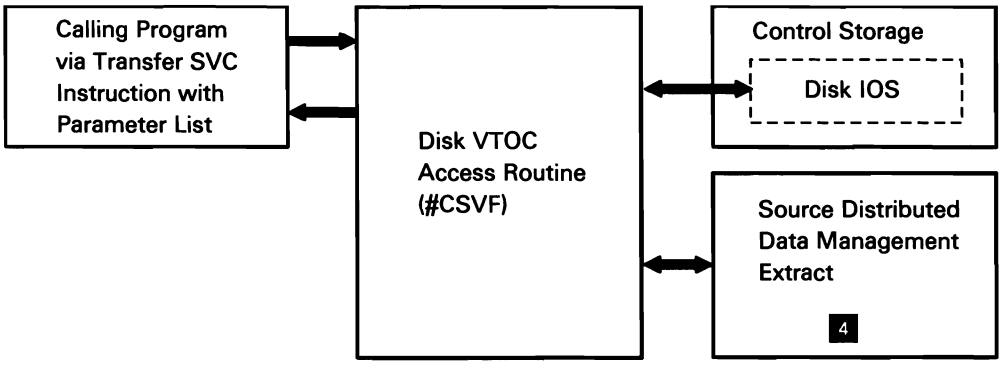


Chart 6.2.4 Disk VTOC Access Control Flow

Diskette VTOC Read/Write

Diskette VTOC read/write performs the following functions:

- Prepare the diskette
- Read/write format 1
- Convert format 1's to header 1's
- Orient autoloader device carriage

Diskette VTOC read/write is called via a transfer SVC instruction and a parameter list pointed to by XR2.

Diskette VTOC read/write processes are listed below:

- Determine the function requested.
- Prepare diskette:
  - If autoloader diskette drive, select diskette.
  - Read and verify the volume label and place volume label and physical attribute indicators in the SCA.
  - Create and write the lock sector.
  - Convert header 1's into format 1's and write to diskette VTOC area on disk.
- Process format-1 read or existence test:
  - Read format 1 from the diskette VTOC on disk.
  - If read next, use parameter list to direct disk IOS in VTOC format-1 area scan.
  - If read next-same label, set up scan mask for disk IOS.
  - If requested format 1 not found, set parameter list return code and return to caller; otherwise, place requested format-1 sector/displacement in parameter list.
  - Unless existence test, move format 1 to caller's I/O area.
- Process format-1 write request:
  - Build IOB for disk IOS to read sector specified in parameter list.
  - Move format 1 from caller I/O area into sector just read and direct disk IOS to write sector back to disk.
  - Return to caller.
- Convert format 1's to header 1's:
  - Verify lock sector.
  - Convert format 1's in diskette VTOC area on disk back to header 1's and write back to diskette.
- Return autoload device carriage to home position.

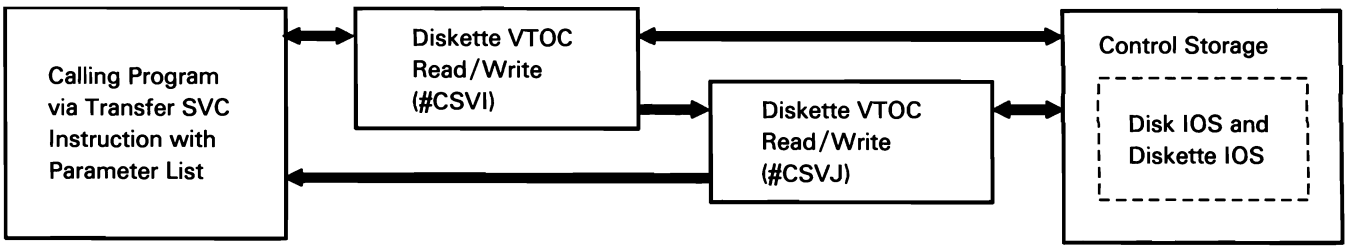


Chart 6.2.5 Diskette VTOC Read/Write Control Flow

Message Retrieve

The message retrieve routine locates the message text for an MIC (message identification code) passed by the calling program. It ensures a valid message retrieve by checking the parameter list for valid indicators and the message member pointer in the appropriate communications region for a nonzero disk sector address. The message retrieve routine is invoked via the transfer SVC instruction with XR2 pointing to the parameter list.

Message retrieve processes are listed below:

- Perform the following for the request:
  - Check message retrieve parameter list for valid indicators.
  - Check message member pointers in appropriate communications region for a nonzero disk sector address (SSN).
- Direct disk IOS to scan message member for sector identified by MIC greater than or equal to the requested MIC.
- Find desired message by reading message member sector.
- Place message in caller's buffer and message length in parameter list.
- If error detected, return error MIC in parameter list.
- Return to caller.

Chart 6.2.6 Message Retrieve Control Flow

SYSIN

SYSIN retrieves records from either the keyboard or from a library procedure member. SYSIN performs substitution and procedure control expression processing. SYSIN is called via a transfer SVC instruction with RIB, and XR2 pointing to the parameter list. The primary users of SYSIN are the initiator (Chart 3.1.1) and the syntax checker (Chart 6.2.14) subfunctions.

The following SYSIN processes are shown in Chart 6.2.7:

1 Perform the following mainline processing, routing where necessary:

- Get statements from procedure (#MASYL).
- If procedure control expression processing is required:
  - Perform // IF ACTIVE existence testing.
  - Perform blocks existence testing.
  - Perform file, switch, member, and volume ID existence testing.
  - Route for IF testing for IF, CANCEL, ELSE, EVALUATE, GOTO, TAG, RESET, and RETURN statements.
  - Perform INQUIRY, JOBQ, DISPLAY, MRTMAX, CONSOLE, and DSPLY, ENABLED, LISTDONE, SECURITY, and EVOKED testing.
- Set return code and return to calling program.

2 Get input from keyboard and place in user buffer.

3 Process all errors detected by SYSIN.

4 Perform detail statement processing.

5 Perform detail statement processing.

6 Perform detail statement processing.

7 Perform detail statement processing.

8 Write records to history file if logging is specifed by LOG statement.

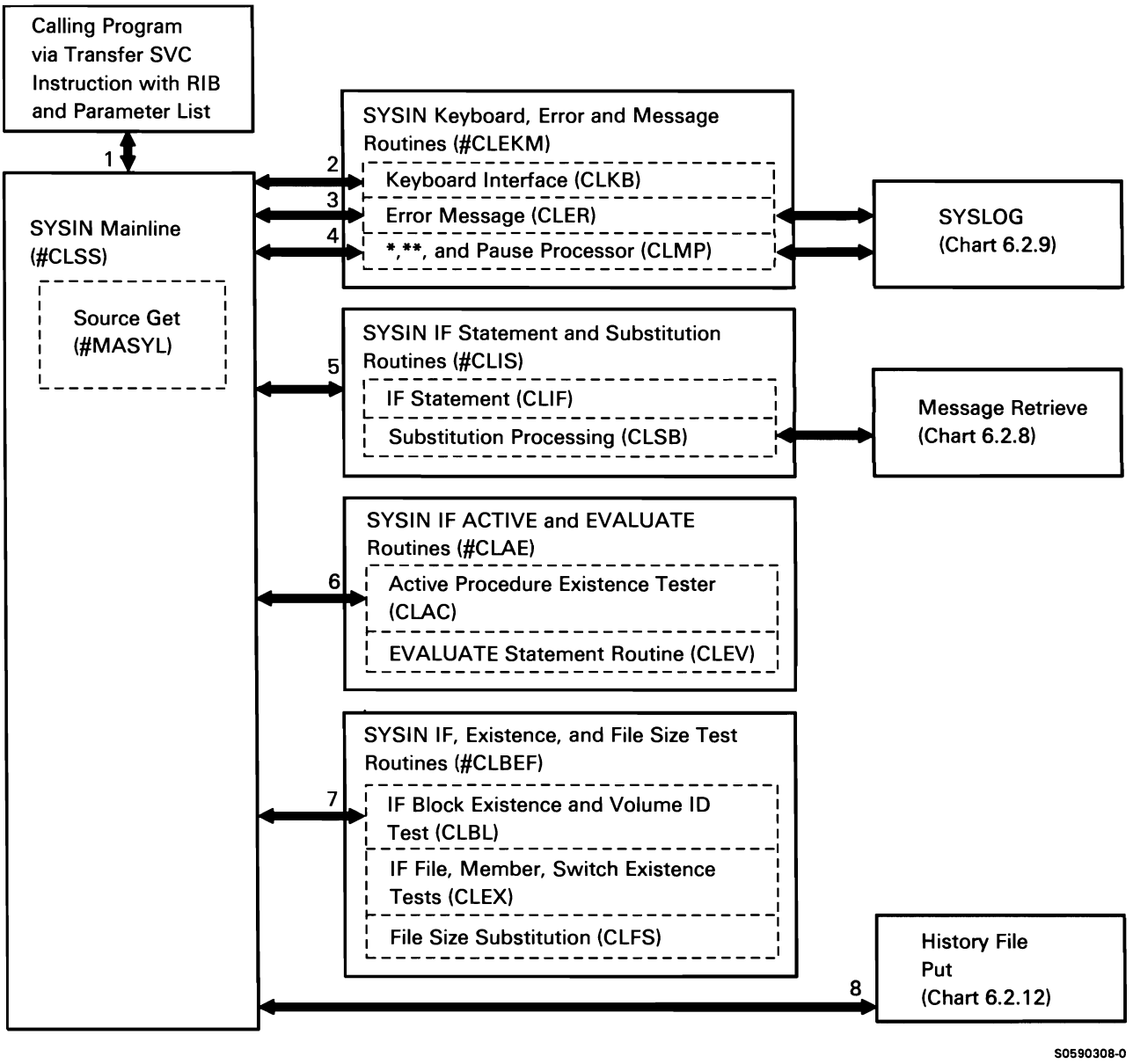


Chart 6.2.7 SYSIN Control Flow

SYSLIST

SYSLIST provides a means of printing or displaying system output to the user. SYSLIST processes two types of system output: Type 1 comes from a message member, and Type 2 comes from a system program. SYSLIST is called via a transfer SVC instruction with RIB, and XR2 pointing to a parameter list. The primary users of SYSLIST are IBM utilities and IBM program products.

The following SYSLIST processes are shown in Chart 6.2.8:

1 Perform the following mainline processing, routing where necessary:

- If Type 1 output (from message member), check parameter list for message member to use, get message from member and place it in SYSLIST buffer.
- If Type 2 output (from system program), use data supplied by caller.
- If SYSLIST device is printer, allocate printer and build buffer, and print message.

2 If SYSLIST device is display (CRT), build the work station data management parameter list and display the message.

3 Get message from appropriate message member.

4 Allocate printer if applicable.

5 Print message.

6 Issue error messages.

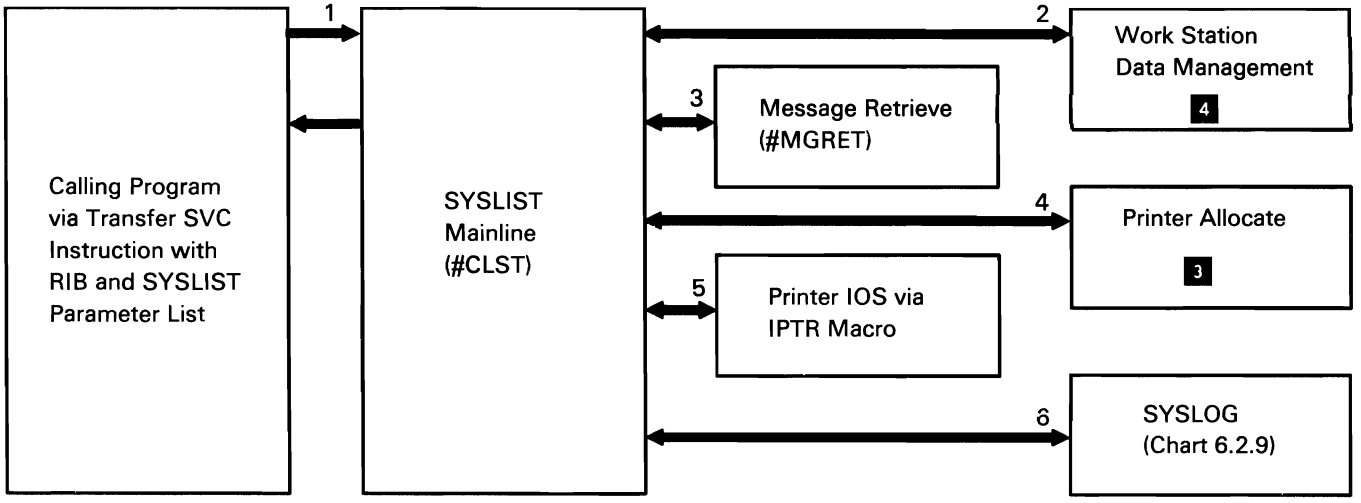


Chart 6.2.8 SYSLIST Control Flow

This page is intentionally left blank.

SYSLOG

SYSLOG provides a method of printing or displaying messages. SYSLOG processes six types of messages:

- Type 1: Message member messages not requiring a response and, optionally, with variable-length data inserted
- Type 1R: Message member messages requiring a response
- Type 2: User program messages not requiring a response
- Type 2R: User program messages requiring a response
- Type 3: User program messages with a format-line request
- Type 4: Message member messages with 8 bytes of data inserted
- Type 5: Program interface to MSG command

SYSLOG is called via a transfer SVC instruction with RIB, and XR2 pointing to a parameter list.

The following SYSLOG processes are shown in Chart 6.2.9:

1 Perform the following mainline processing, routing where necessary:

   - If Type 2, 2R, or 3 message, move caller's message to SYSLOG storage.
   - Build format line and log to history file, if required.
   - If Type 1, 1R, or 4 message, move message from message member to main storage.

2 Get copy of message sent to console; send message indicating response taken.

3 Process MSG command.

4 Perform PUT/GET to work station data management (WSDM).

5 If message's alertable flag is on and alert task is active, issue message to host.

6 If requested, log messages to the history file.

7 Retrieve message.

8 Terminate.

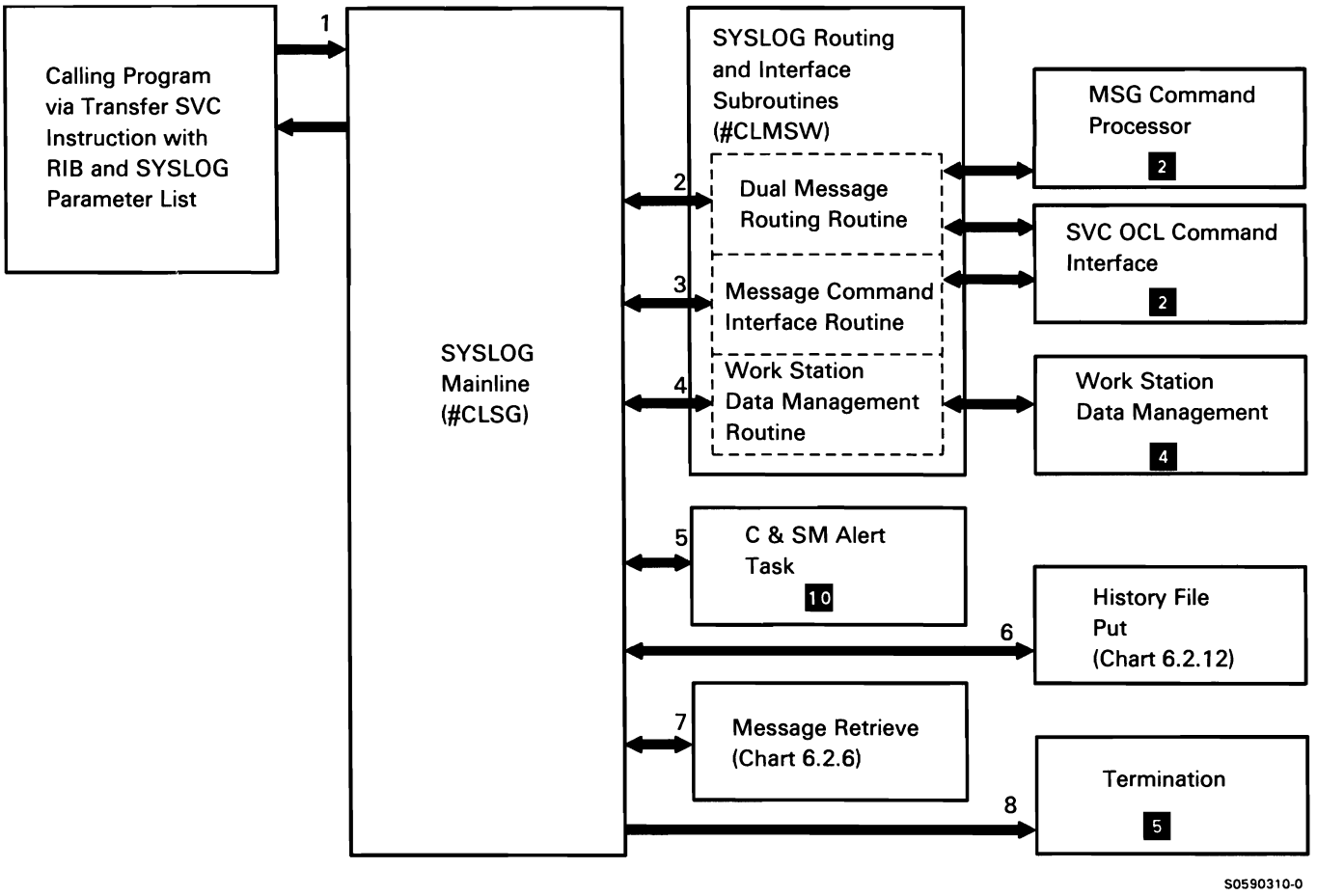


Chart 6.2.9 SYSLOG Control Flow

History File Put

The history file put subfunction places OCL, error messages, and operator responses into the history file. The history file is not a data file, but it is located in the system area on disk. The following history file status information is maintained in the system communications area (SCA):

- Beginning sector address
- Size in sectors
- Sector containing current entry
- Error condition

The history file put subfunction is invoked by transfer SVC instruction with XR2 pointing to a 10-byte parameter list.

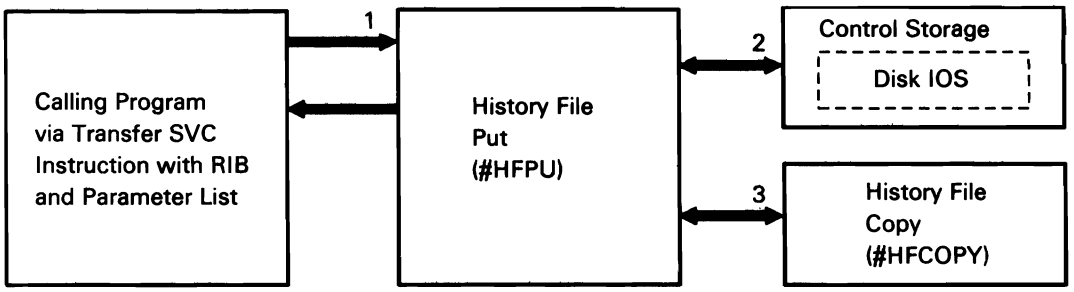
The following history file put processes are shown in Chart 6.2.10:

1 Do the following:

- Get current status from SCA. If no error, get history file start address, size and address of current entry.
- Get input text for current entry, strip-off blanks, and place in buffer.
- Add the following information to the entry in the buffer:
  - Terminal ID from TUB.
  - User ID from JCB or TUB.
  - Job ID from JCB.
  - Time-of-day from control storage timer routine.
  - Message type control bits.
- When entire entry is built, place in sector (in I/O buffer) following the previous current entry; if sector overflows, get new sector and update status information.
- Direct disk IOS to write to disk.
- Route for the remaining processes.

2 Get current sector from history file and place in I/O buffer.

3 If history file no-wrap was specified and number of sectors remaining in system history file is less than 20 per cent, write to HISTCOPY disk file.



S0590311-0

Chart 6.2.10 History File Put Control Flow

Supervisor Task Attach

Supervisor task attach performs the necessary processes to attach a new task to the system. The supervisor task attach subfunction is called by a transfer SVC instruction, with XR1 pointing to the parameter list. The task TB is build during the transfer to supervisor task attach.

Supervisor task attach performs the following:

- Initialize work areas.
- Calculate storage required to load or execute the program.
- Ensure that task can be attached.
- If load request, build and initialize a new program block (PB).
- Set the TB and JCB for task initiation.
- Set registers for load of, or transfer to, the task.
- Issue load or transfer.
- Return the TB address in XR1 for successful attach; otherwise, return the error return code.

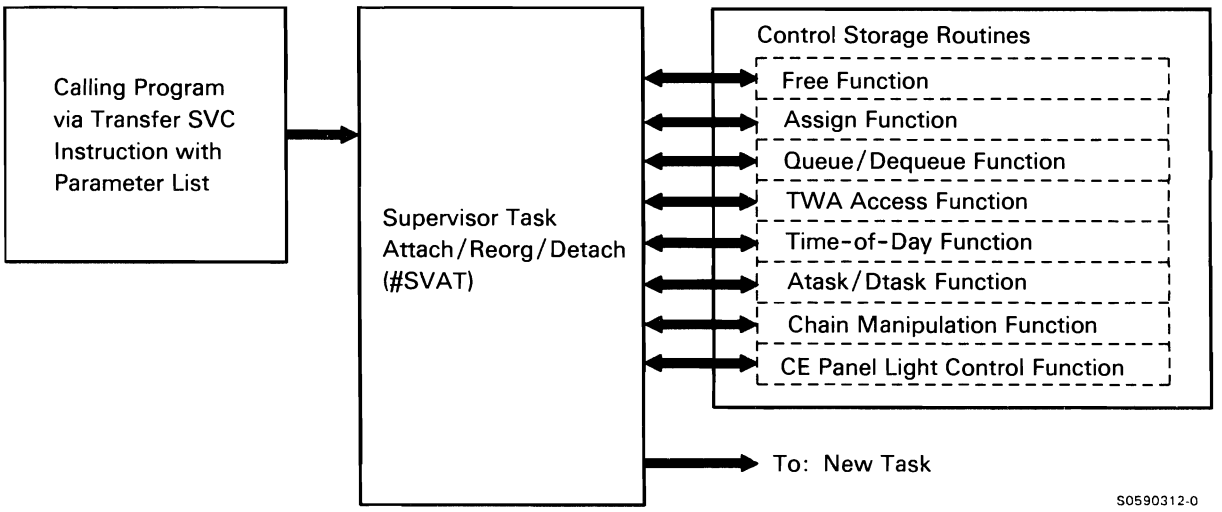


Chart 6.2.11 Supervisor Task Attach Control Flow

Supervisor Task Detach/Change Origin Point

Supervisor task detach/change origin point performs the processing required to detach a task from the system or to change a task's main storage point of origin. It is called by a transfer SVC instruction with XR1 pointing to a parameter list.

The following processes are performed for a task detach request:

- Initialize all work areas.
- Find the largest swappable region size in the system (not including this task).
- Decrement any counters incremented by the task being detached.
- Free any FSBs or format 1's associated with the task.
- If task was an evoked or MRT task, free JCB and WSWA.
- Release any other RBs associated with the task.
- Mark the task as a subtask and allow subtask termination to purge the system of the TB and the last request block-program block (RB-PB) combination.

The following processes are performed for a change in origin:

- Initialize all work areas.
- Find the largest swappable region size in the system (not including this task).
- Calculate sizes required for this task.
- Determine if enough system resources are available to allow the change in origin.
- If this is a load request, initialize the PB.
- Set registers for load or transfer to the task.
- Issue load or transfer.

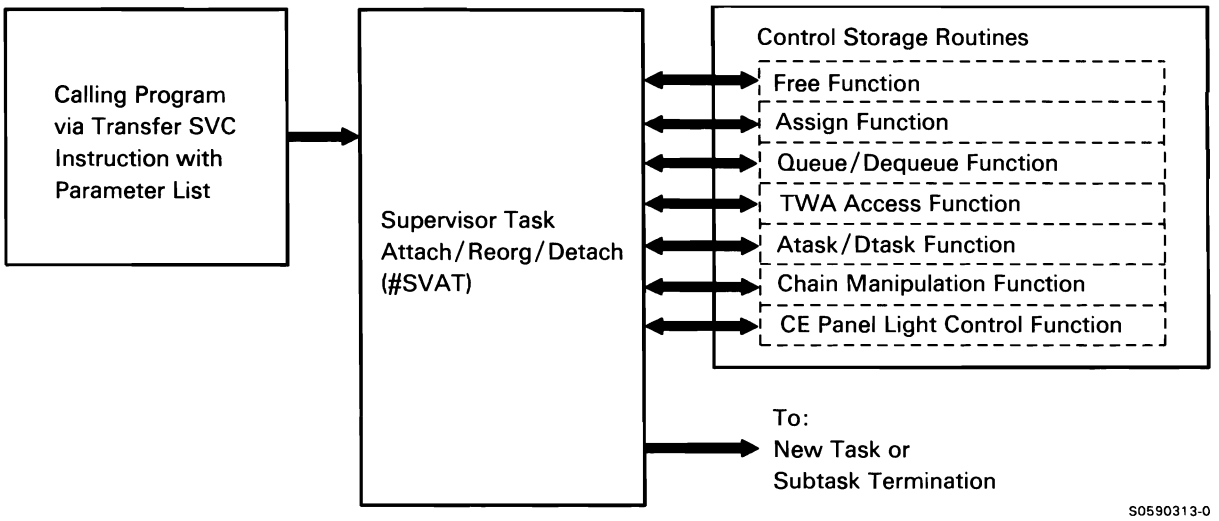


Chart 6.2.12 Supervisor Task Detach/Change Origin Point Control Flow

Syntax Checker

The syntax checker performs the following functions:

- Checks for valid verb in the utility control statement
- Indicates in the communications table the verb and parameter(s) specified in the control statement and any syntax errors detected
- Checks that parameter values are valid and are used in valid combinations
- Places parameter values (or values that are to be substituted for parameter values) in the communications table output area

The syntax checker is called with XR2 pointing to the syntax checker parameter list, which points to the verb list and the communications table. The verb list defines which verb, or verbs, are valid for this particular call to the syntax checker; this allows the caller to define the order in which statements may be coded. The communications table specifies the name of the syntax specification module (if any) that should be loaded from the system library.

The following syntax checker processes are shown in Chart 6.2.13:

- Read control statements.
- Find and load specification module.
- Analyze verbs.
- Process parameters if parameters present.
- Perform end-of-statement check.
- Return to caller.

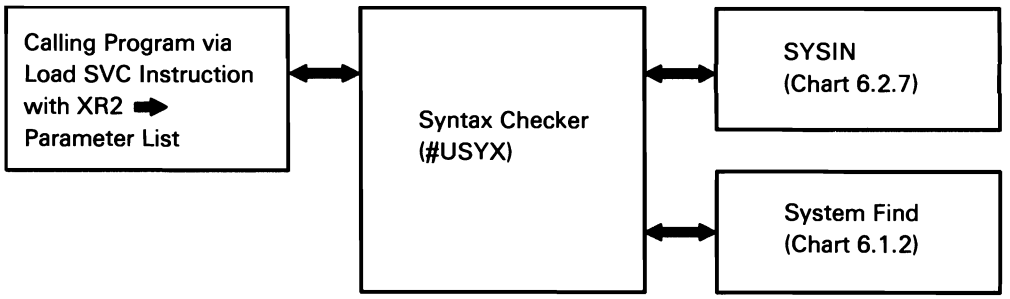


Chart 6.2.13 Syntax Checker Control Flow

Information Retrieval

The information retrieval transient allows user programs to access certain fields within privileged control blocks or the local area on disk. User programs invoke the transient with the transfer SVC instruction and a parameter list (pointed to by XR2) specifying the operation to be performed and the data area to be accessed.

The following information retrieval processes are shown in Chart 6.2.14:

Process user parameter list:

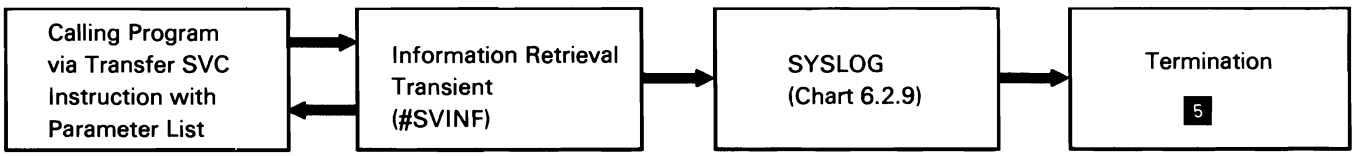
- Move list to transient area.
- Verify fields in parameter list; if invalid entries, issue an option 3-only message.
- Get JCB address from the TB.

If put request, update any one of the following:

- UPSI switch value.
- Program message member 1 and 2.
- System and user local data areas.
- Compiler information block.
- User message member 1.
- JCB return code field.

If get request, return one of the following:

- Date format.
- Program date.
- Session date.
- UPSI switch value.
- Inquiry latch indicator.
- System and user local data areas.
- Compiler information block.
- Program attribute indicator (for NEP).
- Maximum number of requesters (for MRT).
- Number of line per page.
- System list device.
- Unpacked Julian date.
- Spool ID.



S0590315-0

Chart 6.2.14 Information Retrieval Control Flow

Snap Dump

The snap dump transient provides a formatted main storage dump when it is invoked by the transfer SVC instruction with XR2 pointing to the parameter list. The user either dumps the entire region of main storage, or specifies storage limits for the dump.

Snap dump processes are listed below:

- Move the parameter list to temporary area.
- Initialize the work area.
- Calculate the dump limits; if limits are invalid, issue error message and return to caller.
- Get headings messages and dump limits for dump header.
- Set up and print dump, line by line with SYSLIST.
- Return to calling program.

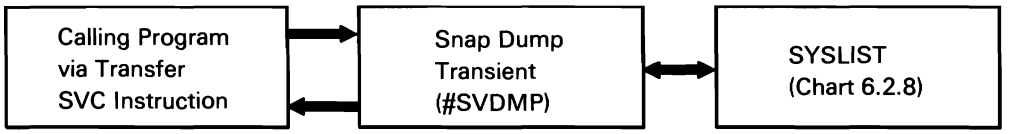


Chart 6.2.15 Snap Dump Control Flow

Diskette Data Save/Restore

The diskette data save/restore routine performs two main functions:

- Saves a disk file on diskette
- Restores a disk file (saved on diskette) to disk

The diskette data save/restore routine is called via a transfer SVC instruction with XR2 pointing to the input file DTF. The output file DTF is chained to the input DTF.

The following diskette data save/restore processes are shown in Chart 6.2.16:

1 Determine the function requested.

2 If save request (using main storage data management):

- Open the diskette and disk files.
- Copy the embedded control record to beginning of buffer.
- Call diskette and disk sector data managements to transfer the data.
- When data transfer is complete, close files.
- Return to caller.

3 If save request (using device-to-device subsystem or data compression):

- Build disk and diskette IOBs.
- Copy the embedded control record at beginning of buffer.
- Call diskette IOS with chained IOBs to transfer the data.
- Update file VTOC for the data transferred.
- Return to caller.

4 If embedded control record get request (using main storage data managment or device-to-device subsystem):

- Open the diskette file.
- Read the embedded control record and move it to the caller's buffer.
- Return to caller.

5 If restore request (using main storage data management):

- Open the diskette and disk files.
- Call diskette and disk data managements to transfer the data.
- When data transfer is complete, close files.
- Return to caller.

6 If restore request (using device-to-device subsystem or data compression):

- Build disk and diskette IOBs.
- Call diskette IOS with chained IOBs to transfer the data.
- Update file format 1 to reflect data transferred.
- Return to caller.

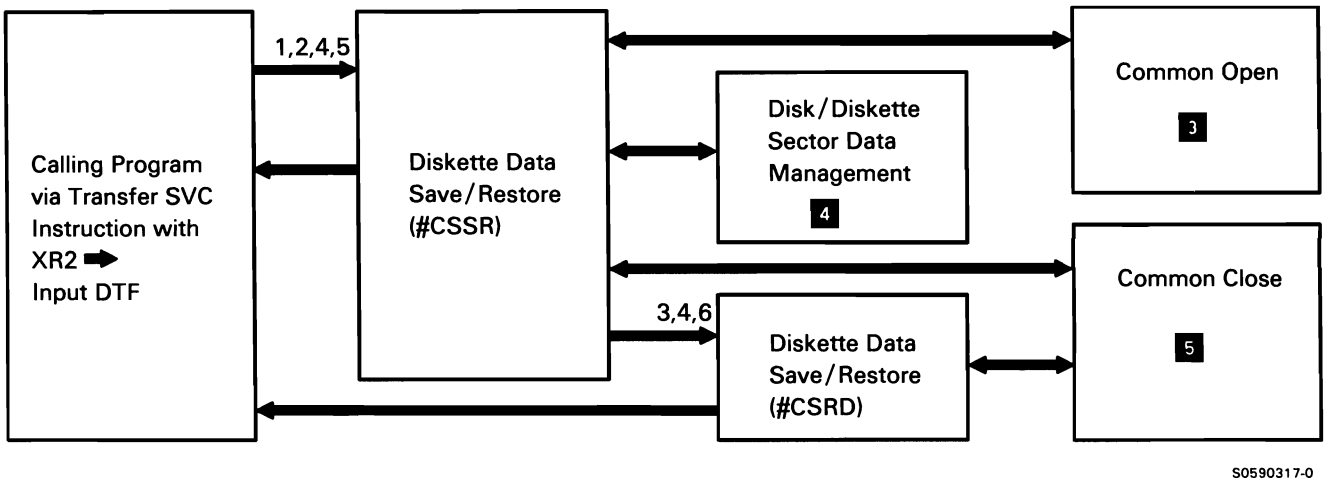


Chart 6.2.16 Diskette Data Save/Restore Control Flow

Tape Data Save/Restore

The tape data save/restore routine performs two main functions:

- Saves disk file(s)/library/folder on tape.
- Restores disk file(s)/library/folder (saved on tape) to disk.

The tape data save/restore routine is called via transfer SVC instruction with XR2 pointing to the input file DTF. The output file DTF is chained to the input DTF.

The tape data save/restore processes are listed below:

1 Acquire TWS to save information across multiple calls in a step; determine the function requested.

2 If save request on 8809 (1/2-inch tape) or 6157 (1/4-inch tape) on 5362 with 9332 disks:

- Build tape HDR1 and HDR2 mask (first pass).
- Build two pairs of tape and disk IOBs (first pass).
- Append UHL1 and UHL2 supplied by the caller to the HDR1 and HDR2.
- Call tape IOS (with nowait), via chained IOBs, to transfer the data.
- If two tape IOB pairs are outstanding, wait for previous IOB's request to complete and return completions to caller.
- If last file transfer request, wait for all oustanding requests to complete and return completions to caller.
- Call close to process END specification from the tape file statement.
- Return to caller.

3 If restore request on 8809 (1/2-inch tape) or 6157 (1/4-inch tape) on 5362 with 9332 disks:

- Build tape and disk IOBs.
- Call tape IOS (with wait) via chained IOBs, to transfer the data.
- If more files are to be restored, return next file's UHL1 and UHL2 to caller's buffer.
- Update file format 1 to reflect data transferred.
- If this is the last file to be restored, call close to process END specification from the tape file statement.
- Return to caller.

4 If save request on 1/4-inch cartridge:

- Build tape HDR1 and HDR2 mask (first pass).
- Build four tape and two disk IOBs (first pass).
- Append UHL1 and UHL2 supplied by caller to the HDR1 and HDR2.
- Call tape IOS and disk IOS by using double buffering to transfer data to tape.
- If not the last file in group, return to caller when the last tape data IOB has been issued.
- If the last file in group, return to caller after all IOBs have completed and call close to handle the END specification from the file statement.

5 If restore request on 1/4-inch cartridge:

- Build four tape and two disk IOBs.
- Call tape IOS and disk IOS using double buffering to transfer data to disk.
- If more files are to be restored, return next file's UHL1 and UHL2 to caller's buffer.
- Update file format 1 to reflect data transferred.
- If this is the last file to be restored, call close to process END specification from the tape file statement.

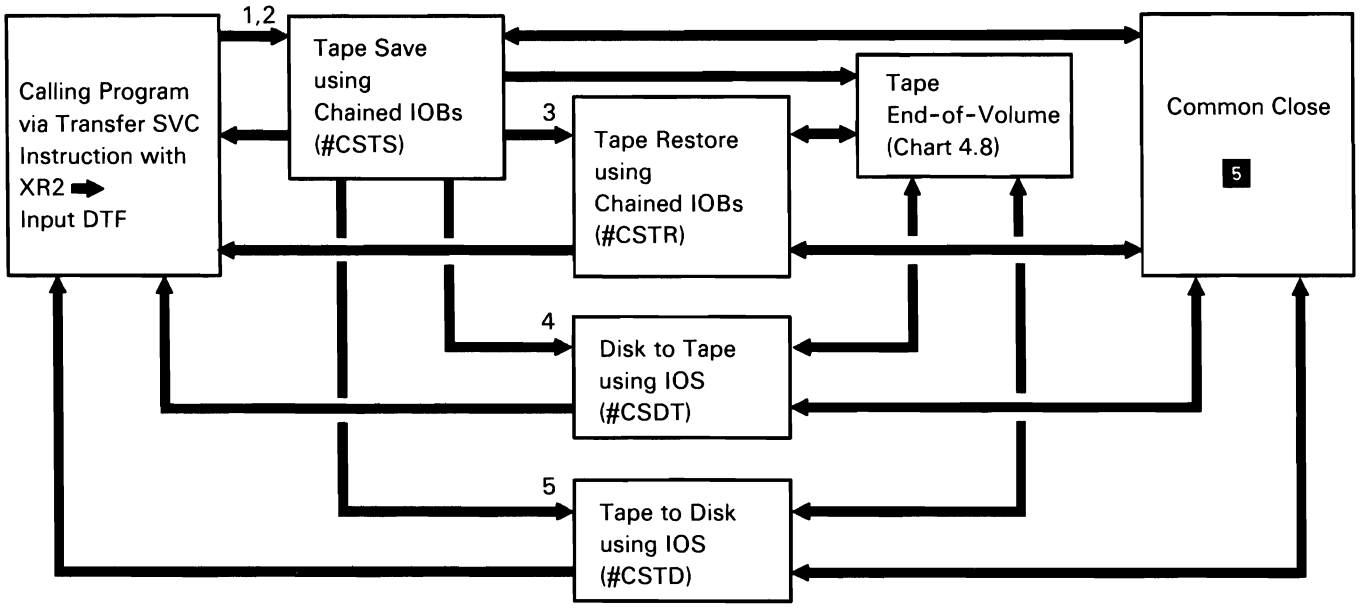


Chart 6.2.17 Tape Data Save/Restore Control Flow

Diskette Magazine Drive Search Routine

The diskette magazine drive search routine performs two main functions:

- Checks for the existence of the specified diskette
- Searches for the specified file

Diskette magazine drive search is called via a transfer SVC instruction with XR2 pointing to the parameter list.

Diskette magazine drive search processes are listed below:

- Determine the function required.
- If search for specific file:
  - If the search is for the first volume of the file, search the range of diskettes until the first or the only volume is found.
  - If the search is for the last volume of the file, search the range of diskettes until the last or the only volume is found.
  - If the search is for any volume of the file, examine the location logically following the original location to determine whether the volume of the specified file is mounted.
- If search for diskette existence:
  - If the search is for the first diskette, with skip empty slots specified, search the range of diskettes until the first diskette is found. If the request does not include the skipping of empty slots, issue a message directing the operator to insert the diskette in the requested slot.
  - If the search is for any volume of the file, examine the location logically following the original location to determine whether the volume of the specified file is mounted.

Chart 6.2.18 Diskette Magazine Drive Search Routine Control Flow

This page is intentionally left blank.

Diagnostic Aids Function 7

The diagnostic aids function consists of the main storage programs provided to assist CEs/CSRs and PSRs in diagnosing and, where feasible, correcting system failures caused by a system or task malfunction. This function includes the following programs which run under the diagnostic aids mainline ($FEAIDS) and the diagnostic aids router ($FECNT):

- Mainline and router processing — Chart 7.1.0
- Error Recording Analysis procedure (ERAP) — Chart 7.1.1
- Dump utilities — Chart 7.1.2.n
- Patch — Chart 7.1.3
- Trace — Chart 7.1.4
- APAR — Chart 7.1.5
- SETDUMP — Chart 7.1.6
- Dump file analysis — Chart 7.1.7
- Diskette diagnostic utility — Chart 7.1.8
- Tape volume statistics — Chart 7.1.9

The diagnostic aids function also includes the following programs that are invoked independent of the diagnostic aids mainline and router programs:

- PTF apply/copy — Chart 7.2
- System measurement facility — Chart 7.3.n
- Service logging — Chart 7.4.n
- Test request — Chart 7.5
- Communications test — Chart 7.6
- Online problem determination — Chart 7.7

MAINLINE AND ROUTER SUBFUNCTION

Requests for diagnostic aids are passed to the diagnostic aids mainline program ($FEAIDS). The mainline program checks for the validity of the requests and initializes for the diagnostic aids shown in Chart 7.1.0.

The following chart shows routing based on the request received from the applicable procedure (with listed keyword):

Request	Utility Called
APAR	$FEAPR, $FEALK, $FESLK, FETLK
DFA	$FEDFA
DIAGCOPY	$FECPY
DUMP CONTROL	$FECON
DUMP DISK	$FEDSK
DUMP IOC	$FEIOC
DUMP MAIN	$FESTR
DUMP MCODE	$FEMCD
DUMP PLD	$FEPDD
DUMP PTF	$FEPTF
DUMP SERVLOG	$FESRV
DUMP STATUS	$FEMNT
DUMP TRACE	$FEDTR
DUMP TAPE	$FETAP
DUMP (TWA, VTOC, SPOOL, or JOBQ)	$FESYS
ERAP	$ERAP
I1DIAG	$DAMP
LOGPLD	$FEPDL
PATCH	$FEDSK
PTF COPY/APPLY	$FECHK
PTF REMOVE	$FEPRM
SERVLOG	$FECNT
SETDUMP	$FEACD
TAPESTAT	$TSB1
TRACE CRT	$FETRC
TRACE BATCH	$FEBTR

Chart 7.1.0 shows the control flow for diagnostic aids mainline and router processing:

Note: Diagnostic aids mainline contains common subroutines used by all diagnostic aids that execute under it. These subroutines are shown in Chart 7.1.0, but are not shown in detail in other 7.1.n charts.

1 Perform common initialization for diagnostic aids utilities execution:

- Initialize common data areas.
- Initialize base subroutines for use by requested utility.
- If necessary alter MIC0001 to show SSP release and modification level, date, and execution-time information.
- Allocate and open requester's work station file.
- If caller is Test Request or System test, load and pass control to ERAP utility; otherwise, load and pass control to the diagnostic aids router.

2 Process utility control statements.

3 Use $FETAB (syntax table for diagnostic aids utility control statements) to check syntax.

4 If APAR diskette input, preprocess diskette file.

5 If APAR tape input, preprocess tape file.

6 If call is for DUMP MAIN, DUMP TRACE, DFA, or APAR, then display dump file status.

7 Process requests for service log recording.

8 Terminate diagnostic aids (the individual diagnostic aid utilities call the ENDIT routine in $FEAIDS to reload the router which calls termination).

9 Perform requested utility processing.

10 Perform subroutine ($FEAIDS) functions as required.

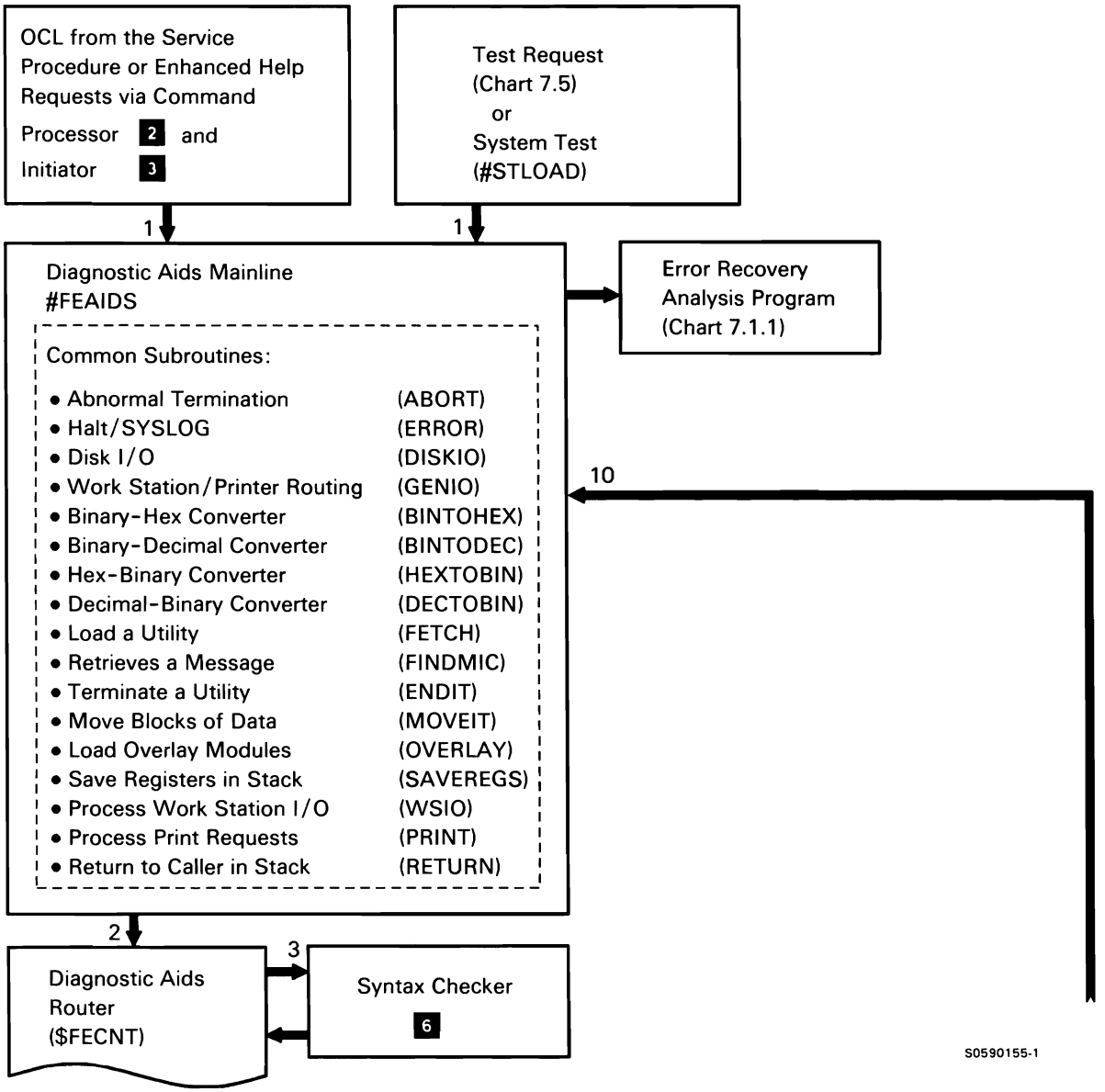


Chart 7.1.0 (Part 1 of 2) Diagnostic Aids Mainline and Router Control Flow

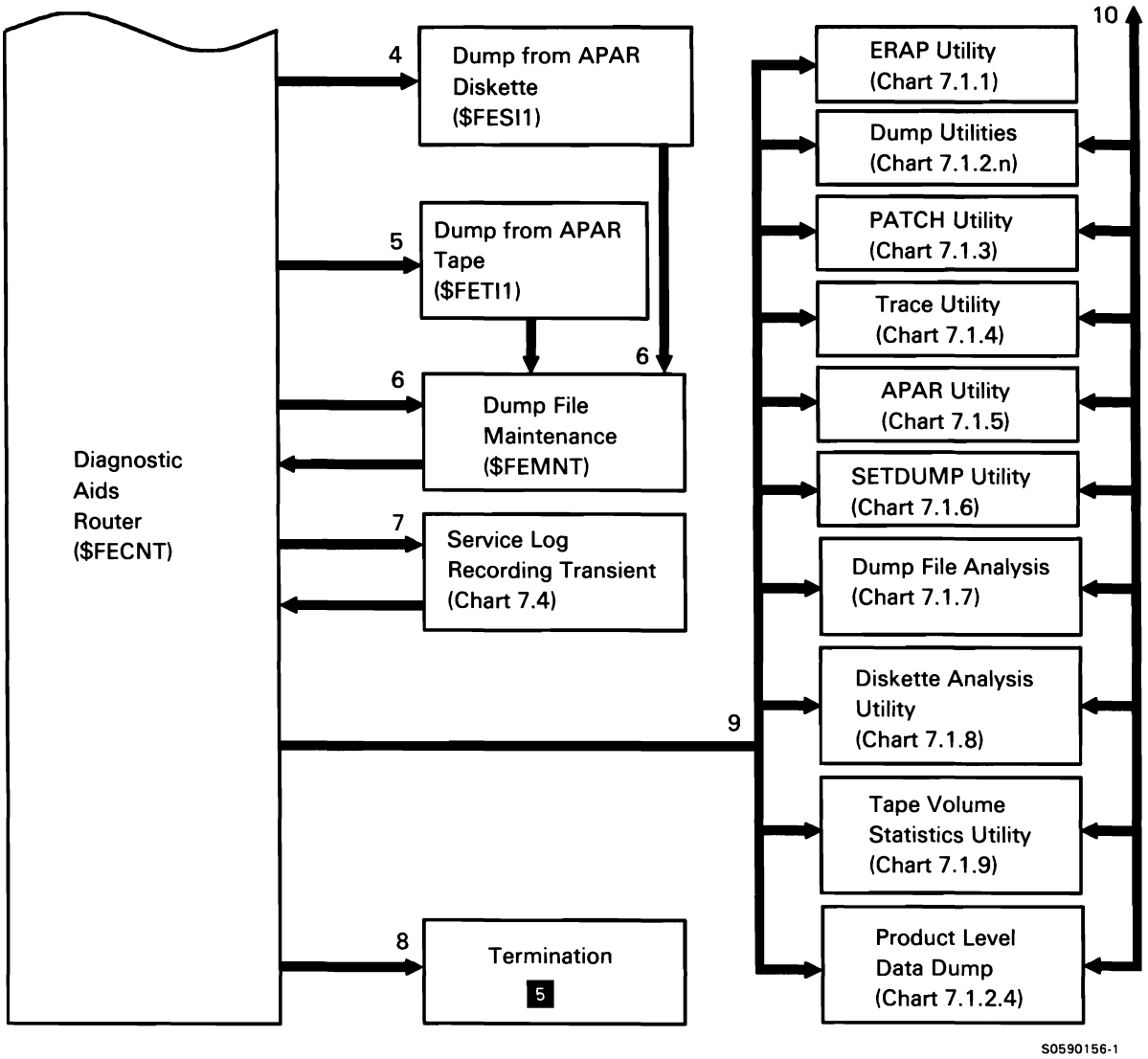


Chart 7.1.0 (Part 2 of 2) Diagnostic Aids Mainline and Router Control Flow

ERROR RECORDING ANALYSIS PROCEDURE (ERAP) SUBFUNCTION

The ERAP subfunction is used to display or print data that is logged for all devices attached to the system. Device-specialized ERAP modules format information from I/O counter tables and error logging tables for their respective devices. The information printed or displayed by the mainline ERAP module is dependent on user-selected options.

All ERAP modules use subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0.

The following ERAP processes are shown in Chart 7.1.1:

1 Do the following:

- Prompt for device.
- Prompt for report options (summary, elapsed time counters, all devices).
- Route for specific device.

2 Do the following by device:

- Format I/O counter table, error counter table, and error history table for specified device.
- Reset I/O counter table or error counter table if requested.

3 Use $FEAIDS common subroutines to do the following:

- Put formatted tables to printer.
- Put formatted tables to display station.
- If called by system test or test request, route control back; otherwise, terminate.

4 Perform the following conversions, as required:

- Timer units to time-of-day.
- Time-of-day to timer units.
- Multiplication.
- Division.

5 Print or display summary formatted report.

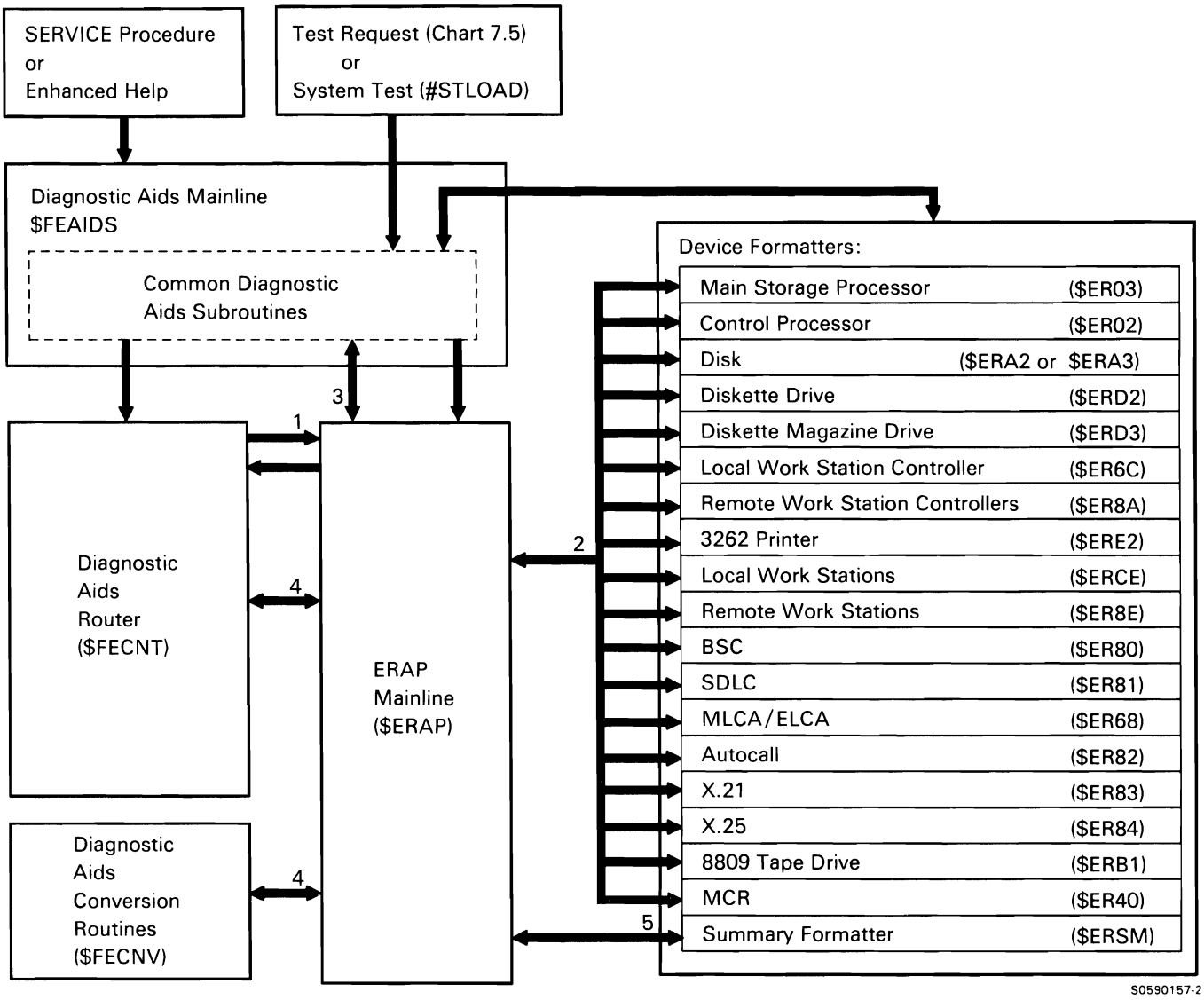


Chart 7.1.1 ERAP Processing Control Flow

DUMP UTILITIES SUBFUNCTION

The dump utilities are used to display system and task dump files from disk, diskette, or tape, or to display other areas frequently required to diagnose an abnormal termination problem.

All dump utilities use subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other dump utilities processes are shown in Charts 7.1.2.1 through 7.1.2.12:

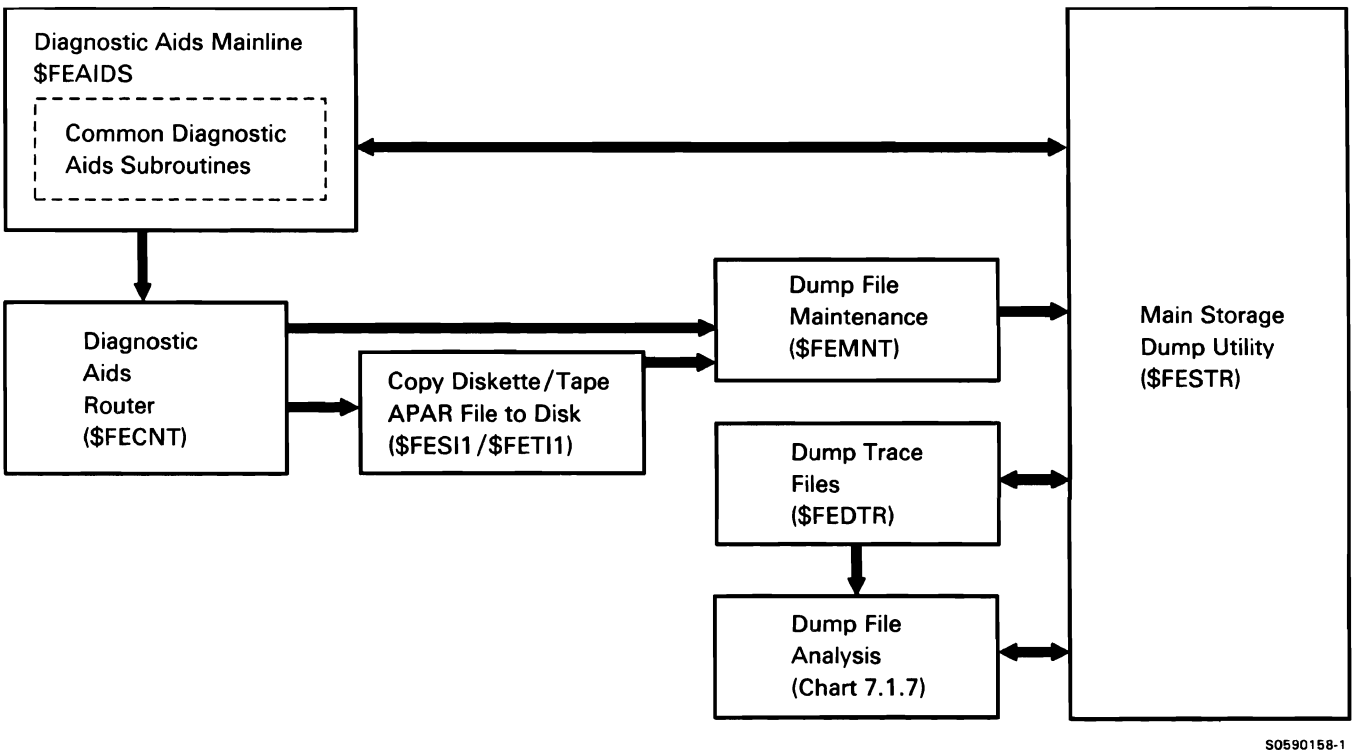


Chart 7.1.2.1 Display System or Task Dump Control Flow

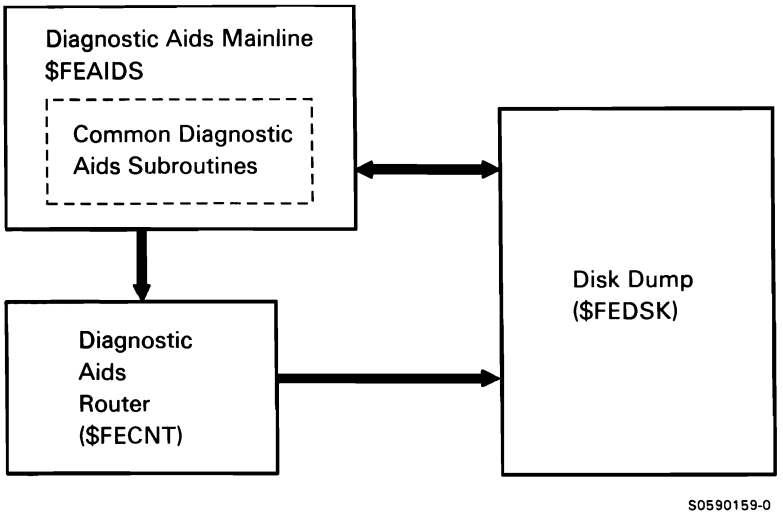


Chart 7.1.2.2 Disk or Diskette Storage Areas Dump Control Flow

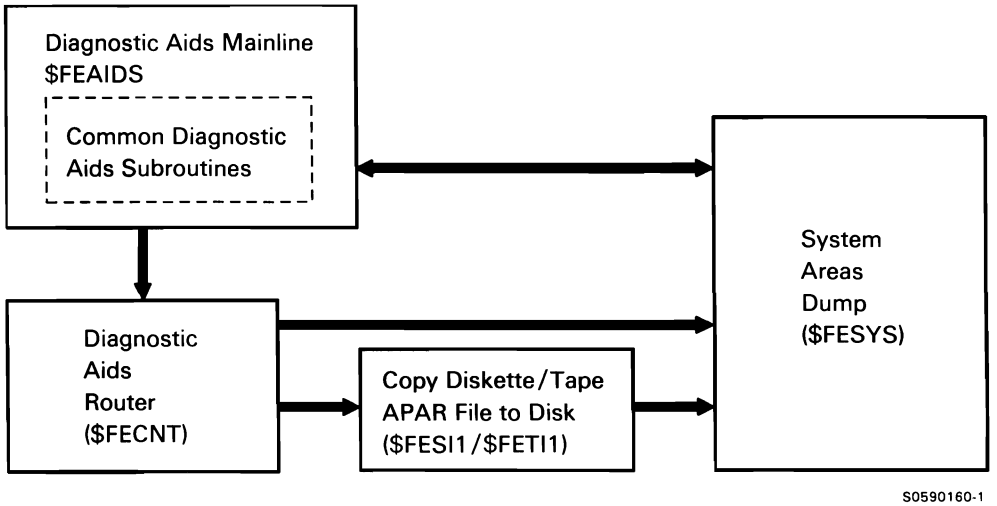


Chart 7.1.2.3 VTOC, TWA, Spool File, or Job Queue Dump Control Flow

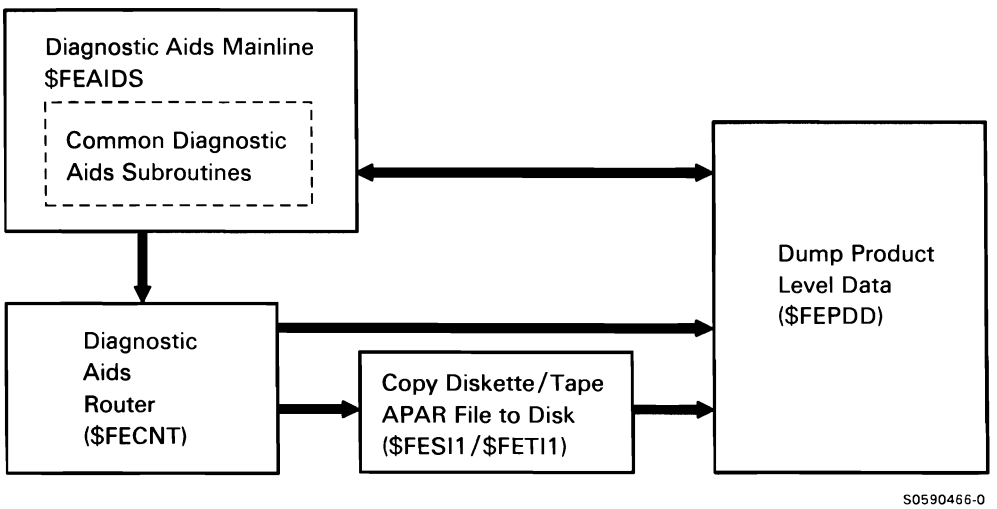


Chart 7.1.2.4 Display Product Level Data Dump Control Flow

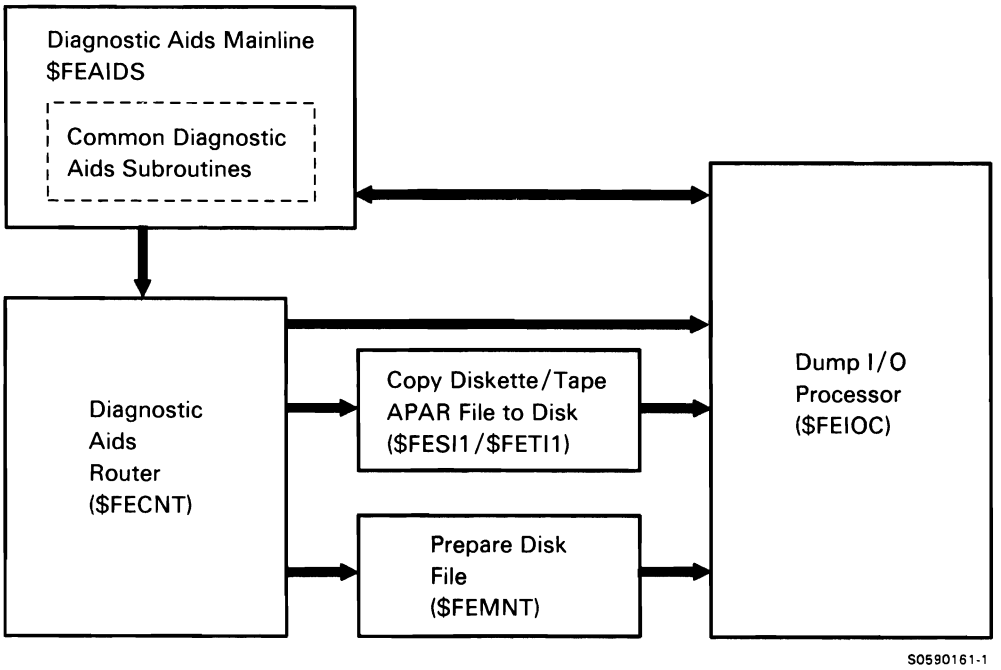


Chart 7.1.2.5 I/O Controller Storage Dump Area Dump Control Flow

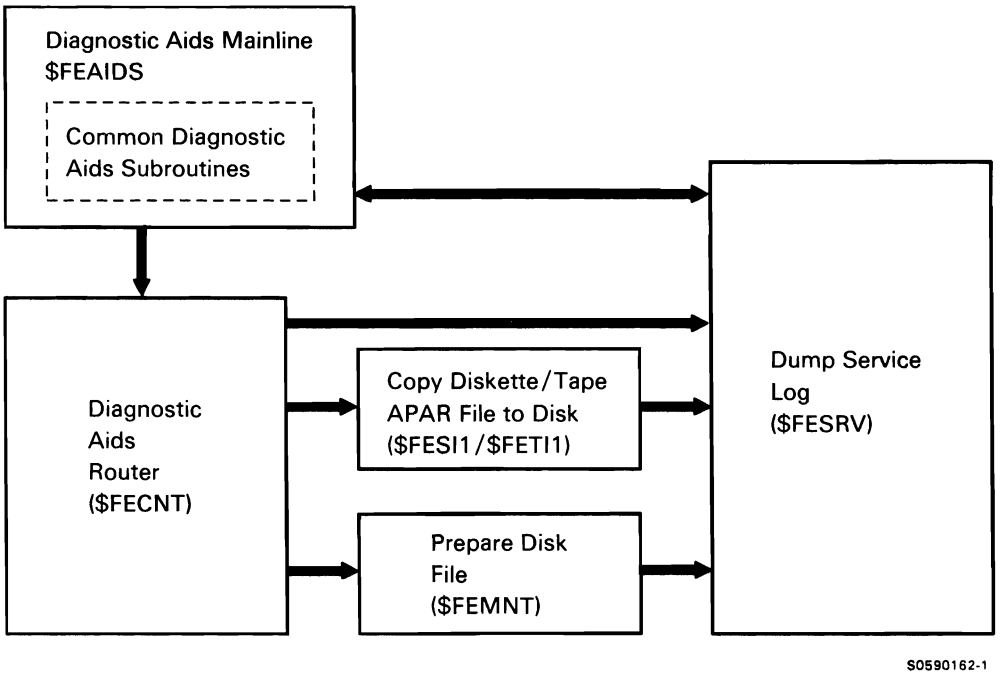


Chart 7.1.2.6 Service Log Dump Control Flow

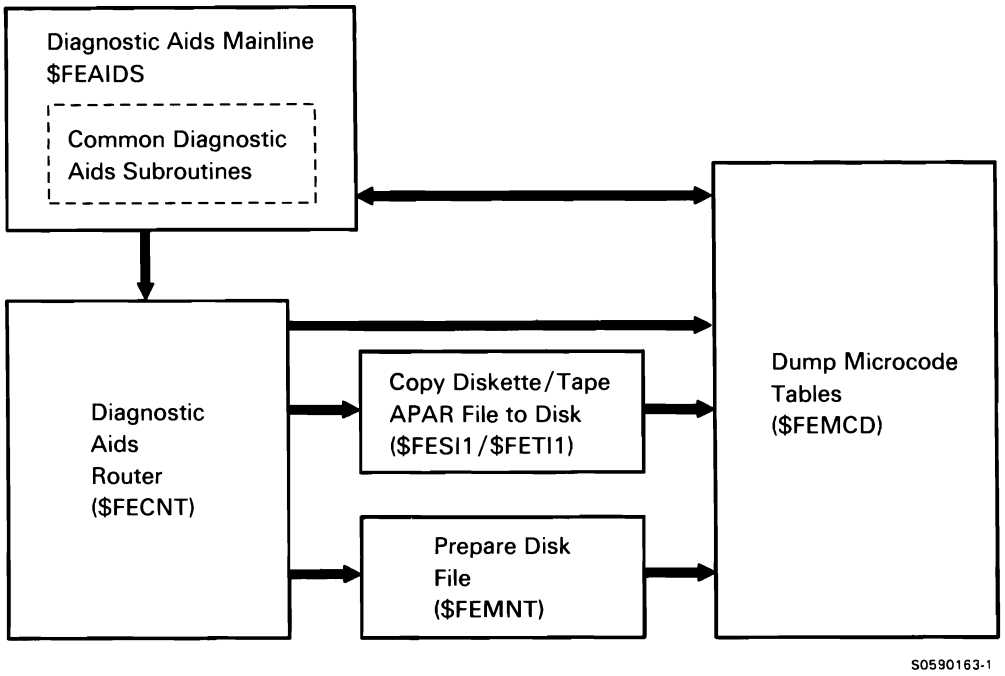


Chart 7.1.2.7 Microcode Patch Table Dump Control Flow

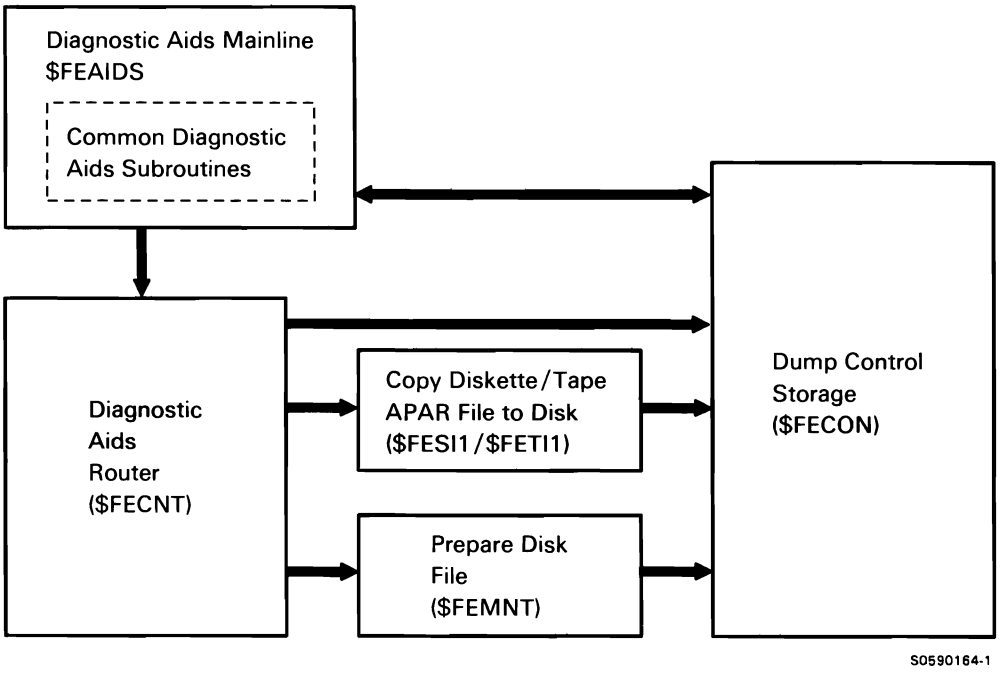


Chart 7.1.2.8 Control Storage Dump Area Dump Control Flow

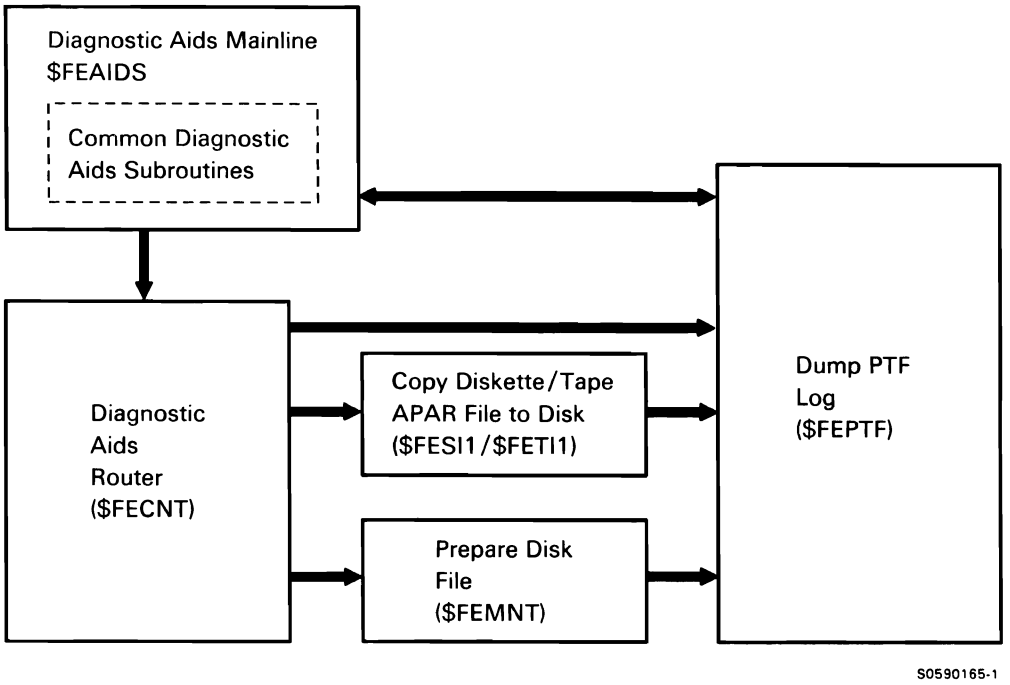


Chart 7.1.2.9 PTF Logs Dump Control Flow

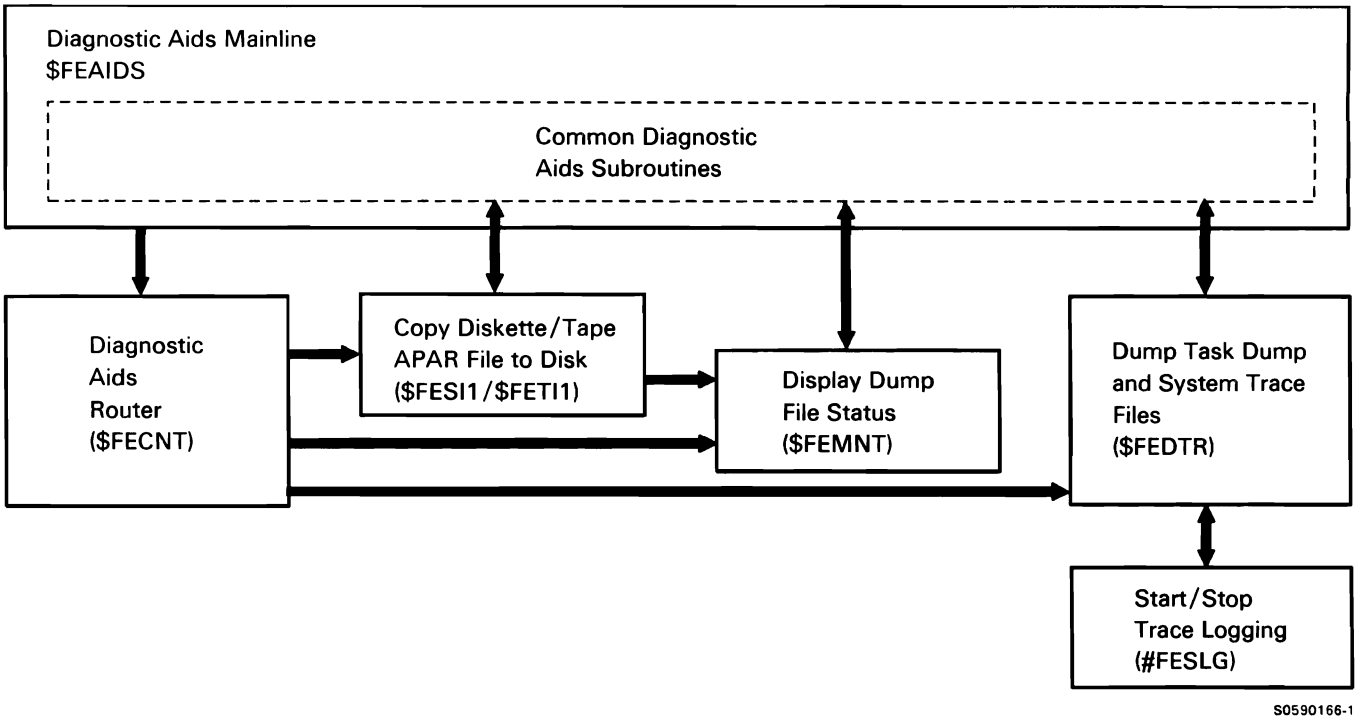


Chart 7.1.2.10 Dump Trace Files Dump Control Flow

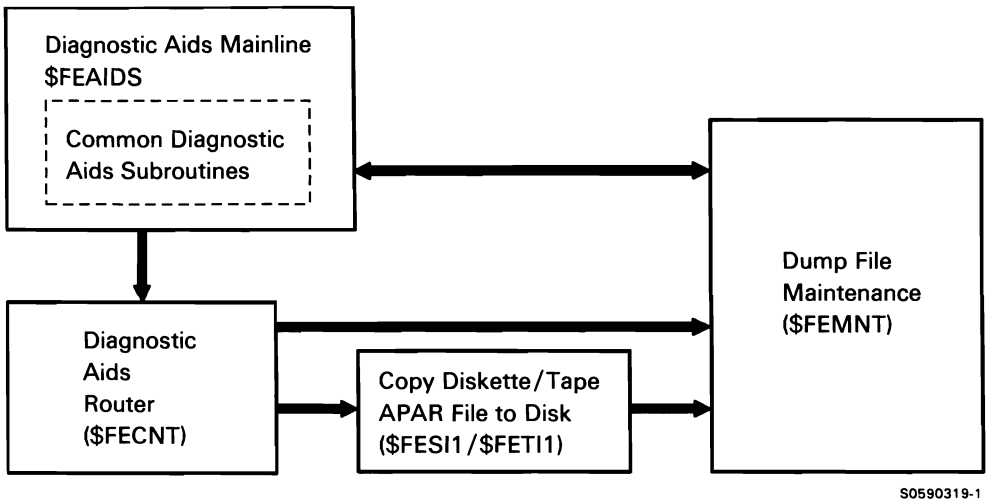


Chart 7.1.2.11 Dump Status Control Flow

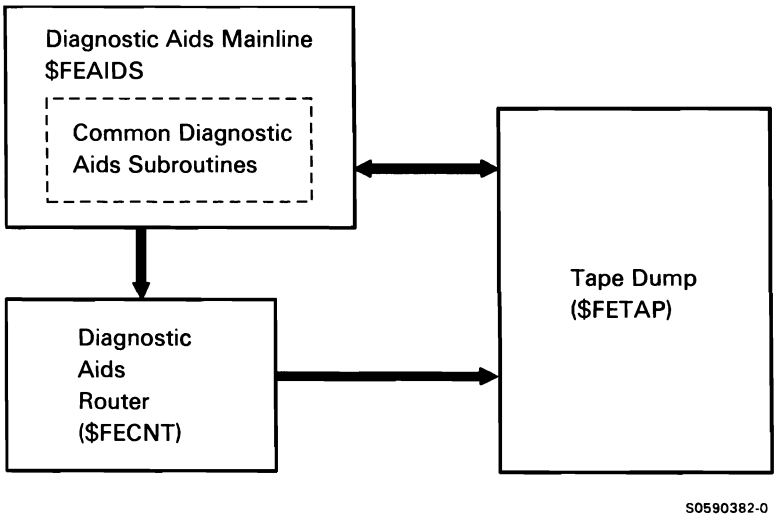


Chart 7.1.2.12 Tape Storage Areas Dump Control Flow

PATCH SUBFUNCTION

The patch utility is used to display or modify disk or diskette storage.

Patch uses subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other patch subfunction processes are shown in Chart 7.1.3:

1 Read control statements and determine that patch was requested.

2 Perform patch processing:

- If request is for disk data, do the following:
  - Get data from disk (via DISKIO subroutine).
  - Modify the data as requested.
  - Write data back to disk (via DISKIO subroutine).

- If request is for diskette, do the following:
  - Get data from diskette (via diskette IOS).
  - Modify the data as requested.
  - Write data back to diskette (via diskette IOS).

3 Terminate via ENDIT subroutine.

4 Perform subroutine processes required by $FEDSK.

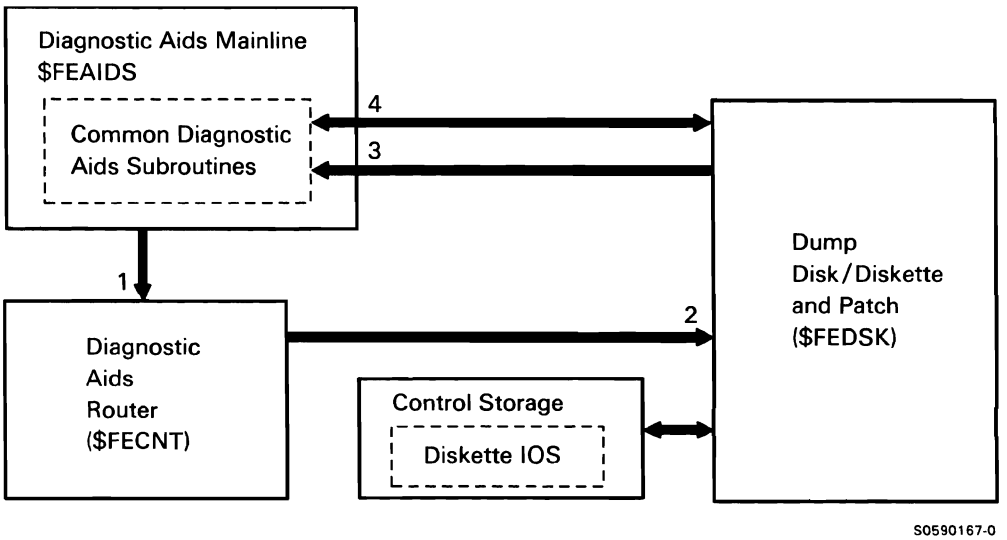


Chart 7.1.3 Patch Utility Control Flow

TRACE UTILITY SUBFUNCTION

The trace utility is used to select events to be traced. Trace uses subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other trace subfunction processes are shown in Chart 7.1.4:

1 Read control statements and determine TRACE requested.

2 Perform, and route for, trace processing requested.

3 Get information on trace profile selected by requester.

4 Initiate requested trace function.

5 Perform subroutine processes required by trace utility.

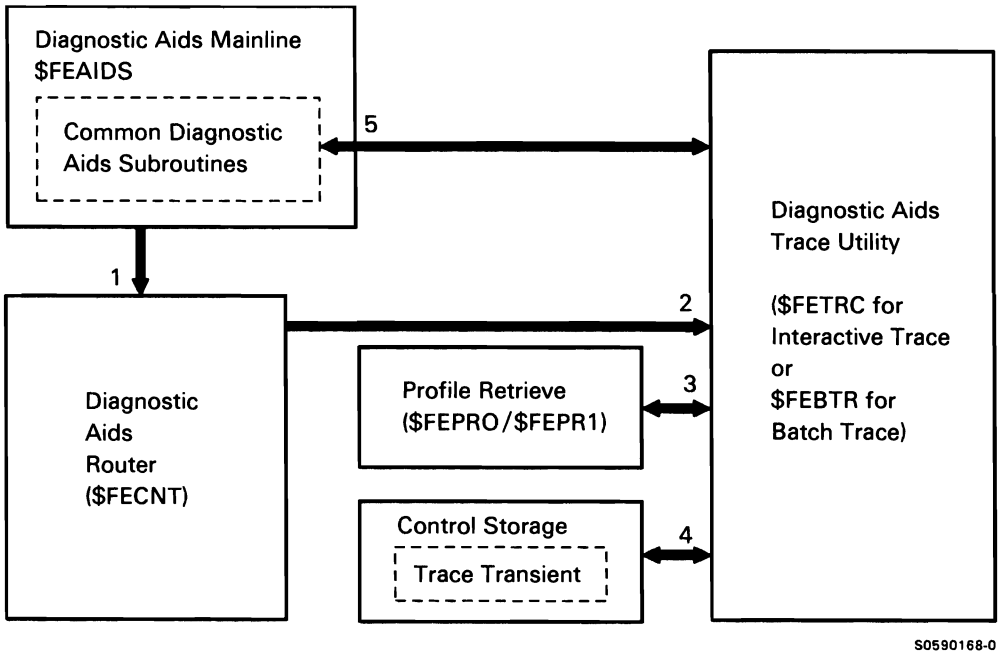


Chart 7.1.4 Trace Utility Control Flow

APAR UTILITY SUBFUNCTION

The APAR utility is used to collect information for APAR submission or to diagnose a problem other than in a real-time environment.

APAR uses subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other APAR subfunction processes are shown in Chart 7.1.5:

1 Read control statements and route control to APAR utility.

2 Perform APAR processing requested:

- Analyze dump files.
- If there is a work station requester, and no dump file name was entered in the procedure, put up dump file status screen.
- Select the task dump to copy to APARFILE.
- If the DUMPFILE parameter is entered in the invoking procedure, select it for APARFILE.

3 Route for required APAR processing.

4 Stop logging trace files to disk, so files can be copied.

5 Copy system data areas to diskette or tape via diskette or tape data management.

6 Perform additional (module 2) APAR processing.

7 Perform additional (module 3) APAR processing (copy spool file).

8 Perform termination processing.

9 Perform subroutine processes required by APAR utility.

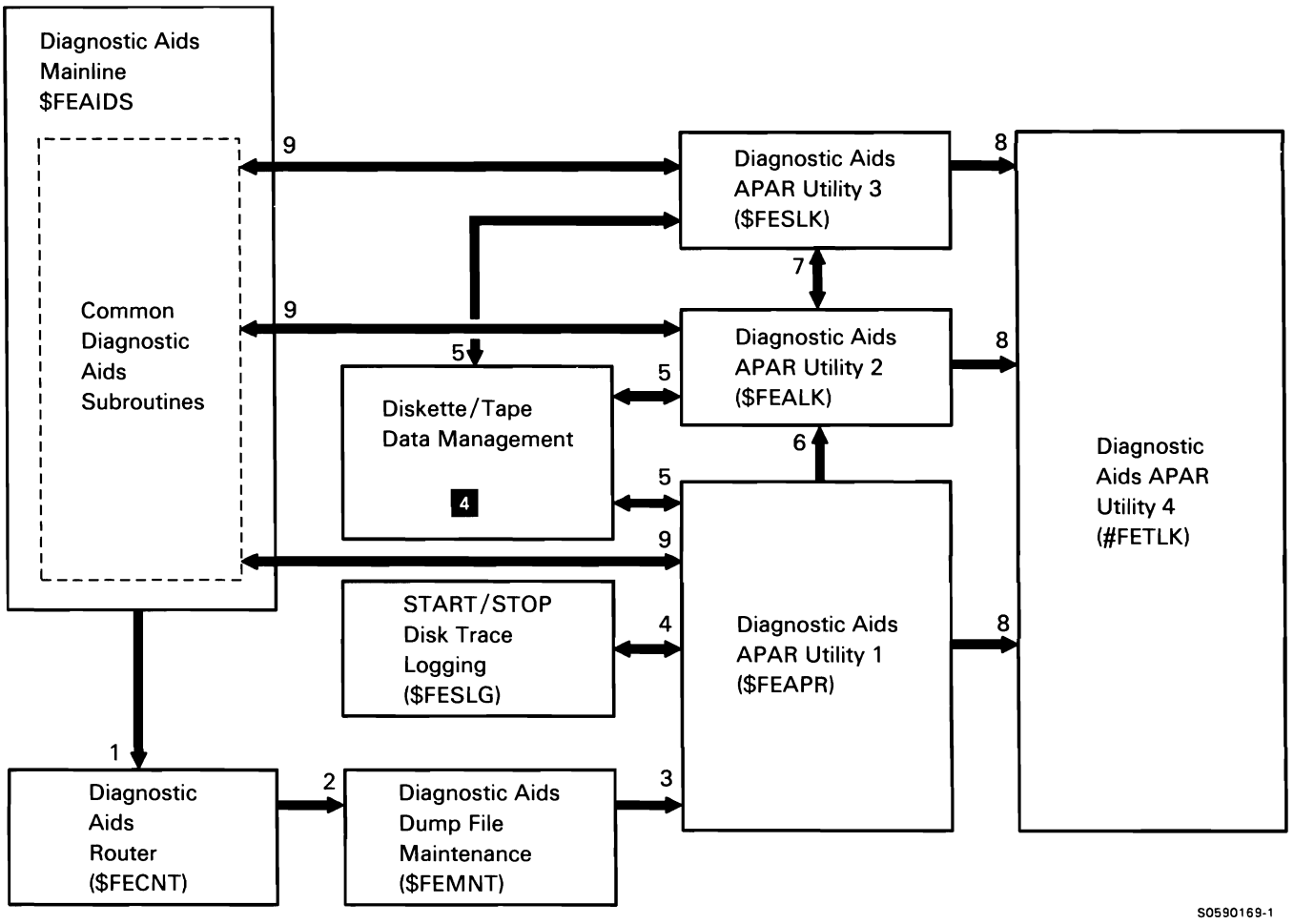


Chart 7.1.5 APAR Utility Control Flow

SETDUMP SUBFUNCTION

SETDUMP is used for the following:

- To specify the hardware and software address, compare values to stop the main storage processor (MSP) long enough to take a task dump at a specific main storage address.
- To suspend a task after a task dump has been taken.
- To force a task dump of a specific task.

SETDUMP uses subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other SETDUMP subfunction processes are shown in Chart 7.1.6:

1 Read control statements and route control to SETDUMP.

2 Perform requested SETDUMP processing:

- Resume any jobs suspended by address compare.
- Validate hardware and software compare values.
- Create parameter list for control storage dump transient ($ACDUMP).

3 Process dump request from SETDUMP.

4 Terminate via ENDIT subroutine.

5 Use FEAIDS subroutines as required.

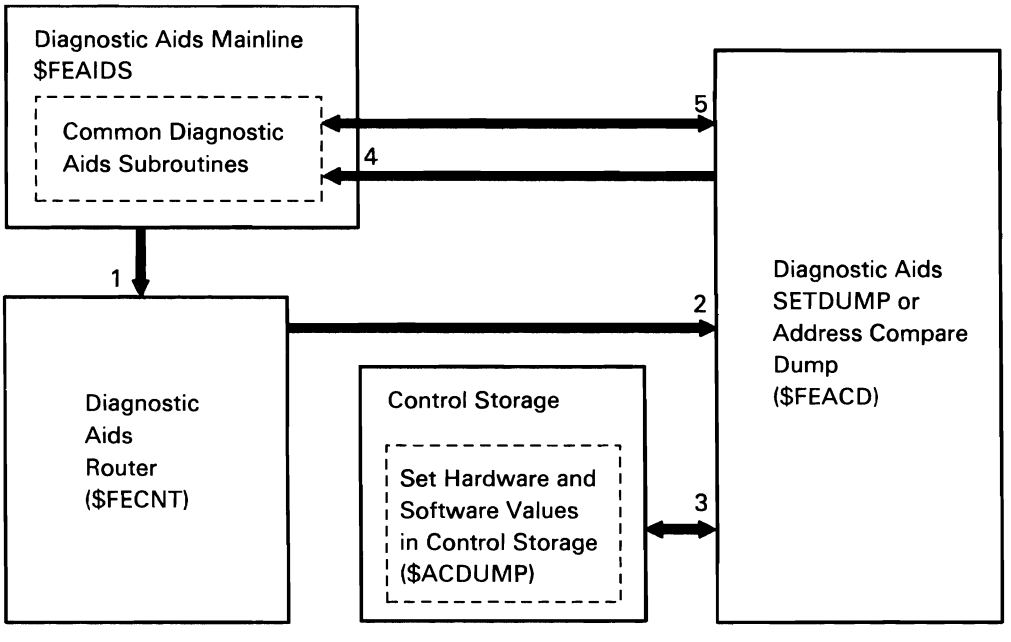


Chart 7.1.6 SETDUMP Utility Control Flow

DUMP FILE ANALYSIS (DFA) SUBFUNCTION

Dump file analysis (DFA) formats and displays selected areas of a storage dump.

DFA uses subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other DFA subfunction processes are shown in Chart 7.1.7:

1 Read control statements and determine function requested.

2 Select system or task dump file to process.

3 Format data and put to printer or display.

4 Perform $FEAIDS subroutine processing, as required.

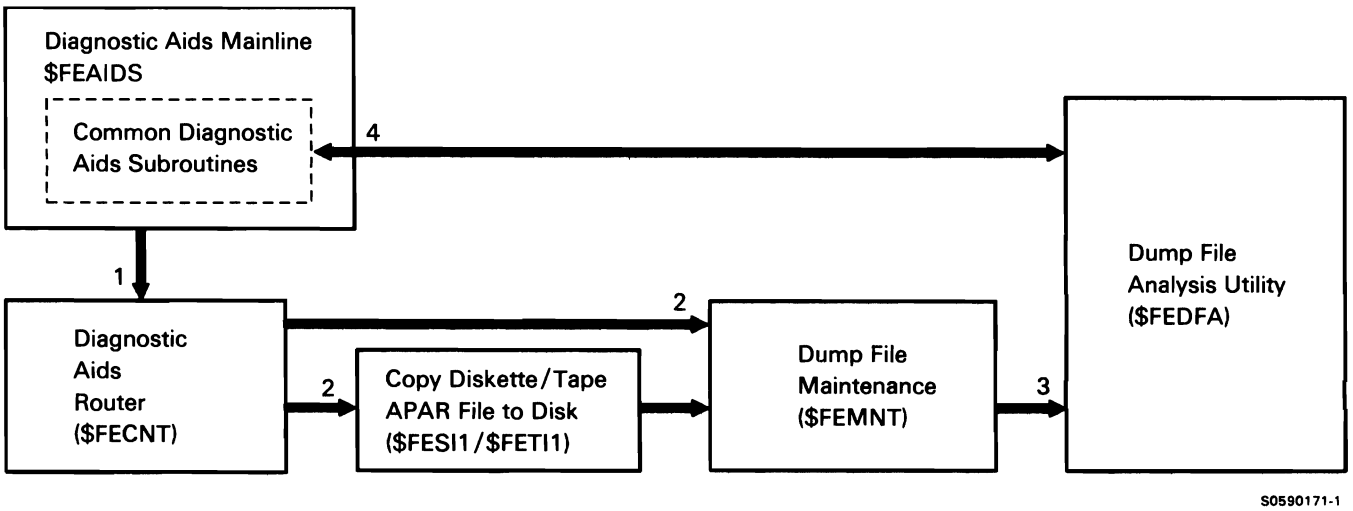


Chart 7.1.7 Dump File Analysis (DFA) Control Flow

DISKETTE DIAGNOSTIC UTILITY SUBFUNCTION

The diskette diagnostic utility helps the user to isolate diskette (media) failures from diskette drive (hardware) failures. The diskette diagnostic utility executes under diagnostic aids mainline ($FEAIDS) and requires a special test diskette.

The diskette diagnostic utility uses subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other diskette diagnostic utility processes are shown in Chart 7.1.8:

1 Read control statements and determine that diskette diagnostic utility was requested.

2 Perform the following mainline processing:

- Allocate the diskette drive.
- Route for menu item selected:
  - 1, *Test diskette drive.*
  - 2, *Scan diskette for errors.*
  - 3, *Print diskette error log.*
  - 4, *Recover data on a diskette.*
  - 5, *Copy data from one diskette to another.*
  - 6, *Display or change diskette sectors.*
- Terminate.

3 Process menu item 1 by writing and reading, and by reading the alignment tracks.

4 Process menu item 2 by reading all diskette sectors and printing summary report documenting errors.

5 Process menu item 3 by displaying statistical information record.

6 Process menu item 4 by performing multiple reads and optional writes to the specified sector.

7 Process menu item 5 by copying diskette.

8 Process menu item 6 by patching diskette.

9 Use $FEAIDS subroutines as required.

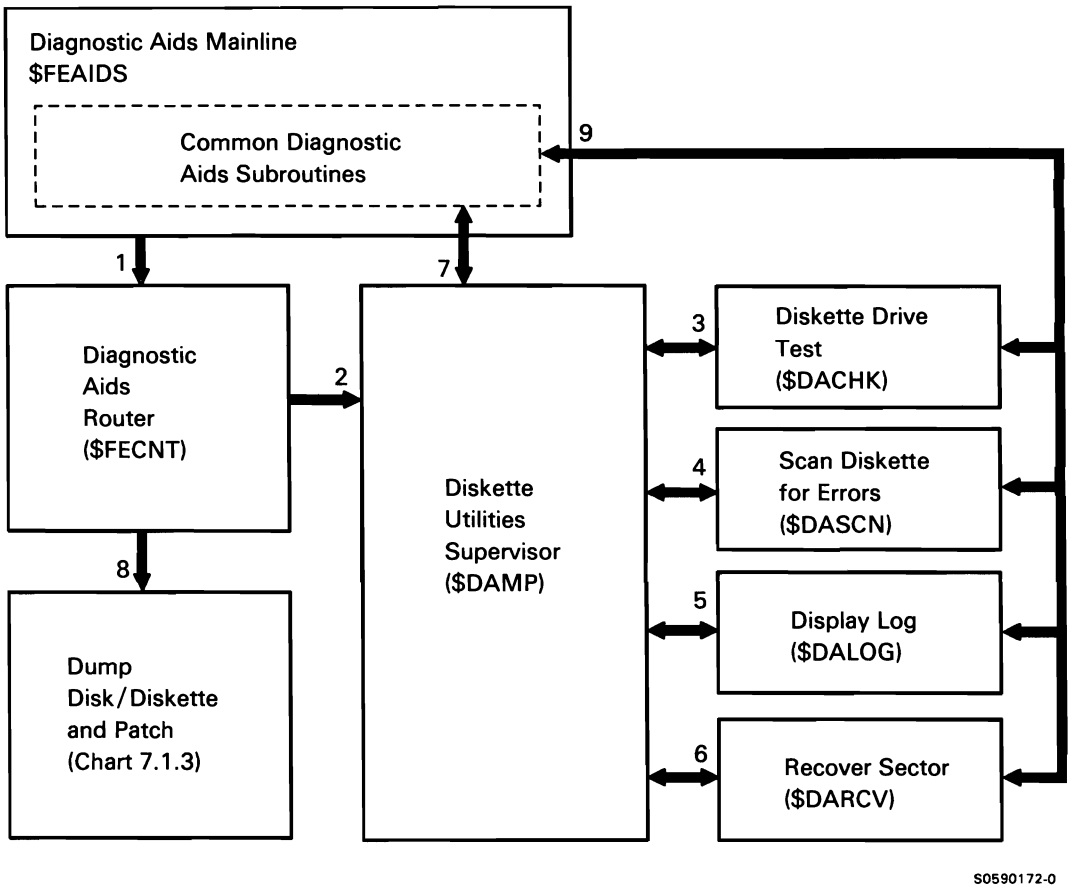


Chart 7.1.8 Diskette Utility Control Flow

TAPE VOLUME STATISTICS UTILITY SUBFUNCTION

The tape volume statistics utility formats and prints tape volume statistical information for all tape devices on the system. The tape volume statistics utility uses subroutines provided by diagnostic aids mainline ($FEAIDS); these subroutines are shown in Chart 7.1.0. Other tape volume statistics processes are shown in Chart 7.1.8:

1 Read control statements and determine that tape volume statistics utility was requested.

2 Read tape volume statistics table from disk, format statistics, and display or print.

3 Use FEAIDS subroutines as required.

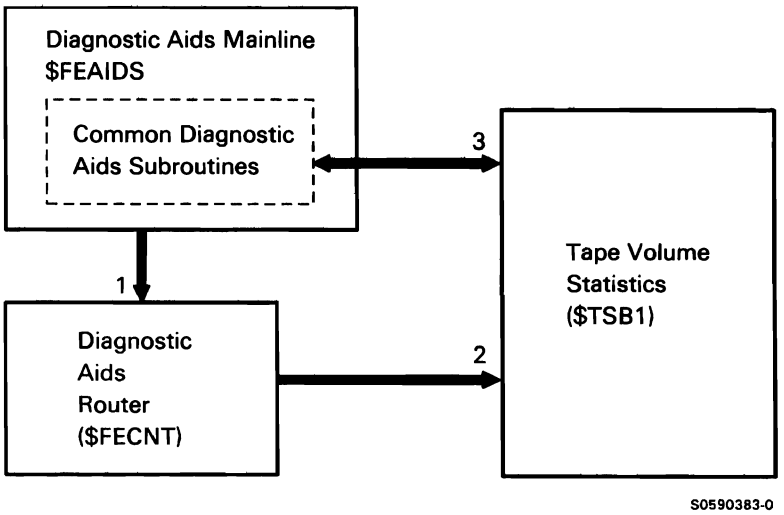


Chart 7.1.9 Tape Volume Statistics Utility Control Flow

PTF APPLY/COPY SUBFUNCTION

The PTF apply/copy subfunction creates a library with the selected module replacements and checks to ensure that all prerequisite PTFs are also present. After creating the library of modules, it replaces the appropriate system modules with their respective replacement modules.

The following PTF apply/copy processes are shown in Chart 7.2:

1 If PTF copy selected, determine which files should be copied from diskette to work library.

2 If PTF copy selected, build library of PTFs from the PTF file on diskette.

3 If PTF apply/copy selected, verify release level and check to ensure that all prerequisite PTFs are to be copied/applied.

4 For PTF copy, build PTF library and copy all specified modules; for PTF apply, apply module replacements to specified library.

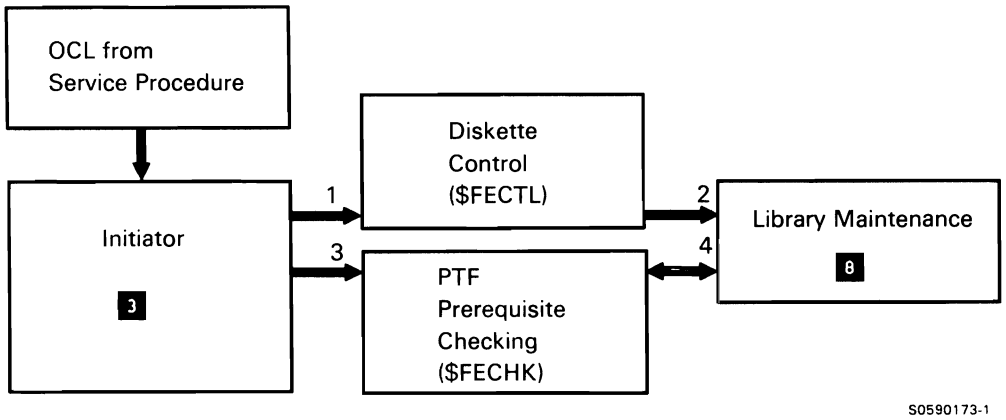


Chart 7.2 PTF Apply/Copy Control Flow

SYSTEM MEASUREMENT FACILITY (SMF) SUBFUNCTION

The system measurement facility (SMF) collects system-resource-usage data that can enable the user to identify causes of poor system performance. SMF consists of three components: *data collection* collects data while user applications are executing and writes the data to the SMF data file on disk; the *report-writer* uses the data collected to print the type of report specified by the user (all, detail, mini, or summary); and the *report file program* uses the data collected to format the type of disk file specified by the user (all, detail, mini, or summary).

SMF Data Collection

SMF data collection collects statistical data about interactive and batch programs, the spooling function, the main storage and control storage processors, the disk drive, and the communications lines. SMF data collection is activated by the SMFSTART procedure command and is deactivated by the SMFSTOP procedure command.

Note: The SMFSTOP procedure loads an additional copy of the SMF data collection program that terminates the currently executing copy.

The following data collection processes are shown in Chart 7.3.1:

1 Determine function requested.

2 If SMFSTOP request, post active data collection task (see preceding note) for end of job.

3 Process SMFSTART request by performing initialization:

- Syntax check utility control statements.
- Initialize control storage data areas.
- Allocate SMF data file.
- If data communications SMF specified, allocate SMF data communications microcode.
- If I/O and SEC data by task was selected, build a system measurement block for each task.
- Collect the time-of-day date and system configuration information and write them to the SMF data collection file.

4 Collect data:

- Wait for SMF data collection time interval to elapse.
- Determine if an SMF STOP request has been received.
- Collect time-of-day, main storage usage information, and task work area usage data.
- Call control storage SMF microcode to collect system event counters, I/O counters, device and data storage attachment (DSA) information.
- If communications was selected:
  - Collect communications configuration information if there are any new lines active for this snapshot.
  - Collect communications utilization data for all active lines that are not new this snapshot.
- Collect information on each active task:
  - If data-by-task was selected, collect I/O and SEC data by task from system measurement block.
  - If data-by-task was selected, collect data from each task that has terminated since last snapshot.
- Write the data collected to the the SMF data collection file.

5 Perform EOJ cleanup of SMF control storage data areas.

6 Issue error message for SMF-detected errors.

7 Exit to terminate.

8 If I/O or SEC data collection by task was in progress when a task terminates, clean up counts related to that task.

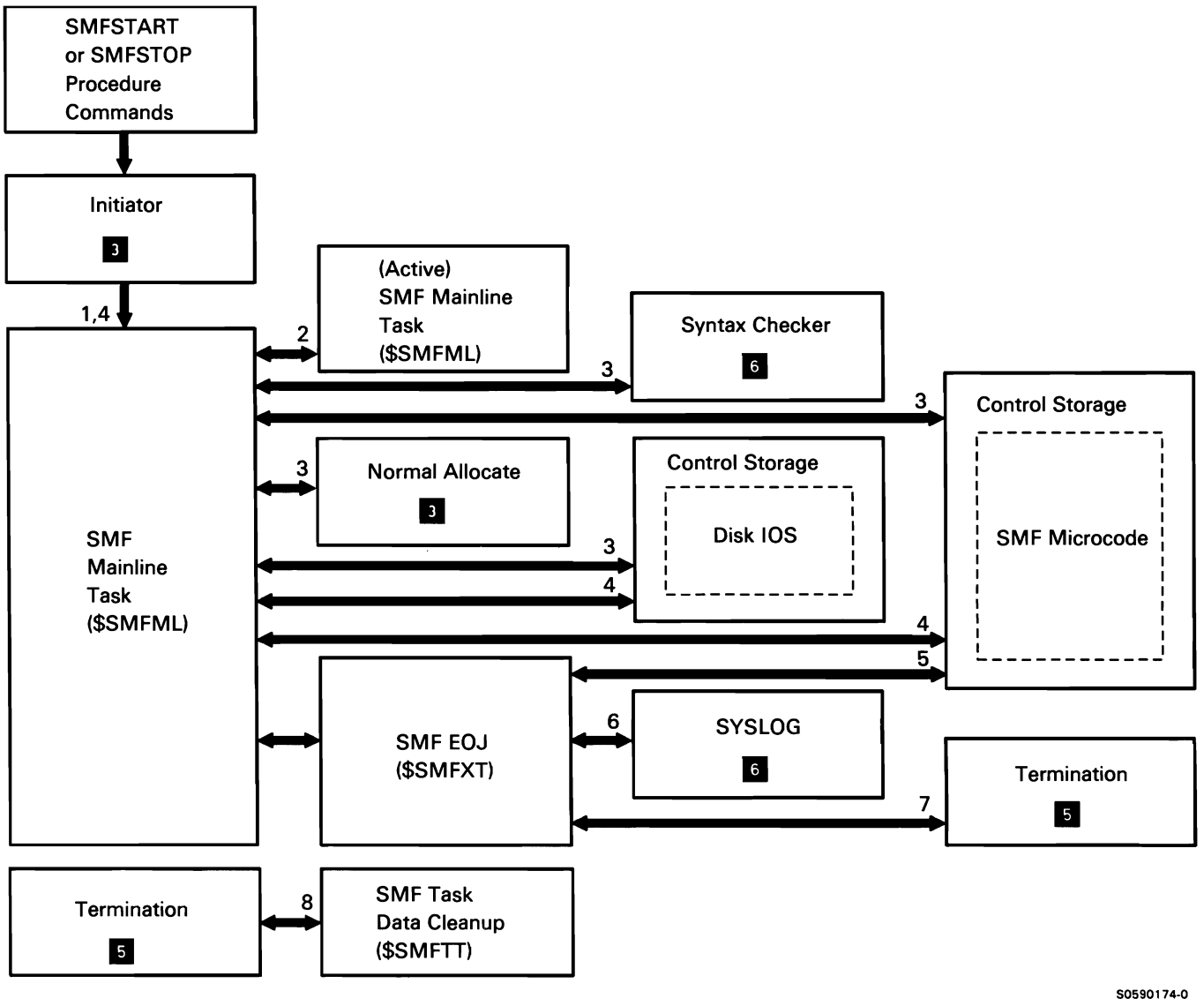


Chart 7.3.1 SMF Data Collection Control Flow

SMF Report Writer

The SMF report writer formats and prints the performance statistics accumulated by SMF data collection. The report writer is a PLS program that prints records retrieved from the SMF data file on disk. The SMF utility report writer is invoked by either the SMF or SMFPRINT procedure command.

The following SMF report writer processes are shown in Chart 7.3.2:

1 Perform initialization.

2 Read next logical record.

3 Check for limits processing.

4 Process logical record.

5 Retrieve heading line to print from #SM#M2 message member.

6 Print SMF report.

7 Terminate program.

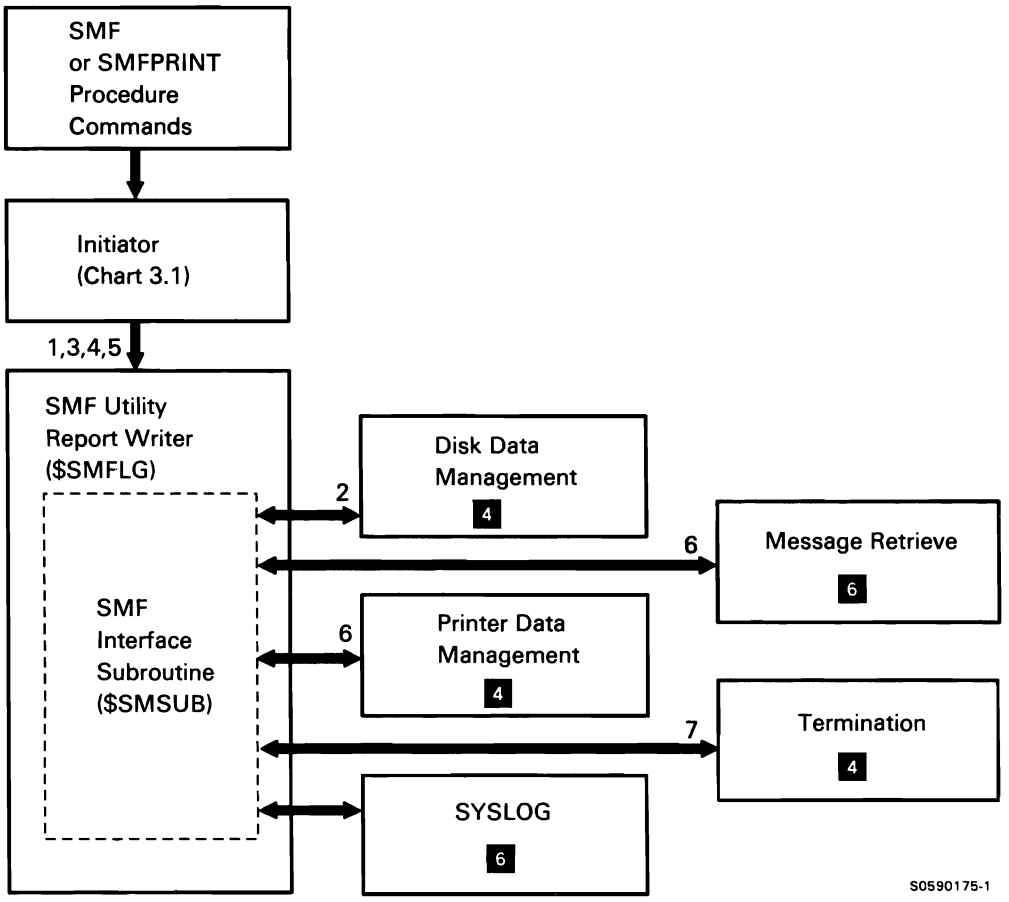


Chart 7.3.2 SMF Utility Report Writer Control Flow

SMF Report File Program

The SMF report file program formats and writes the performance statistics accumulated by SMF data collection to the report file. The report file program is a PLS program that writes records retrieved from the SMF data file on disk. The SMF utility report file program is invoked by either the SMF or SMFDATA file program.

The following SMF report file program processes are shown in Chart 7.3.3:

1 Perform initialization.

2 Read next logical record.

3 Check for processing limits.

4 Process logical record.

5 Write SMF report file.

6 Terminate program.

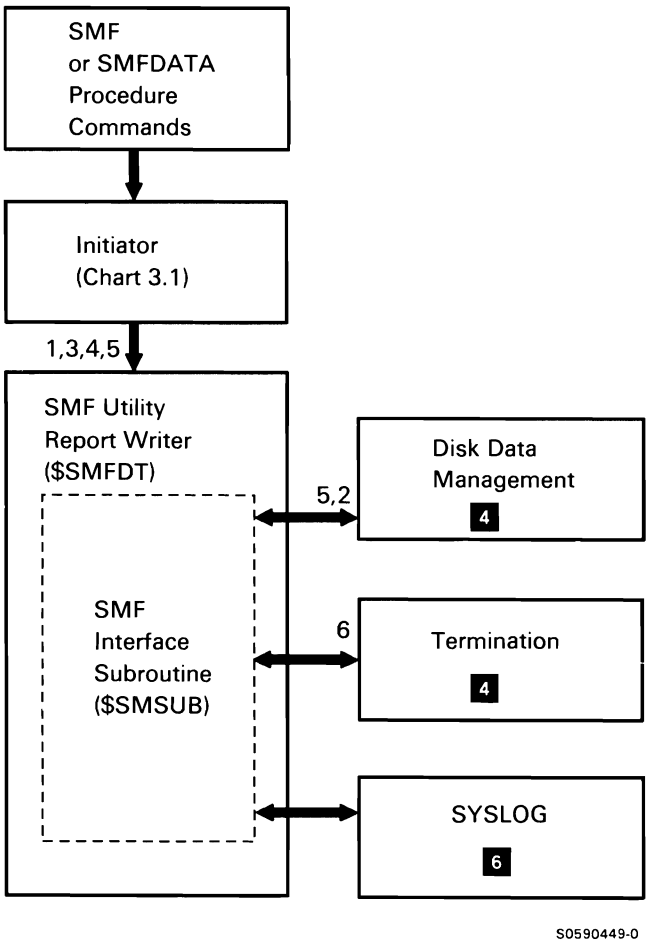


Chart 7.3.3 SMF Utility Report File Program Control Flow

SERVICE LOGGING SUBFUNCTION

The service logging subfunction records system maintenance-related information for use by the service representative. The information is recorded in #SERVLOG, a variable-length record, system file. Entries can be entered automatically by the configuration, IPL, task termination, and PTF apply/copy subfunctions, or manually, by the service representative, via the SERVLOG procedure. Service logging consists of service logging recording and service logging allocate/terminate subfunctions.

Information recorded by the service logging function is displayed or printed by the dump service log utility ($FESRV), which is described under the *Dump Utilities Subfunction.*

Service Log Recording Transient

The service logging recording transient is called by configuration, IPL (for overrides and system dump), task termination, and PTF apply/copy, to automatically record entries in the service log; service representatives can also make manual entries to the service log via the SERVLOG procedure. All callers invoke the recording transient via a transfer SVC instruction with RIB hex 39.

The following service log recording processes are shown in Chart 7.4.1:

1 Process user requests:

- If logging session does not exist, call allocate/deallocate.
- Assign buffer space.
- Add new record to end of file.
- Check service log for a wrap condition and update wrap table.
- Return to caller.

2 Allocate/deallocate service logging session.

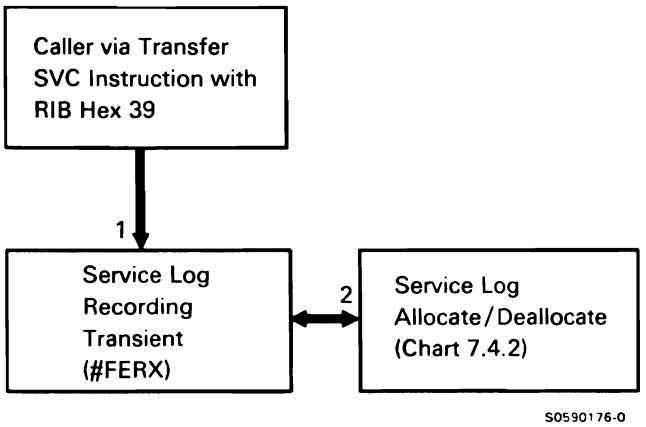


Chart 7.4.1 Service Log Recording Control Flow

Service Logging Allocate/Terminate

Service logging allocate/terminate processes requests from service log recording or from termination. Requests are made via a transfer SVC instruction with a RIB of hex 3A.

The following service logging allocate/terminate processes are shown in Chart 7.4.1:

1 Process request to allocate logging session.

2 Process request to terminate logging session.

3 Allocate the logging session:

- Allocate service log file (#SERVLOG).
- Build the service log control block.
- Increment use count in file format 1.
- Update current record pointers in service log control block.
- Deallocate the format 1.

4 Terminate the logging session:

- Reallocate the service log file.
- Update the current record pointers in the service log file.
- Decrement use count in format 1.
- Deallocate and remove the format 1.
- Free service log control block.

5 Return to caller.

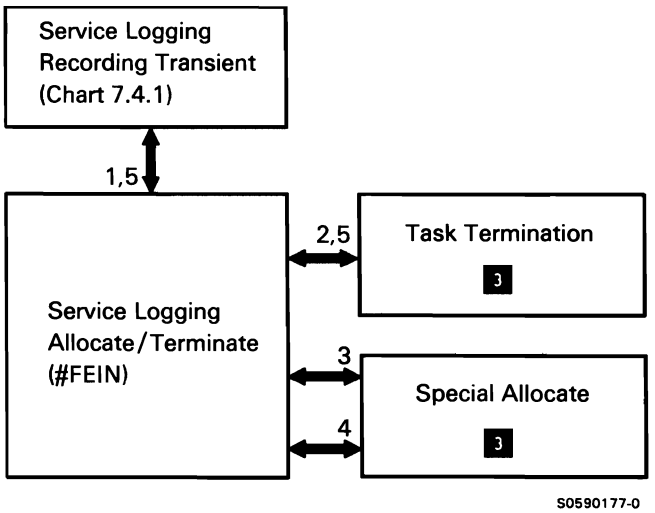


Chart 7.4.2 Service Logging Allocate/Terminate Control Flow

TEST REQUEST SUBFUNCTION

The test request subfunction prompts for and executes various work station testing functions. These include display/printer verification, configuration data confirmation, ERAP for work stations, keyboard data key verification, and link test. Test request is invoked by either the Test Request command key or by the TESTREQ command.

The following test request processes are shown in Chart 7.5:

1 Prompt and route for work station verification type requested.

2 Prompt for and execute appropriate display verification test (menu option 1):

- Display characteristics of a specified attribute.
- Display all displayable characters.
- Display characteristics and/or function on any input field.
- Test function keys.
- Display characteristics of a specified color attribute.

3 Prompt for and execute appropriate printer verification test (menu option 2):

- Prompt for test.
- Load the appropriate data areas module for printer type:
  - 5256 (#ST31).
  - 5224/5225 (#ST32).
  - 5224/5225 (#ST33).
  - 5219 (#ST34).
- Execute test.
- When canceled, return to #ST01.

4 Alternately display configuration data for local and online remote work stations (menu option 3); when canceled, return to #ST01.

5 Load and run ERAP (menu option 4).

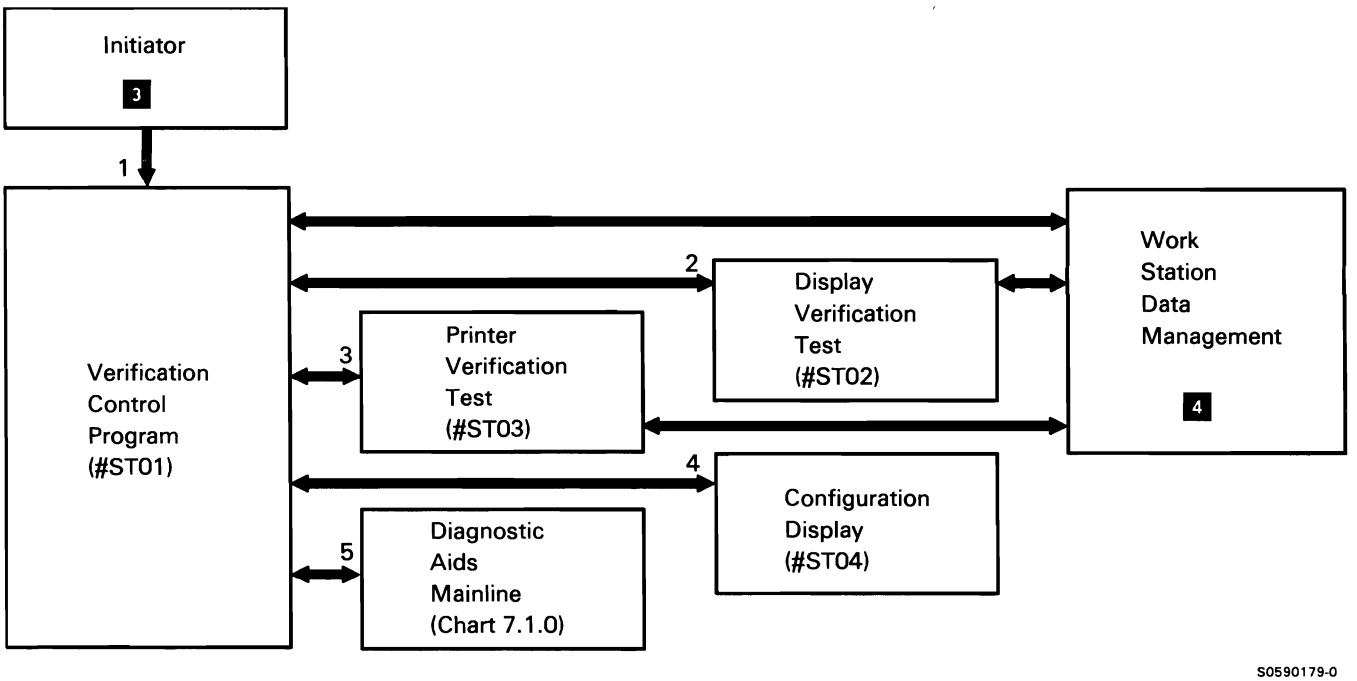


Chart 7.5 Test Request Control Flow

This page is intentionally left blank.

COMMUNICATIONS TEST SUBFUNCTION

The communications test subfunction prompts for and executes eight line communications attachment (ELCA), multiline communications attachment (MLCA), or single line communications attachment (SLCA) tests. If MLCA or ELCA is configured, communications test prompts for the line number the test is to be run on. The communications test is invoked by the COMMTEST procedure command.

The following communications test processes are shown in Chart 7.6:

1 Prompt and route for communications test functions requested:

- ELCA.
- MLCA.
- SLCA.

2 Set up for controller diagnostic test:

- Allocate all lines to prevent their use.
- Set up error recovery block (ERB) for wrap.
- Set error IOB and POSTI command processor for error recovery (forces wrap).

3 Perform requested MLCA/ELCA operation:

- BASIC communications diagnostic test.
- Link problem determination aids.
- Remote loopback test.

4 Perform SLCA BASIC communications diagnostic test processing.

5 Perform selected SLCA test processing:

- Link problem determination aids.
- Remote loopback test.

6 Allocate/deallocate specified lines and load appropriate microcode.

7 Print trace areas:

- Print disk trace area 2.
- Print disk trace area 1.

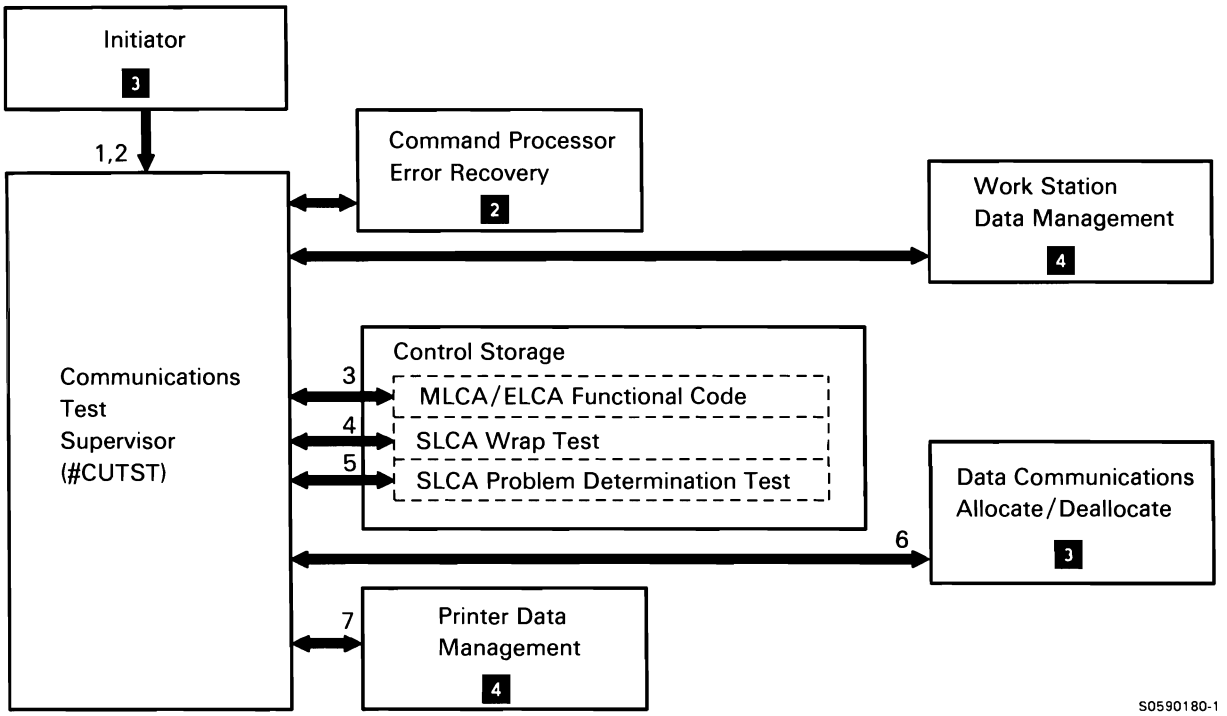


Chart 7.6 Communications Test Control Flow

ONLINE PROBLEM DETERMINATION (OLPD) SUBFUNCTION

The online problem determination (OLPD) subfunction presents information that is used by the customer in performing problem determination. OLPD is driven by OLPD control records; these records control the flow of OLPD, based on user responses to OLPD displays. For more information on OLPD control records, see *Online Problem Determination Control File* in the *System Data Areas* manual.

OLPD consists of procedures, data areas, files, and programs. The remainder of this topic describes only the programs, and the procedures, data areas, and files *directly* applicable to those programs.

The program portions of OLPD gather information for use by the customer in performing problem determination.

The OLPD driver is evoked via the PROBLEM procedure from either the command displays or from a help menu. The local data area (LDA) is used by all OLPD modules to pass data to other modules and by the OLPD driver to control program flow. (For more information on OLPD use of the LDA, see *Online Problem Determination Local Data Area* in the *System Data Areas* manual.) All OLPD-executable code is written in assembler language and RPG II and is stored, with its associated data areas in #ONLPD, the online problem determination library.

The following online problem determination processes are shown in Chart 7.7:

1 Determine online problem determination function required and route accordingly:

   - Set up LDA for function required.
   - Route control with return set in LDA.

2 Determine communications adapter type attached and store 4 bytes of configuration data in LDA to be added to 'next' field of control record.

3 Get line number for work station logical ID passed and build index for use in 'next' field of control record; store index in LDA.

4 Check to determine if previously issued VARY OFF command was successful; build index to be added to 'next' field of control record and store in LDA.

5 Check to determine if previously issued VARY ON command was successful; build index to be added to 'next' field of control record and store in LDA.

6 Get work station communications configuration from WSWA and and place 4 bytes of data in LDA for use by #OPFCH.

7 Using communications line configuration data placed in LDA by #OPFWA, build index for 'next' field of control record; store in LDA.

8 Set up for control storage remote communications tests and route control.

9 Based on logical work station ID passed, get and display work station address and work station port; if remote work station, get and display ID, address, and location description of associated controller (if any). Build index to be added to 'next' field of control record, based on information displayed, and store in LDA.

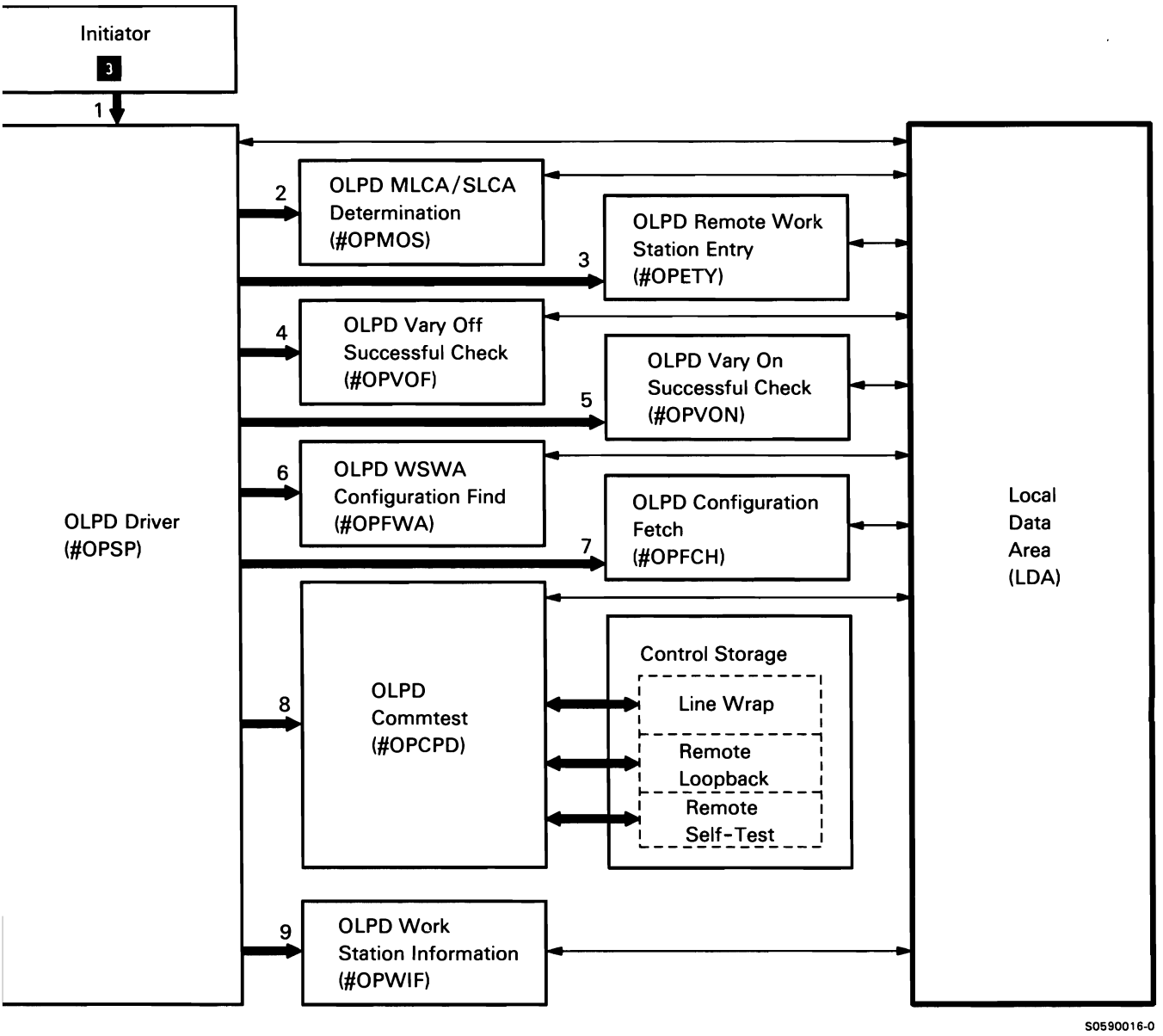


Chart 7.7 Online Problem Determination (OLPD) Control Flow

SSP System Utility Programs Function 8

The SSP system utility programs function consists of programs that the user invokes through SSP-supplied commands or procedures, command keys, function keys, or user-supplied OCL. Each set of load-run statements is accompanied by utility control statements that specify which utility options are requested and what user data is supplied as source for the utility. All system utilities run in user storage as user tasks.

The SSP system utility programs function includes the following utilities:

• $ARSP	Auto response	Chart 8.1
• $BICR	Basic exchange and l-exchange	Chart 8.2.0
• $BMENU	Build menu	Chart 8.3
• $BUILD	Sector rebuild	Chart 8.4
• $COPY	Disk copy/display	Chart 8.5.0
• $CPPE	Procedure error	Chart 8.6
• $DCOPY	Diagnostic diskette copy	Chart 8.37
• $DDST	Keysort	Chart 8.7
• $DELET	File delete	Chart 8.8
• $DUPRD	Diskette copy	Chart 8.9
• $FBLD	File build	Chart 8.10
• $HELP	Help	Chart 8.11
• $HIST	History file display	Chart 8.12
• $IDSET	SSP-ICF define ID	Chart 8.13
• $INIT	Diskette labeling and initialization	Chart 8.14
• $LABEL	VTOC display	Chart 8.15
• $MAINT	Library maintenance	Chart 8.16.0
• $MGBLD	Message build	Chart 8.17
• $MMSP	Stop BSC monitor	Chart 8.18
• $MMST	Start BSC monitor	Chart 8.19
• $PACK/ $FREE	Disk reorganization	Chart 8.20
• $PNLM	Define autocall phone list	Chart 8.21
• $POST	Post	Chart 8.22
• $PRxxx/ $RRxxx	Security support	Chart 8.23.n
• $RENAM	File rename	Chart 8.24
• $SETCF	Work station configuration	Chart 8.25
• $SETCP	Communications configuration	Chart 8.26
• $SFGR	Screen-format generator	Chart 8.27
• $UASC	Display spool file entries	Chart 8.28
• $UASF	User access to spool file	Chart 8.29
• $TINIT	Tape initialization	Chart 8.30
• $TCOPY	Tape copy	Chart 8.31.0
• $XREST	Extended character file restore	Chart 8.32
• $XSAVE	Extended character file save	Chart 8.33
• #SINR	Network resources directory	Chart 8.34
• $TMSERV	Folder management services (FMS) utility	Chart 8.35
• #DSIN	Interactive data definition utility (IDDU)	Chart 8.36

AUTO RESPONSE UTILITY ($ARSP)

The auto response utility updates message load members in a disk library based on input read from a source member. The utility finds the requested message in the message load member, inserts the requested auto response value and severity level, and writes the updated message back to disk. The auto response utility is invoked by the RESPONSE procedure or equivalent OCL, control, and specification statements.

The following auto response processes are shown in Chart 8.1:

1 Perform the following, routing where necessary:

- Process utility control statements.
- Get source member from user or #LIBRARY.
- Get specification statement from input source member.
- Find specified message, insert auto response value, and write message back to disk.
- Get next specification statement and repeat above process.
- Issue error messages as required.
- Terminate program.

2 Read and syntax check utility control statements.

3 Find requested user library.

4 Find requested source member and get from library.

5 Read specification statements from input source member.

6 Find specified message member.

7 Protect member while updating.

8 Read/write disk.

9 Issue requested error messages.

10 Terminate.

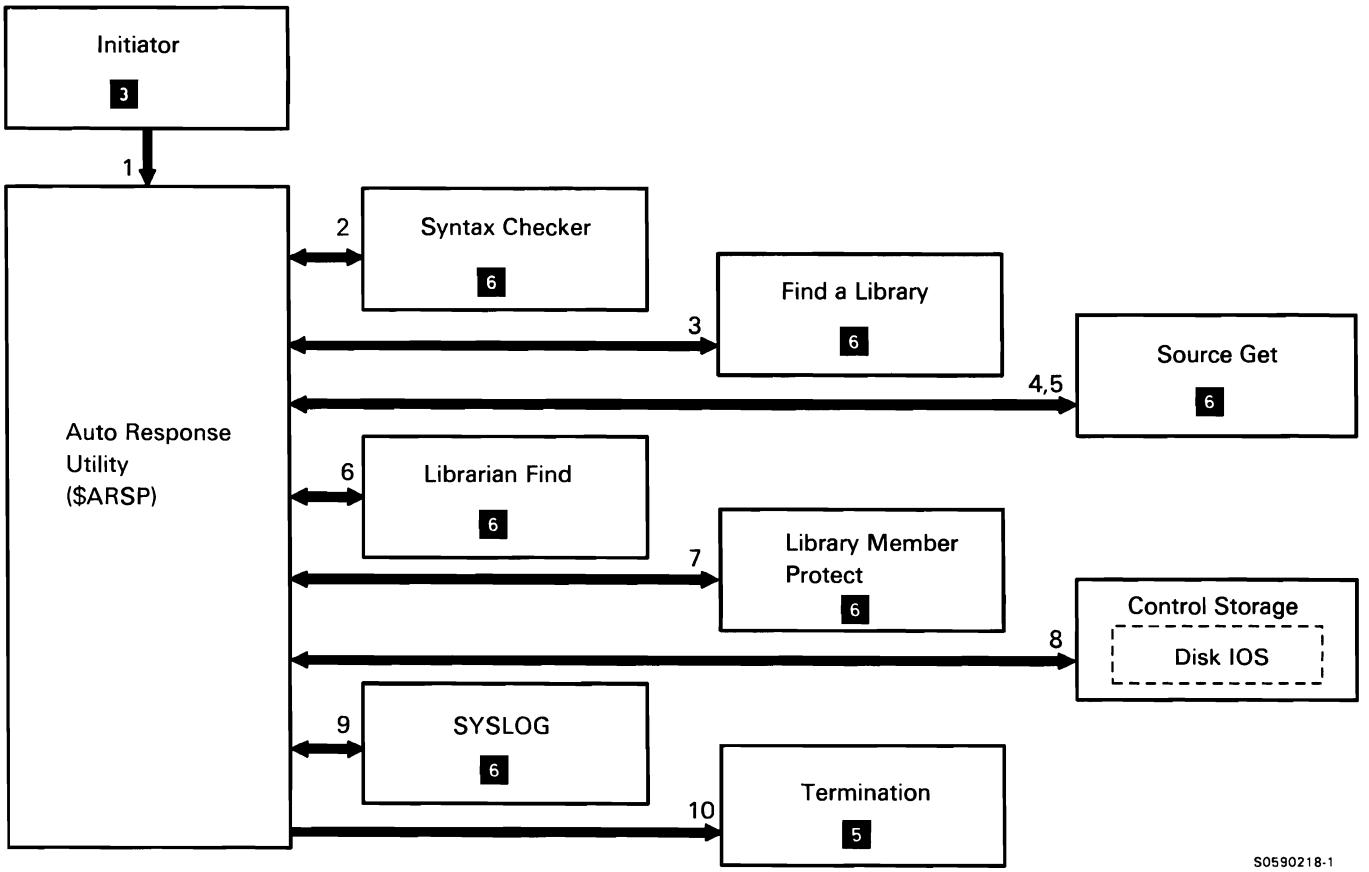


Chart 8.1 Auto Response Utility Control Flow

BASIC EXCHANGE AND I-EXCHANGE UTILITY ($BICR)

The basic exchange and I-exchange utility can convert either a disk file to basic or I-exchange file on diskette or a basic or I-exchange diskette file to a sequential or indexed disk file. It can also add a basic or I-exchange file to a sequential disk file or display all or part of a diskette basic or I-exchange file on the SYSLIST device. The basic exchange and I-exchange utility can be called by the TRANSFER procedure or by equivalent OCL statements.

The following basic exchange and I-exchange utility overview processes are shown in Chart 8.2.0:

1 Read and check syntax of control statements.

2 Get format 1 for diskette COPYIN file.

3 If requested, display basic exchange or I-exchange diskette file.

4 If requested, perform diskette-to-disk copy.

5 If requested, perform disk-to-diskette copy.

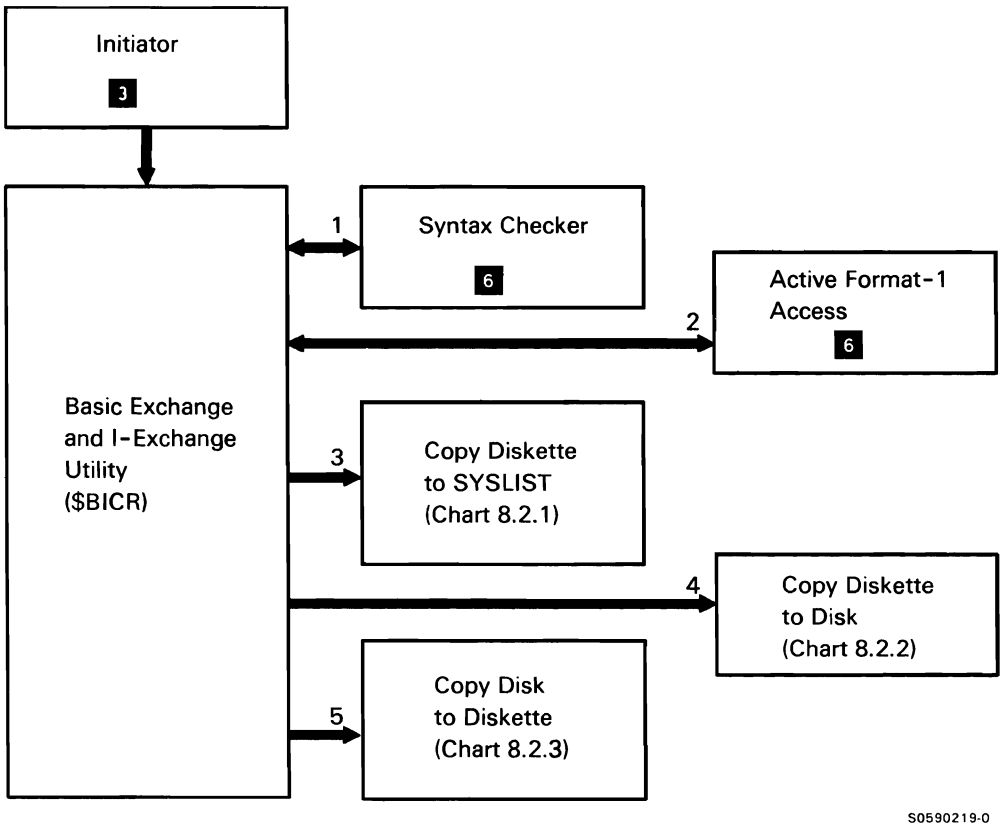


Chart 8.2.0 Basic Exchange and I-Exchange Utility Overview Control Flow

The following basic exchange and I-exchange utility diskette-to-SYSLIST copy processes are shown in Chart 8.2.1:

1 Perform basic exchange diskette to SYSLIST copy.

2 Perform I-exchange diskette to SYSLIST copy.

3 Find diskette file on magazine drive.

4 Allocate COPYIN file and open diskette DTF.

5 Read records.

6 If records are within limits specified, print/display output to SYSLIST device.

7 Issue any required messages.

8 Close diskette file and terminate this job step.

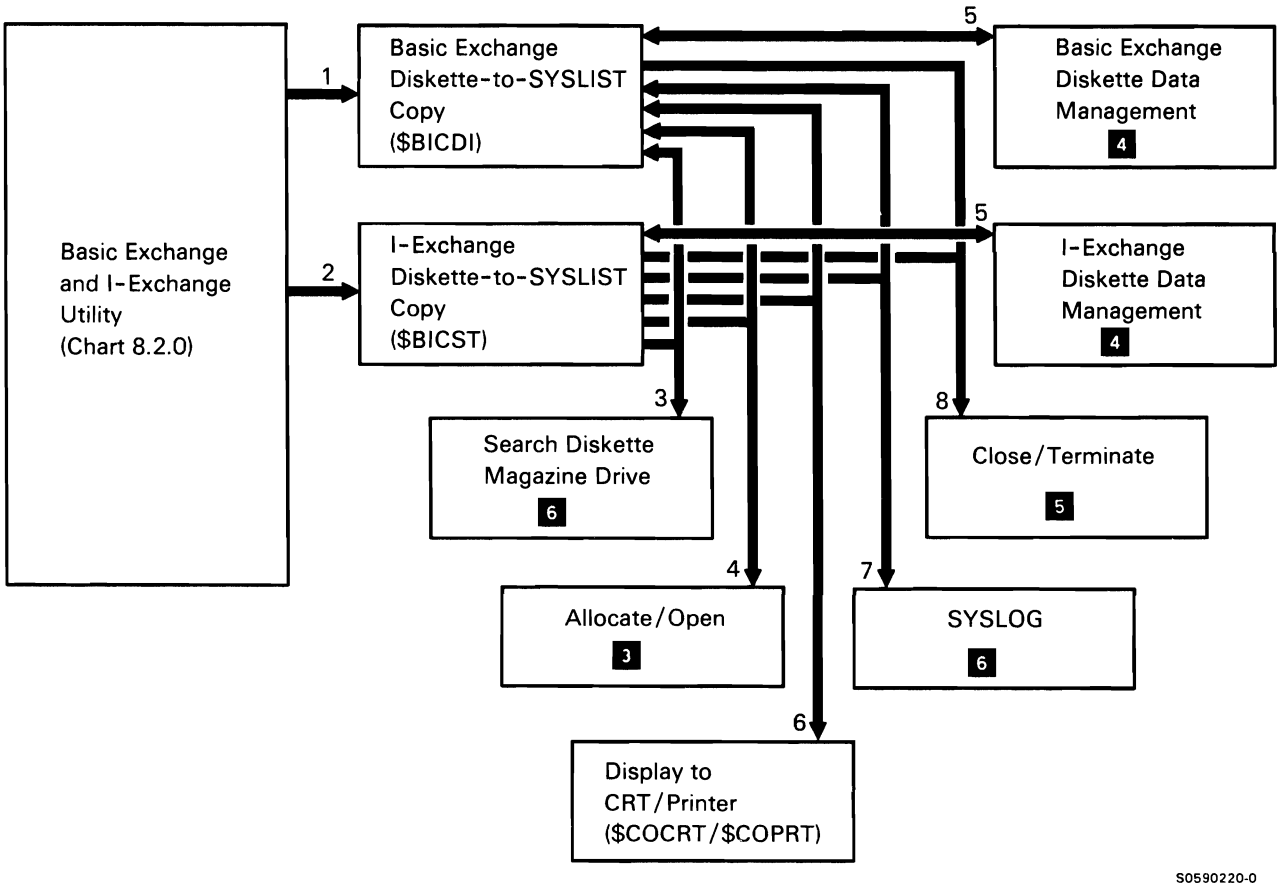


Chart 8.2.1 Basic Exchange and I-Exchange Utility Diskette-to-SYSLIST Display Control Flow

The following basic exchange and I-exchange utility diskette-to-disk copy processes are shown in Chart 8.2.2:

1 Perform basic exchange diskette to disk copy.

2 Perform I-exchange diskette to disk copy.

3 Get format 1 for diskette COPYIN file.

4 Find diskette file on magazine drive.

5 Allocate diskette input file, disk output file, and open DTFs.

6 Read diskette file records.

7 Write disk records.

8 Issue any required messages.

9 Close input and output files and terminate this job step.

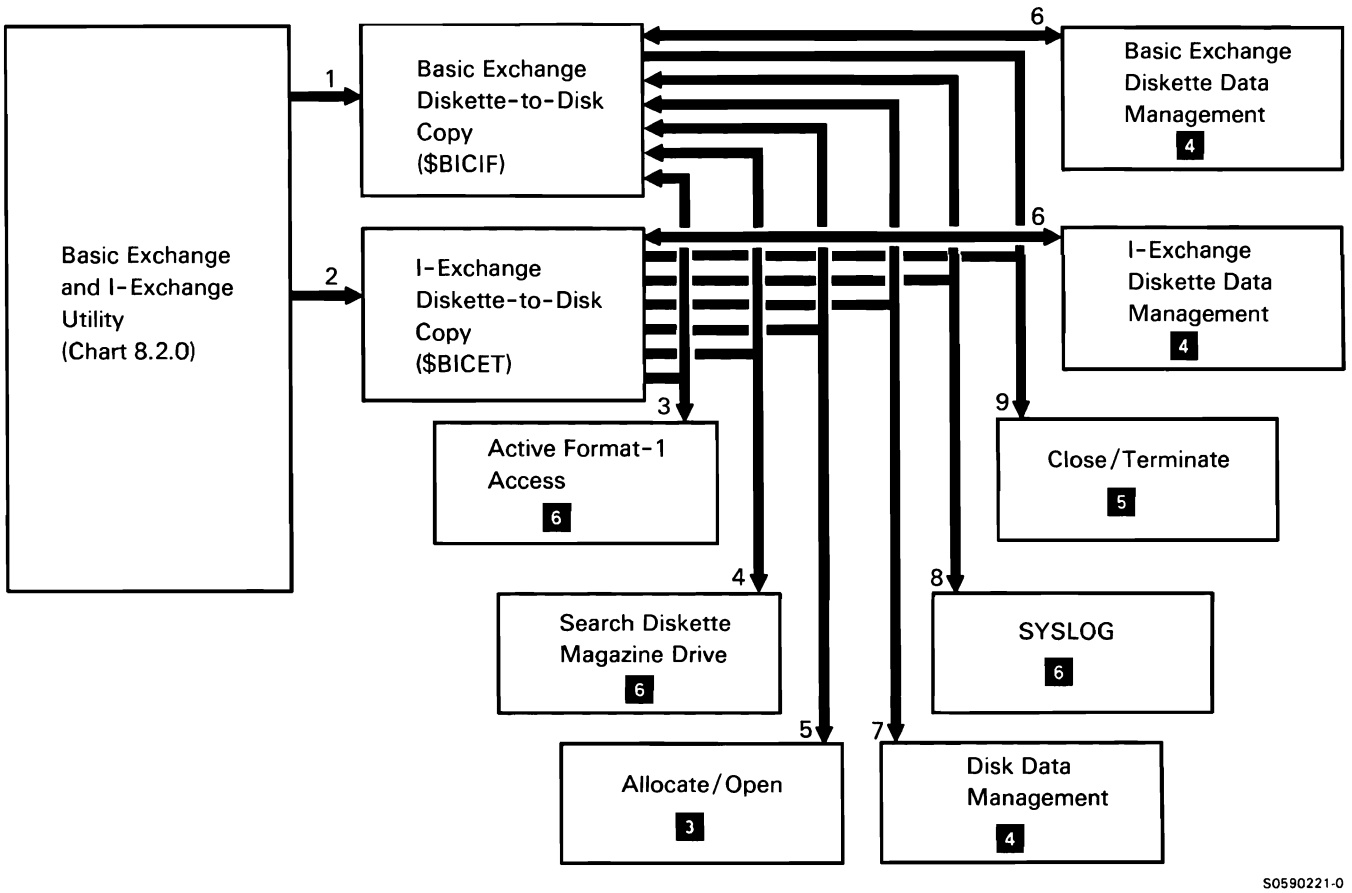


Chart 8.2.2 Basic Exchange and I-Exchange Utility Diskette-to-Disk Copy Control Flow

The following basic exchange and I-exchange utility disk-to-diskette copy processes are shown in Chart 8.2.3:

1 Perform disk to basic exchange diskette copy.

2 Perform disk to I-exchange diskette copy.

3 Get format 1 for disk COPYIN file.

4 Allocate disk input file, diskette output file, and open DTFs.

5 Find diskette file on magazine drive.

6 Read disk file records.

7 Write diskette file records.

8 Issue any required messages.

9 Close input and output files and terminate this job step.

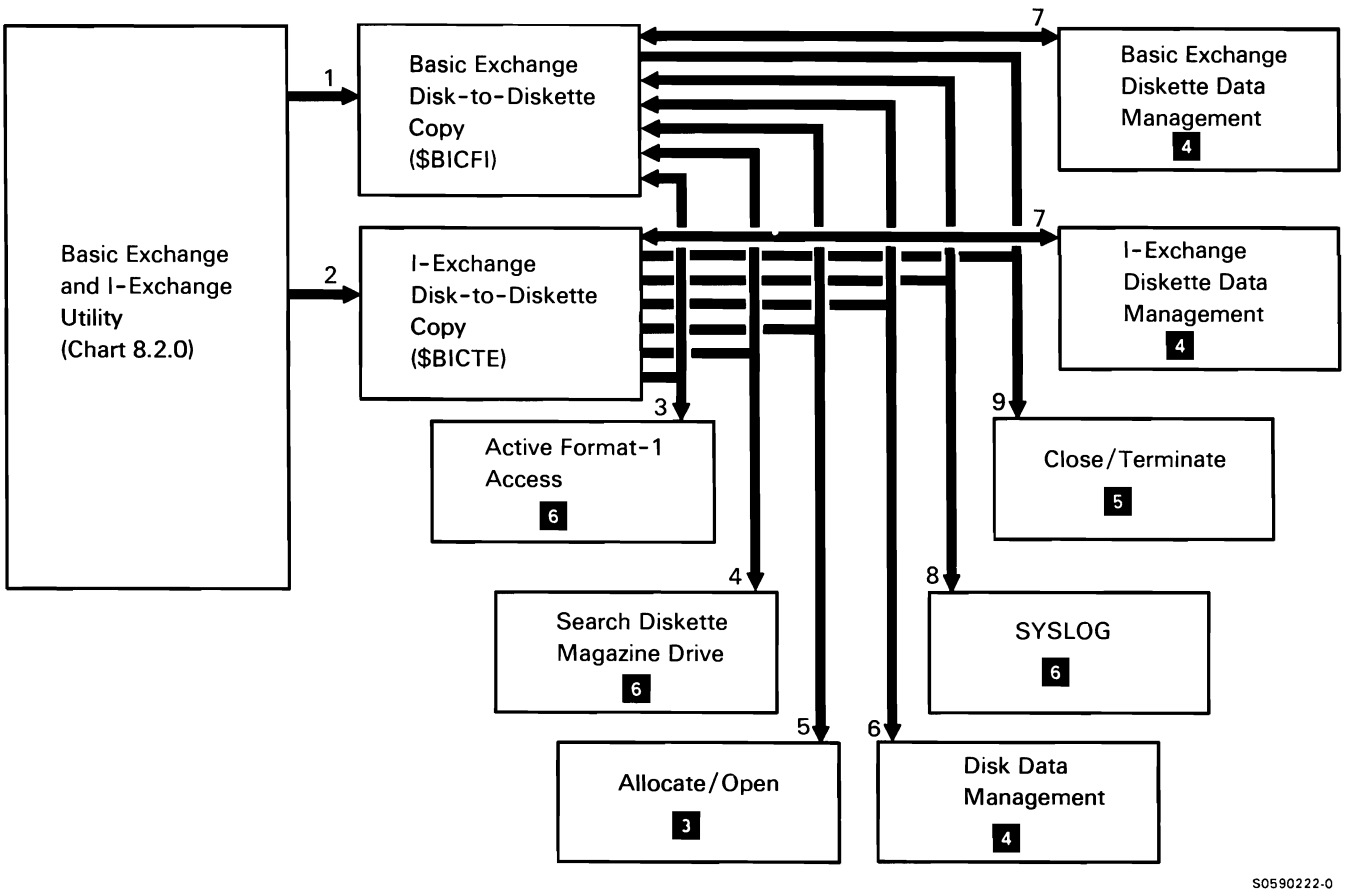


Chart 8.2.3 Basic Exchange and I-Exchange Utility Disk-to-Diskette Copy Control Flow

BUILD MENU UTILITY ($BMENU)

The build menu utility creates the library members required to display a menu. The user calls the utility with the BLDMENU procedure or with appropriate OCL and utility control statements.

The following build menu utility processes are shown in Chart 8.3:

1 Do the following, routing control where necessary:

- Process utility control statements.
- Set up the work file.
- Build source records.
- Build the screen format generator main storage communications area.
- Call the screen format generator utility to build required members for the menu.

2 Read and syntax check the utility control statements.

3 Prepare the work file to build the screen format generator source records.

4 Build temporary screen format source records reflecting the menu being built. Print the menu that was designed on the screen by the user.

5 Build the menu.

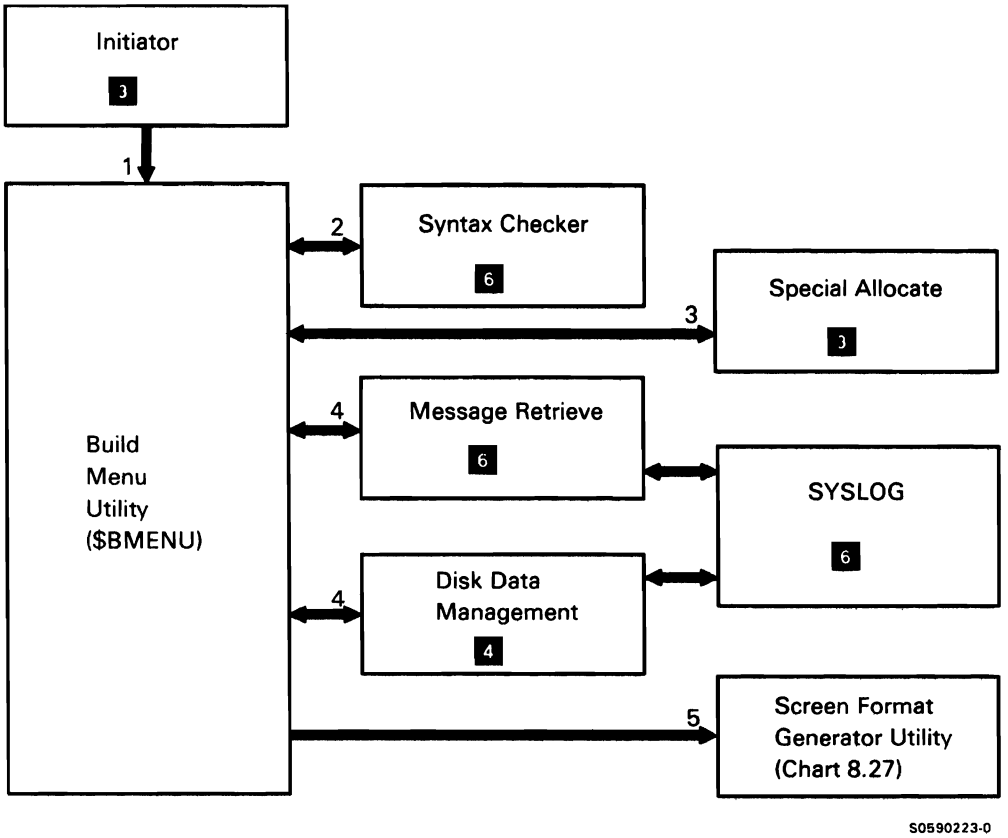


Chart 8.3 Build Menu Utility Control Flow

SECTOR REBUILD UTILITY ($BUILD)

The sector rebuild utility allows the user to correct possibly erroneous disk data. This data can result when a disk I/O error occurs and the data recovery attempts were not successful. The user invokes the sector rebuild utility with the BUILD procedure or appropriate OCL.

The following sector rebuild utility processes are shown in Chart 8.4:

1 Issue printer forms change message to system operator.

2 Find and load $FEKEY or $FEIGC.

3 Load page heading messages.

4 Read verify all sectors on disk. If any possibly defective sectors are found, do the following, remaining processes.

5 Read VTOC format-1 records to determine if the sector is in a file.

6 On SYSLIST, list the sector ahead of the possibly defective one, the possibly defective sector, and the sector following the possibly defective one.

7 Display the possibly defective sector to the operator, allowing for modification of the data.

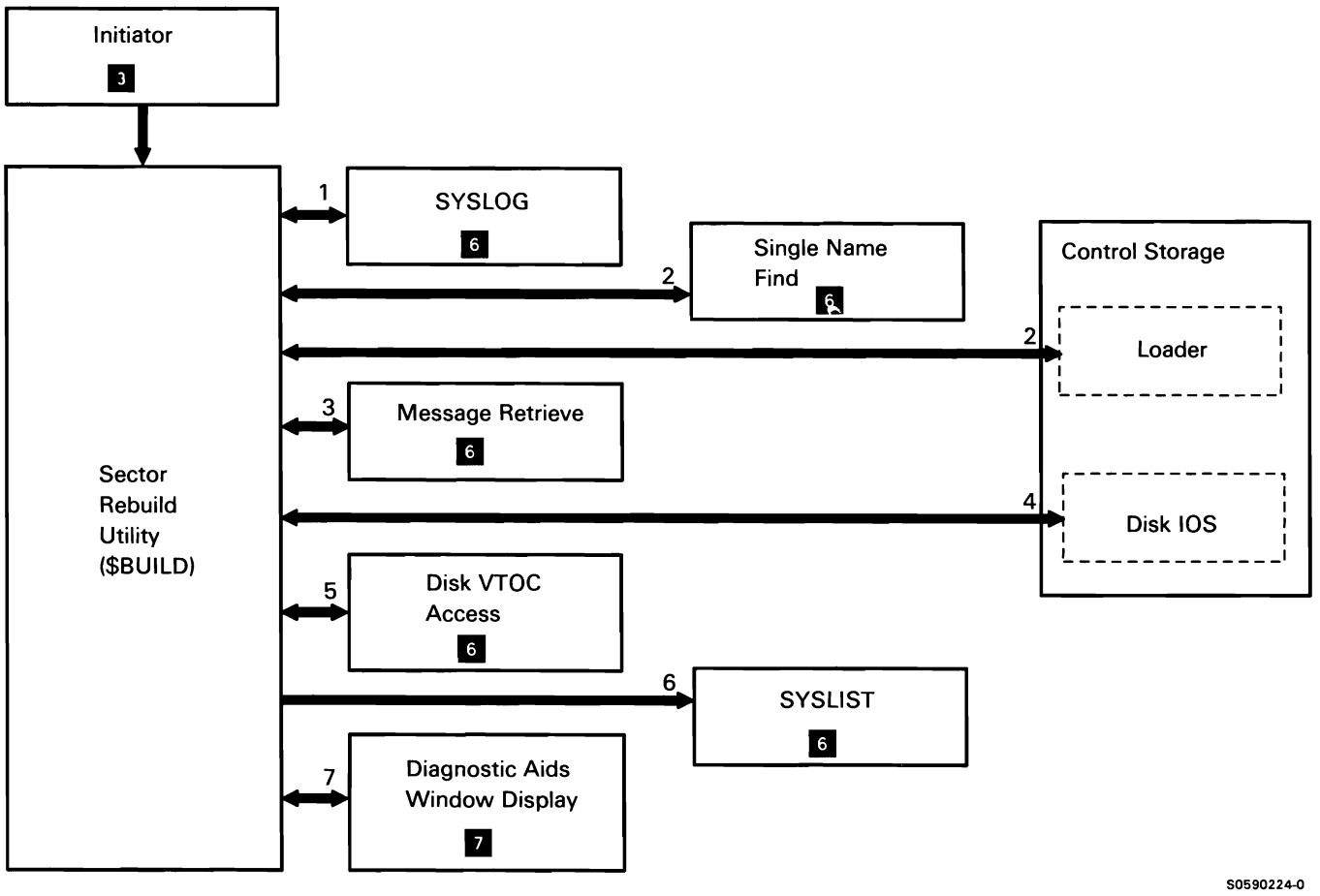


Chart 8.4 Sector Rebuild Utility Control Flow

DISK COPY/DISPLAY UTILITY ($COPY)

The disk copy/display utility copies requested files between disk and disk, diskette, or tape. The utility copies a requested portion of a file between disk and disk or diskette. The utility also copies a requested file or portion of a file from disk, diskette, or tape to the SYSLIST device. If a file is defined in the network resource directory (NRD), the requested operation is done in record mode and the data is copied to or from the remote system. The utility is invoked by appropriate procedures or OCL and utility control statements.

The following copy/display overview processes are shown in Chart 8.5.0:

1 Read and check the syntax of utility control statements.

2 If required, allocate printer or SYSLIST device.

3 Issue any required messages.

Determine type of copy required and route control accordingly:

4 Process for record mode and remote file copy.

5 Add disk file to existing diskette file.

6 Process for disk-to-disk high-speed copy.

7 Process for diskette-to-disk high-speed copy.

8 Process for tape-to-disk high-speed copy.

9 Process for disk-to-diskette high-speed copy.

10 Process for disk-to-tape high-speed copy.

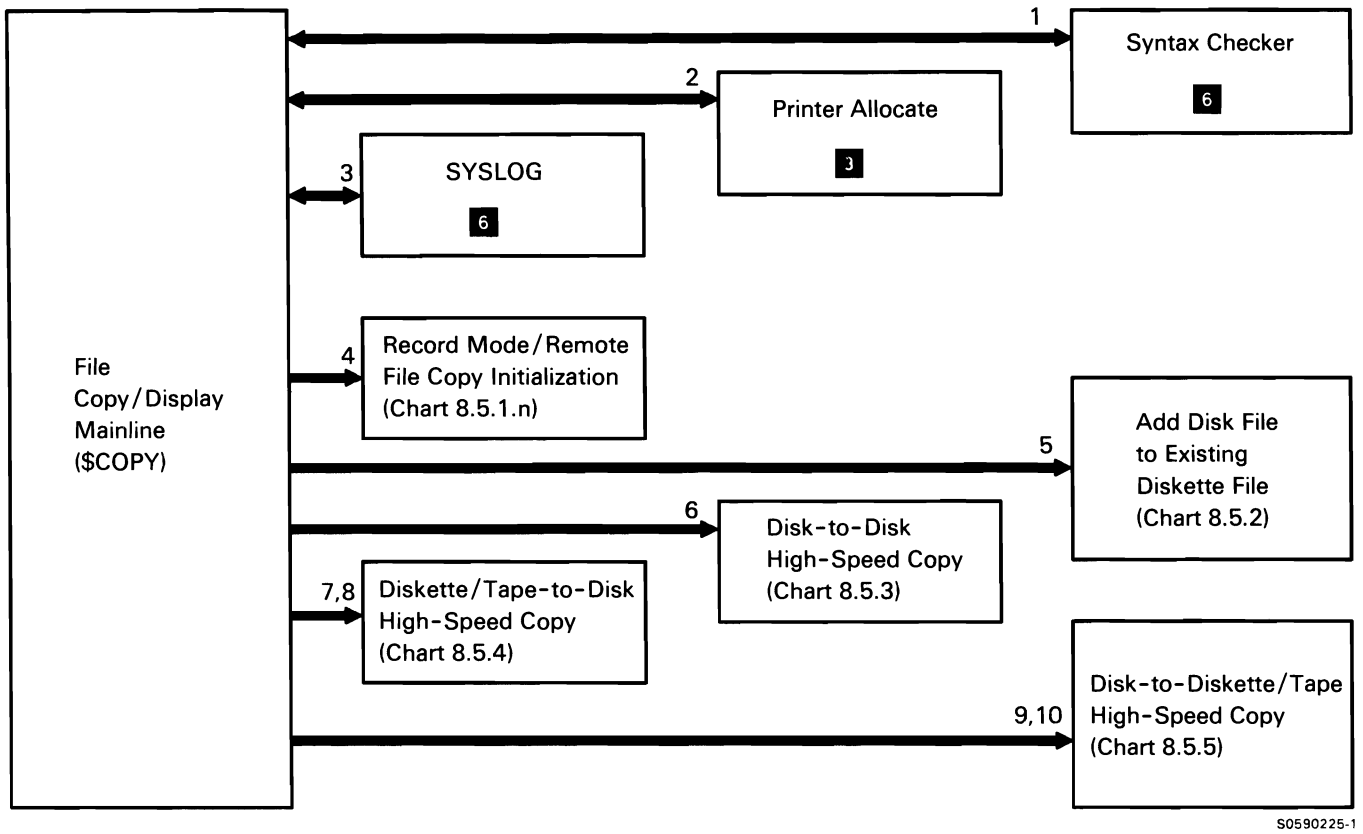


Chart 8.5.0 Disk Copy/Display Utility Overview Control Flow

Record Mode/Remote File Copy

Record mode/remote file copy performs the copy by file record to allow analysis of individual records for any of the following purposes:

- File reorganization
- Display or printing of selected records
- Record selection based on content, key value (for indexed files), or position in input data file
- Remote file processing directly or indirectly through distributed data management (DDM)

The following record mode/remote file copy input initialization processes are shown in Chart 8.5.1.1:

1 Check if remote system supports DDM remote file transfer operations.

2 Allocate diskette drive and search for specific diskette.

3 Allocate input disk, diskette, or tape file.

4 Open input diskette file.

5 Get additional main storage for tape cartridge.

6 Lock data dictionary and get file definition level.

7 Get data dictionary information from disk VTOC.

8 Issue error messages.

9 Close input diskette file (depending on error).

10 Terminate this job step (depending on error).

11 Load and route control to the process for record mode/remote file copy output initialization.

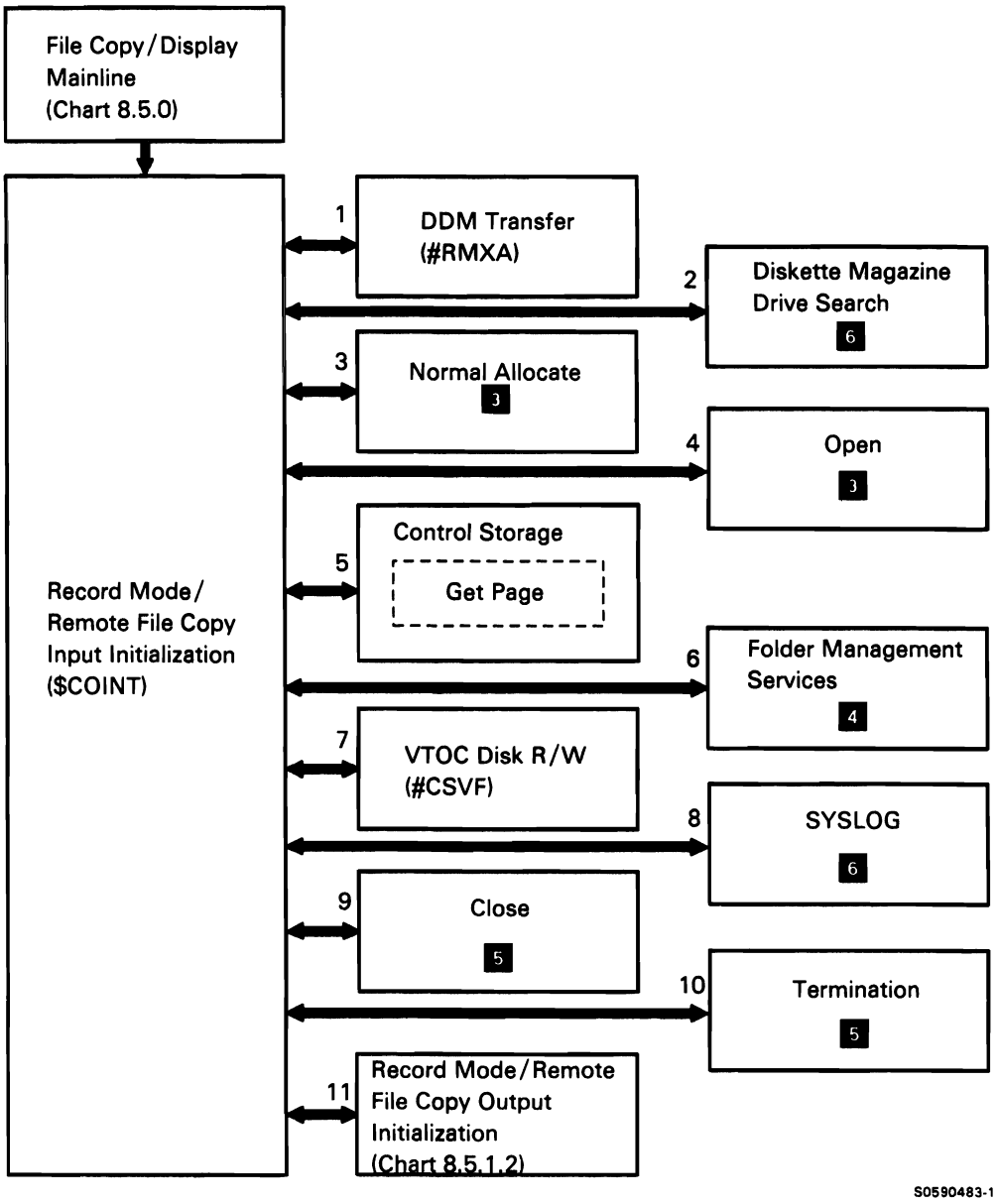


Chart 8.5.1.1 Record Mode/Remote File Copy Input Initialization Control Flow

The following record mode/remote file copy output initialization processes are shown in Chart 8.5.1.2:

1 Special allocate the output disk file.

2 Write output disk file format 1 to VTOC.

3 Close data dictionary.

4 Allocate diskette drive and search for specific diskette.

5 Allocate output diskette file.

6 Open output diskette file.

7 Get data dictionary information from disk VTOC.

8 Issue error messages through SYSLOG or the DDM remote message handler.

9 Close input or output diskette file (depending on error).

10 Terminate this job step (depending on error).

11 Open input and/or output disk file(s); open input tape file.

12 Determine the type of copy and route control accordingly:

- Process for record mode/remote file copy with I/O interface through disk data management.
- Process for remote file copy through direct I/O interface with distributed data management (DDM).

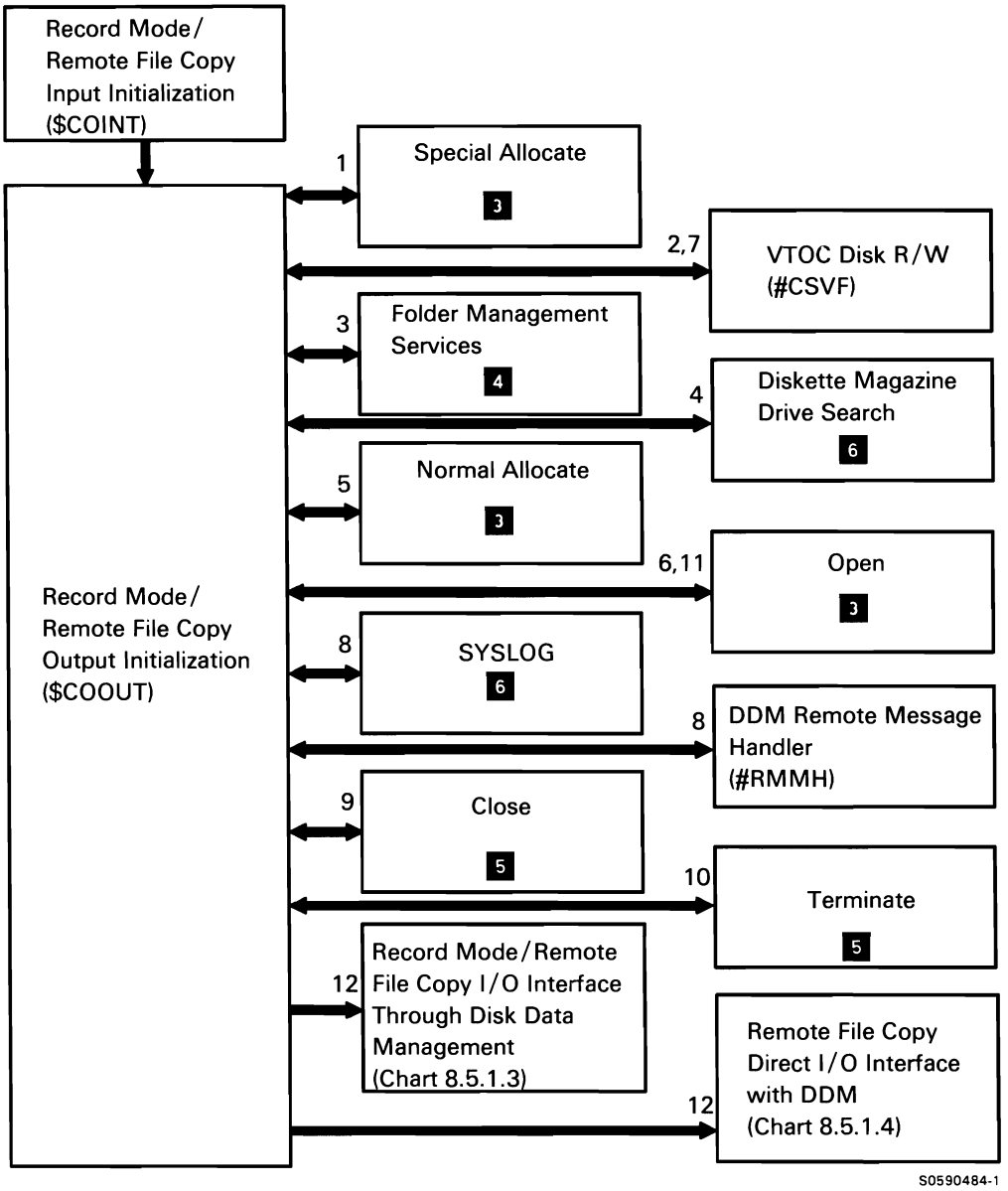


Chart 8.5.1.2 Record Mode/Remote File Copy Output Initialization Control Flow

The following record mode/remote file copy I/O interface through disk data management processes are shown in Chart 8.5.1.3:

1 Get and/or put records from/to local/remote disk file.

2 Get or put records from/to diskette file.

3 Get records from tape file.

4 Process a record for printing, if requested.

5 On first call to $COPRT, get the printer image.

6 Print records.

7 Process a record for display to the CRT, if requested.

8 Display records to the CRT.

9 Process records from $COGET and direct $COGET in performing WSU-required processing.

10 Determine if record should be included or omitted from output.

11 Issue error messages through SYSLOG or the DDM remote message handler.

12 Close the disk, diskette, or tape file(s).

13 Terminate this job step.

Note: When processing a remote disk file, the I/O processes (allocate, open, get, put, close) interface with DDM to perform the operations on the remote system.

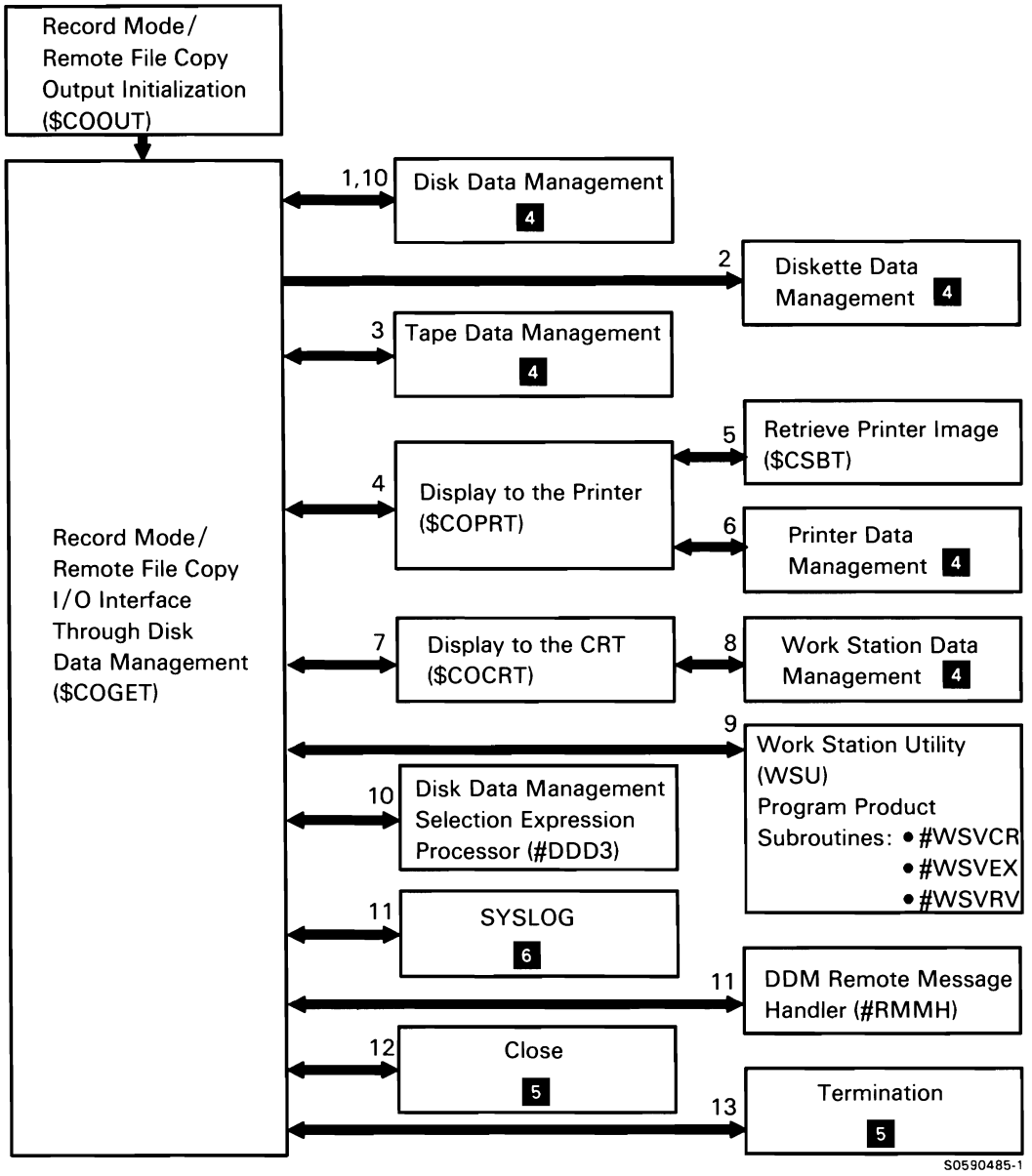


Chart 8.5.1.3 Record Mode/Remote File Copy I/O Interface Through Disk Data Management Control Flow

The following remote file copy direct I/O interface with distributed data management (DDM) processes are shown in Chart 8.5.1.4:

1 Open DDM remote file transfer operation(s).

2 Get or put records directly from/to remote disk file.

3 Get or put records from/to local disk file or indirectly from/to remote disk file.

4 Get or put records from/to diskette file.

5 Process a record for printing, if requested.

6 On first call to $COPRT, get the print image.

7 Print records.

8 Process a record for display to the CRT, if requested.

9 Display records to the CRT.

10 Determine if record should be included or omitted from output.

11 Issue error messages through SYSLOG or the DDM remote message handler.

12 Close DDM remote file transfer operations.

13 Close the disk or diskette file.

14 Terminate this job step.

Note: When processing a remote disk file indirectly through disk data management, the I/O processes (allocate, open, get, put, close) interface with DDM to perform the operations on the remote system. When processing a remote disk file directly through DDM remote file transfer operations (except for file allocation), $COPY directly interfaces with DDM to perform the operations on the remote system.

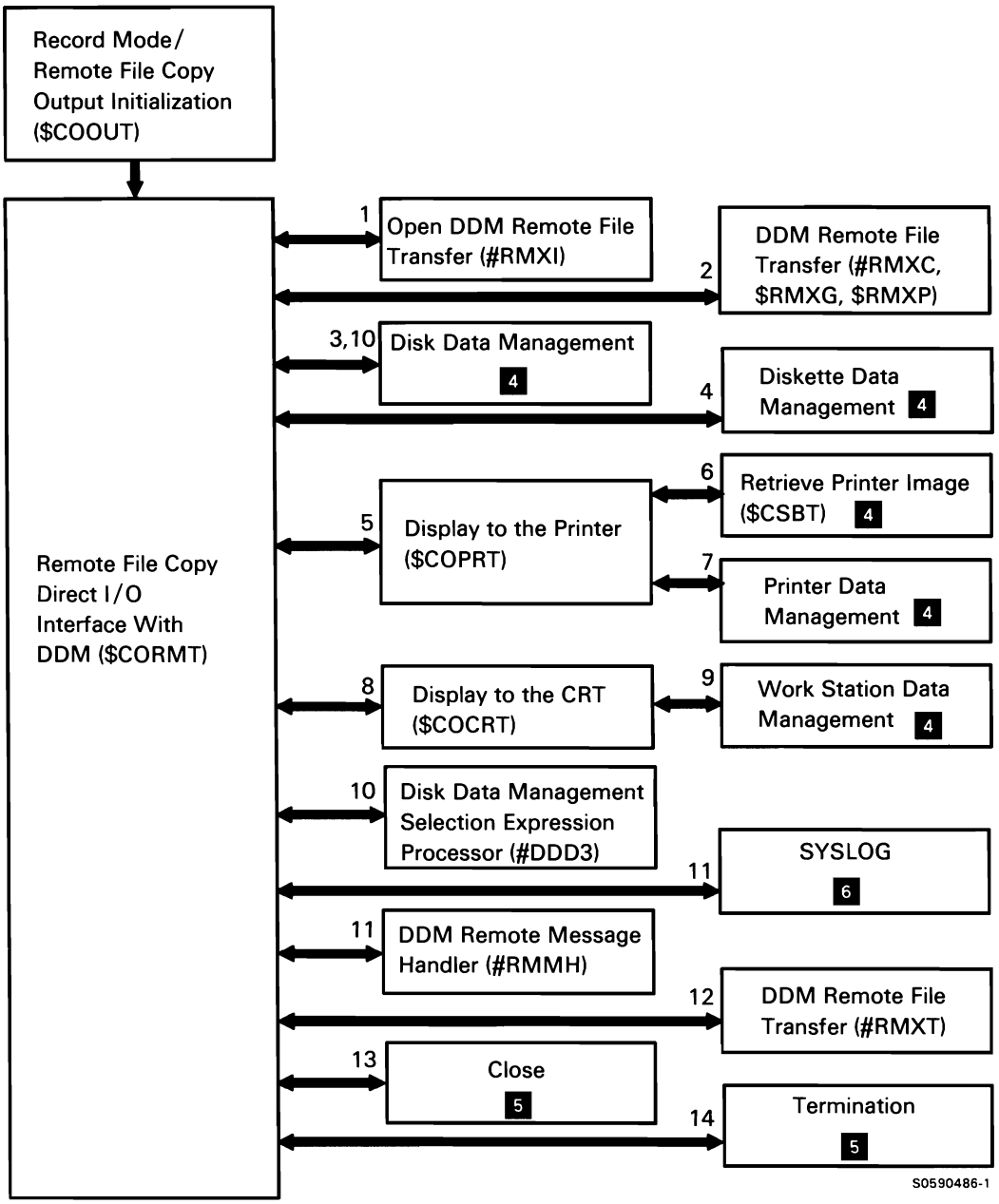


Chart 8.5.1.4 Remote File Copy Direct I/O Interface with DDM Control Flow

Add Disk File to Diskette File

The add disk file to diskette file subfunction adds an existing disk file to an existing diskette file. Control is routed to this subfunction from disk copy/display mainline, based on requests received from the user.

The following add disk file to diskette file processes are shown in Chart 8.5.2:

1 Allocate diskette drive and search for specific diskette.

2 Allocate disk and diskette files.

3 Open disk and diskette files.

4 Get sectors of data from disk file.

5 Put sectors of data to diskette file. (Repeat processes 4 and 5 until all the data has been copied to diskette.)

6 Issue error messages if necessary.

7 Close disk and diskette files.

8 Terminate this job step.

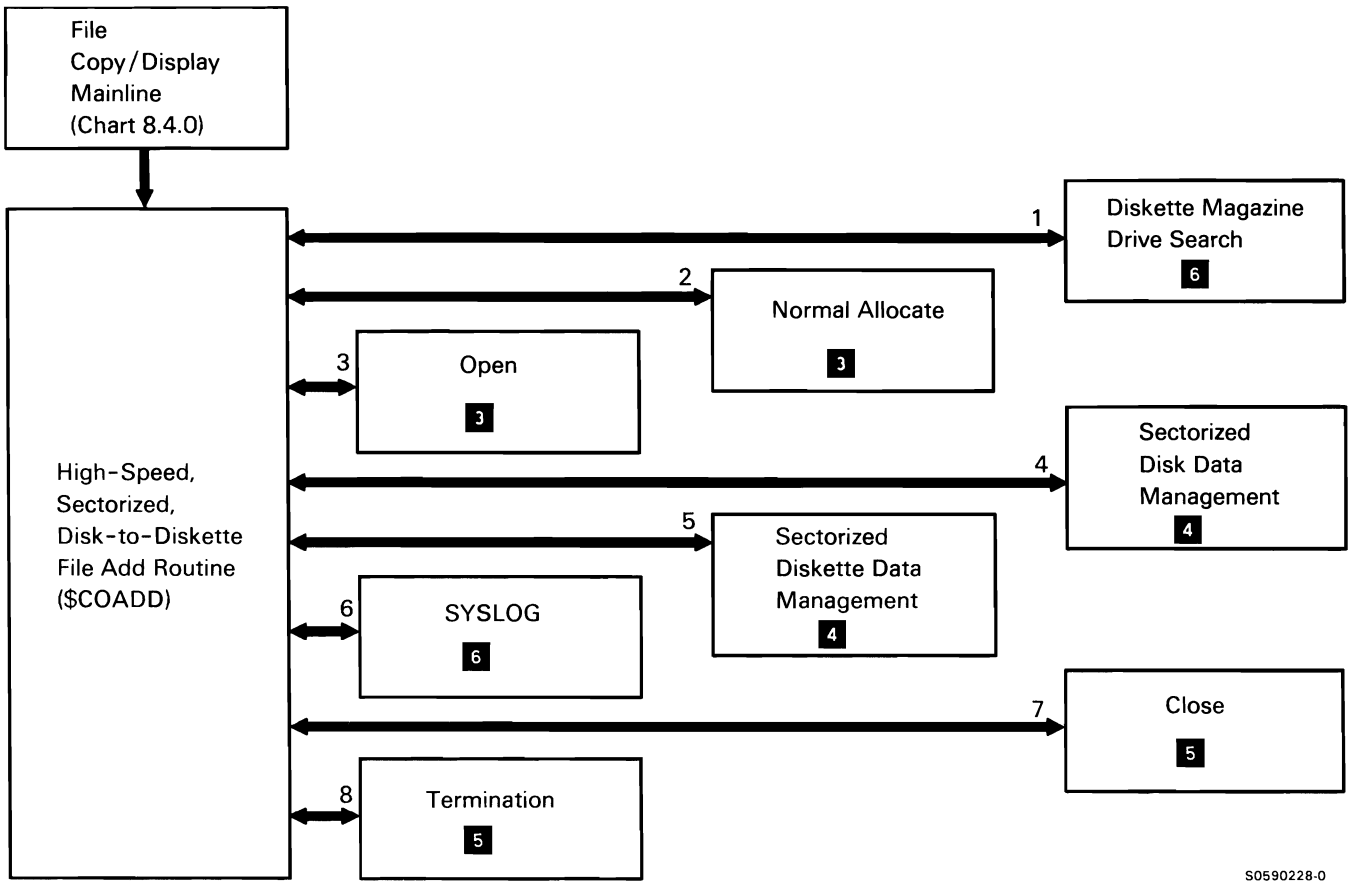


Chart 8.5.2 Add Disk File to Diskette File Control Flow

High-Speed Disk File to Disk File Copy

The high-speed disk file to disk file copy subfunction copies one disk file to another. Control is routed to this subfunction from disk copy/display mainline, based on requests received from the user. High-speed disk file to disk file copy copies multiples of 256-byte sectors without regard to individual records; it interfaces directly to control storage disk IOS, without going through disk data management.

The following high-speed disk file to disk file copy processes are shown in Chart 8.5.3:

1 Allocate input disk file.

2 Special allocate output disk file.

3 Copy sectors of data from input to output file.

4 If indexed output file, extract keys and build index.

5 Issue error messages if necessary.

6 Terminate this job step.

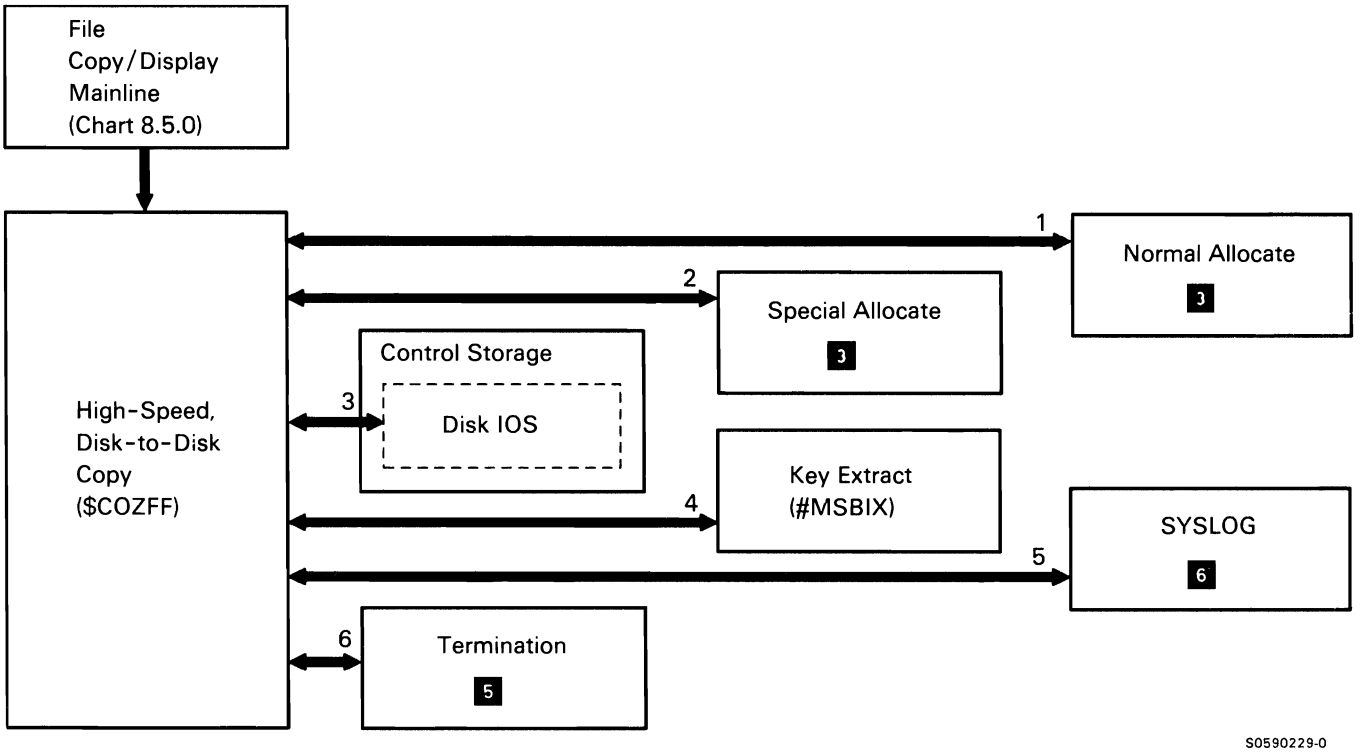


Chart 8.5.3 High-Speed Disk File to Disk File Copy Control Flow

High-Speed Diskette/Tape File to Disk File Copy

The high-speed diskette/tape file to disk file copy subfunction copies diskette or tape files to disk in sector mode. Control is routed to this subfunction from disk copy/display mainline, based on requests received from the user. High-speed diskette/tape file to disk file copy is used to restore a diskette/tape file(s) to disk; it uses disk and diskette sector data managements or the data storage controller to copy multiples of 256-byte sectors without regard to individual records. It also acquires the necessary work space for diskette data compression.

The following high-speed diskette/tape file to disk file copy processes are shown in Chart 8.5.4:

1 Allocate and prepare multiple diskette drive slots.

2 Prepare single slot diskette drive.

3 Normal allocate input diskette drive, tape drive, and diskette data compression control storage microcode.

4 Special allocate output disk file.

5 Copy sectors of data from diskette to disk.

6 Copy sectors of data from tape to disk.

7 Create alternative index file(s) if required.

8 Extract keys to build index portion of indexed files.

9 Deallocate disk file if this is a multiple file restore operation (repeat processes 2 through 8 until all files are restored).

10 Issue any required error messages.

11 Deallocate the diskette data compression control storage microcode.

12 Terminate this job step.

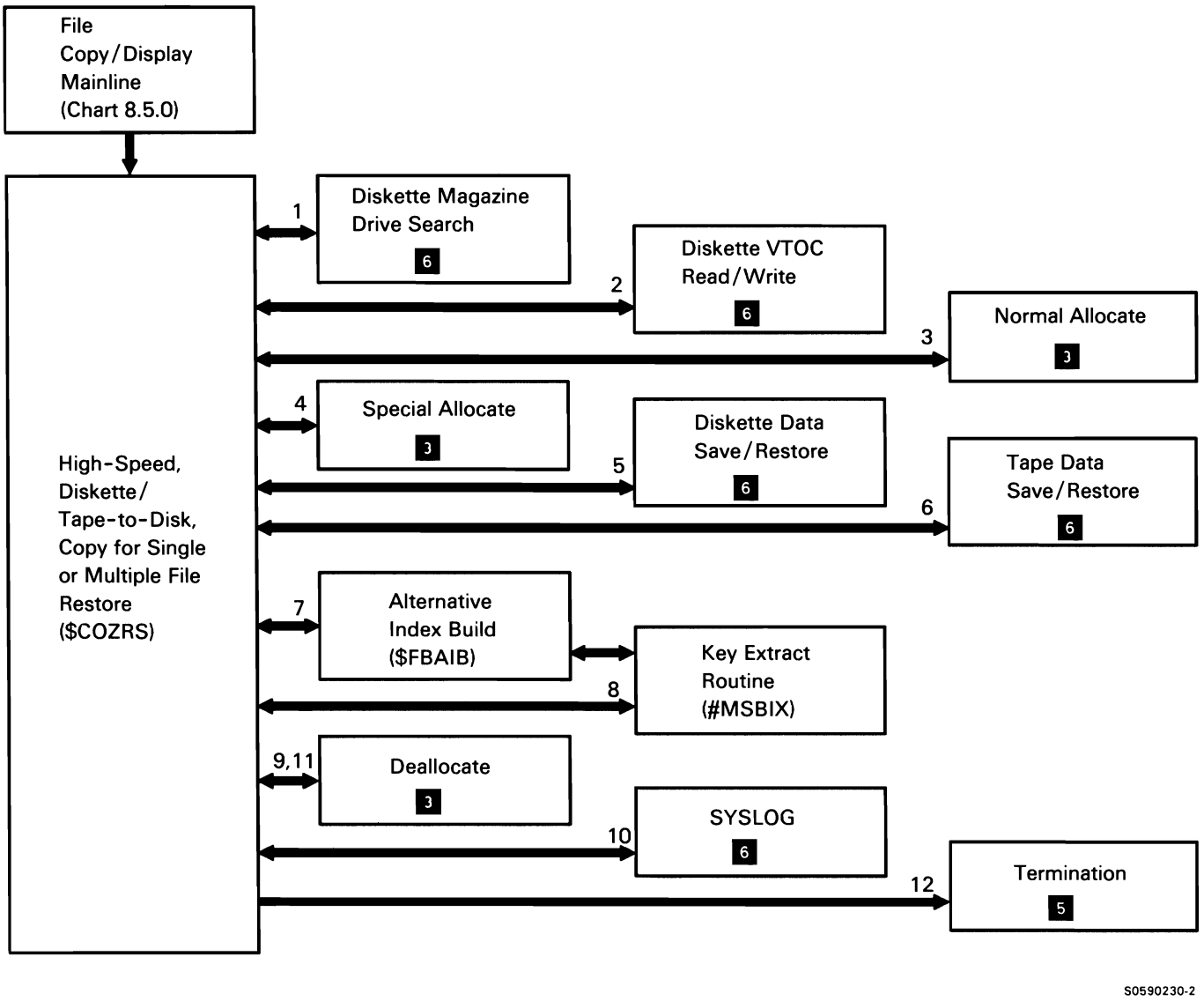


Chart 8.5.4 High-Speed Diskette/Tape File to Disk File Copy Control Flow

High-Speed Disk File to Diskette/Tape File Copy

The high-speed disk file to diskette/tape file copy subfunction copies disk file(s) to diskette or tape in sector mode. Control is routed to this subfunction from disk copy/display mainline, based on requests received from the user. High-speed disk file to diskette/tape file copy is used to save a disk file(s) on diskette or tape; it uses disk and diskette sector managements or the data storage controller to copy multiples of 256-byte sectors without regard to individual records. It also acquires the necessary work space for diskette data compression.

The following high-speed disk file to diskette/tape file copy processes are shown in Chart 8.5.5:

1 If SAVE ALL was specified, find the first disk file to save.

2 Allocate the disk file.

3 If diskette copy, search for the specified diskette to start the save.

4 If diskette copy, prepare the diskette if the system only has one diskette slot.

5 Normal allocate output diskette drive, tape drive, and diskette data compression control storage microcode.

6 If diskette copy, copy sectors of data from disk to diskette.

7 If tape copy, copy sectors of data from disk to tape.

8 Deallocate the disk file if this is a multiple file save. (Repeat processes 1 through 8 until all specified files are saved.)

9 Issue any required error messages.

10 Deallocate the diskette data compression control storage microcode.

11 Terminate this job step.

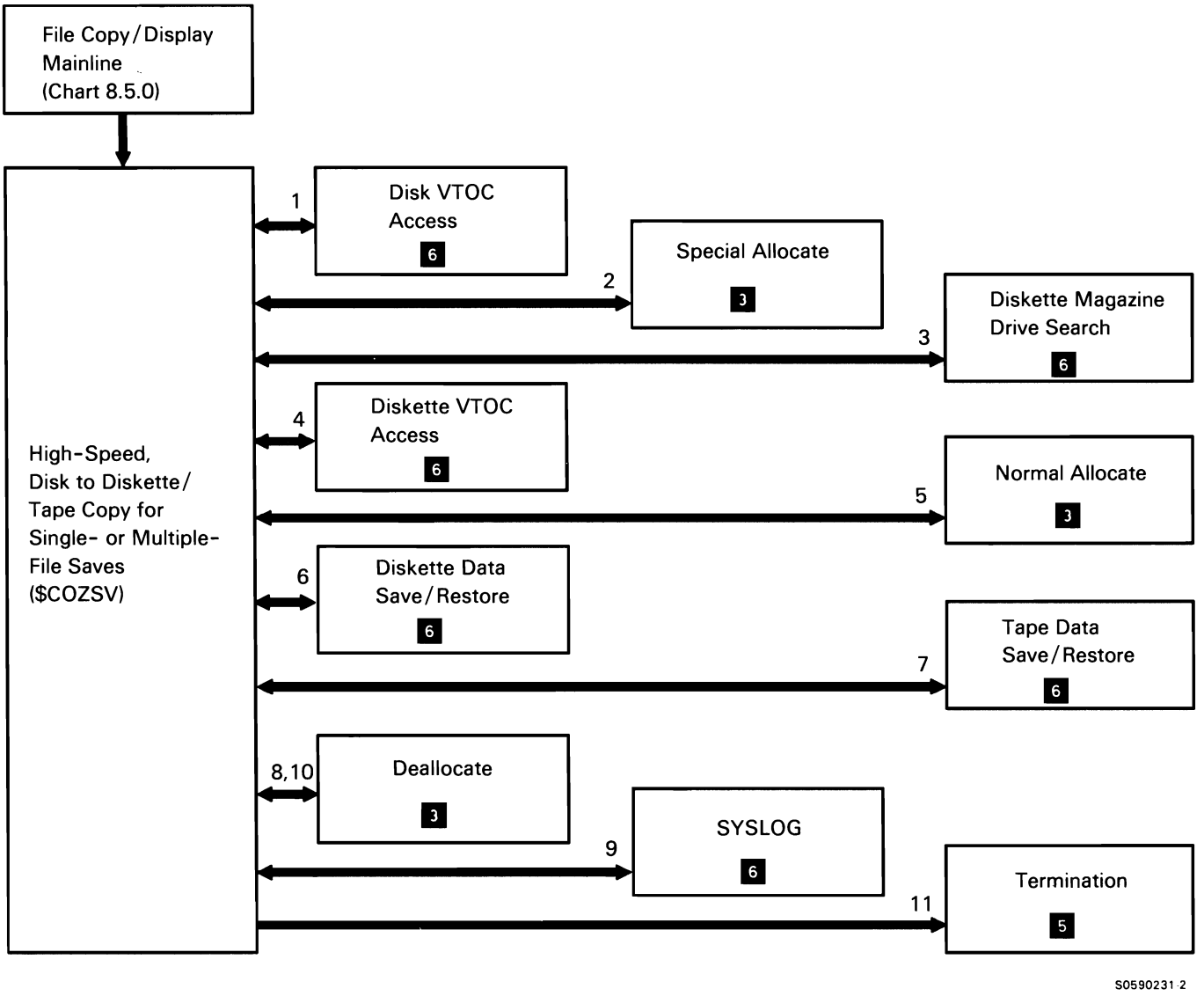


Chart 8.5.5 High-Speed Disk File to Diskette File Sectorized Copy Control Flow

PROCEDURE ERROR UTILITY ($CPPE)

The procedure error utility allows IBM-supplied procedures and user-written procedures to issue error messages. IBM-supplied procedures invoke the error utility with the #ERR system procedure, and user-written procedures invoke the utility with the ERR procedure.

The following procedure error utility processes are shown in Chart 8.6:

1 Read and syntax check utility control statements.

2 Issue error message requested.

3 If option 3, D, or H was not taken (in which case SYSLOG terminated without returning to $CPPE), return option selected to calling procedure.

4 Terminate this job step.

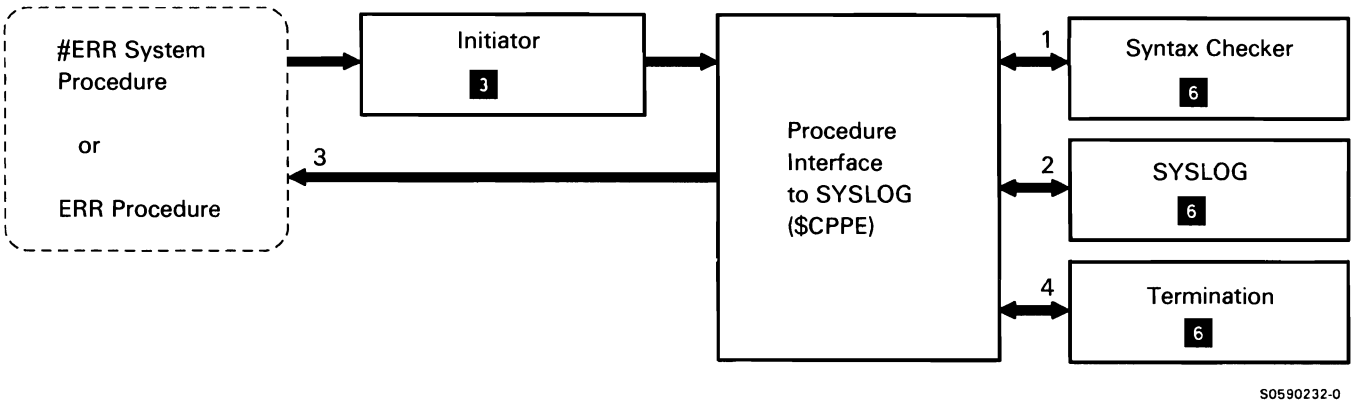


Chart 8.6 Procedure Error Utility Control Flow

KEYSORT UTILITY ($DDST)

The keysort utility sorts the index entries of a specified indexed file or alternative index. The utility is invoked by the KEYSORT system procedure or by equivalent OCL.

The following keysort utility processes are shown in Chart 8.7:

1 Process the keysort request, routing control where necessary.

2 Syntax check utility control statements.

3 Allocate the specified file.

4 Open the file.

5 Keysort the file.

6 Deallocate the file.

7 Close the file.

8 Issue any required messages.

9 Terminate this job step.

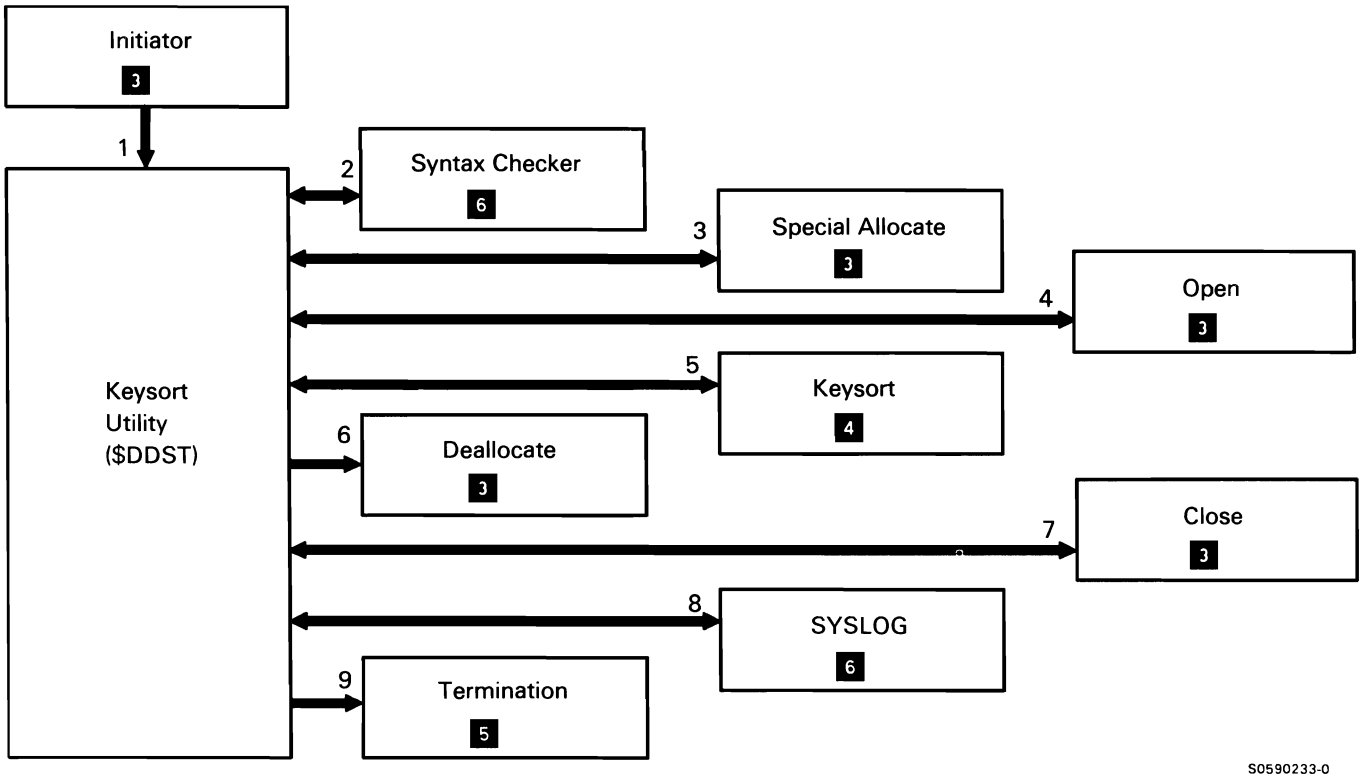


Chart 8.7 Keysort Utility Control Flow

FILE/LIBRARY/FOLDER DELETE UTILITY ($DELET)

The file/library/folder delete utility frees space on a disk or diskette for use by new files, libraries, or folders. If the file is defined in the network resource directory (NRD), the file is deleted from the remote system. The user calls the file/library delete utility with the DELETE procedure or appropriate OCL and utility control statements.

The following file/library delete utility processes are shown in Chart 8.8:

1 Read and syntax check utility control statements.

2 Use the VTOC to find format 1 for file, library, or folder to be deleted. If request is for file or folder, special allocate it. If it is for library, call find a library. Deallocate the file, library, or folder, specifying to delete. If ERASE-YES, deallocate specify to write zeroes over the data. If the file is on a remote system, the special allocate and deallocate interacts with DDM to perform the delete operation.

3 Delete the file from the diskette. For SCRATCH, set the diskette file expiration date to the current date; for REMOVE, remove the file entry from the VTOC; for ERASE, remove the file entry from the VTOC and zero the data.

4 Process required messages to the operator.

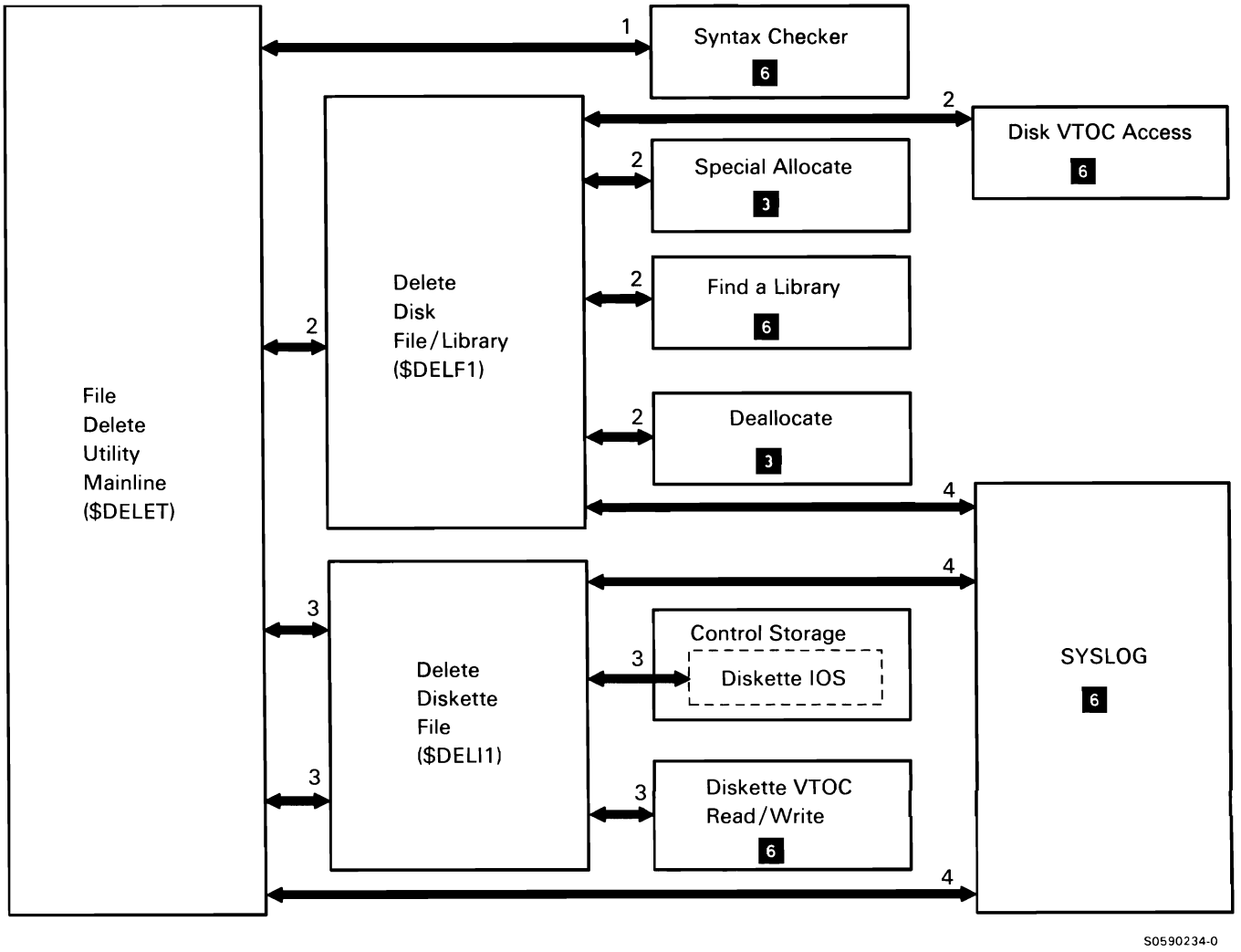


Chart 8.8 File/Library/Folder Delete Utility Control Flow

This page is intentionally left blank.

DISKETTE COPY UTILITY ($DUPRD)

The diskette copy utility copies the files from one or more diskettes onto one or more other diskettes. It can be used to create an extra copy of a diskette file or to gather all unused space on one diskette into a single free area on another diskette. The copy-to and copy-from diskettes must be of the same format.

The diskette copy utility is invoked by the COPYI1 procedure or appropriate OCL and utility control statements.

The following diskette copy utility processes are shown in Chart 8.9:

1 Determine functions requested, then process or route for the following:

- Process utility control statements.
- Allocate required resources.
- Load disk and diskette data managements.
- Load and call appropriate diskette copy module.

2 Syntax check utility control statements and put utility control language data into copy utility communications area.

3 Allocate diskette device.

4 Read specific/all format 1's into storage.

5 Determine single file copy functions requested, then process or route for the following:

- Read and convert header information.
- Copy diskette file to a disk scratch file.
- Fetch and load single file copy output module.
- Process read diskette read error.

6 Read and convert header information into diskette format 1.

7 Read diskette file from the diskette to disk.

8 Read diskette file from disk and write to diskette.

9 If necessary, display error sector on diskette and prompt for corrections.

10 Determine diskette or magazine copy functions requested, then process or route for the following:

- Read and convert header information.
- Copy diskette files to disk scratch files.
- Fetch and load single file copy output module.
- Process read diskette read error.

11 Read header information for each file and create a format 1 for each file on a disk scratch file.

12 Read each diskette file from disk scratch files and write to diskette.

13 Close diskette and disk files.

14 Issue any required messages.

15 Terminate this job step.

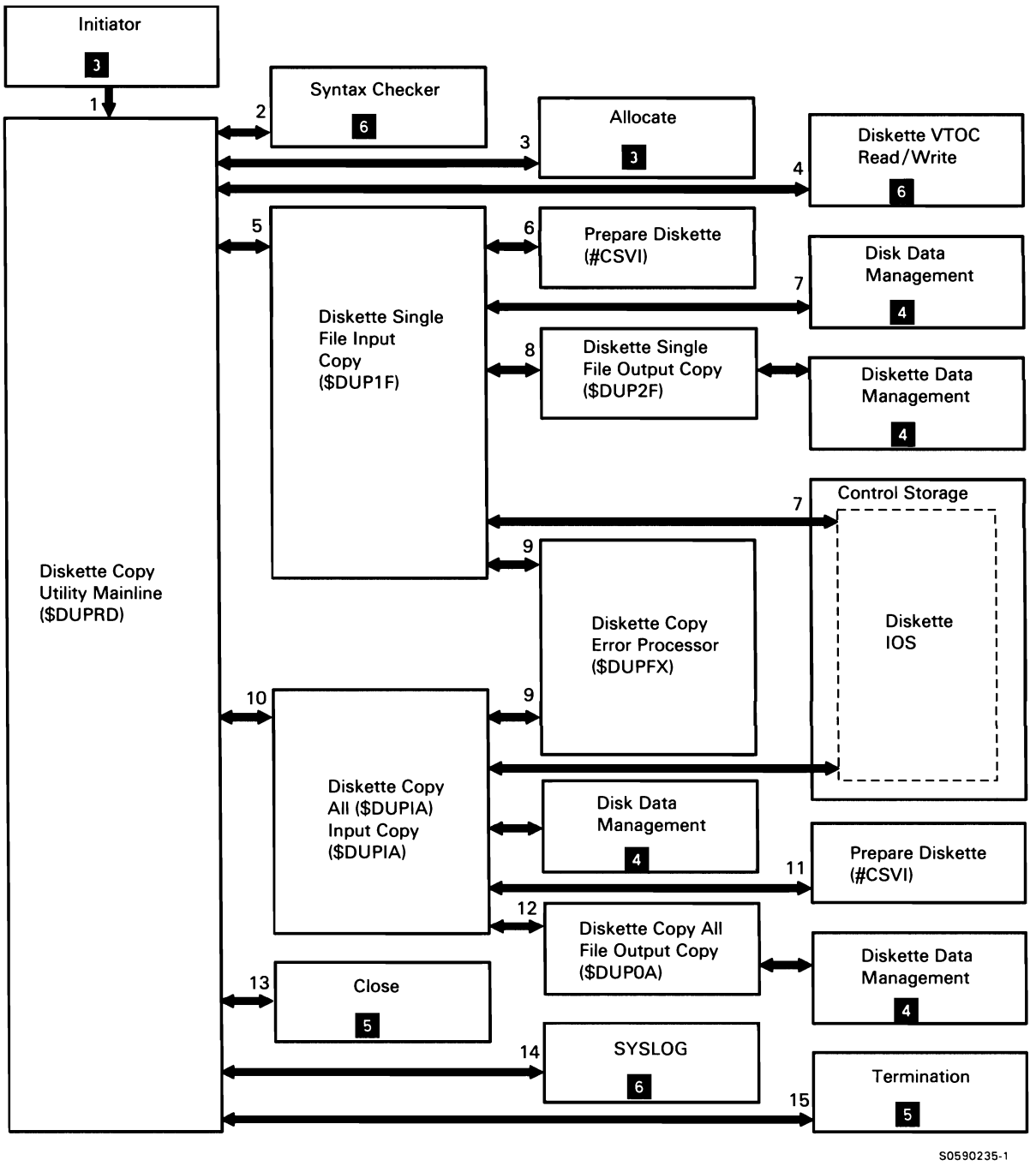


Chart 8.9 Diskette Copy Utility Control Flow

FILE BUILD UTILITY ($FBLD)

The file build utility allows the user to allocate a disk file either on the local system or on a remote system without supplying data records (null file). The utility can also be used to create a new alternative index file, with the index based on data in an existing physical file. If the physical file is on the local system, the alternative index will be created on the local system. If the physical file is on a remote system, the alternative index will be created on the remote system.

The user calls the file build utility with the BLDFILE or BLDINDEX procedures or with the equivalent OCL and utility control statements.

Note: When creating a remote file, the allocate and deallocate interacts with DDM to perform the operations on the remote system.

The following file build utility processes are shown in Chart 8.10:

1 Read and syntax check utility control statements.

2 Allocate a new null data file with the characteristics specified in the utility control statements.

3 Deallocate the null data file allocated in the process reference above.

4 Allocate the physical file that the alternative index file is based on.

5 Perform the following processes to create an alternative index file.

   If the physical file is on the remote system, the following processes are performed:

   - Check if the user is authorized to create an alternative index file over it.
   - Give control to the DDM Source Manager to create the alternative index on the remote system.

   If the physical file is on the local system, the following processes are performed:

   - Enqueue an interlock on the physical file.
   - Allocate space for the alternative index file.
   - Extract the alternative keys from the data in the physical file.
   - Deallocate the new alternative index file (the keys in the file may be sorted).
   - Read the highest key in the alternative index file and build a high-key-bucket.
   - Indicate in the VTOC entry for the physical file that the file has an alternative index based on it.
   - Dequeue the interlock on the physical file.

6 Deallocate the physical file that the alternative index is based on.

7 Issue any required messages.

8 Terminate this job step.

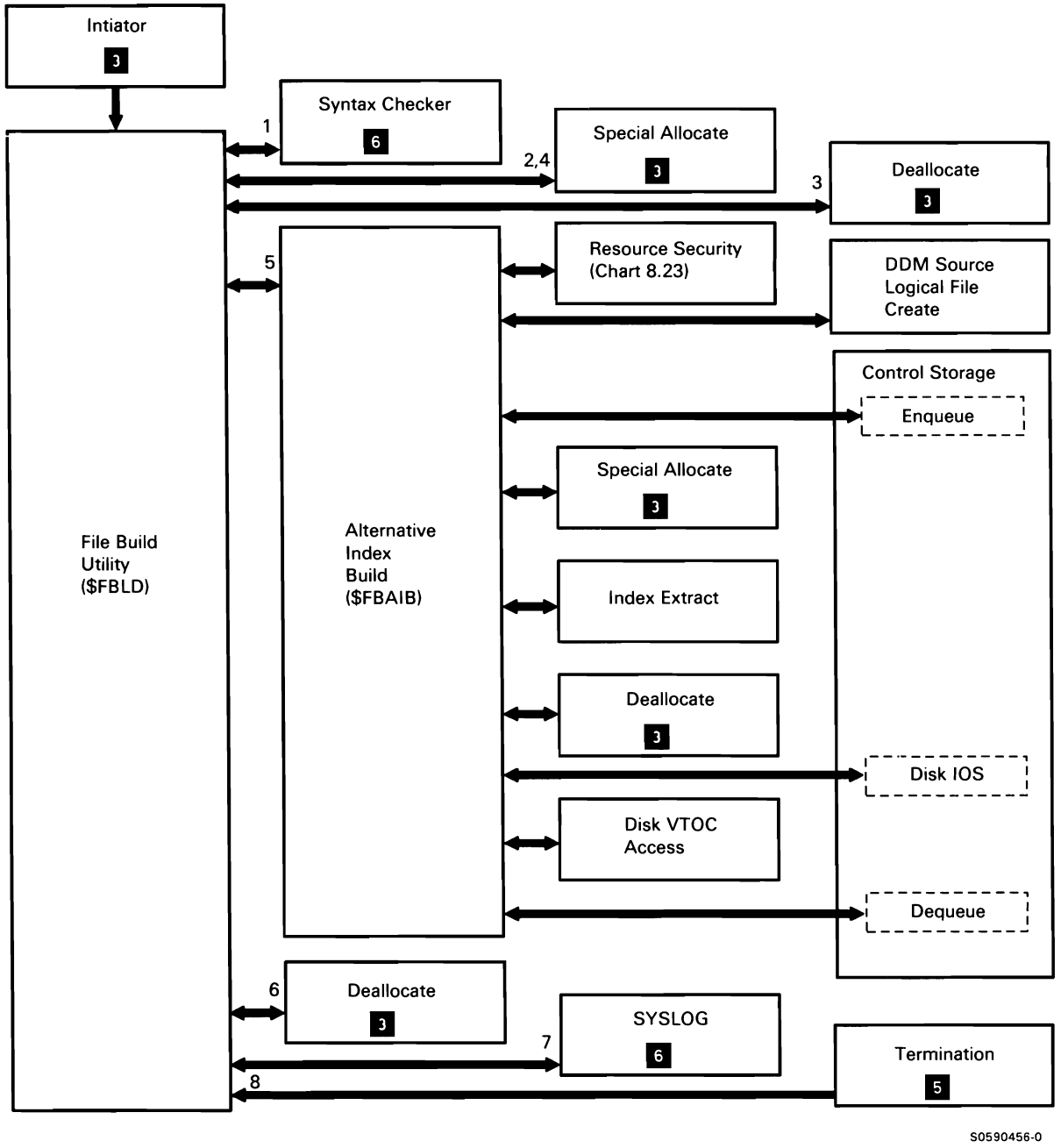


Chart 8.10 File Build Utility Control Flow

HELP UTILITY ($HELP)

The help utility assists a user in running IBM-supplied procedures without referring to hard copy information on the detail parameters required. The utility guides the user by prompting for procedure parameters and displays information about OCL and procedure control expression (PCE) statements. The help utility displays three types of screens:

- Screens that prompt for procedure parameters.
- Screens that list OCL or PCE statements from which a specific statement can be selected.
- Screens that provide information on specific OCL or PCE statements.

The help utility works in conjunction with command processor help facilities to respond to user requests. If help cannot perform a user-requested function from within a procedure, it passes control to command processor help via the command processor interface (#CPIN). Command processor help calls the help utility ($HELP) when it cannot perform the function requested by the user from the keyboard.

The user can invoke the help utility in any of the following ways:

- Pressing the Help key when a procedure name is entered on the input line.
- Entering 'HELP XXXXXXXX', where XXXXXXXX is a system procedure name.
- Selecting an option from the command processor help menu.
- Selecting the H option from a message displayed by a system procedure or program.

In all of the above cases, the command processor invokes the help utility via the initiator.

The following help utility processes are shown in Chart 8.11:

1 Initialize for help support requested:
   - Determine where call was from.
   - Initialize task work area (TWA).
   - Initialize procedure parameter save area (PPSA).

2 Get appropriate help table module(s) based on user option specified.

3 Set up for requested help display.

4 Put help displays; get responses.

5 Issue any required messages.

6 Issue a job queue request or check for a valid command when the user help request is not in the help utility's list of procedures.

7 Put/get PPSA.

8 Terminate this job step.

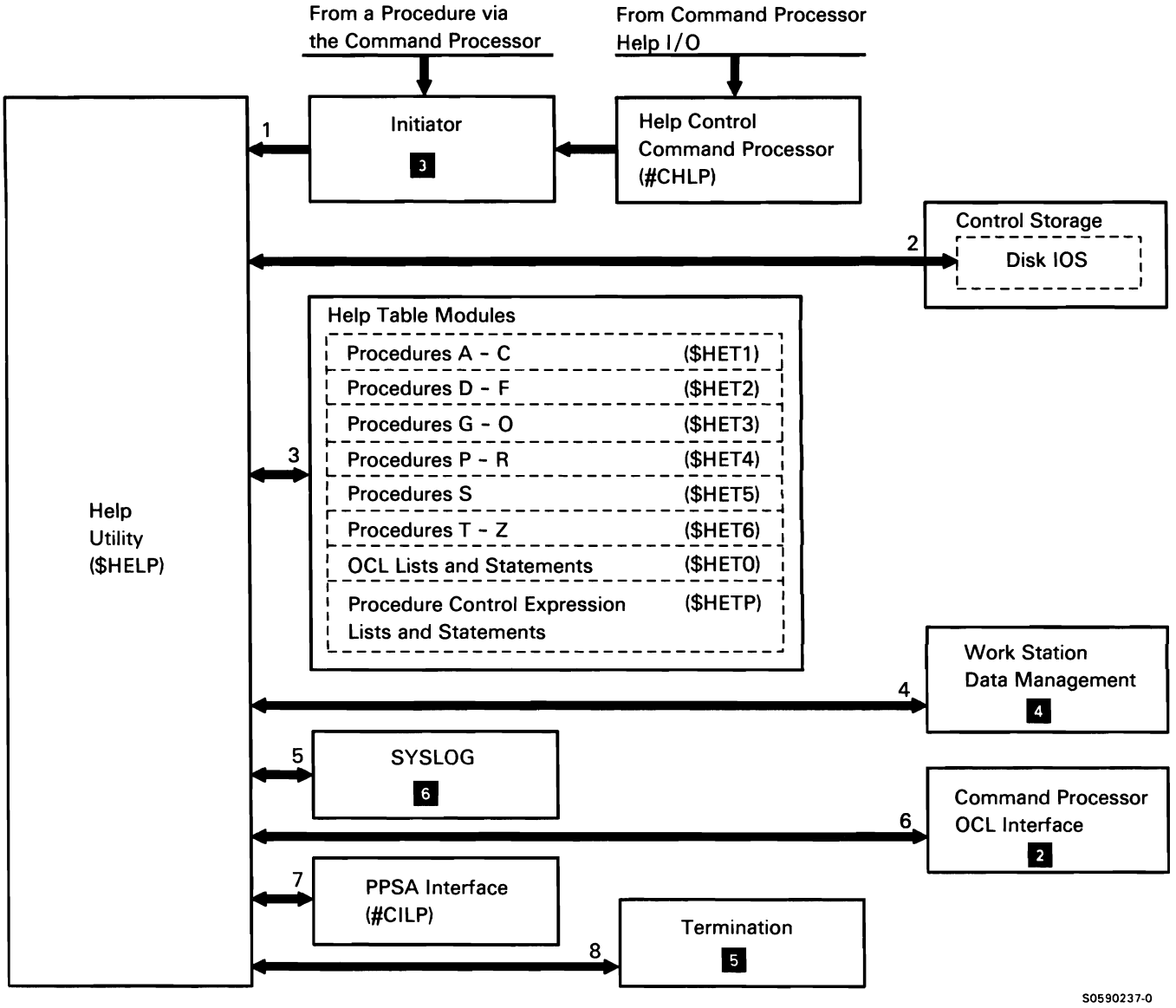


Chart 8.11 Help Utility Control Flow

HISTORY FILE DISPLAY UTILITY ($HIST)

The history file display utility allows the user to display the system history file (via the SYSLIST device), erase the system history file, copy the system history file to a user file, or scroll either the history file or a previously copied user file.

The following history file display utility processes are shown in Chart 8.12:

1 Read and syntax check the utility control statements.

2 Display the system history file.

3 Display a user file.

4 Copy the system history file to a user file, if requested.

5 Scroll the system history file.

6 Scroll a user file.

7 Read/write disk, as required.

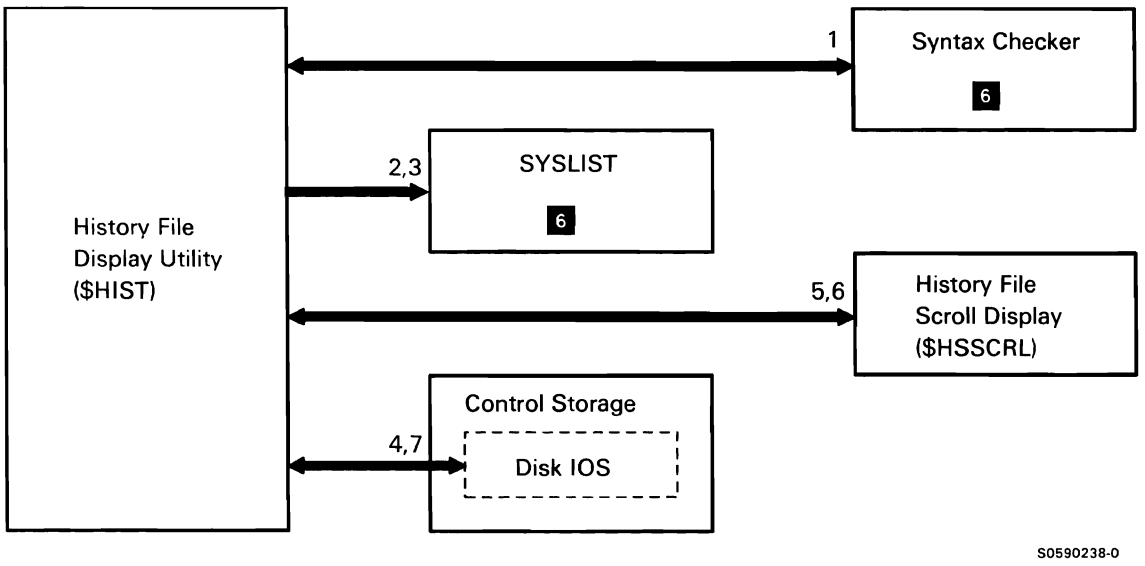


Chart 8.12 History File Display Utility Control Flow

SSP-ICF DEFINE ID UTILITY ($IDSET)

The SSP-ICF define ID utility is used to display, update, or delete remote ID file entries for a switched communications line that uses the SSP-ICF BSCEL subsystem 10. The switched line ID file (#IBSRID) cannot be altered when the BSCEL subsystem is enabled. The define ID utility is invoked by the DEFINEID system procedure or equivalent OCL.

The following define ID utility processes are shown in Chart 8.13:

1 Route for syntax checking Route for system file maintenance:

- Set XR1 to indicate user request (display, update, or delete).
- Load and branch to $IDSM.

2 Syntax check utility control statements.

3 Perform user-requested file maintenance:

- Allocate file.
- Write file to storage.
- Display, update, or delete according to user request.
- If update, write file to disk.
- Deallocate file.

4 Issue any required messages.

5 Terminate this job step.

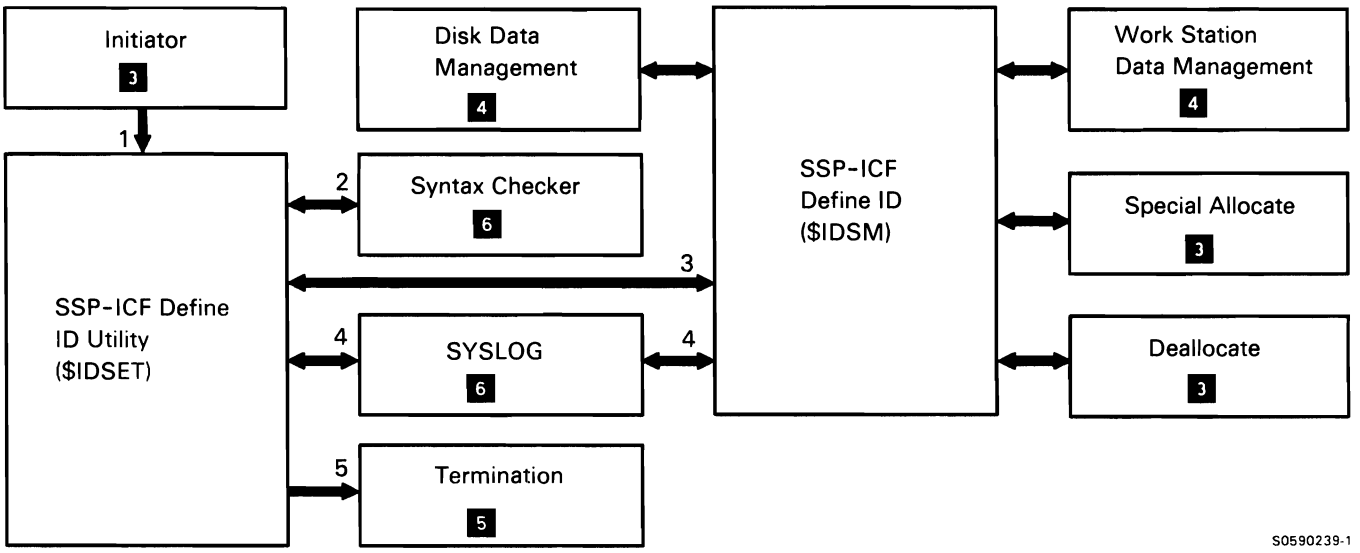


Chart 8.13 ICF Define ID Utility Control Flow

DISKETTE LABELING AND INITIALIZATION UTILITY ($INIT)

The diskette labeling and initialization utility formats a diskette, renames a diskette, or deletes files from a diskette. While formatting a diskette, the utility assigns alternative tracks for tracks found to be defective. A maximum of two alternative tracks are allowed; if more than two defective tracks are found, the diskette is considered unusable.

The following diskette labeling and initialization utility processes are shown in Chart 8.14:

1 Read and syntax check utility control statements.

2 Allocate diskette drive.

3 Perform requested function.

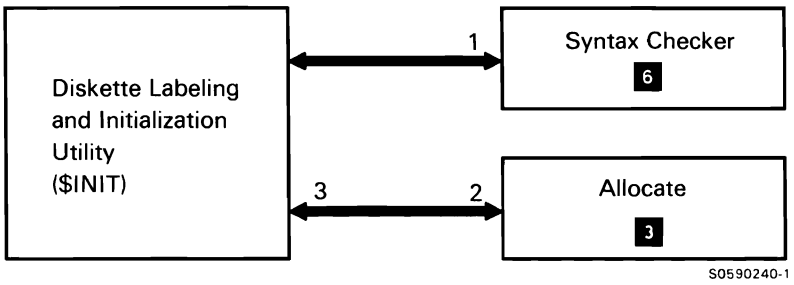


Chart 8.14 Diskette Labeling and Initialization Utility Control Flow

VTOC DISPLAY UTILITY ($LABEL)

The VTOC display utility displays or prints disk/diskette VTOC information, tape label information, network resource directory (NRD) information, and remote system VTOC information. The user calls VTOC display with the CATALOG procedure or appropriate OCL and utility control statements.

The following VTOC display utility processes are shown in Chart 8.15:

1 Read and syntax check utility control statements.

2 If request is for the entire disk VTOC, sort VTOC entries for display:

3 Allocate a work file for VTOC entry sort.

4 Build entries for work file using VTOC and NRD.

5 Issue any required messages.

6 If request is for a specific disk file, format and display disk VTOC entires. If file is on a remote system, format and display NRD entry.

7 Format and display disk VTOC entries:

8 Get headings for VTOC display.

9 Get VTOC entries from work file.

10 Read VTOC entries for display data.

11 Display or print VTOC information.

12 Deallocate work file.

13 If DDM is installed, display or print NRD entries.

14 Issue any required messages.

15 If request is for diskette VTOC, format VTOC entries for display:

16 Allocate diskette drive.

17 Read diskette VTOC entries.

18 Read diskette track label for VTOC information.

19 Get headings for VTOC display.

20 Display or print VTOC information.

21 Issue any required messages.

22 If request is for tape, allocate the tape drive:

23 If request is for tape sort by name:

24 Allocate a work file for the tape label sort entries.

25 Read the tape labels.

26 Read/write entries to/from the work file.

27 Deallocate unused work file.

28 If request is for a specific tape file or if request is for a sort by location:

29 Format and display tape entries:

30 Get headings for tape label display.

31 If a work file was used, read the entries from the work file.

32 If a work file was not used, read the tape labels.

33 If a work file exists, deallocate it.

34 Display or print the tape label information.

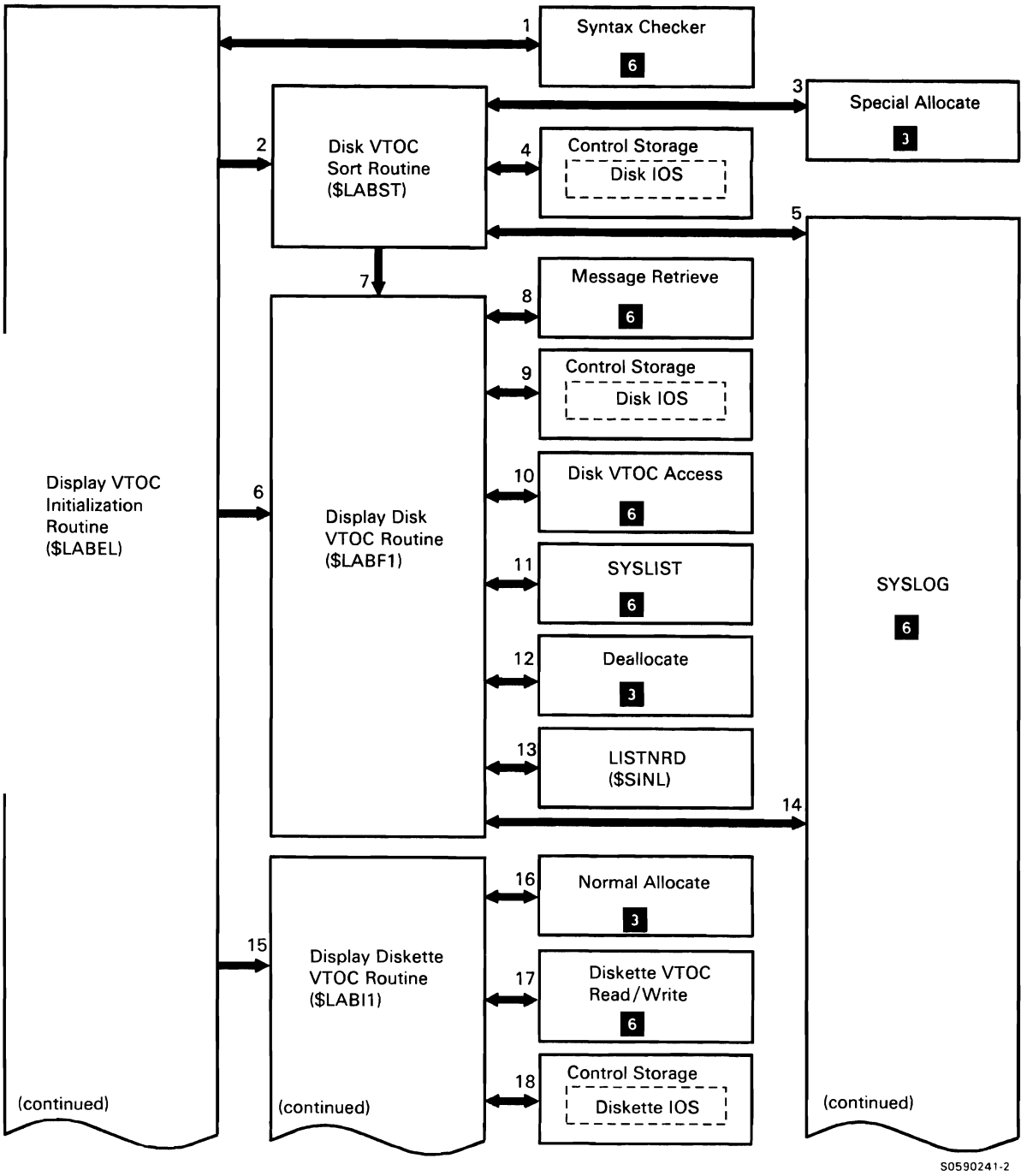


Chart 8.15 (Part 1 of 2) VTOC Display Utility Control Flow

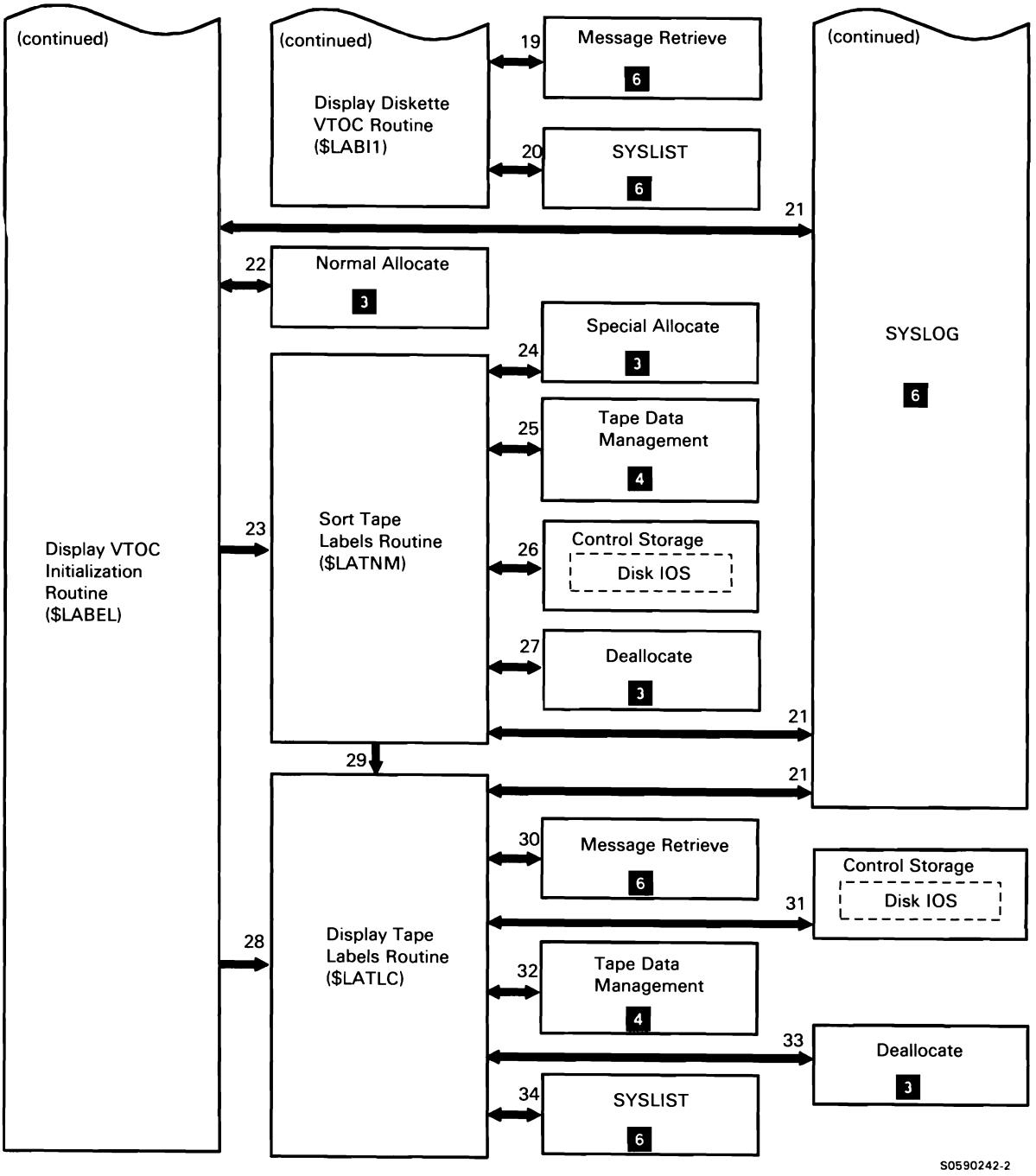


Chart 8.15 (Part 2 of 2) VTOC Display Utility Control Flow

LIBRARY MAINTENANCE UTILITY ($MAINT)

The library maintenance utility can be used for any of the following:

- Copy specified members to and from libraries and files.
- Create libraries and reallocate them according to user size specifications.
- Save and restore entire libraries to and from diskette and tape.
- Add or delete members from libraries.
- Display library information.
- Compress unused space within a library.
- Change specified member's name, subtype, and reference number.

The library maintenance utility is invoked by several different system procedures and it can also be evoked by equivalent OCL and utility control statements.

The following library maintenance utility mainline processes are shown in Chart 8.16.0:

1 Read and syntax check utility control statements.

2 Process request to save a library from disk to diskette.

3 Process request to restore a library from diskette to disk.

4 Issue any required messages.

5 Process request to reallocate a library so members in an extent can be placed in library. If required, process new directory and and library sizes.

6 Process request to allocate a user library or change size (or directory size) of a user library.

7 If request involves library transfer, find the library.

8 Process request to compact a library.

9 Process request to delete member or members from a library.

10 Process request to display library and library member information.

11 Process request to copy or add source or procedure members from library to file in record mode.

12 Process request to copy members from library to file in sector mode.

13 Process request to copy members from one library to another.

14 Process request to display members from a file in record mode.

15 Process request to copy members from file to library in record mode.

16 Process request to display library and library member information from a file.

17 Process request to copy members from file to library in sector mode.

18 Process request to copy records from reader to a library member.

19 Process request to change a member's name, subtype, and reference number.

20 When all utility control statements have been processed, terminate this job step.

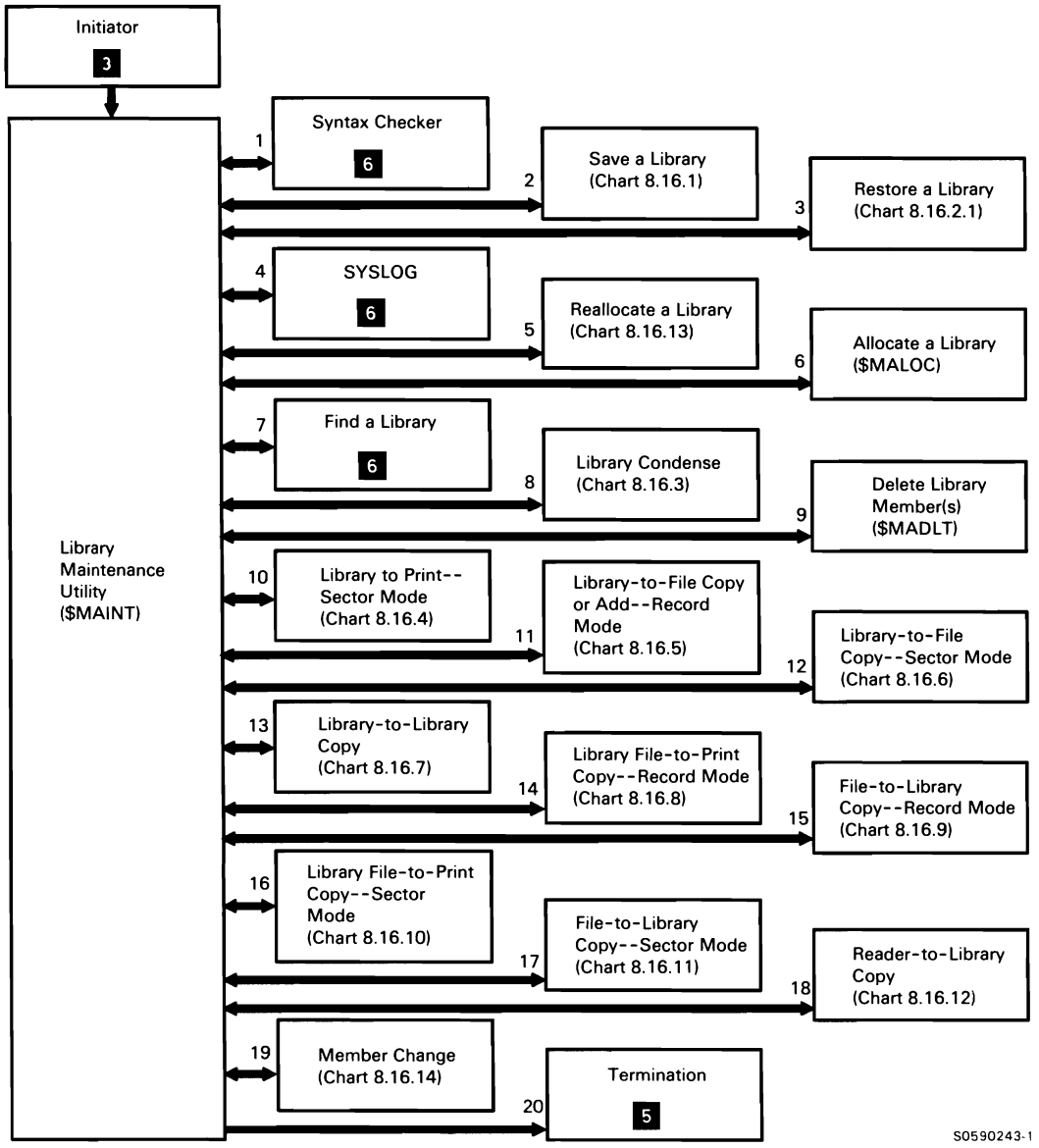


Chart 8.16.0 Library Maintenance Utility Mainline Control Flow

Save a Library

Save a library processes the SAVELIBR procedure command, saving all members from a library to a diskette file.

The following save a library processes are shown in Chart 8.16.1:

1 Find the specified library.

2 Read the library control sector (LCS).

3 Compact the library directory.

4 Allocate the library diskette or tape DTF; if library is #LIBRARY, also allocate a bootstrap DTF.

5 Open the diskette or tape DTF(s).

6 For a save to diskette, read the library directory and build a matrix table (1 bit for each sector allocated).

7 If library is #LIBRARY, copy heading messages to the bootstrap file. If heading messages are not found, issue an error message and return to mainline.

8 For a save to diskette, copy the LCS, compressed directory, and compressed members to the library diskette file.

9 If library is #LIBRARY, copy the IPL bootstrap (#MSNIP), the security module (@PRLD) and the reload screen formats (##FLOD) to diskette or tape; if any of these modules are not found, issue an error message and return to mainline.

10 For a save to tape, copy library to tape.

11 Whenever possible, use streaming mode.

12 Issue any required messages.

13 Close the diskette or tape DTF(s) and return to mainline.

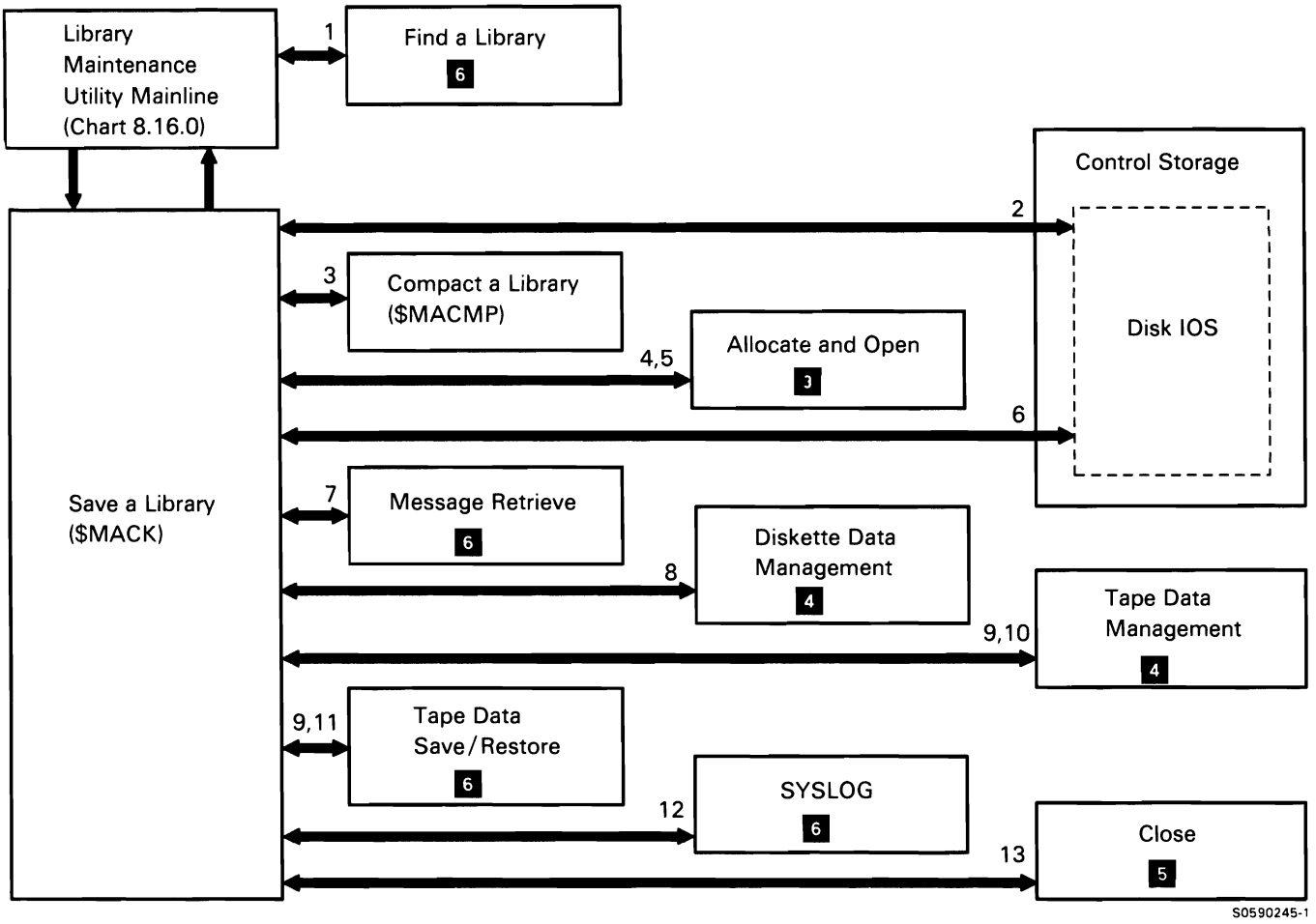


Chart 8.16.1 Save a Library Control Flow

Restore a Library

Restore a library processes the RESTLIBR procedure command, restoring to disk library members that were saved with the SAVELIBR procedure.

The following restore a library processes are shown in Chart 8.16.2.1.

Note: If restoring a library via the RESTLIBR procedure, the #IPLPROC procedure will be called and processes 1 through 14 *will not* occur. Tape or diskette may be used. If restoring a library via utility control statements, tape may not be used. Processes 1 through 14 *will* occur.

1 Ensure that library is not in use; if library is #LIBRARY, ensure that system is dedicated; if library in use or system is not dedicated, issue message.

2 Find the specified library on disk.

3 Allocate and open the diskette or tape library DTF.

4 If library is #LIBRARY, restore it.

5 Read the library control sector (LCS) from diskette or tape.

6 If library already exists, issue message. If user continues, allocate scratch file.

7 If library does not exist, allocate a new disk file.

8 Issue any required messages.

9 Write the diskette or tape LCS to disk.

10 Read the rest of the diskette or tape library.

11 Write the diskette or tape library to disk.

12 Close the diskette or tape DTF.

13 If library being restored already exists, point existing F1 to scratch file and point scratch file F1 to existing library.

14 If IBM-supplied members were copied, set OXREF indicator on (so that cross-reference resolver 6 will be run later), and return to mainline.

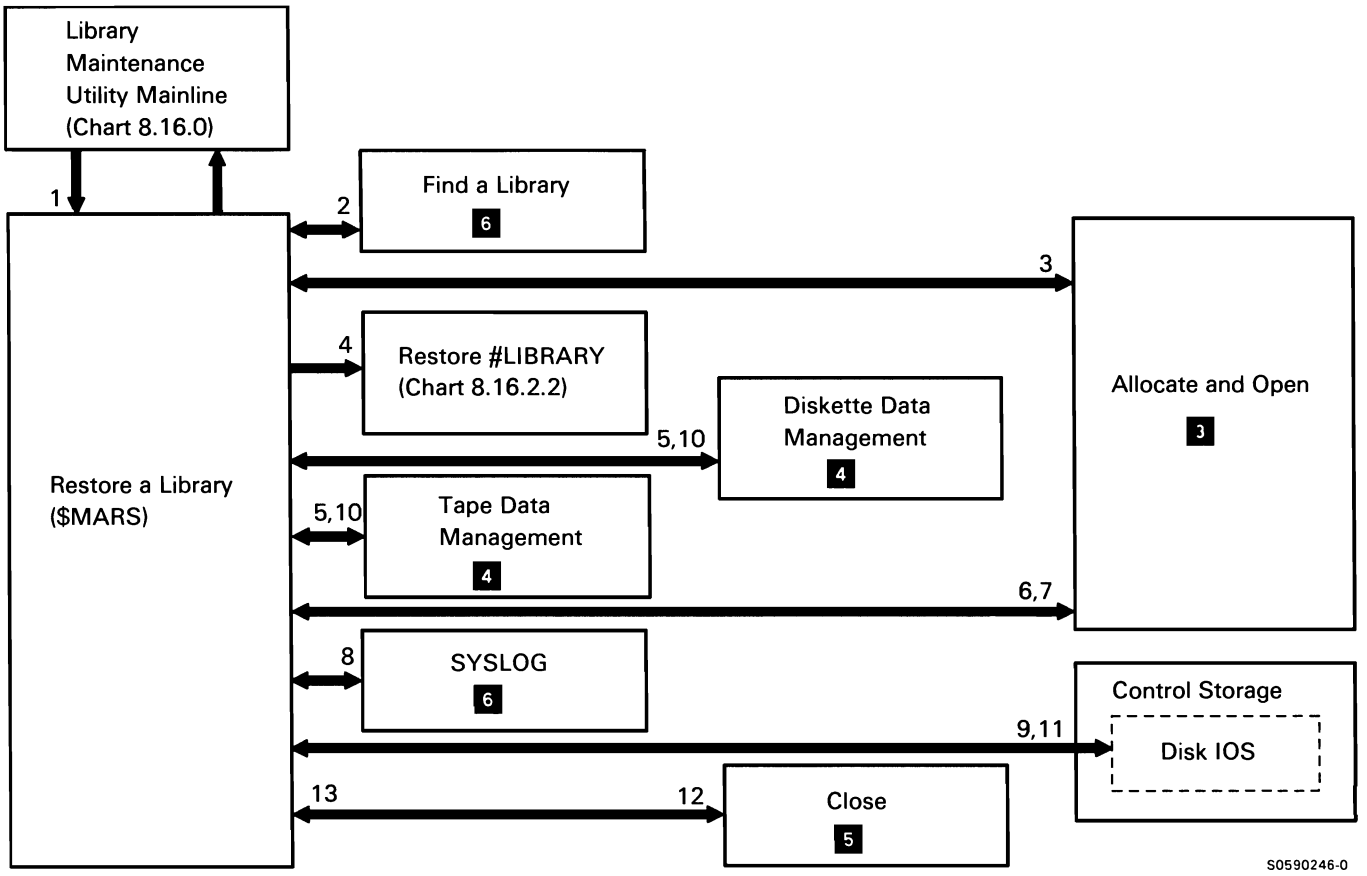


Chart 8.16.2.1 Restore a Library Control Flow

The following restore #LIBRARY processes are shown in Chart 8.16.2.2.

Note: If restoring #LIBRARY via the RESTLIBR procedure, the #IPLPROC procedure will be called, and processes 1 through 11 *will not* occur. If restoring #LIBRARY via utility control statements, processes 1 through 11 *will* occur.

1 Read bootstrap file messages and display formats from diskette.

2 Read the disk format 1 for #LIBRARY.

3 Suspend the command processor.

4 Write updated format 1 to disk.

5 Read the configuration record from disk.

6 Write updated configuration record to disk.

7 Copy the bootstrap file (#MSNIP) to disk.

8 Read the library on diskette.

9 Write the library to disk.

10 Resolve module disk relocations.

11 Terminate this job step.

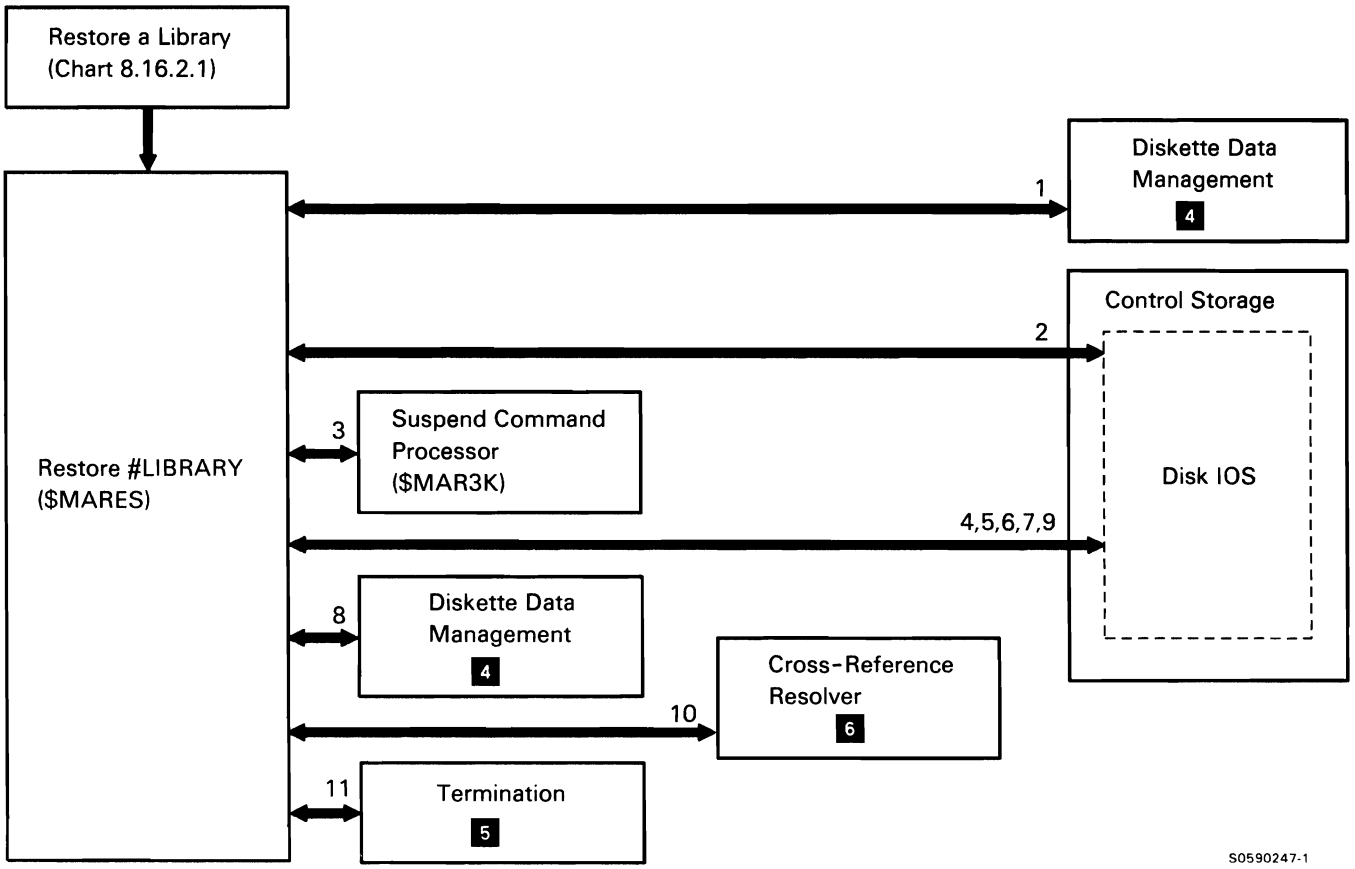


Chart 8.16.2.2 Restore #LIBRARY Control Flow

This page is intentionally left blank.

Library Condense

Library condense removes gaps in the member space of the requested library. It is evoked by the CONDENSE procedure command, via library maintenance mainline.

The following library condense processes are shown in Chart 8.16.3:

1 Find the specified library.

2 If the library is in use or if the specified library is #LIBRARY and the system is not dedicated, issue message.

3 If the specified library is #LIBRARY, suspend the command processor.

4 Read the library control sector (LCS).

5 Compact the library directory.

6 Read the directory and build a matrix table (one entry for each library member sector).

7 Process the matrix table, moving the members and building the hole table.

8 Update and write the LCS.

9 If there is now enough room, move the extent into the main library.

10 If there is still not enough room to move the extent, issue a message.

11 Update message member addresses in the JCB and update the menu member pointer in the JCB.

12 Update the procedure parameter save area (PPSA), using hole table data.

13 Update format index and WTG tables.

14 If another pass is required, load $MARCK again; otherwise:

15 Return to mainline.

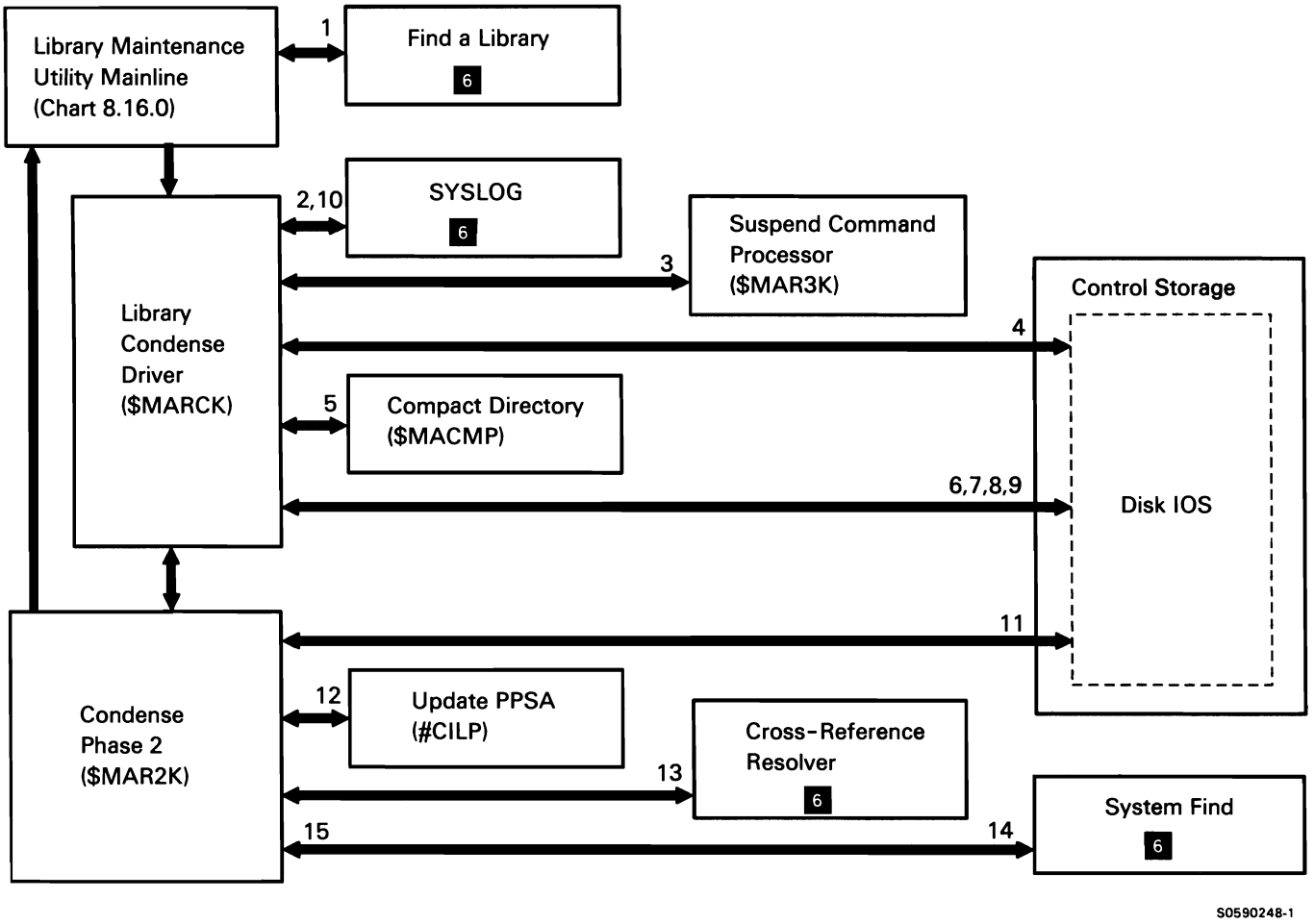


Chart 8.16.3 Library Condense Control Flow

Library-to-Print Copy–Sector Mode

The library-to-print copy–sector mode routine displays the status and directory information of a user library or of the system library. It also displays members by name, type, or subtype. It is evoked by the LISTLIBR procedure command, via library maintenance mainline.

The following save a library-to-print copy–sector mode processes are shown in Chart 8.16.4:

1 Find the specified library.

2 Read the configuration record for the chain image to see which characters are displayable.

3 Retrieve all headings for the report.

4 Print necessary headings.

5 Read the library control sector (LCS) to get status information on the specified library.

6 If necessary, compact the library directory.

7 Print all specified status and directory information.

8 Find the specified directory entries for printing members.

9 Read the member's directory entries into the buffer.

10 Print the requested information.

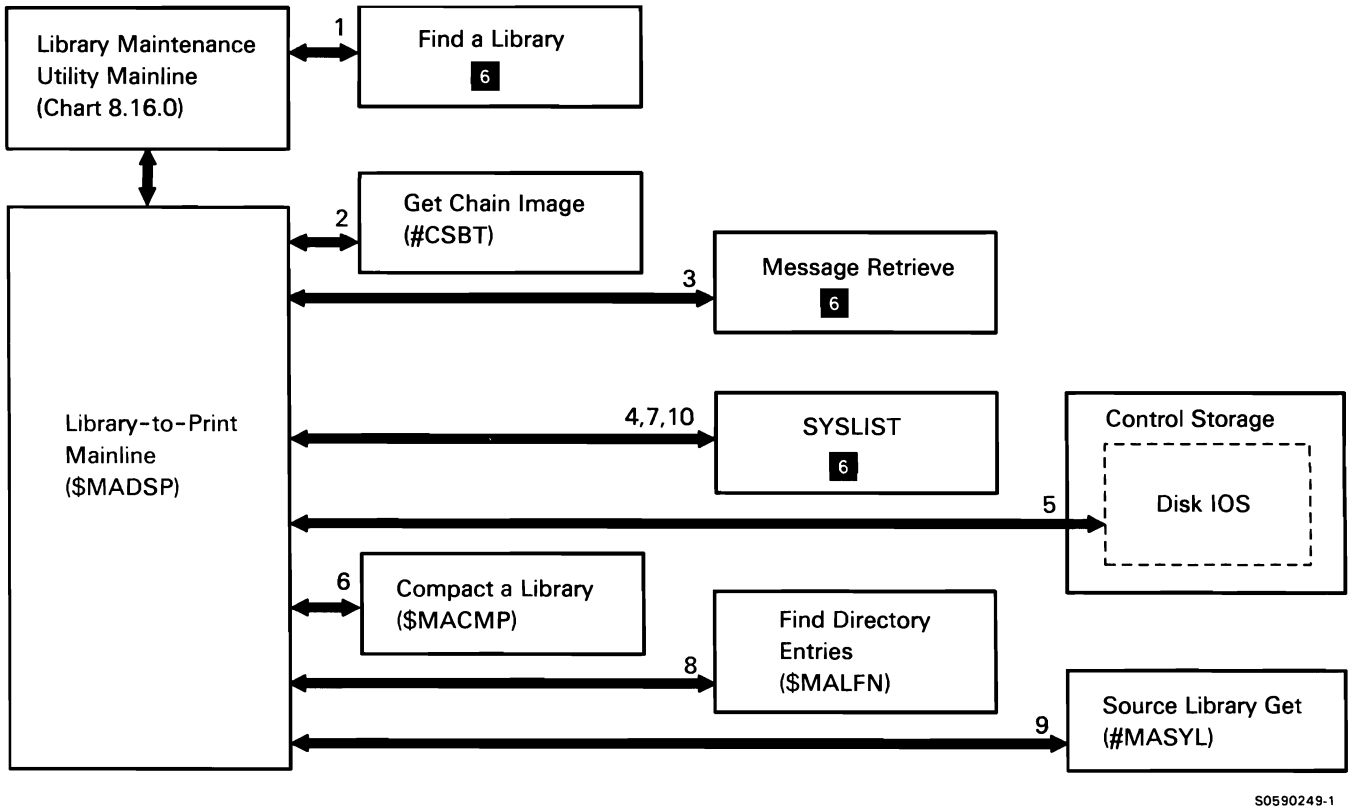


Chart 8.16.4 Library-to-Print Copy–Sector Mode Control Flow

Library-to-File Copy or Add–Record Mode

The library-to-file copy or add–record mode routine copies or adds source and/or procedure members from the library to a file in record mode. It is evoked by the FROMLIBR procedure command via library maintenance mainline.

The following library-to-file copy or add–record mode processes are shown in Chart 8.16.5:

1 Determine the device type of the output file. Load disk, diskette, or tape data management.

2 Open the output file.

3 Find the library member to be copied.

4 Build the copy control statement.

5 Write the copy control statement to the file.

6 Get library member records.

7 Write records to file.

8 When all requested members are copied, close output file and return to mainline.

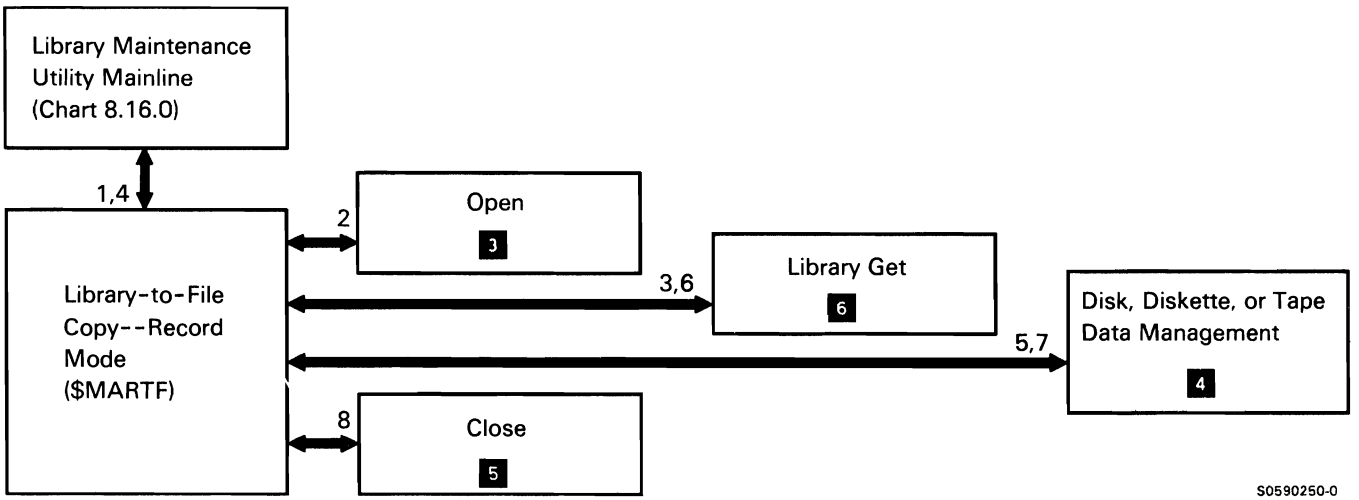


Chart 8.16.5 Library-to-File Copy or Add–Record Mode Control Flow

Library-to-File Copy–Sector Mode

The library-to-file copy–sector mode routine copies specified members from a library to a disk, diskette, or tape file. This routine is evoked by the FROMLIBR command procedure via library maintenance mainline.

The following library-to-file copy–sector mode processes are shown in Chart 8.16.6:

1 Find the specified library.

2 Open the input file.

3 Find the directory entries to be copied.

4 Open the library and copy the member(s) from the library.

5 If applicable, copy to disk.

6 If applicable, copy to diskette.

7 If applicable, copy to tape.

8 Issue any required messages.

9 Close the input file and return to mainline.

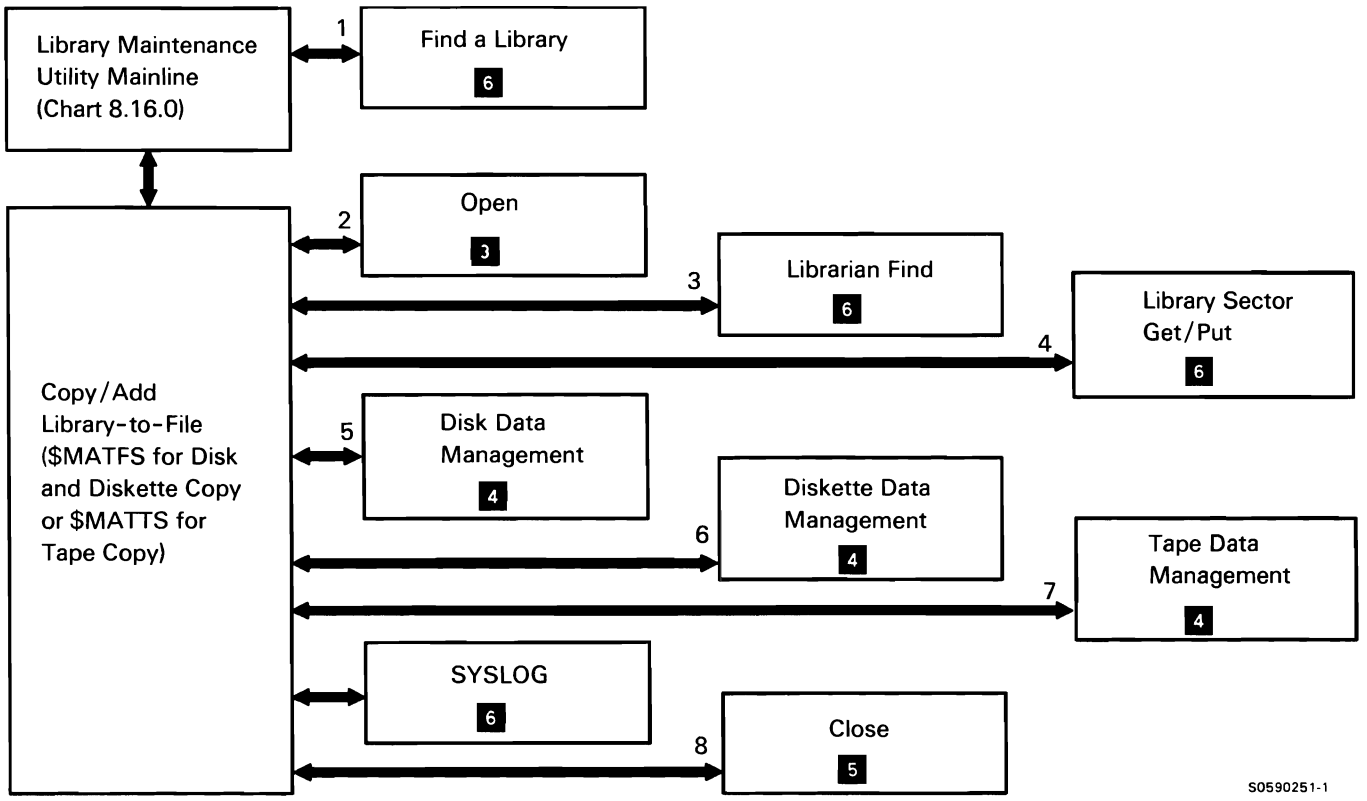


Chart 8.16.6 Library-to-File Copy–Sector Mode Control Flow

Library-to-Library Copy–Sector Mode

The library-to-library copy–sector mode routine copies specified members from one library to another. It also copies and renames from one library to another or within the same library. Library-to-library copy is evoked by the LIBRLIBR procedure command via library maintenance mainline.

The following library-to-library copy–sector mode processes are shown in Chart 8.16.7:

1 Find the specified library.

2 Find the directory entries of the member to be copied.

3 If the copy-to library is #LIBRARY and the member being copied is an IBM-supplied member, ensure system is dedicated; if system is not dedicated, terminate this job step.

4 Issue any required messages.

5 Open the specified libraries.

6 Read the library control sector (LCS).

7 Copy the specified members to the buffer, then to the specified library.

8 Close the libraries.

9 Update the directory entries.

10 Write the LCS and return to mainline.

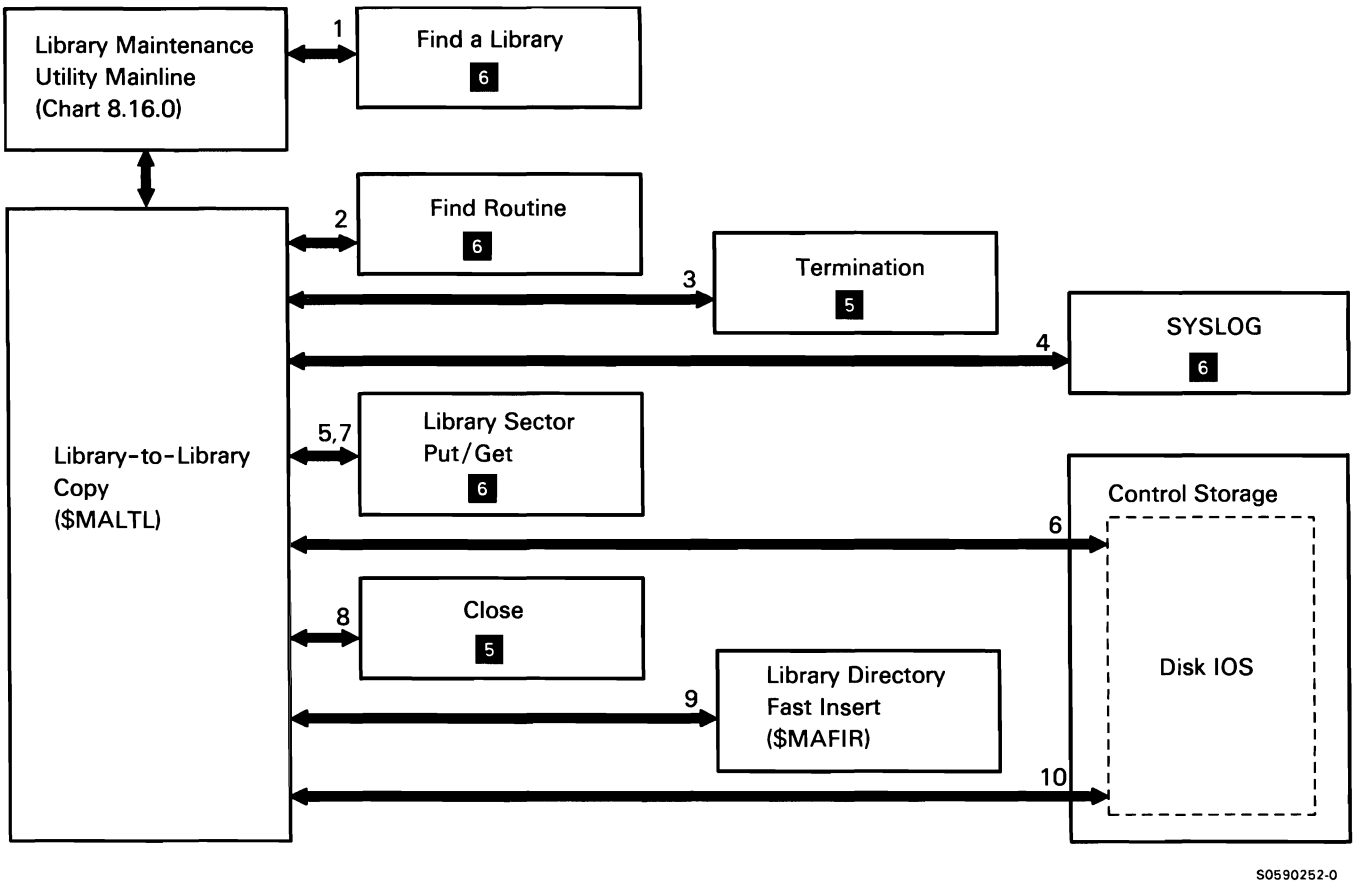


Chart 8.16.7 Library-to-Library Copy–Sector Mode Control Flow

Library File-to-Print Copy–Record Mode

The library file-to-print copy–record mode routine lists the member type and name of all members in a record mode file. Library file-to-print copy is evoked by the LISTFILE procedure command via library maintenance mainline.

The following library file-to-print copy–record mode processes are shown in Chart 8.16.8:

1 Get headings and current date and time.

2 Open the input file.

3 Print headings.

4 If input from disk, read record from input file.

5 If input from diskette, read record from input file.

6 If input from tape, read record from input file.

7 Check for valid copy control statement.

8 Print type and name from copy control statement.

9 At end of file, close input file and return to mainline.

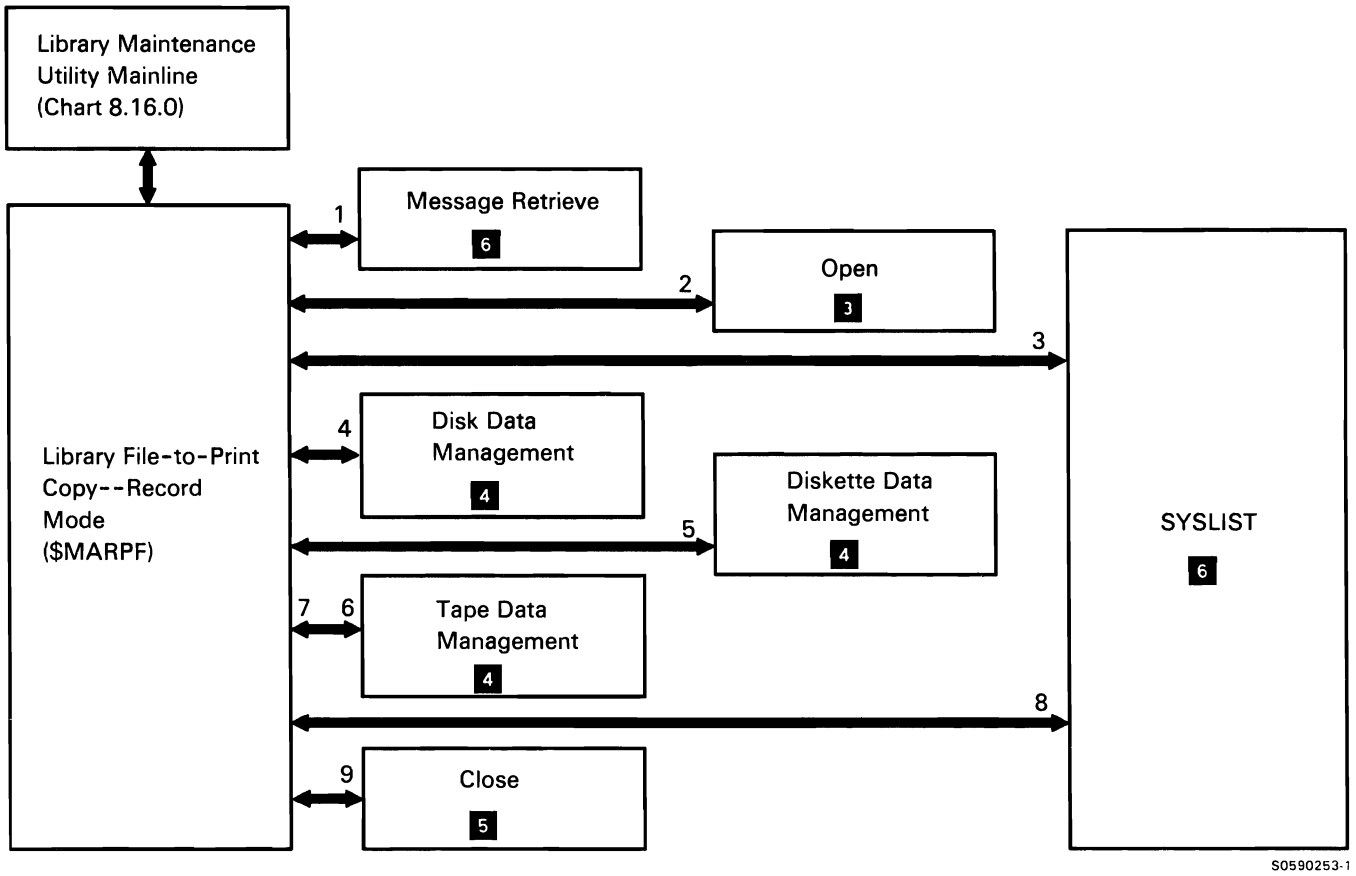


Chart 8.16.8 Library File-to-Print Copy–Record Mode Control Flow

File-to-Library Copy–Record Mode

The file-to-library copy–record mode routine copies source and procedure members from a disk file or a tape, or it copies a basic exchange diskette file directly to a library. This routine is evoked by the TOLIBR procedure command via library maintenance mainline.

The following file-to-library copy–record mode processes are shown in Chart 8.16.9:

1 Determine the type of input file device.

2 Open the input file.

3 If input file is from disk, get copy control statement; check syntax and save attributes.

4 If input file is from diskette, get copy control statement; check syntax and save attributes.

5 If input file is from tape, get copy control statement; check syntax and save attributes.

6 Open the library for new member.

7 If input file is from disk, get record.

8 If input file is from diskette, get record.

9 If input file is from tape, get record.

10 Copy record to library.

11 At end of file, close input file and return to mainline.

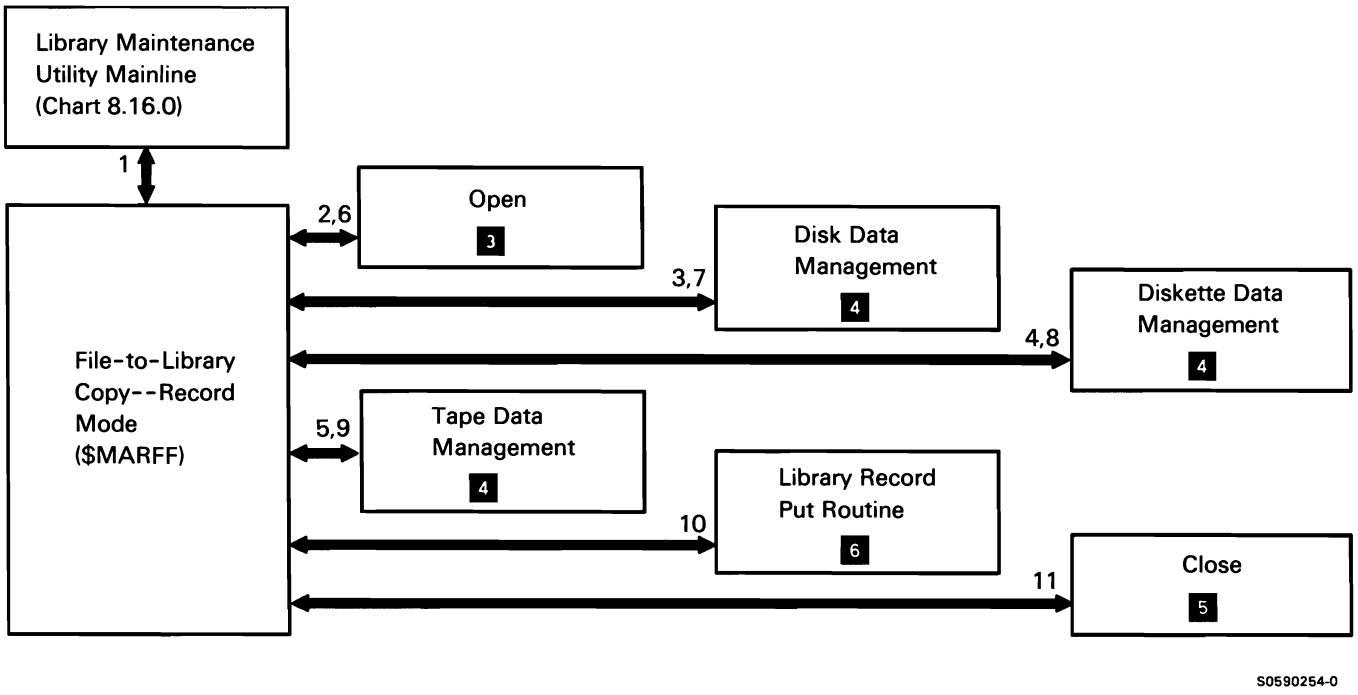


Chart 8.16.9 File-to-Library Copy–Record Mode Control Flow

Library File-to-Print Copy–Sector Mode

The library file-to-print copy–sector mode routine displays directory information for a disk, diskette, or tape file for files created by the FROMLIBR and SAVELIBR procedures. This routine is evoked by the LISTFILE procedure command via library maintenance mainline.

The following file-to-print copy–sector mode processes are shown in Chart 8.16.10:

1 Open the input file.

2 Retrieve all headings for the report.

3 Print the headings.

4 If input file is from disk, read to fill buffer.

5 If input file is from diskette, read to fill buffer.

6 If input file is from tape, read to fill buffer.

7 At end of file, close input file and return to mainline.

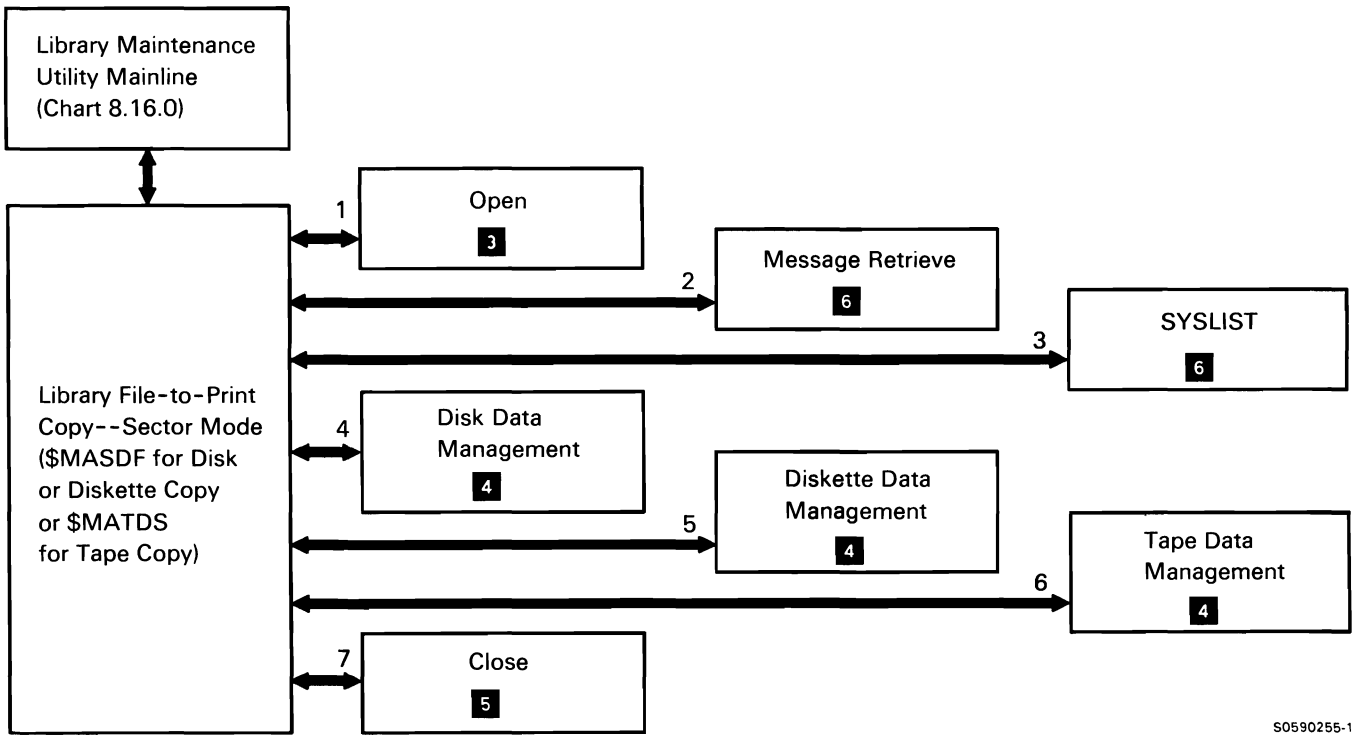


Chart 8.16.10 Library File-to-Print Copy–Sector Mode Control Flow

This page is intentionally left blank.

File-to-Library Copy–Sector Mode

The file-to-library copy–sector mode routine copies specified members from a disk, diskette, or tape file to a library. It can only be used for files created by the FROMLIBR procedure command. File-to-library copy is evoked by the TOLIBR procedure command via library maintenance mainline.

The following file-to-library copy–sector mode processes are shown in Chart 8.16.11:

1 Find the specified library.

2 Open the input file.

3 Read the library control sector (LCS).

4 Open the library.

5 If input file is from diskette, read the member from the file to the buffer.

6 If input file is from disk, read the member from the file to the buffer.

7 If input file is from tape, read the member from the file to the buffer.

8 Write the buffer to the library.

9 Close the library.

10 Update the directory.

11 Write the updated LCS to disk.

12 Resolve disk address changes.

13 Close the input file and return to mainline.

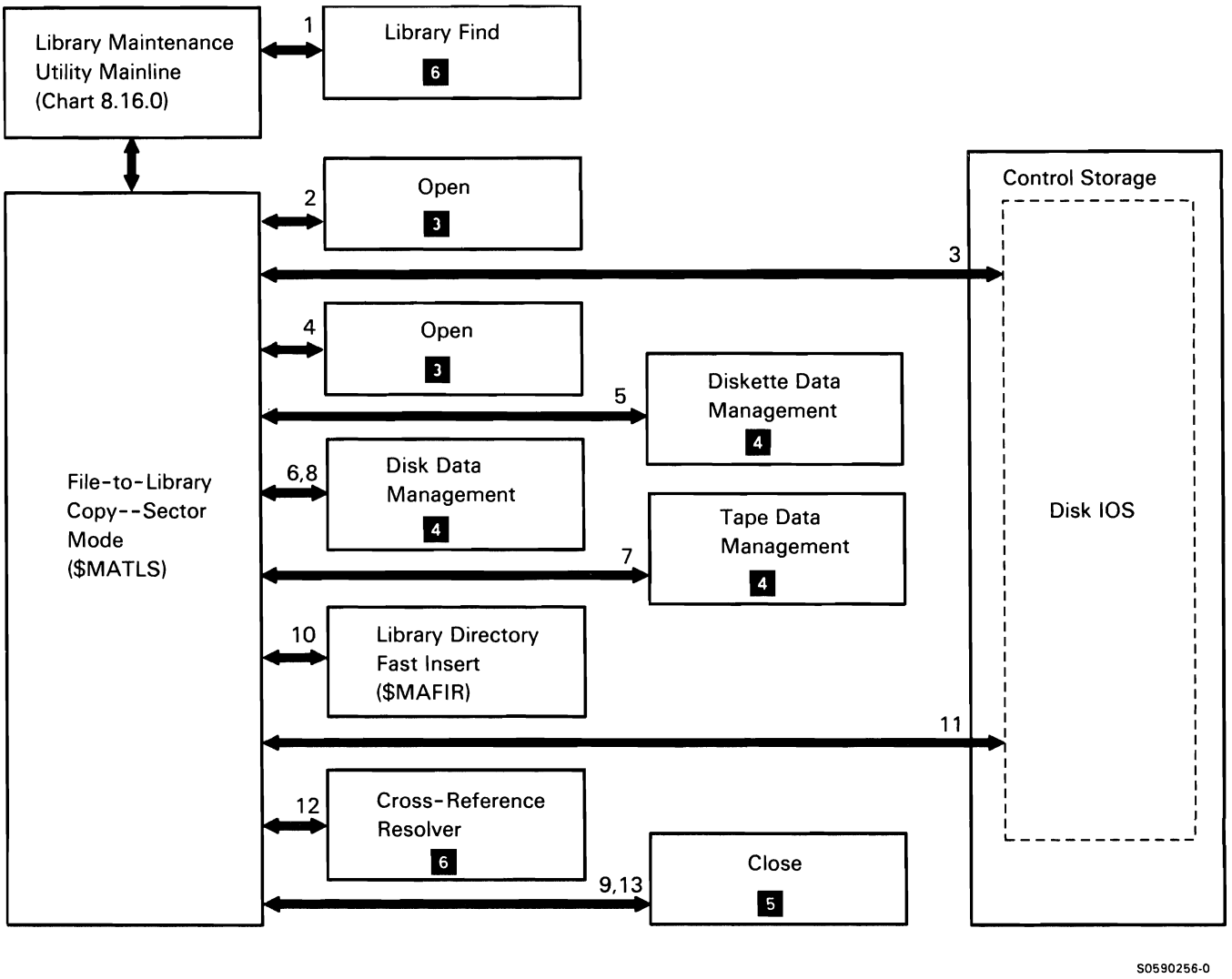


Chart 8.16.11 File-to-Library Copy–Sector Mode Control Flow

Reader-to-Library Copy

The reader-to-library copy routine copies records from the SYSIN device to the specified library. This routine is evoked via library maintenance mainline.

The following reader-to-library copy processes are shown in Chart 8.16.12:

1 Read the library control sector (LCS).

2 Validate library member name.

3 Open the library.

4 Protect library member being created.

5 Get a record from SYSIN.

6 Put the record into the library.

7 When end of file is reached, close the library and return to mainline.

8 If no records were put in the library, do not place the member in the library; issue a halt message and return to mainline.

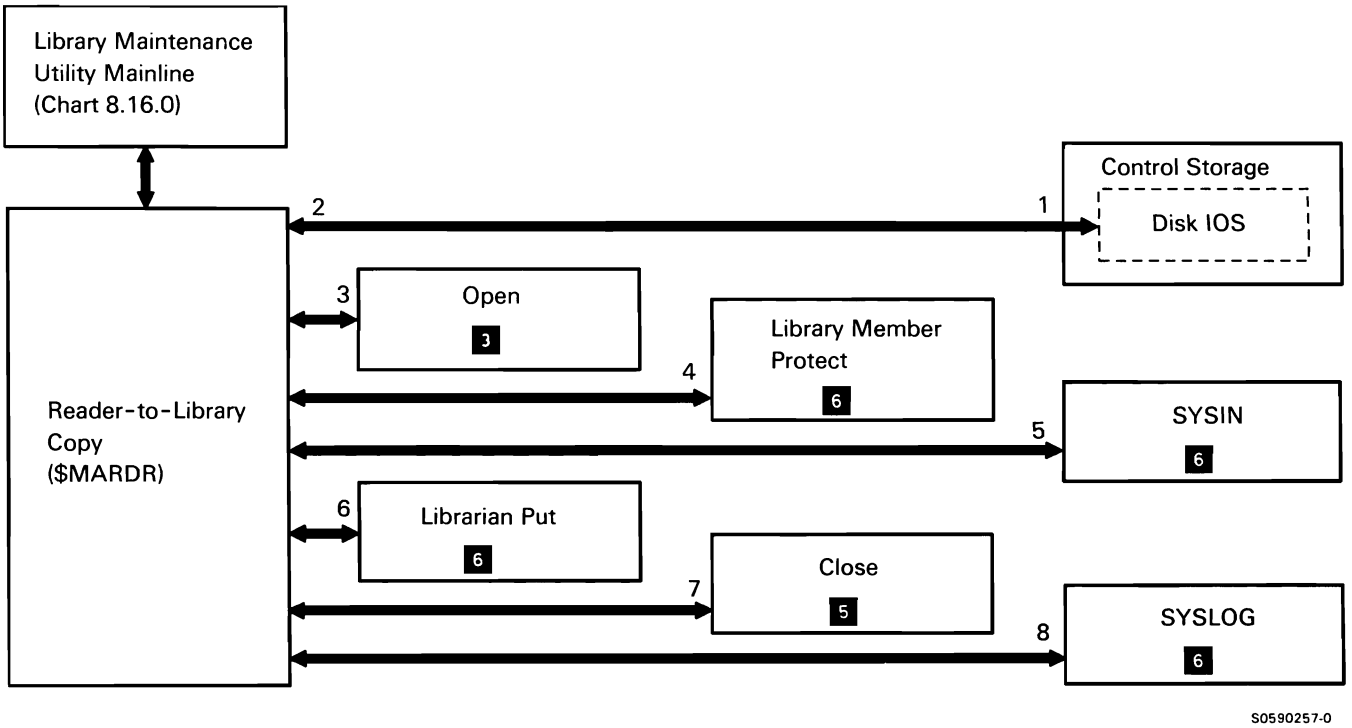


Chart 8.16.12 Reader-to-Library Copy Control Flow

Library Reallocate

The library reallocate routine reallocates a specified library to allow members located in an extent to be placed in the library. The user can specify new library and directory sizes for the library to be reallocated, even if an extent does not exist. The library reallocate routine is evoked by the ALOCLIBR procedure command via library maintenance mainline.

The following reader to library copy processes are shown in Chart 8.16.13:

1 Resource enqueue the library.

2 Read the library control sector (LCS).

3 Allocate disk space to move the library.

4 Copy the directory, the members and, if it exists, the file extent to the allocated disk space.

5 Update the LCS.

6 Fill any unused directory sectors with hex FF.

7 Update the directory for any members moved from the file extent.

8 Update the library format 1 and the extent format 1 and write them back to the VTOC.

9 Zero the old library and its extent.

10 Deallocate the old library and its extent.

11 Return to mainline.

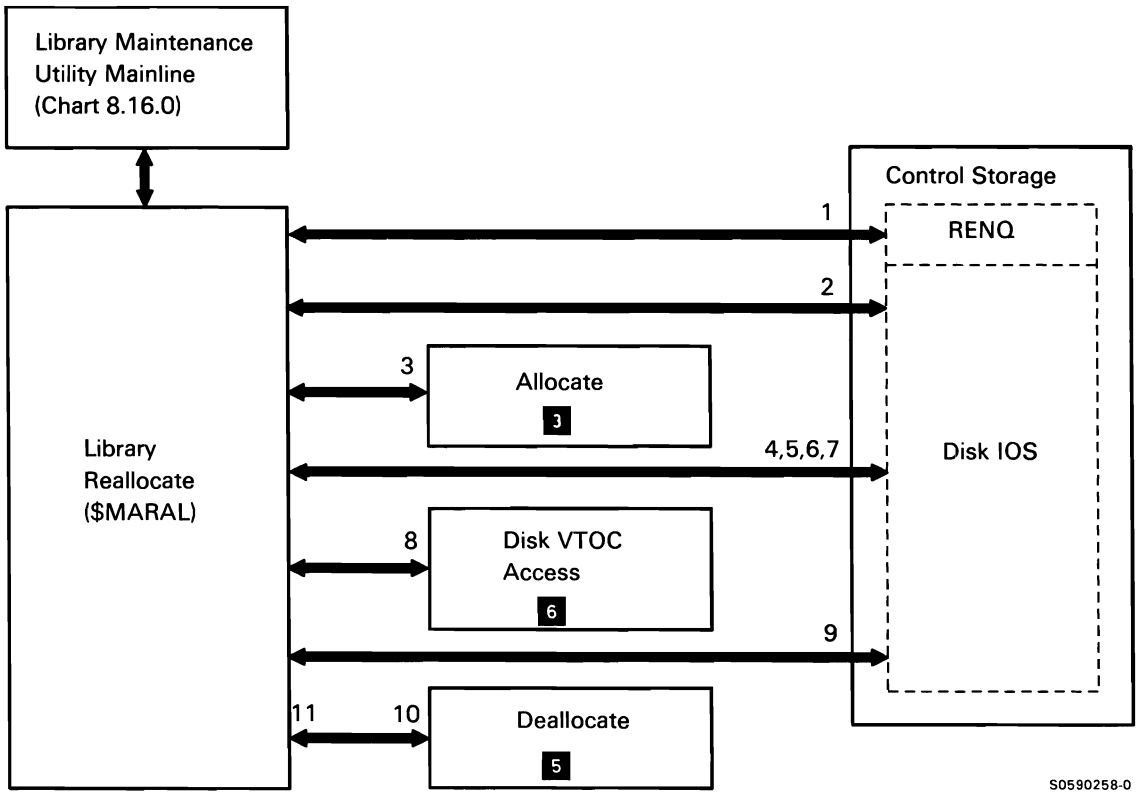


Chart 8.16.13 Library Reallocate Control Flow

Library Member Change

The library member change routine can be used to change a library member's name, subtype, and/or reference number. The library member change routine is evoked by the CHNGEMEM procedure.

The following member change processes are shown in Chart 8.16.14:

1 Enqueue the library as a resource.

2 Read the library control sector (LCS).

3 If necessary, compact the directory.

4 If name change requested, scan the directory for name match.

5 If subtype and/or reference number change only, find member to be changed and insert subtype and/or reference number in directory entry.

6 If subtype and/or reference number change only, return to mainline.

7 For rename only, check for member protection.

8 If rename only, check for valid new name and scan for duplicate.

9 Check for protection of members with new name.

10 For rename only, mark old member directory entry as deleted.

11 For rename only, insert new directory entry.

12 For rename only, write LCS.

13 Return to mainline.

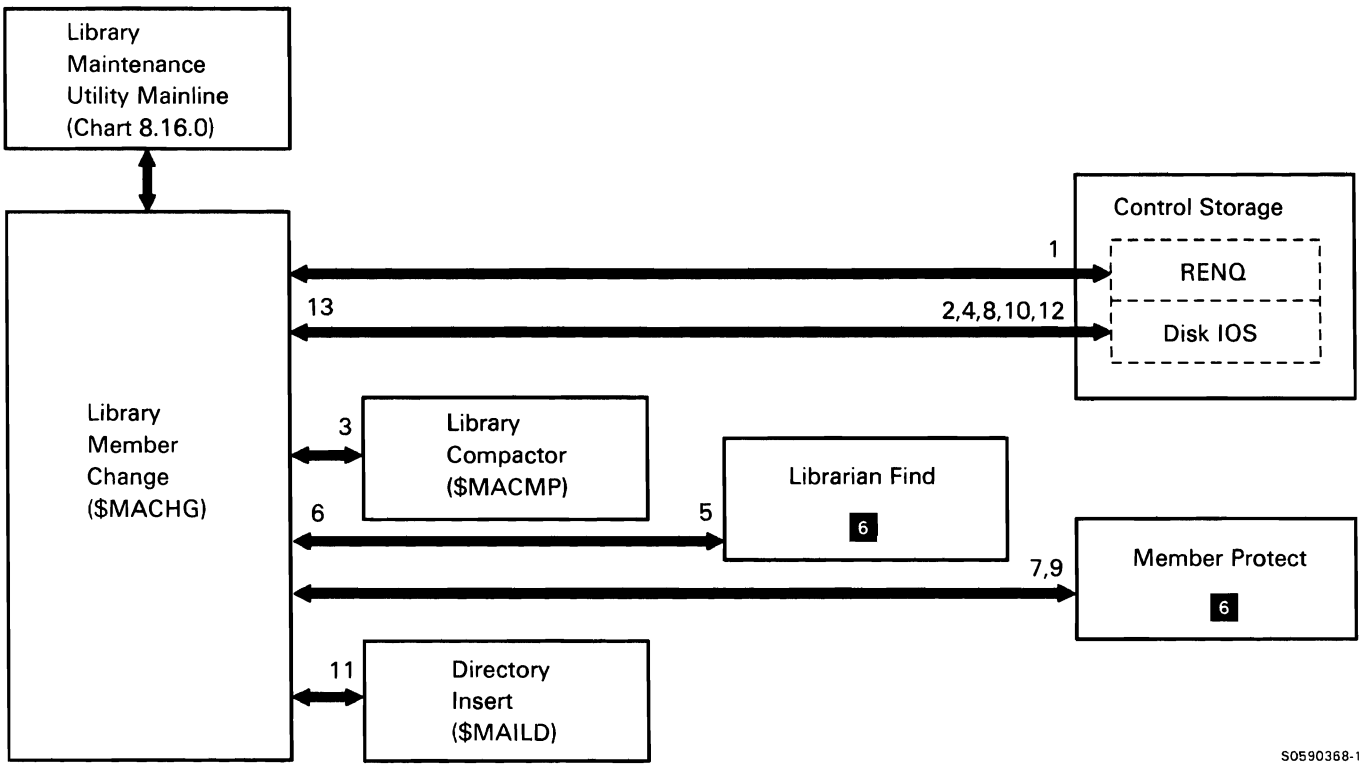


Chart 8.16.14 Library Member Change Control Flow

This page is intentionally left blank.

MESSAGE BUILD UTILITY ($MGBLD)

The message build utility creates message load members in a library. The load member contains formatted message text records that are retrieved by SSP message retrieve (#MGRET), via user-specified message identification codes (MIC). The message build utility is called by the CREATE procedure or by user-supplied OCL and utility control statements.

The following message build utility processes are shown in Chart 8.17:

1 Determine functions requested, then route, if necessary, for the following:

- Process utility control information:
  - Check syntax.
  - Save REPLACE and SSP parameters in library control block (LCB). If REPLACE not specified, ensure duplicate name does not exist.
  - If user library specified, find library.
  - Find and get source member.
- Process source member control and source statements:
  - Save member name in LCB.
  - Allocate and open work file.
  - Read message text statements and build messsage records in format required by #MGRET.
  - Write records from I/O buffer to work file when buffer is full.
- Create message member in library:
  - Set up work file for sector input to disk.
  - Fill I/O buffer with sectors from work file.
  - Write I/O buffer to disk.
- Terminate.

2 Read and syntax check utility control statements.

3 Find specified library members for validity checking and get operations.

4 Issue required messages.

5 Find specified libraries for validity checking and member-get operations.

6 Get source member.

7 Allocate work file on disk.

8 Open work file.

9 Terminate message build utility.

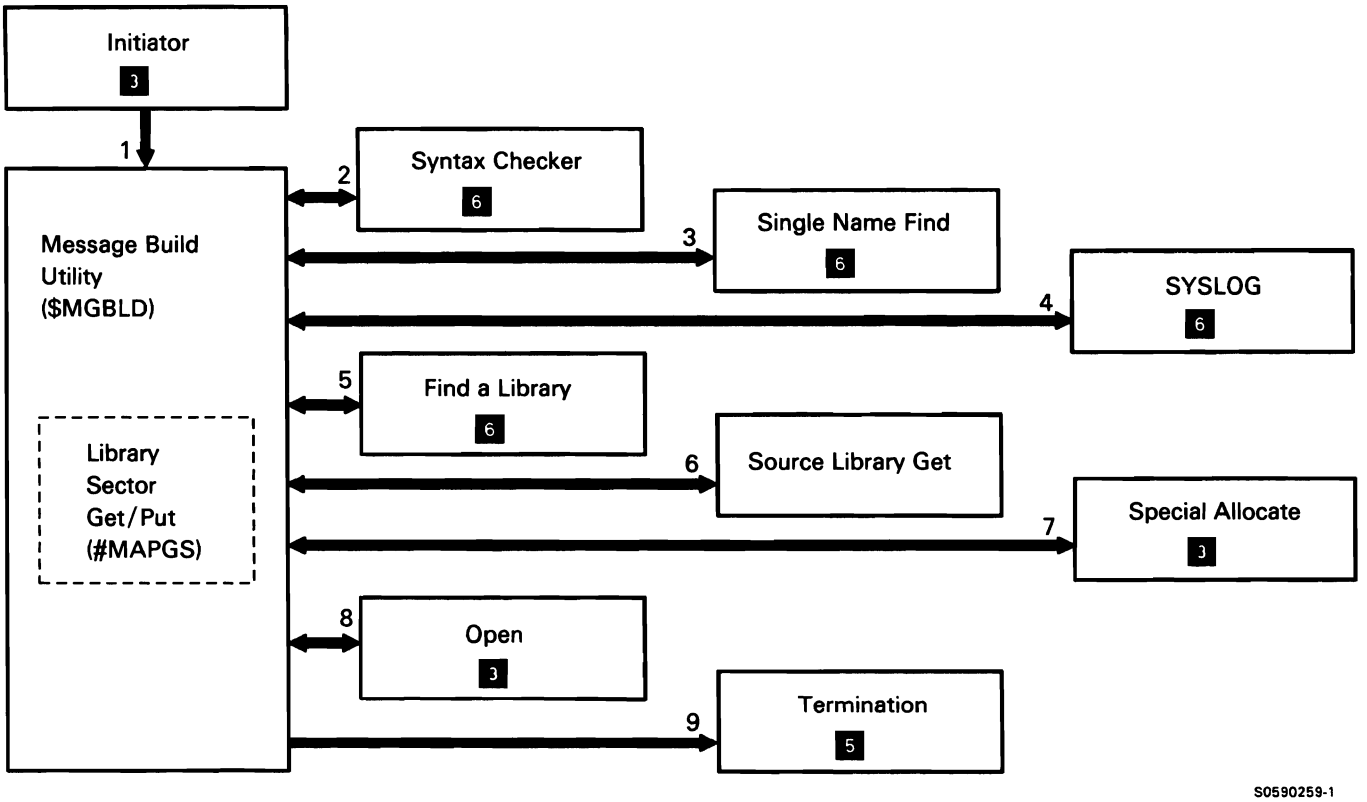


Chart 8.17 Message Build Utility Control Flow

STOP BSC MONITOR UTILITY ($MMSP)

The stop BSC monitor utility stops the automatic monitoring function of one of possibly four BSC multipoint lines. The stop BSC monitor utility is evoked by the STOPM system procedure or equivalent OCL.

The following stop BSC monitor utility processes are shown in Chart 8.18:

1 Do the following:

- Route for syntax checking.
- Determine line number.
- Issue disable IOB to stop line monitoring.

2 Syntax check utility control statements.

3 Stop line monitoring.

4 Issue any required messages.

5 Terminate this job step.

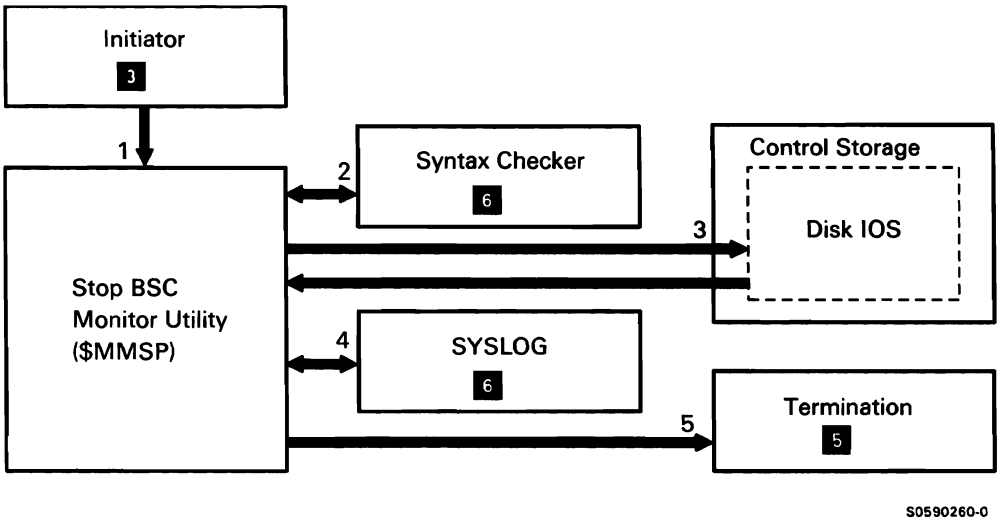


Chart 8.18 Stop BSC Monitor Utility Control Flow

START BSC MONITOR UTILITY ($MMST)

The start BSC monitor utility starts the automatic monitoring function of one of possibly four BSC multipoint lines. The start BSC monitor utility is invoked by the STARTM system procedure or equivalent OCL.

The following start BSC monitor utility processes are shown in Chart 8.19:

1 Do the following:

- Route for syntax checking.
- Determine line number.
- Determine code type (EBCDIC or ASCII).
- Determine station specified.
- Enable the line.
- Issue an IOB to start line monitoring.

2 Syntax check utility control statements.

3 Perform line monitoring.

4 Issue any required messages.

5 Terminate this job step.

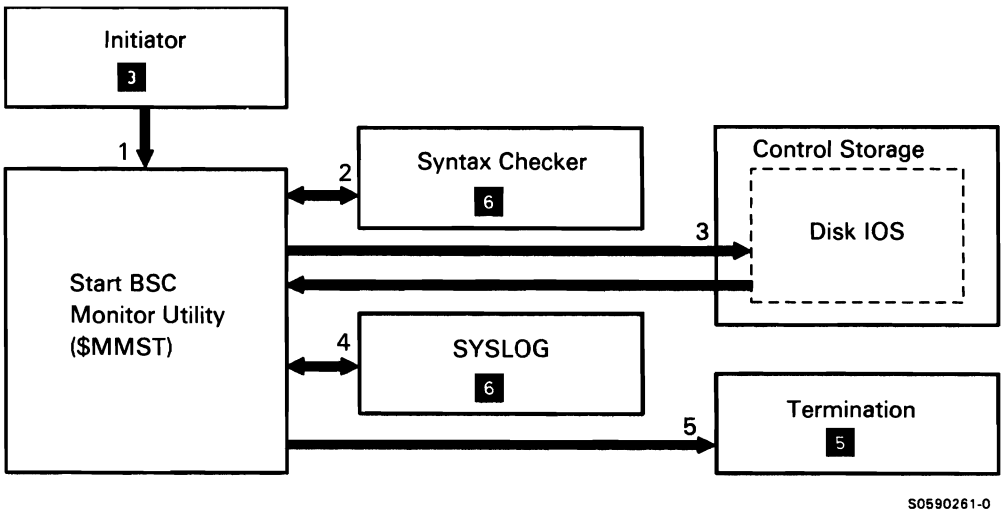


Chart 8.19 Start BSC Monitor Utility Control Flow

DISK REORGANIZATION UTILITY ($PACK/$FREE)

The disk reorganization utility reorganizes disk files so that free space on the disk is accumulated as specified by the user. The user calls the disk reorganization utility with the COMPRESS procedure command or with equivalent OCL and utility control statements; IPL file rebuild can also call disk reorganization via the transfer SVC instruction.

The following disk reorganization utility processes are shown in Chart 8.20:

1 Initialize the utility with either $PACK (for System/32 compatibility) or with $FREE, noting whether the utility was called by user or by IPL file rebuild.

2 If called by user for $FREE, read and syntax check the utility control statements.

3 Issue any required messages.

4 Create a table in TWS for each format-1 entry found on the system.

5 If called by IPL file rebuild, update the format 5s to reflect the locations and quantity of free space.

6 If FREEHIGH is specified (or implied by DISK-A1, DISK-ALL, or COMPRESS specified), accumulate free space at the high end of one or both disk drives.

7 If FREELOW is specified (or implied by DISK-A2), accumulate free space at the low end of one or both disk drives.

8 If DISK-ALL is specified (or implied by COMPRESS statement default) accumulate free space at the low end of disk A2.

9 Save COMPRESS request specifications in VTOC; need to restart and read format-1 entries to build the table.

10 Read for format-1 entries of files to be moved and, after moving the file (to fill a gap on disk) update the format 1 and its table entry.

11 Issue any required messages.

12 Update the where-to-go tables of any IBM-supplied modules in moved user libraries.

13 Update the spool file, job queue, message file, and security system files on any error exits.

14 Issue any required messages.

15 Update the format 5s to reflect the locations and quantities of free space, after the free space has been accumulated.

16 Write the updated format 5s to disk.

17 Update the PPSA addresses of any active procedures moved.

18 If called from IPL file rebuild, return there; otherwise, transfer to termination exit to end the job step.

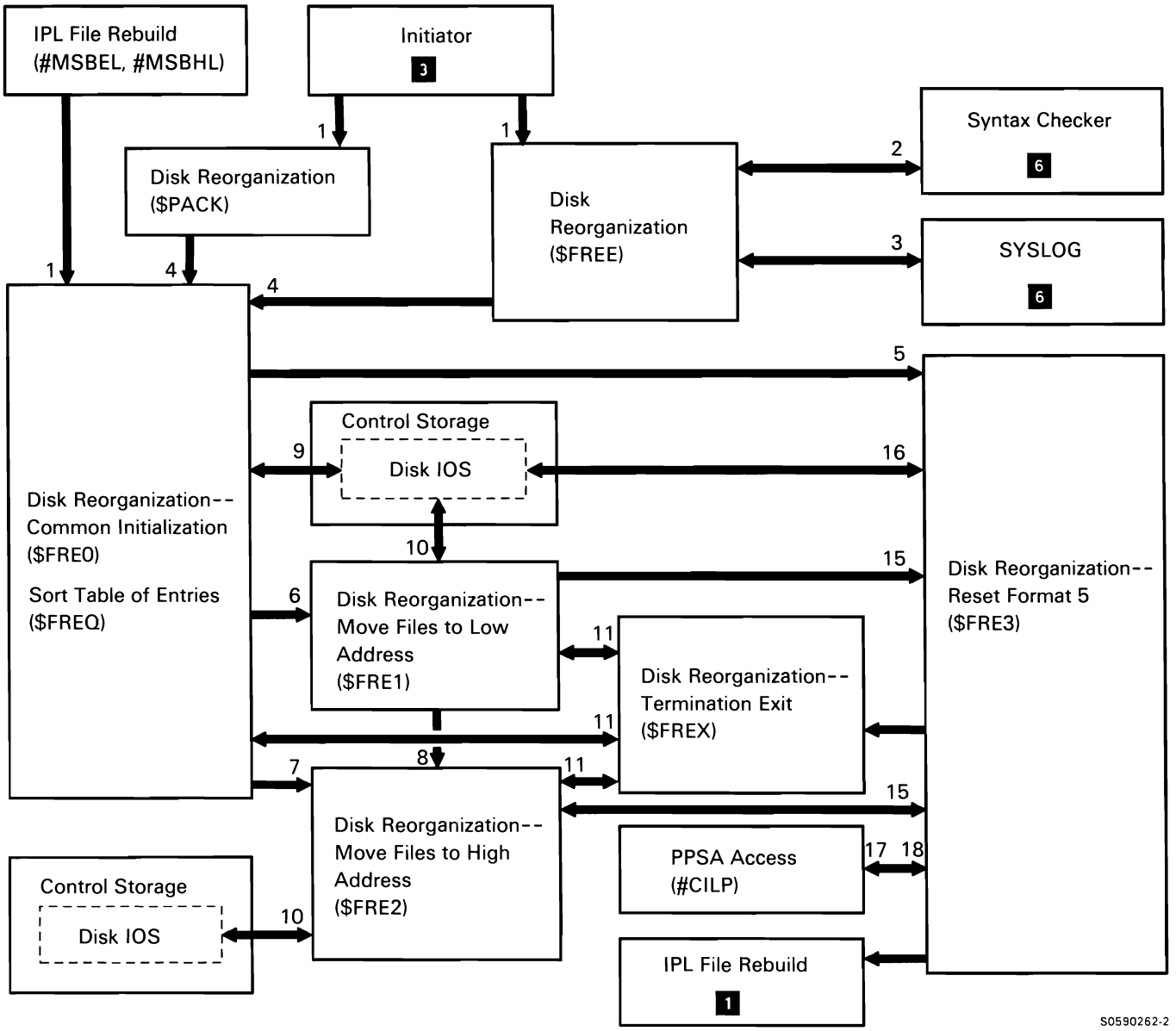


Chart 8.20 Disk Reorganization Utility Control Flow

DEFINE AUTOCALL PHONE LIST UTILITY ($PNLM)

The define autocall phone list utility either creates a new autocall phone list load member or alters an existing phone list load member. The define autocall phone list utility is invoked by the DEFINEPN system procedure or equivalent OCL.

The following define autocall phone list utility processes are shown in Chart 8.21:

1 Do the following:

- Route for syntax checking.
- Determine member program(s) specified.

2 Syntax check utility control statements.

3 Issue read/write requests for phone list load members.

4 Issue any required messages.

5 Terminate this job step.

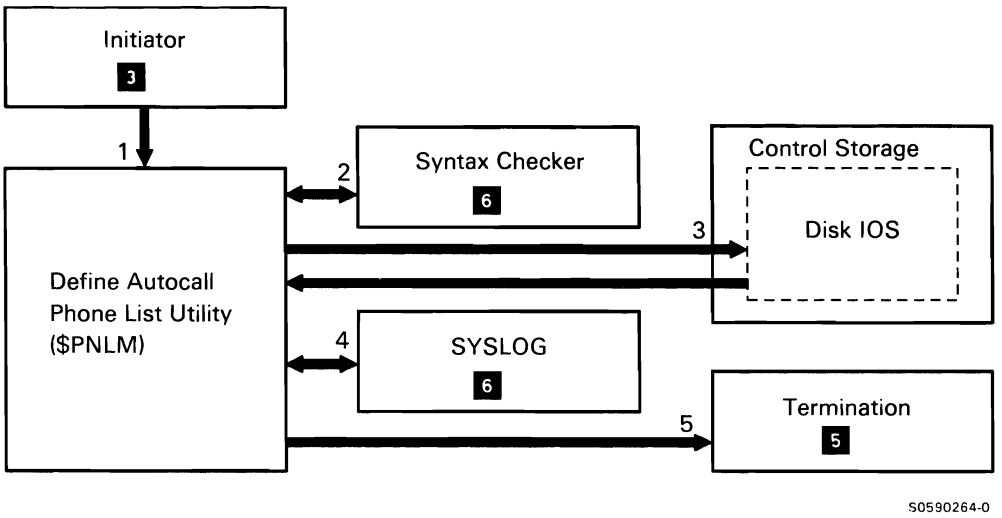


Chart 8.21 Define Autocall Phone List Utility Control Flow

This page is intentionally left blank.

POST UTILITY ($POST)

The post utility can convert a disk file to a diskette basic exchange file, convert diskette basic exchange file to a disk file, or can add a diskette basic exchange file or diskette special E file to an existing disk file. If the *input* file is a disk file, it can be a consecutive, indexed, or direct file; if the *output* file is a new disk file, it can be consecutive or indexed; if the *output* file is an existing disk file, it must be a consecutive file. The post utility is evoked by the POST procedure command or by equivalent OCL and utility control statements.

The following post utility processes are shown in Chart 8.22:

1 Read and syntax check utility control statements.

2 Get active format 1 for COPYIN file and determine operation to be performed.

3 Perform special E-diskette-to-disk copy:

  4 Find diskette file to be copied.

  5 Allocate diskette COPYIN file and disk COPYO file; open DTF for each.

  6 Read records from the diskette file.

  7 Write records to the disk file.

  8 Close the COPYIN and COPYO files.

  9 Terminate this job step.

10 Issue any required messages.

11 Perform basic exchange disk to diskette copy.

12 Perform basic exchange diskette to disk copy.

13 Perform basic exchange diskette to SYSLIST copy.

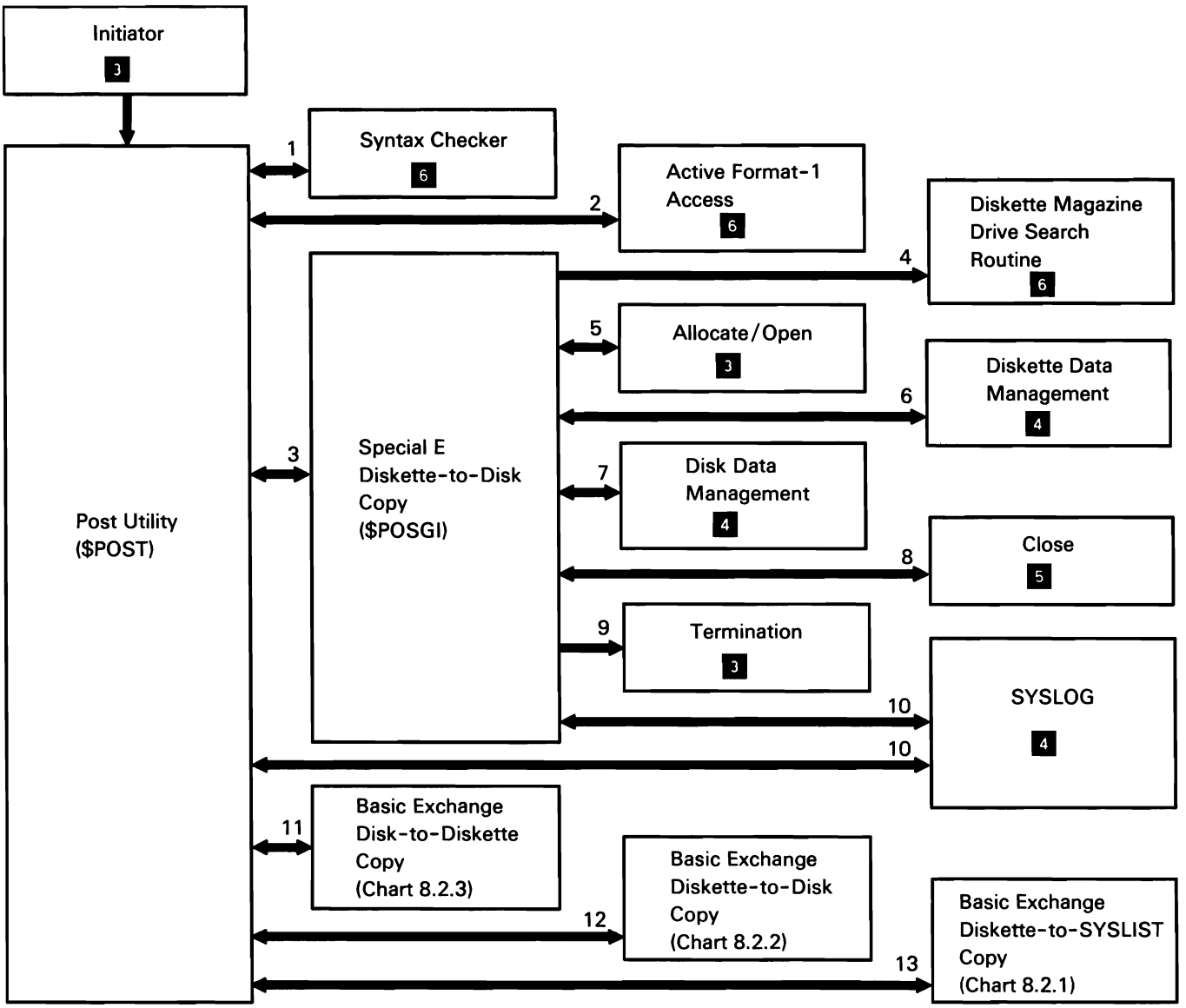


Chart 8.22 Post Utility Control Flow

SECURITY SUPPORT

Security support consists mainly of the security utilities that are evoked by IBM-supplied procedures. Also included under this topic are the security support modules; they are evoked by a variety of functions and are included here only as a convenience to the reader.

Security Utilities

The security utilities provide for password, badge, communications, and resource security. They are called by the following commands and parameters, or by equivalent OCL and utility control statements:

- SECDEF USERID evokes the user identification file definition utility ($PRUID).
- SECEDIT USERID evokes the user profile edit utility ($PRUED).
- SECSAVE USERID evokes the user identification file save utility ($PRUSV).
- SECREST USERID evokes the user identification file restore utility ($PRURS).
- SECLIST USERID evokes the user profile list utility ($PRLST).
- PASSWORD CHANGE evokes the password change utility ($PRPWD).
- SECDEF RESOURCE evokes the resource security file definition utility ($RRESC).
- SECEDIT RESOURCE evokes the resource security edit utility ($RREDT) for files, libraries, and groups.
- SECSAVE RESOURCE evokes the resource security file save utility ($RRSAV).
- SECREST RESOURCE evokes the resource security file restore utility ($RRSTR).
- SECLIST RESOURCE evokes the resource security list utility ($RRLST) for files, libraries, and groups.
- SECEDIT COMM evokes the location profile edit utility ($PRCED).
- SECLIST COMM evokes the location profile list utility ($PRCLT).
- SECEDIT OFFICE evokes the resource security edit utility ($RRTED) for folders and folder members.
- SECLIST OFFICE evokes the resource security list utility ($RRTLT) for folders.
- SECCONV converts the System/34 password security file and/or the System/34 resource security file to System/36 security file(s) ($PRCVT).

For restrictions on the use of these procedure commands, refer to the System/36 *System Security Guide*.

The following security utilities overview processes are shown in Chart 8.23.1.0:

1 Process SECDEF USERID procedure command.

2 Process SECEDIT USERID procedure command.

3 Process SECSAVE USERID procedure command.

4 Process SECREST USERID procedure command.

5 Process SECLIST USERID procedure command.

6 Process PASSWORD CHANGE procedure command.

7 Process SECDEF RESOURCE procedure command.

8 Process SECEDIT RESOURCE procedure command.

9 Process SECSAVE RESOURCE procedure command.

10 Process SECREST RESOURCE procedure command.

11 Process SECLIST RESOURCE procedure command.

12 Process SECEDIT COMM procedure command.

13 Process SECLIST COMM procedure command.

14 Process SECEDIT OFFICE procedure command.

15 Process SECLIST OFFICE procedure command.

16 Process SECCONV procedure command.

All disk accesses to the user identification file and the resource security file are processed by user identification file data management (@PRUDM) and resource security file data management (@RRDMT), respectively. These modules are loaded into the caller's address space and are branched to, like subroutines. The modules use control storage ASGN to obtain an area of SQS to contain the disk IOB and control storage disk IOS for scanning, reading, or writing the file data.

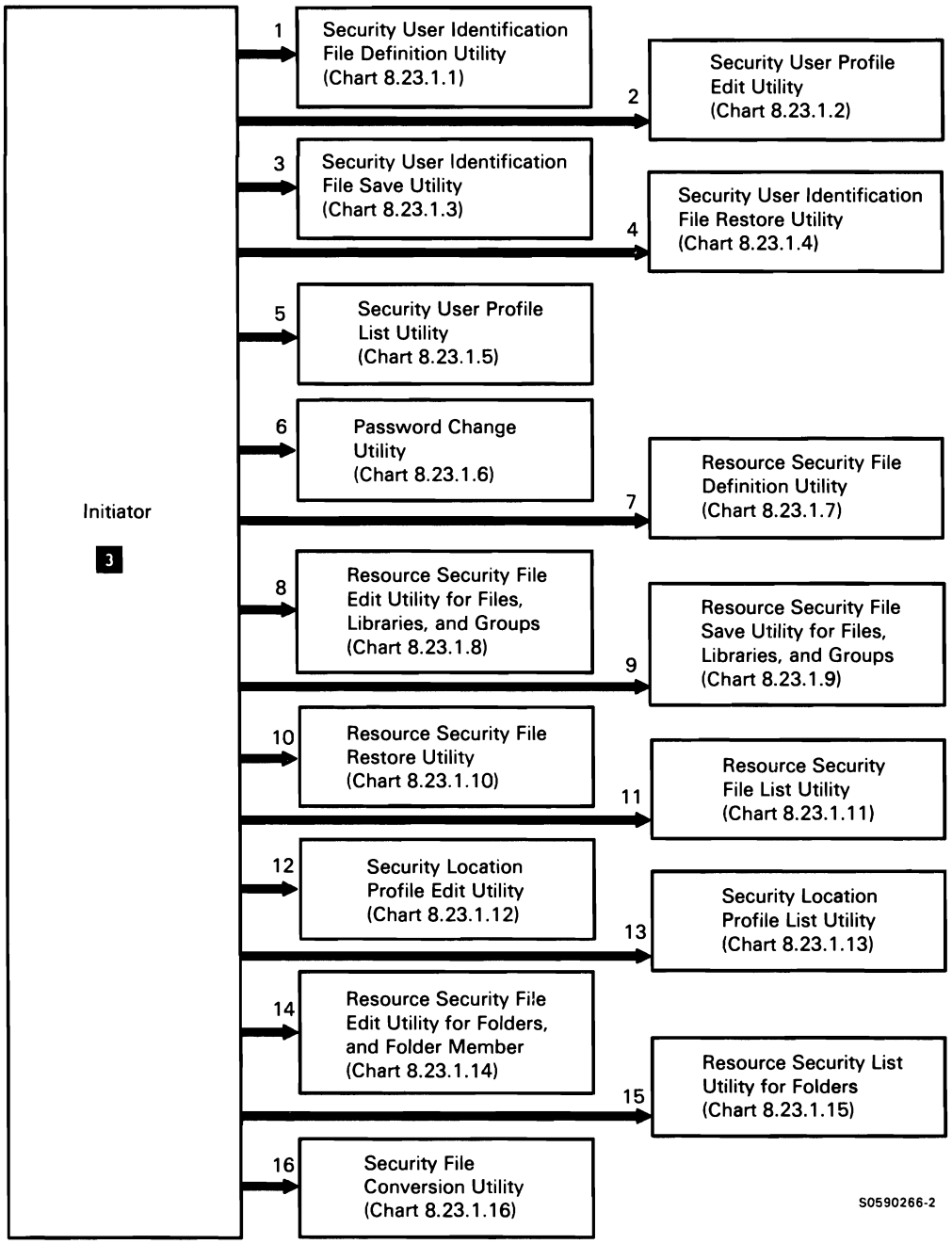


Chart 8.23.1.0 Security Utilities Overview Control Flow

Security User Identification File Definition Utility

The security user identification file definition utility creates and deletes the user identification file, activates and deactivates password security, activates and deactivates badge security, and activates and deactivates password date checking.

The following security user identification file definition utility processes are shown in Chart 8.23.1.1:

1 *Read and syntax check* utility control statements.

2 Get the utility interlock to ensure another security utility is not running.

3 Allocate the user identification file.

4 Read or write information from disk.

5 Read or write to display station.

6 Get/put, scramble/unscramble, user identification file records.

7 Release the utility interlock so another security utility can run.

8 Issue any required messages.

9 Terminate this job step.

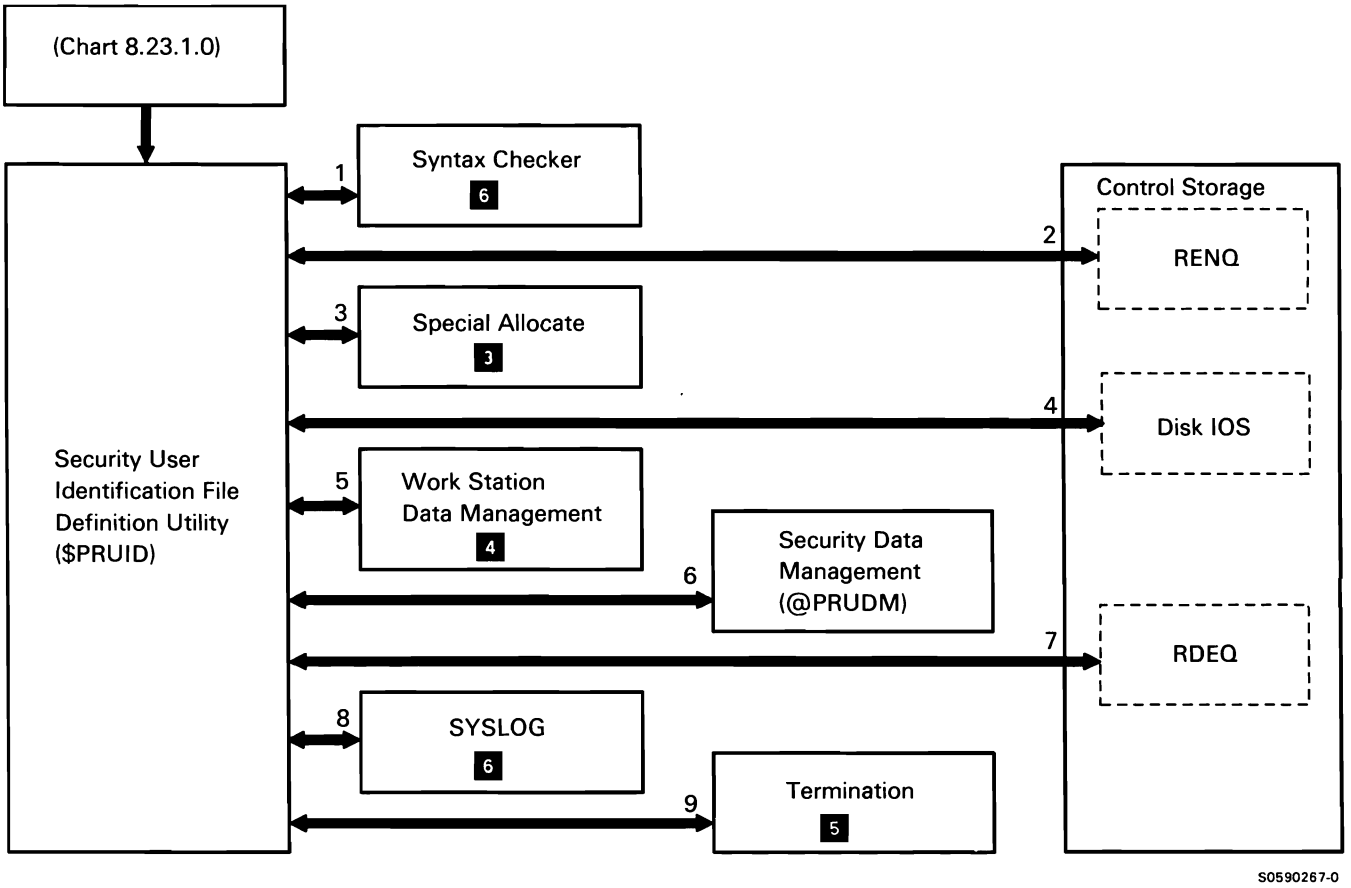


Chart 8.23.1.1 Security User Identification File Utility Control Flow

This page is intentionally left blank.

Security User Profile Edit Utility

The security user profile edit utility updates, adds, deletes, and locates user profiles in the user identification file.

The following security user profile edit utility processes are shown in Chart 8.23.1.2:

1 Get the utility interlock to ensure another security utility is not running.

2 Open the display file at the display station.

3 Build the termination exit block (TEB) in the TB.

4 Read or write to display station.

5 Get/put, scramble/unscramble, user identification file records.

6 Special allocate the user identification file for an extend.

7 Assign extend parameter list.

8 Extend resource security file.

9 Issue any required messages.

10 If abnormal termination occurs while TEB is active, perform cleanup function.

11 Deactivate TEB in TB so termination will not take error exit.

12 Release the utility interlock so another security utility can run.

13 Terminate this job step.

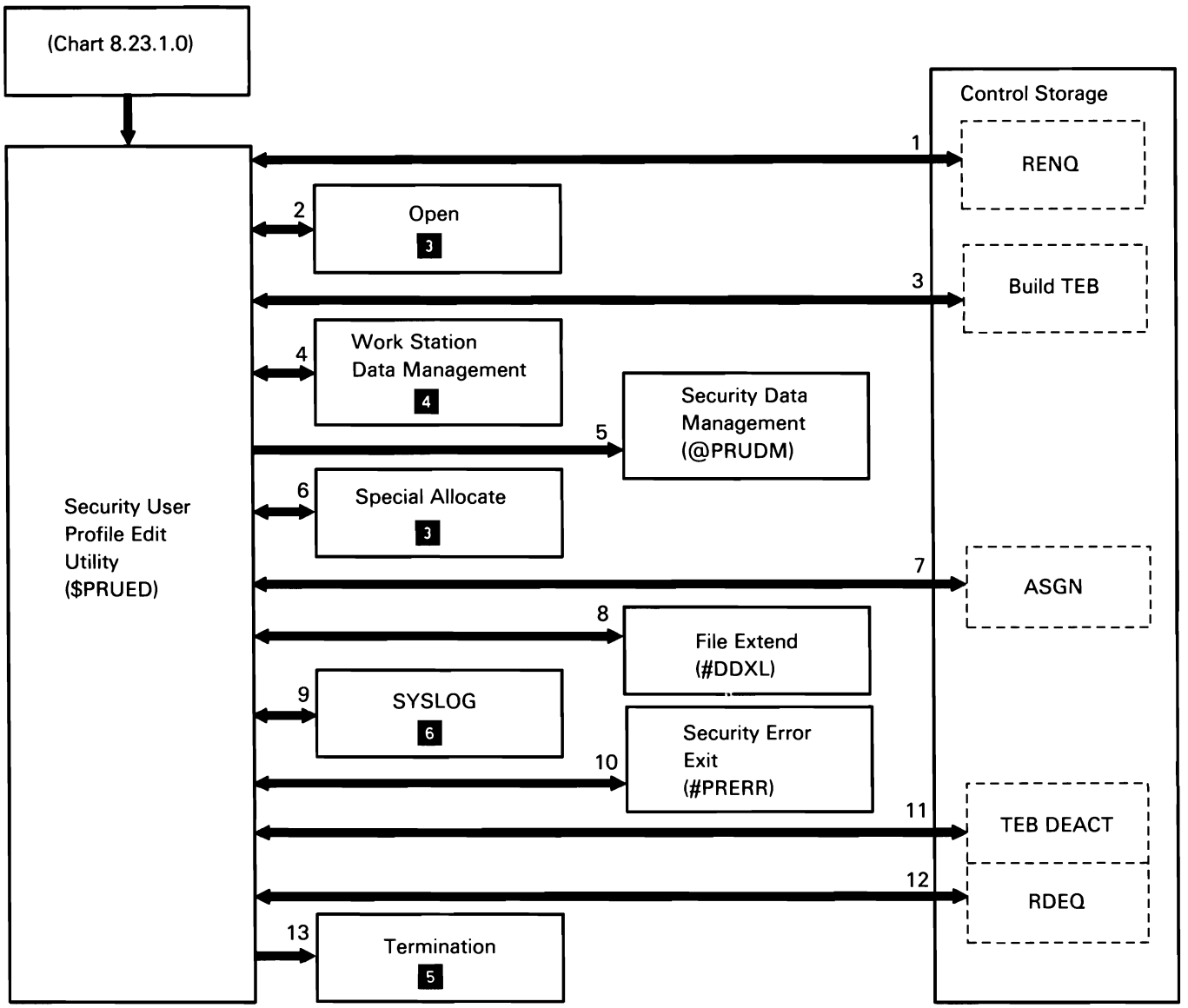


Chart 8.23.1.2 Security User Profile Edit Utility Control Flow

User Identification File Save Utility

The user identification file save utility saves the user identification file on disk, diskette, or tape.

The following user identification file save utility processes are shown in Chart 8.23.1.3:

1 Get the utility interlock to ensure another security utility is not running.

2 Read from, or write to, disk, diskette, or tape.

3 Allocate the job file (#SECCOPY) to receive the copy of the user identification file.

4 Get/put, scramble/unscramble, user identification file records.

5 Release the utility interlock so another security utility can execute.

6 Terminate this job step.

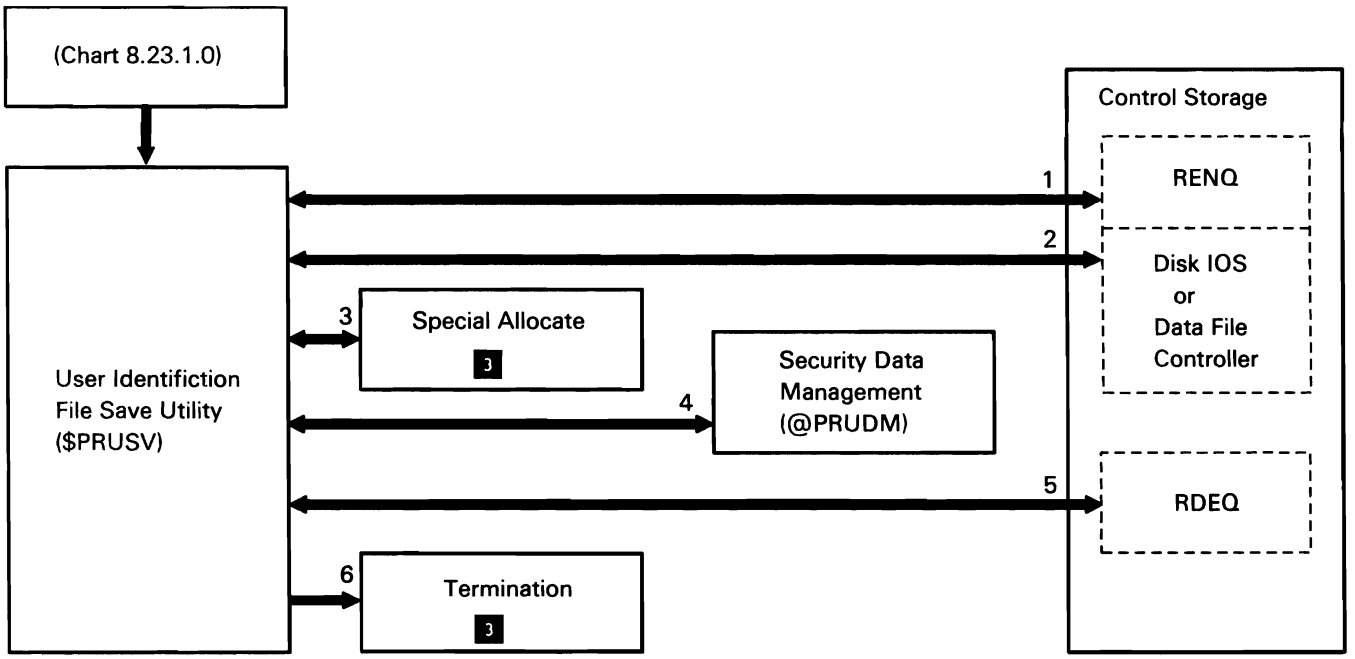


Chart 8.23.1.3 User Identification File Save Utility Control Flow

This page is intentionally left blank.

User Identification File Restore Utility

The user identification file restore utility restores the user identification file on disk from a disk, diskette, or tape file created by SECSAVE.

The following user identification file restore utility processes are shown in Chart 8.23.1.4:

1 Read and syntax check the utility control statements.

2 Get the utility interlock to ensure another security utility is not running.

3 Activate the termination exit block (TEB) in the TB.

4 Read from, or write to, disk, diskette, or tape.

5 Allocate the new user identification file.

6 Allocate the previously created job file (#SECREST).

7 Get/put, scramble/unscramble, user identification file records.

8 Issue any required informational message(s).

9 Issue any required message(s).

10 Delete the user identification file.

11 Deactivate TEB in TB so termination will not take error exit.

12 Release the utility interlock so another security utility can execute.

13 Terminate this job step.

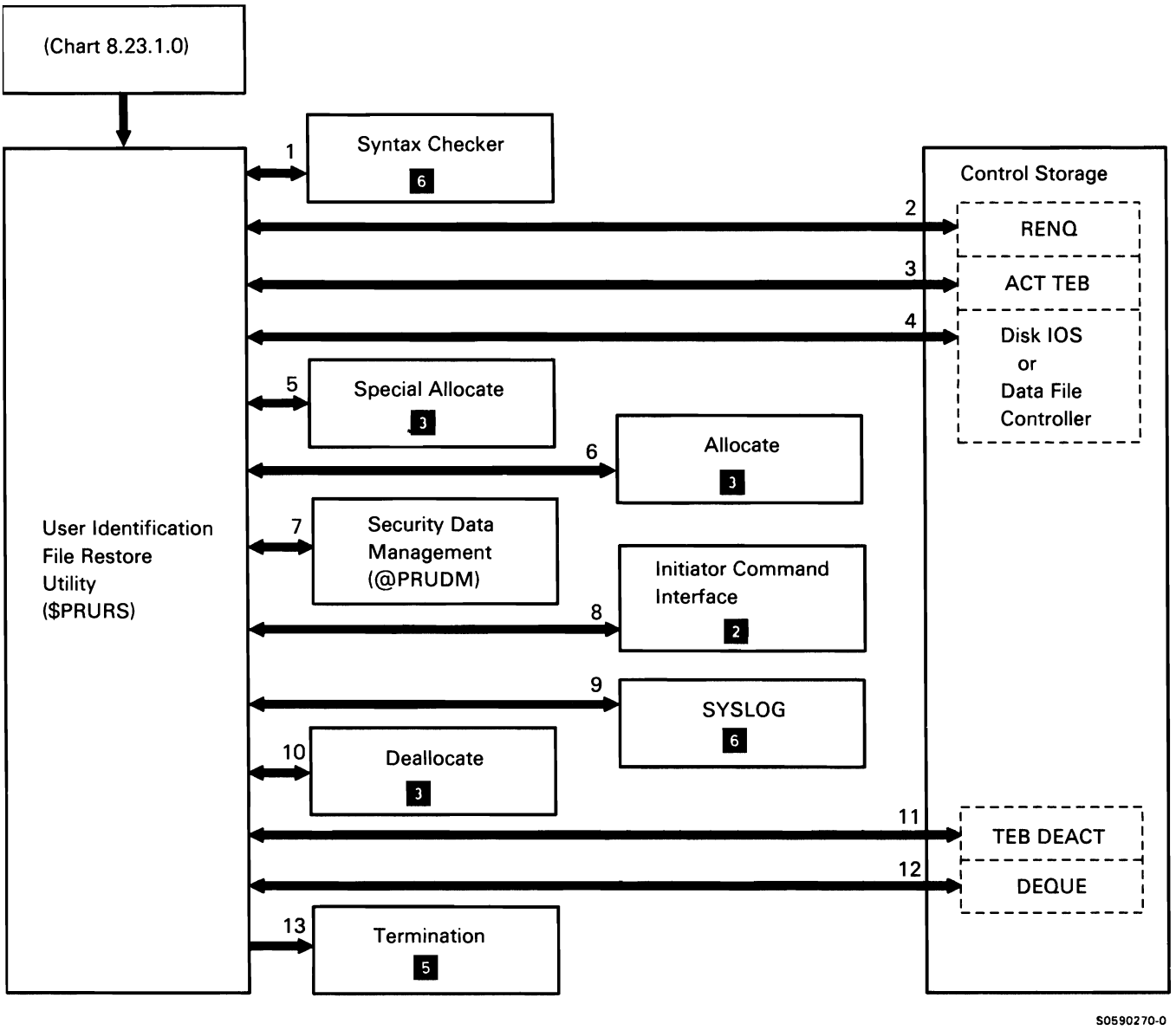


Chart 8.23.1.4 User Identification File Restore Utility Control Flow

Security User Profile List Utility

The security user profile list utility lists user profiles from the user identification file to either the system printer or to the display station. The output is sorted alphabetically within the security class or alphabetically by USERID.

The following security user profile list utility processes are shown in Chart 8.23.1.5:

1 Read and syntax check the utility control statements.

2 Get the utility interlock to ensure another security utility is not running.

3 Allocate and open the disk work file.

4 Activate the termination exit block (TEB) in the TB.

5 Retrieve and unscramble user profiles in the user identification file.

6 Put records to the disk work file.

7 Read configuration sector for security; zero disk work file on error.

8 Get heading lines.

9 Get time and date for headings.

10 Get records from disk.

11 List user identification file records.

12 Deactivate TEB in TB so termination will not take error exit.

13 Release the utility interlock so another security utility can execute.

14 Issue any required messages.

15 If error occurs while TEB active, process error.

16 Terminate this job step.

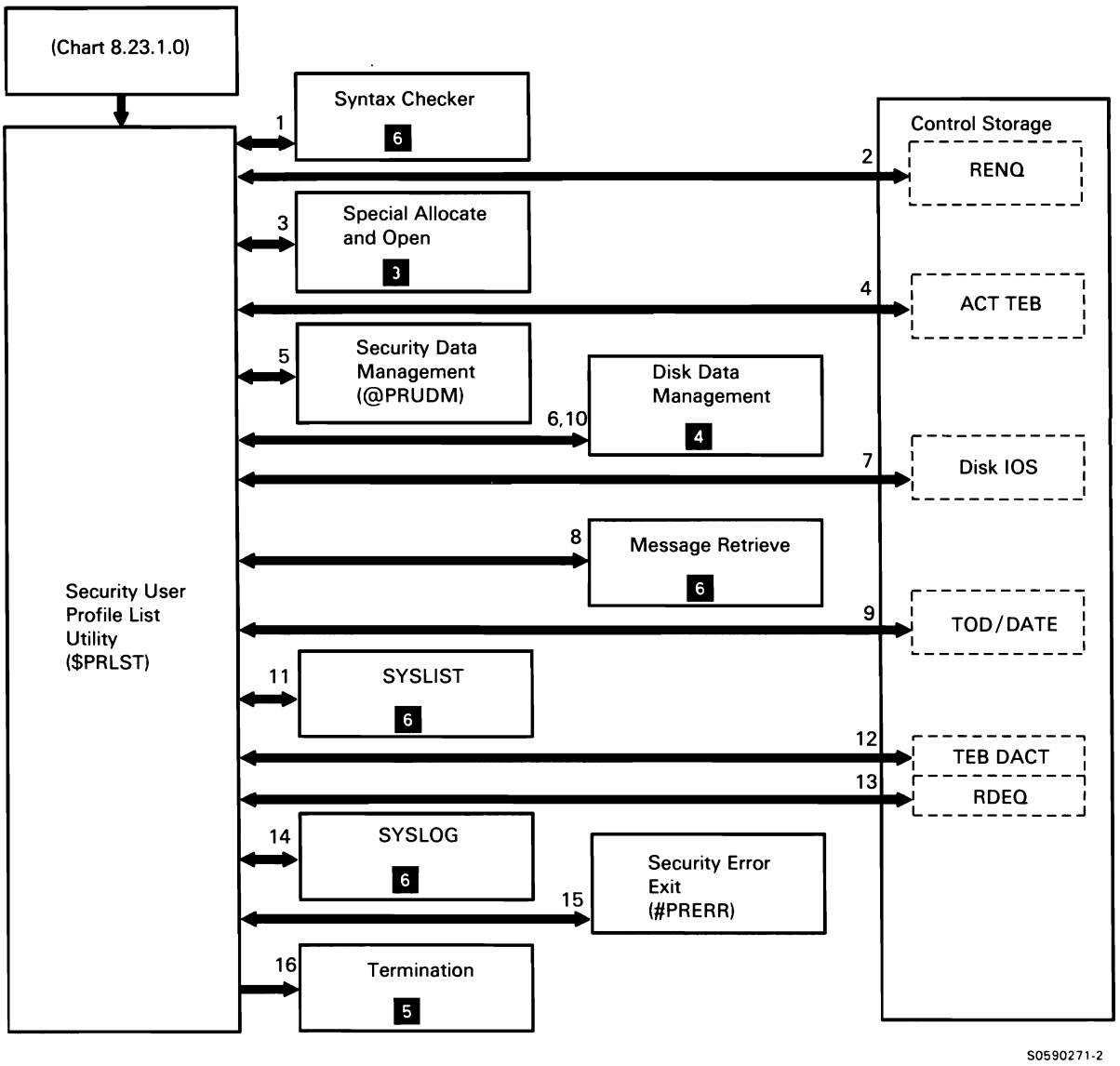


Chart 8.23.1.5 Security User Profile List Utility Control Flow

Password Change Utility

The password change utility allows a user to change his or her password. The user password is in the user profile of the user identification file.

The following password change utility processes are shown in Chart 8.23.1.6:

1 Get the utility interlock to make sure another security utility is not running.

2 Open the display file at the display station.

3 Build the termination exit block (TEB) in the TB.

4 Read and write to display station.

5 Get/put, scramble/unscramble, the user profile of the user requesting to change his or her password.

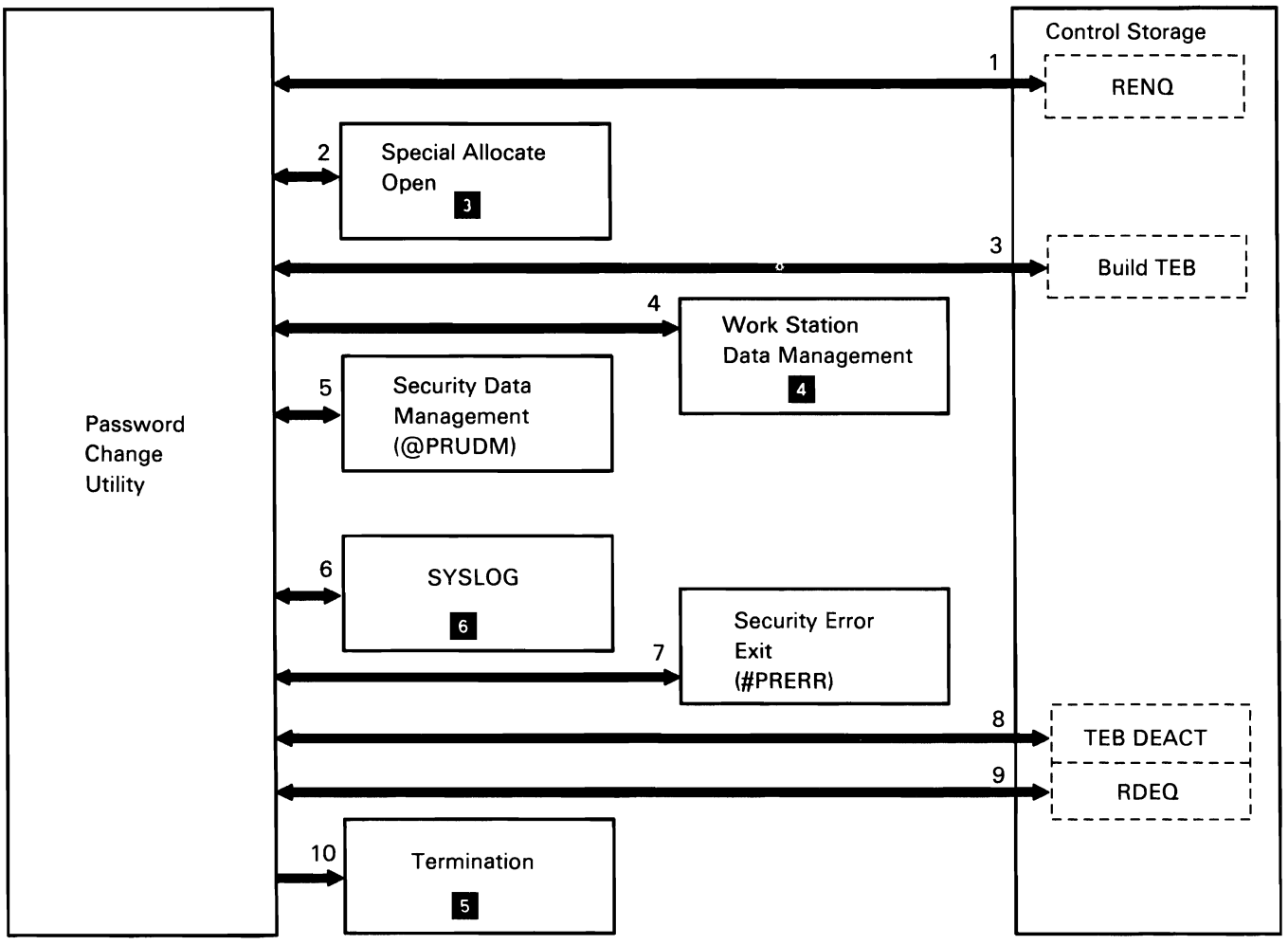
6 Issue any required messages.

7 If abnormal termination occurs while TEB is active, perform cleanup function.

8 Deactivate TEB in TB so termination will not take error exit.

9 Release the utility interlock so another security utility can run.

10 Terminate this job step.



S0590463-0

Chart 8.23.1.6 Resource Security Password Change Utility Control Flow

This page is intentionally left blank.

Resource Security File Definition Utility

The resource security file definition utility creates and deletes the resource security file and activates and deactivates resource security.

The following resource security file definition utility processes are shown in Chart 8.23.1.7:

1 Read and syntax check utility control statements.

2 Get the utility interlock to ensure another security utility is not running.

3 Allocate the new resource security file or allocate the old resource security file so it can be deleted.

4 Open the file.

5 Read from, or write to, disk.

6 Read or write to display station.

7 Get/put, scramble/unscramble, resource security file records.

8 Delete the resource security file from the VTOC.

9 Release the utility interlock so another security utility can execute.

10 Issue any required messages.

11 Terminate this job step.

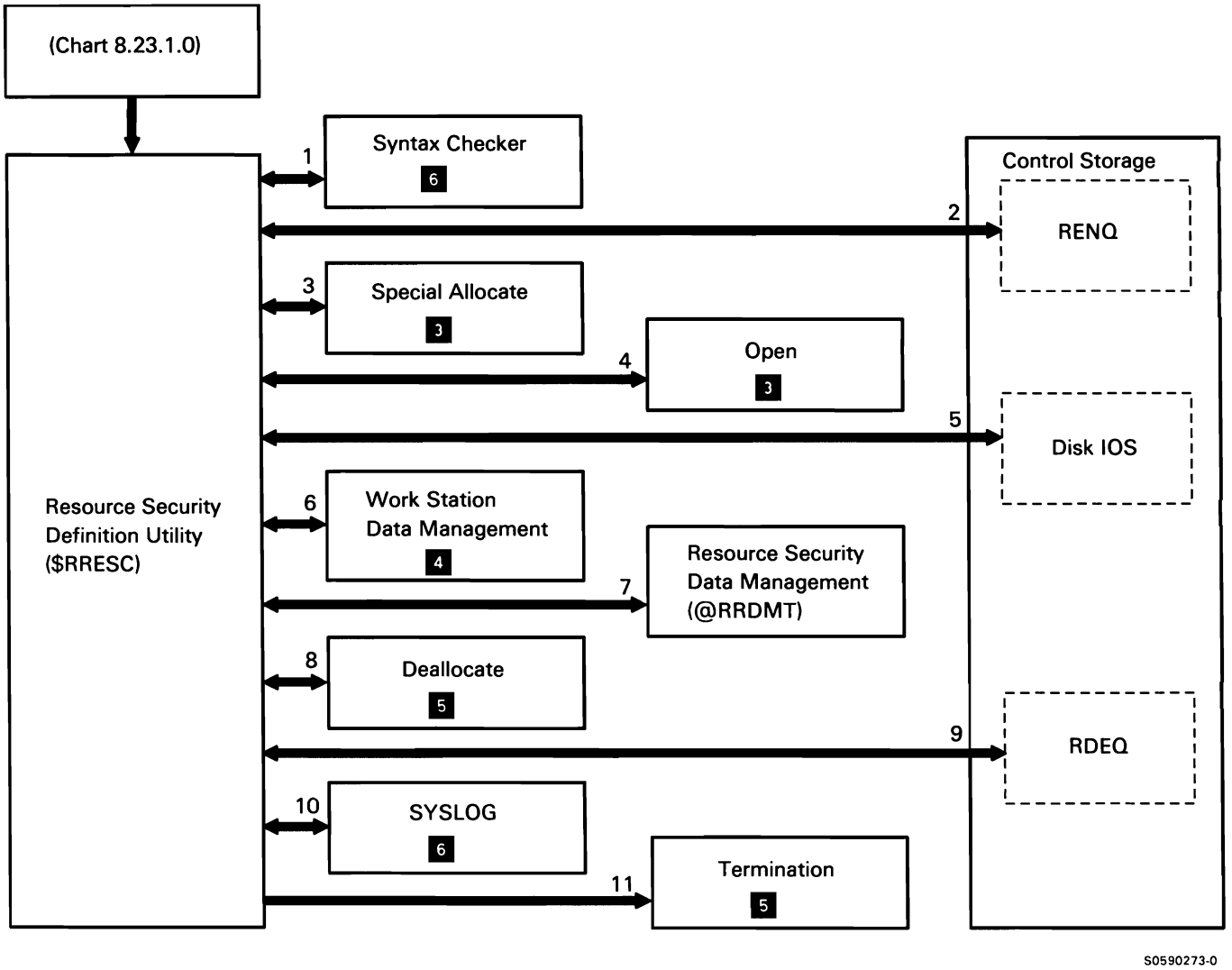


Chart 8.23.1.7 Resource Security Definition Utility Control Flow

Resource Security File Edit Utility

The resource security file edit utility updates, adds, deletes, and finds resource security file information on files, libraries, and groups.

In addition to being called from the security menu, the resource security file edit utility can also be called by privileged programs such as PS/36 to prompt the user for file or library security.

The following resource security file edit utility processes are shown in Chart 8.23.1.8:

1 Get the utility interlock to ensure another security utility is not running.

2 Activate the termination exit block (TEB) in the TB so controlled cancel can be done even if the job is canceled.

3 Open the display format.

4 Read or write to display station.

5 Get/put, scramble/unscramble, resource security file records.

6 Read/write VTOC, marking files and libraries as secure or unsecure.

7 Special allocate the resource security file for an extend.

8 Assign extend parameter list.

9 Extend resource security file.

10 Read/write security configuration sector.

11 Issue any required messages.

12 If abnormal termination occurs while TEB is active, perform cleanup.

13 Deactivate TEB in TB so termination will not take error exit.

14 Release the utility interlock so another security utility can run.

15 If $RREDT has been called by another program, exit.

16 Terminate this job step.

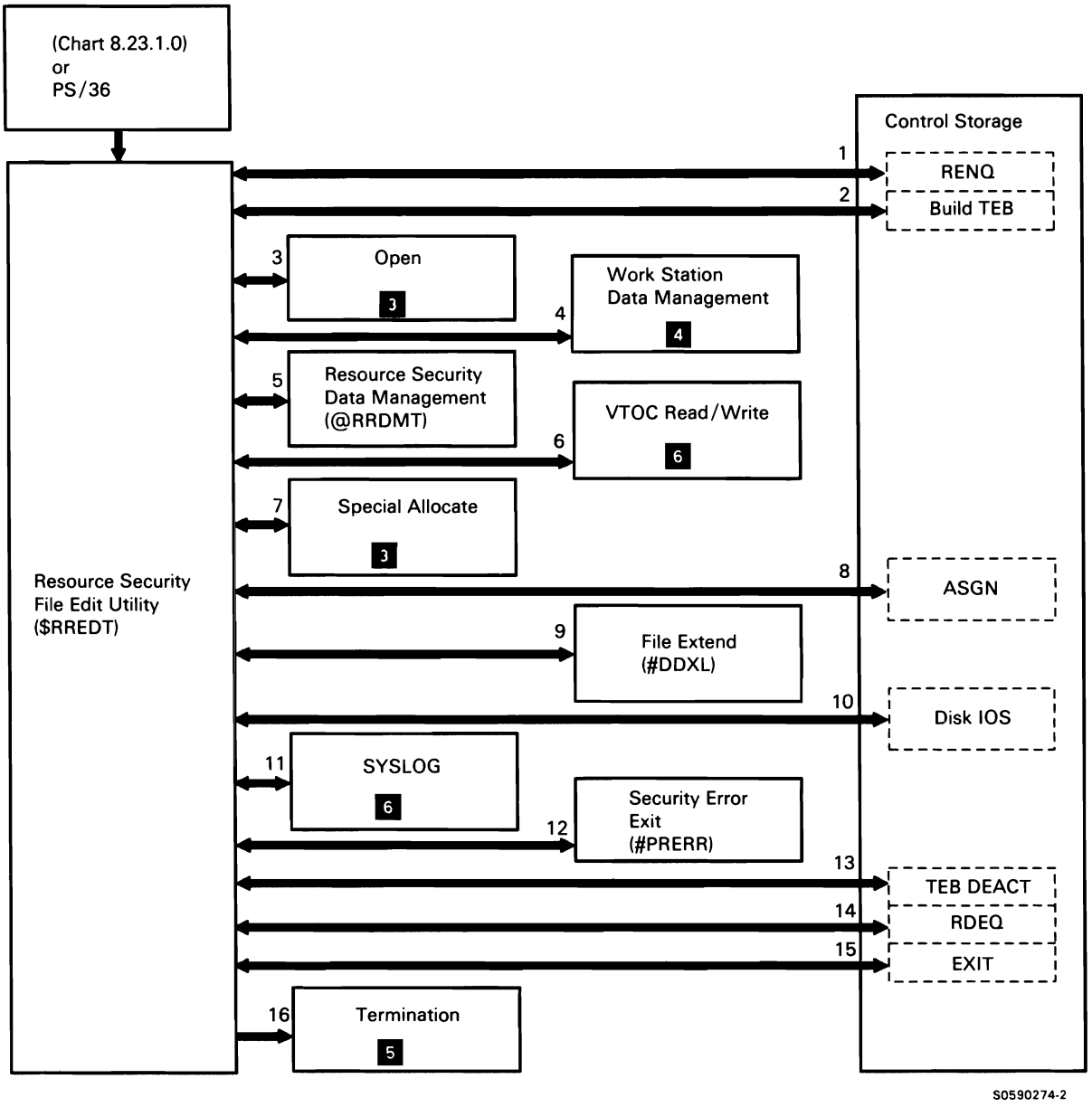


Chart 8.23.1.8 Resource Security File Edit Utility Control Flow

Resource Security File Save Utility

The resource security file save utility saves the resource security file on disk, diskette, or tape.

The following user identification file save utility processes are shown in Chart 8.23.1.9:

1 Get the utility interlock to ensure another security utility is not running.

2 Read from, or write to, disk, diskette, or tape.

3 Allocate the job file (#SECCOPY) where the copy of the resource security file is saved.

4 Get/put, scramble/unscramble, resource security file records.

5 Release the utility interlock so another security utility can execute.

6 Issue any required messages.

7 Terminate this job step.

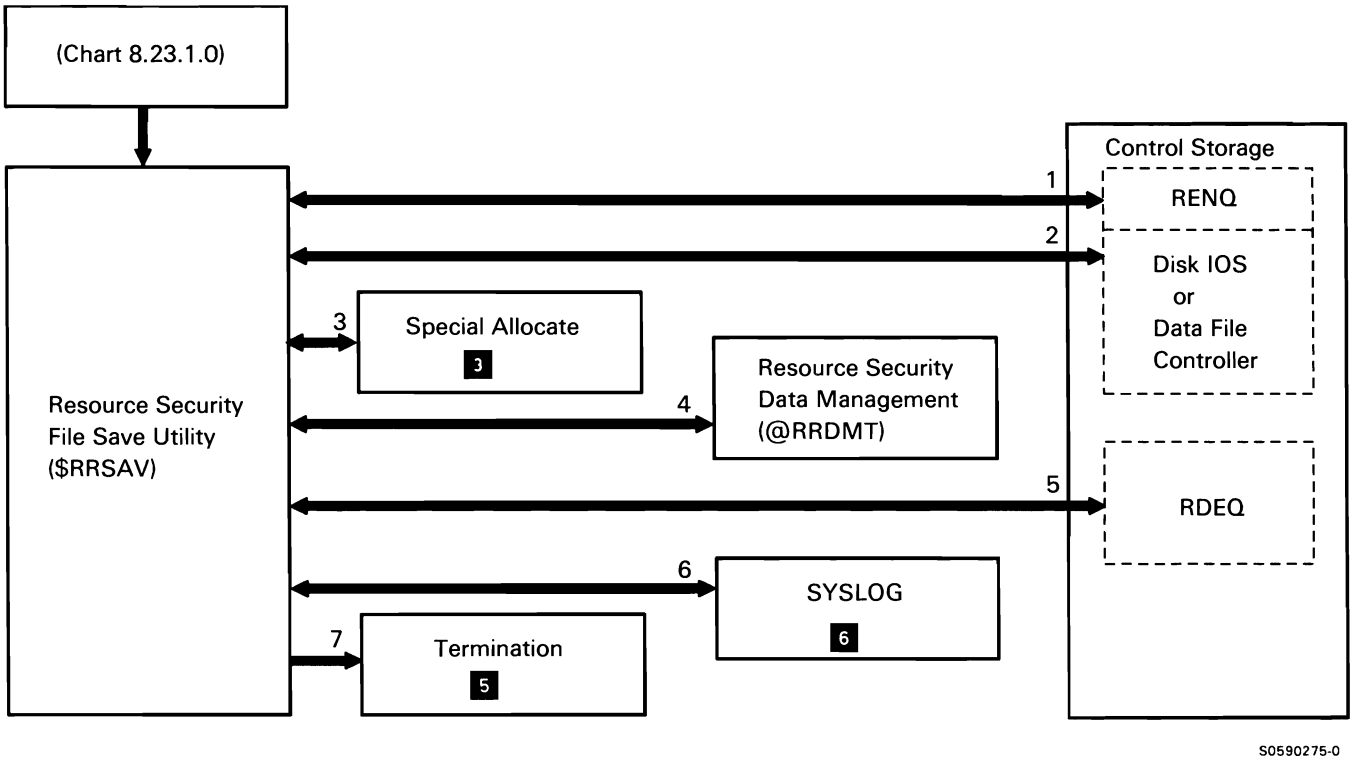


Chart 8.23.1.9 Resource Security File Save Utility Control Flow

This page is intentionally left blank.

Resource Security File Restore Utility

The resource security file restore utility restores, as the current resource security file, a saved copy on disk, diskette, or tape.

The following user identification file restore utility processes are shown in Chart 8.23.1.10:

1 Read and syntax check the utility control statements.

2 Get the utility interlock to ensure another security utility is not running.

3 Activate the termination exit block (TEB) in the TB.

4 Allocate the new resource security file.

5 Allocate the previously created job file (#SECREST).

6 Read from or write to disk, diskette, or tape.

7 Get/put, scramble/unscramble, resource security file records.

8 Delete the old resource security file.

9 Get the VTOC format-1 interlock.

10 Search the VTOC for each file record in the resource security file.

11 Release the disk interlock.

12 Release the utility interlock so another security utility can execute.

13 Issue any required informational messages to the system console.

14 Issue any required messages.

15 If abnormal termination occurs while TEB is active, perform cleanup.

16 Deactivate TEB in TB so termination will not take error exit.

17 Terminate this job step.

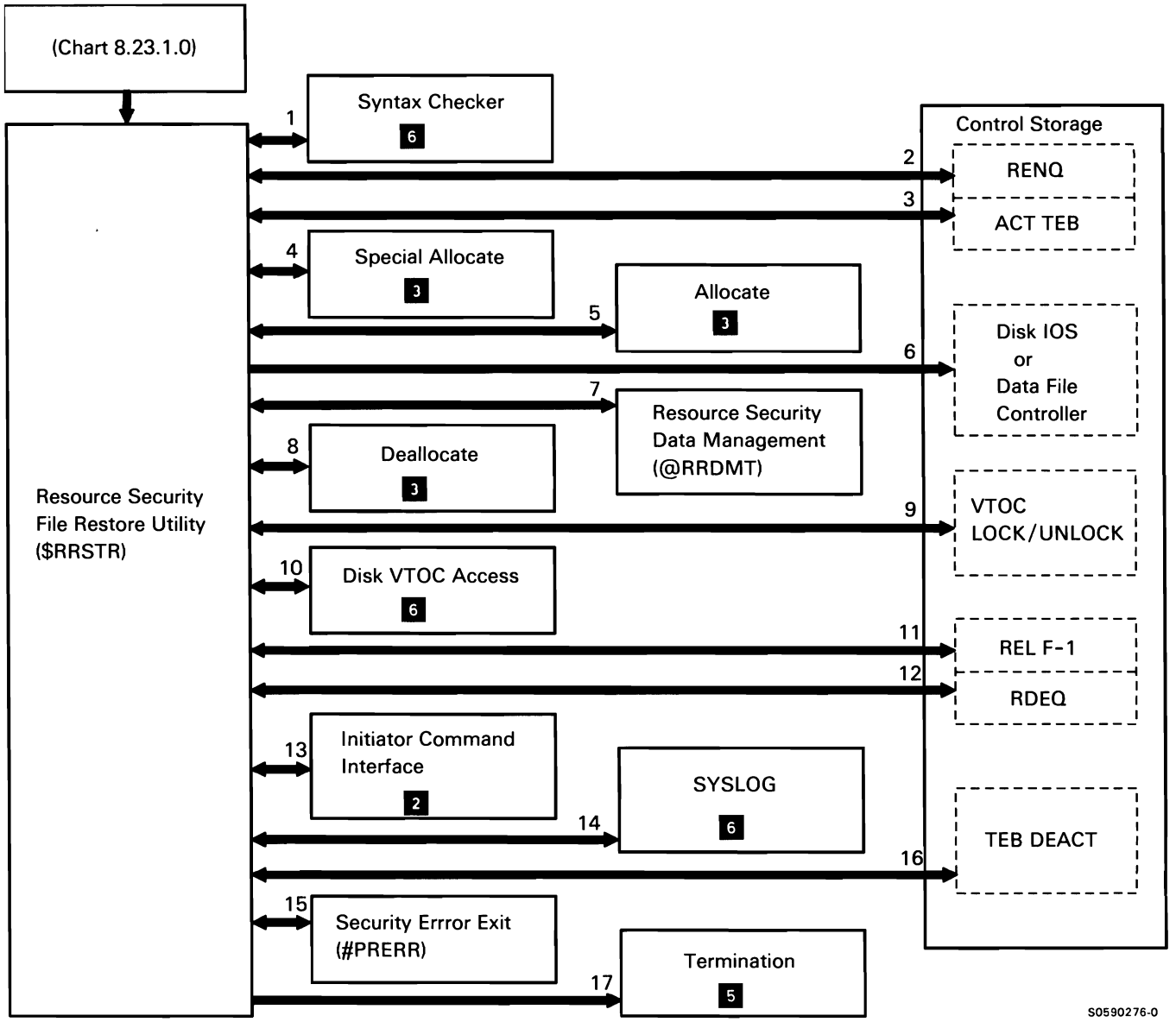


Chart 8.23.1.10 Resource Security File Restore Utility Control Flow

Resource Security File List Utility

The resource security file list utility lists resource security file information on files, libraries, and groups to the SYSLIST device. The output is sorted according to user specifications contained in the utility control statements.

The following resource security file list utility processes are shown in Chart 8.23.1.11:

1 Read and syntax check the utility control statements.

2 Get the utility interlock to ensure another security utility is not running.

3 Allocate and open the disk work file.

4 Activate the termination exit block (TEB) in the TB.

5 Retrieve file records and user records.

6 Search the VTOC for each file record in the resource security file.

7 Put records to the disk work file.

8 Read configuration sector for security; zero disk work file on error.

9 Close the disk work file.

10 Sort the work file alphabetically within security class.

11 Allocate the work file for output.

12 Get heading lines.

13 Get time and date for headings.

14 Get records from disk.

15 List file, library, and group security information.

16 Deactivate TEB in TB so termination will not take error exit.

17 Release the utility interlock so another security utility can run.

18 Issue any required messages.

19 If error occurs while TEB active, process error.

20 Terminate this job step.

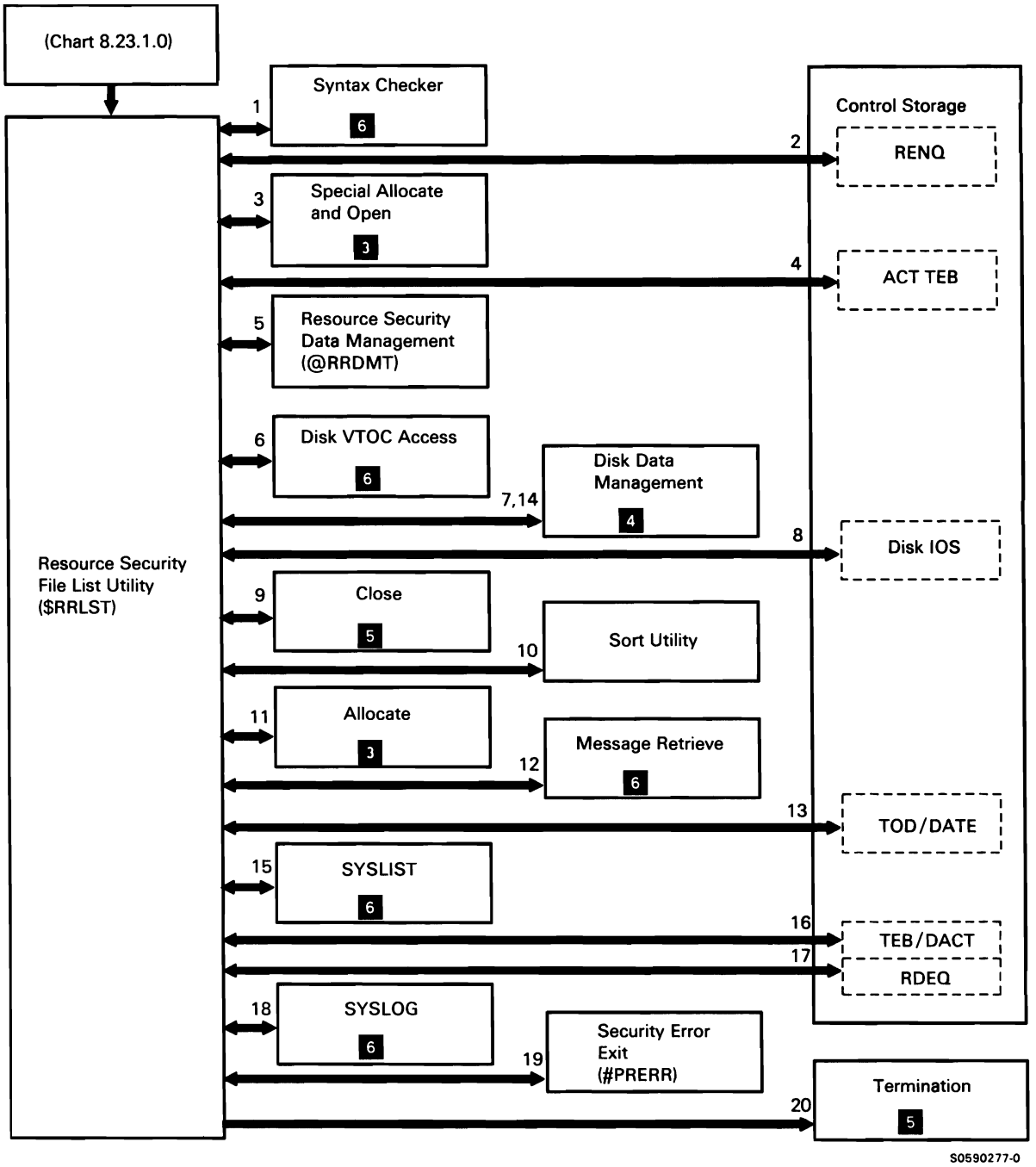


Chart 8.23.1.11 Resource Security File List Utility Control Flow

Security Location Profile Edit Utility

The security location profile edit utility updates, adds, deletes, and finds location profiles in the user identification file.

The following security location profile edit utility processes are shown in Chart 8.23.1.12:

1 Get the utility interlock to ensure another security utility is not running.

2 Open the display file at the display station.

3 Build the termination exit block (TEB) in the TB.

4 Read or write to display station.

5 Get/put, scramble/unscramble, location profiles.

6 Special allocate the user identification file for an extend.

7 Assign extend parameter list.

8 Extend resource security file.

9 Get time/date stamp information.

10 Issue any required messages.

11 If abnormal termination occurs while TEB is active, perform cleanup function.

12 Deactivate TEB in TB so termination will not take error exit.

13 Release the utility interlock so another security utility can run.

14 Terminate this job step.

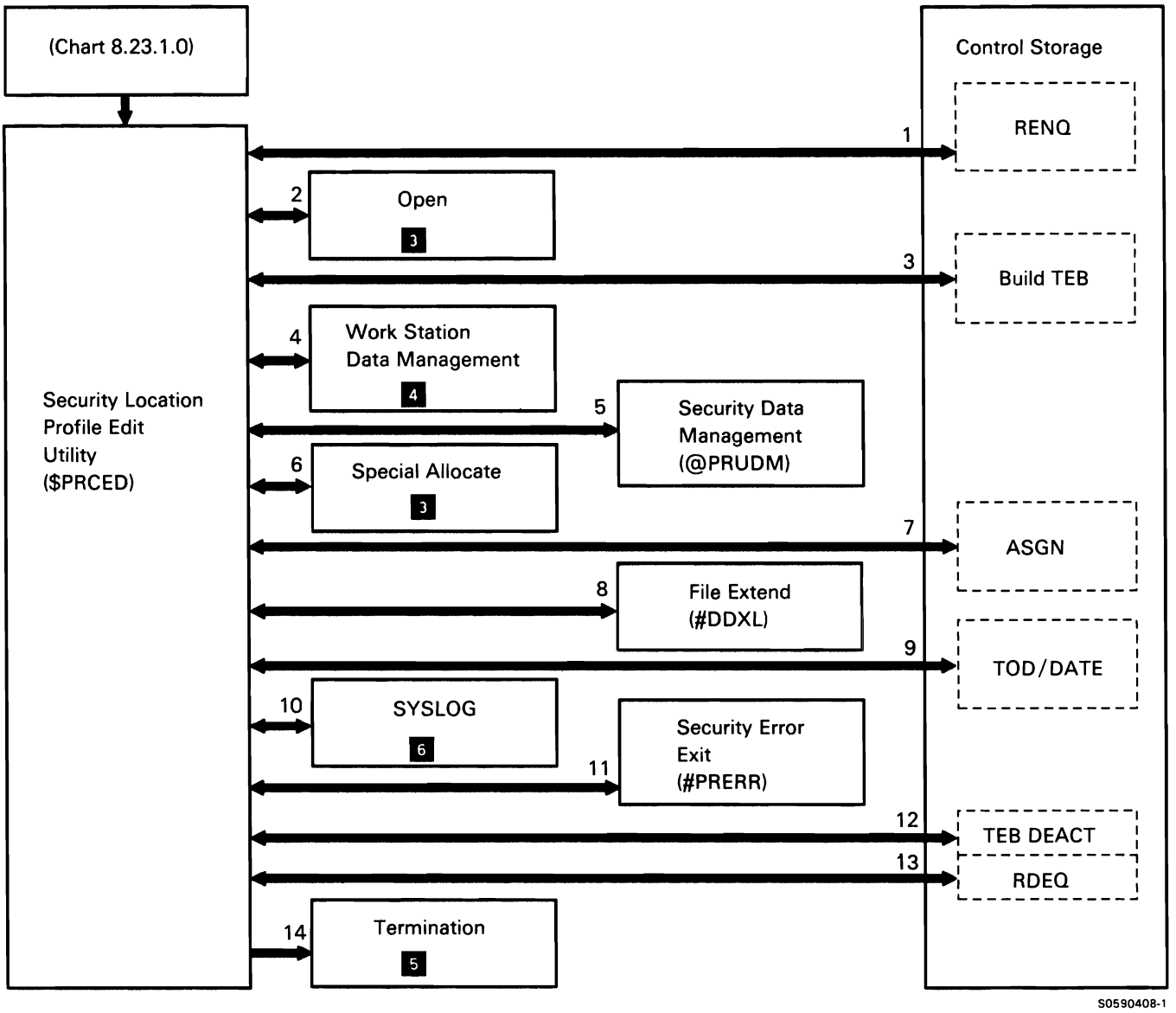


Chart 8.23.1.12 Security Location Profile Edit Utility Control Flow

Security Location Profile List Utility

The security location profile list utility lists location profiles from the user identification file to either the system printer or to the display station. The output is sorted alphabetically by location name.

The following security location profile list utility processes are shown in Chart 8.23.1.13:

1 Get the utility interlock to ensure another security utility is not running.

2 Allocate and open the disk work file.

3 Activate the termination exit block (TEB) in the TB.

4 Retrieve and unscramble location profiles in the user identification file.

5 Put records to the disk work file.

6 Read configuration sector for security; zero disk work file on error.

7 Close the disk work file.

8 Sort the work file alphabetically by location name.

9 Allocate the work file for output.

10 Get heading lines.

11 Get time and date for headings.

12 Get records from disk.

13 List location profiles.

14 Deactivate TEB in TB so termination will not take error exit.

15 Release the utility interlock so another security utility can run.

16 Issue any required messages.

17 If error occurs while TEB active, process error.

18 Terminate this job step.

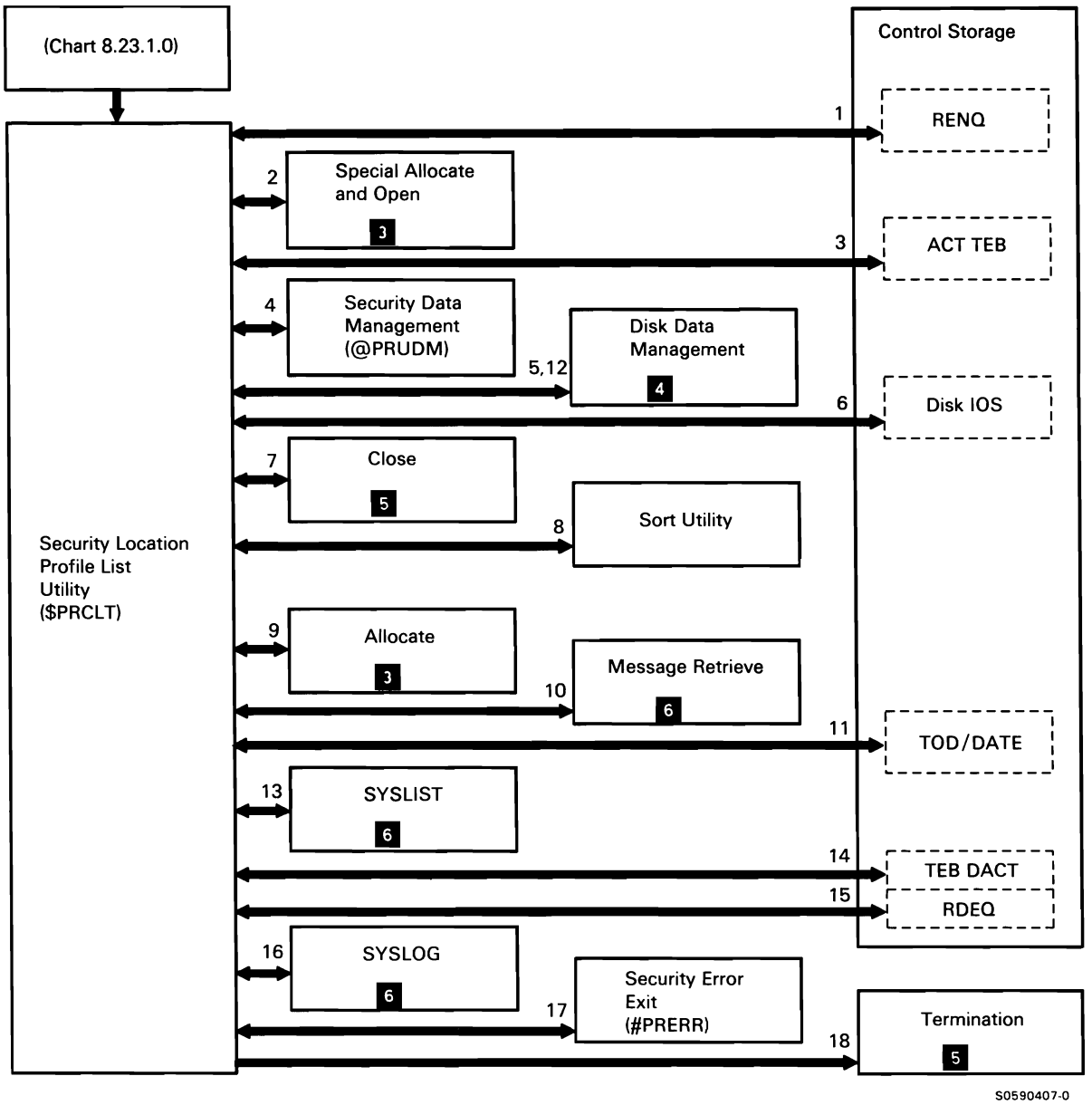


Chart 8.23.1.13 Security Location Profile List Utility Control Flow

Resource Security File Edit Utility for Folders and Folder Members

The resource security file edit utility for folders and folder members updates, adds, deletes, and locates security information on folders and folder members.

In addition to being called from the security menu, the resource security file edit utility for folders and folder members is also called by the OFFICE/36 program products.

The following resource security file edit utility processes are shown in Chart 8.23.1.14:

1 Get the utility interlock to ensure another security utility is not running.

2 Activate the termination exit block (TEB) in the TB so controlled cancel can be done even if the job is canceled.

3 Get heading lines.

4 Open the display format.

5 Special allocate the work files.

6 Open the work files.

7 Get/put, scramble/unscramble, security information for folders and folder members.

8 Get/put entries from/to the work file.

9 Read or write to display station.

10 If called by one of the text processing utilities, map to caller's TMIO parameter list.

11 Open/close folders and folder members and get/put DDA (data descriptor area).

12 Mark VTOC entries for folders secured or unsecured, as required.

13 Issue any required messages.

14 Special allocate the resource security for an extend.

15 Assign extend parameter list.

16 Extend resource security file.

17 Read/write security configuration sector; zero disk work files.

18 If abnormal termination occurs while TEB is active, perform cleanup.

19 Deactivate TEB in TB so termination will not take error exit.

20 Release the utility interlock so another security utility can run.

21 Terminate this job step.

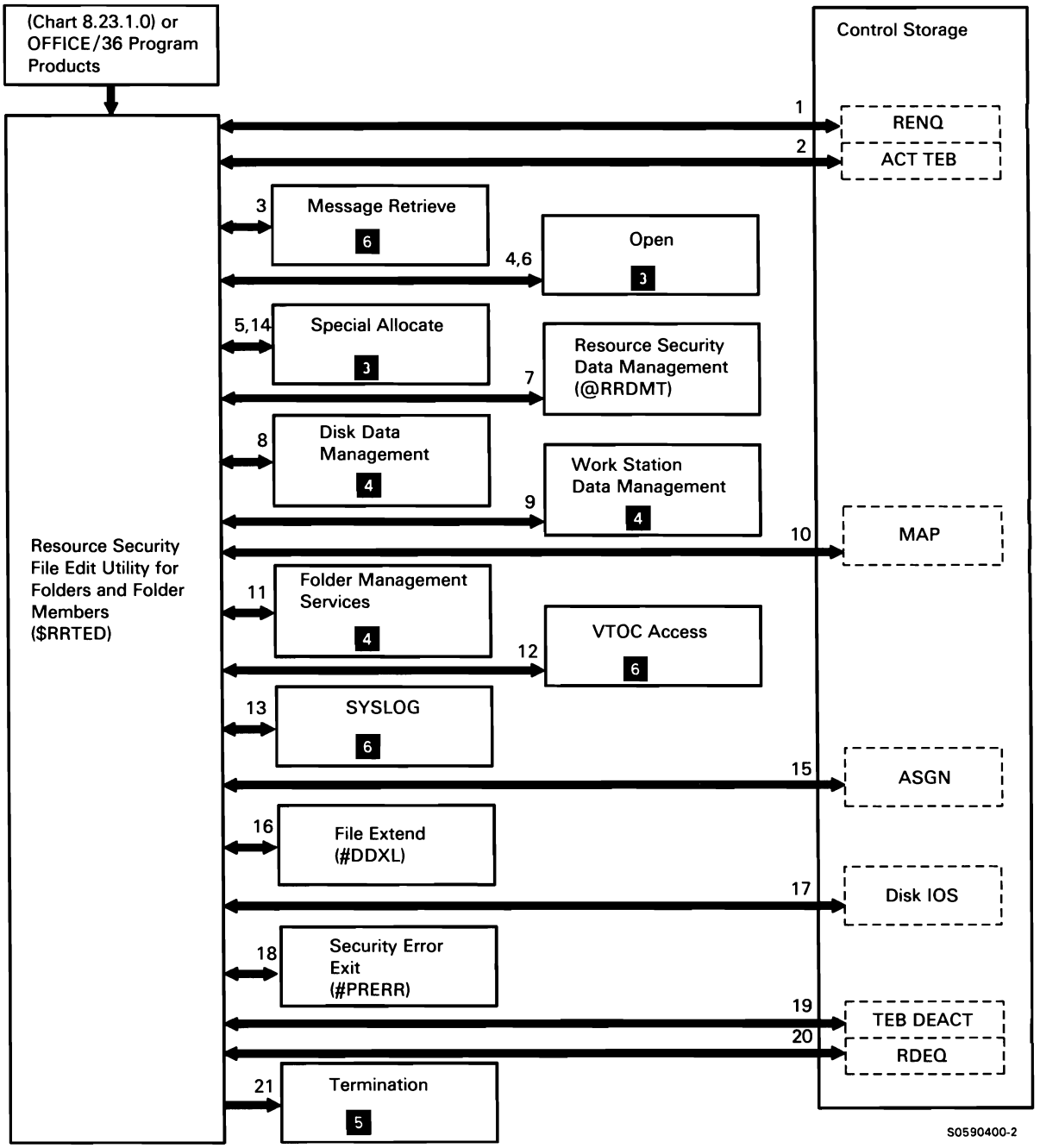


Chart 8.23.1.14 Resource Security File Edit Utility for Folders and Folder Members Control Flow

Resource Security File List Utility for Authorization Lists and Folders

The resource security file list utility for authorization lists and folders lists resource security information about authorization lists and folders to the SYSLIST device.

The following resource security file list utility for folders processes are shown in Chart 8.23.1.15:

1 Read and syntax check the utility control statements.

2 Get the utility interlock to ensure another security utility is not running.

3 Allocate and open the disk work file.

4 Activate the termination exit block (TEB) in the TB.

5 Retrieve authorization lists and folders security information from the user identification file.

6 Search the VTOC for each folder record in the resource security file.

7 Put records in the disk work file.

8 Read configuration sector for security; zero out disk work file on error.

9 Close the disk work file.

10 Allocate the work file for output.

11 Get heading lines.

12 Get time and date for headings.

13 Get records from disk.

14 List security information for folders.

15 Deactivate TEB in TB so termination will not take error exit.

16 Release the utility interlock so another security utility can run.

17 Issue any required messages.

18 If error occurs while TEB is active, process error.

19 Terminate this job step.

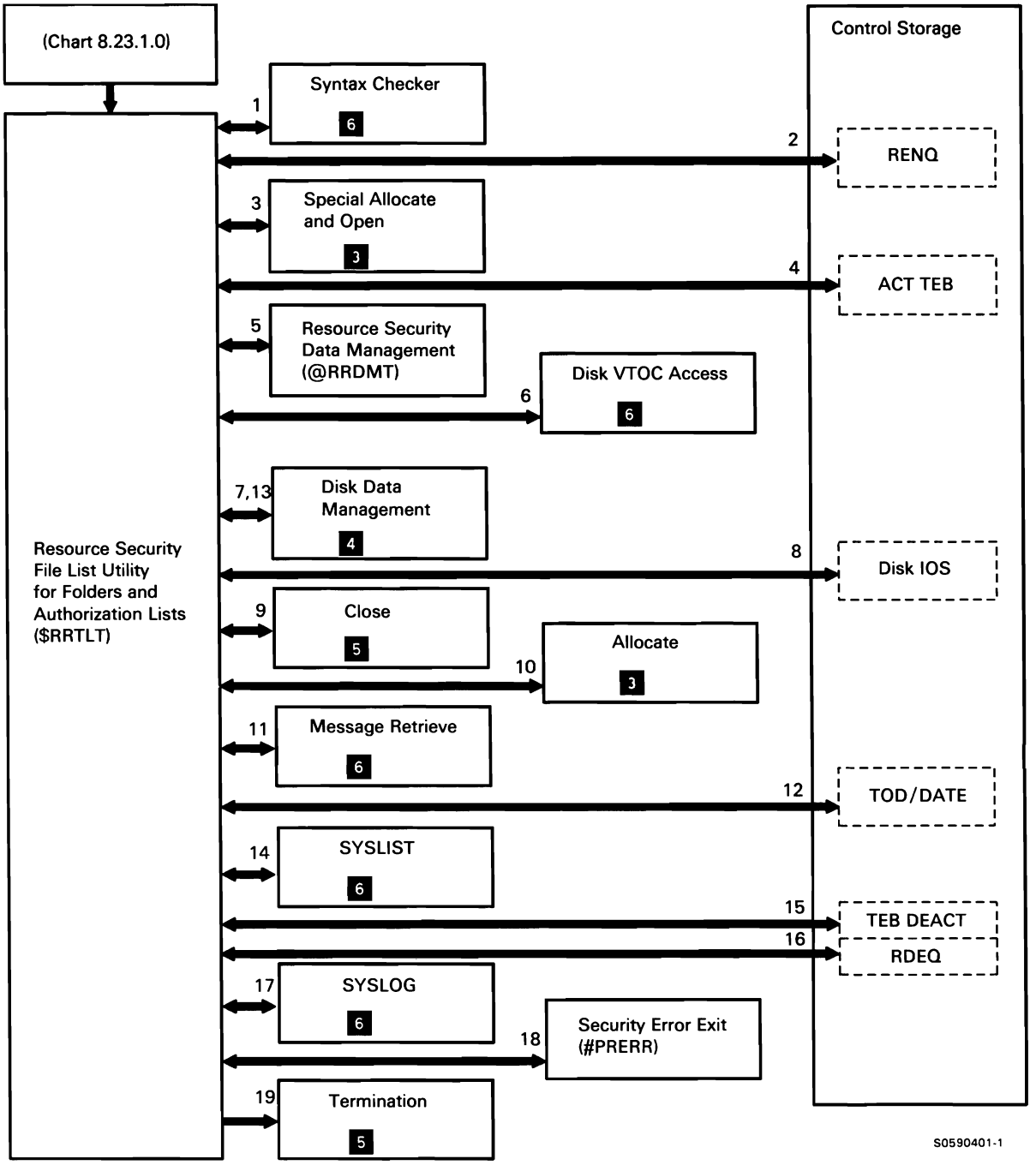


Chart 8.23.1.15 Resource Security File List Utility for Authorization Lists and Folders Control Flow

Security File Conversion Utility

The security file conversion utility converts either or both of the System/34 password security file and resource security file to the System/36 user identification file and resource security file.

The following security file conversion utility processes are shown in Chart 8.23.1.16:

1 Read and syntax check the utility control statements.

2 Allocate and open the System/34 security diskette file.

3 Allocate the security files on disk.

4 Read in a diskette buffer.

5 Open the converted user identification file, put scrambled records, and close the file.

6 Open the converted resource security file, put scrambled records, and close the file.

7 Write the index sector for the converted user ID file and converted resource security file.

8 Close the diskette file.

9 If error, delete the converted security file(s).

10 Issue any required messages.

11 Terminate this job step.

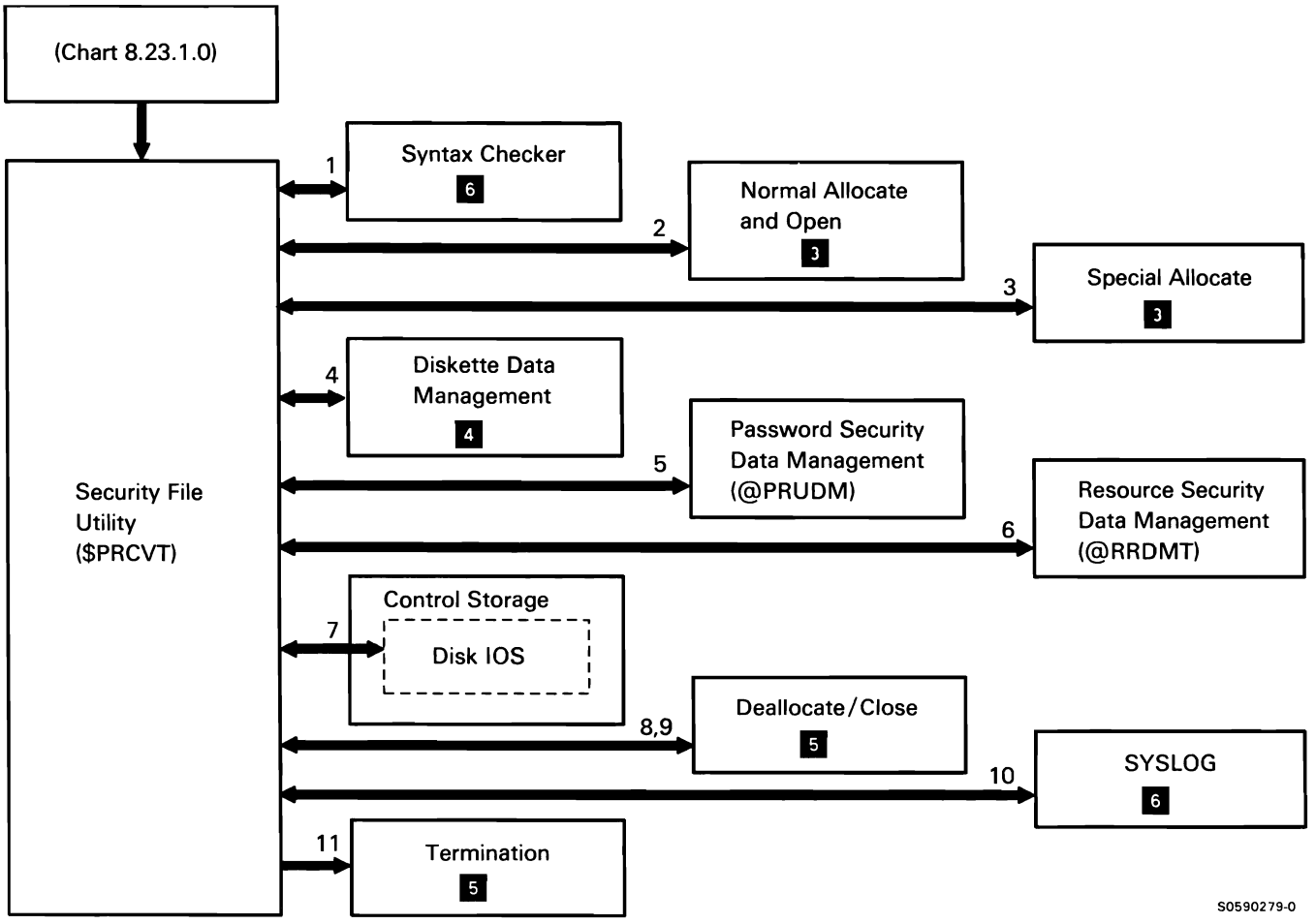


Chart 8.23.1.16 Security File Conversion Utility Control Flow

Security Support Modules

The security support modules perform various security services for other SSP functions. They are *not* evoked as utilities and are included under this topic for reader convenience only.

Security IPL Modules

The following security IPL processes are shown in Chart 8.23.2.1:

1 Do the following, routing control where necessary:

- Check resource security status.
- Set up resource security information in SCA.
- In work area, record any errors detected.

2 Write resource security configuration sector information.

3 Search VTOC for secured files.

4 Search resource security file for secured files.

5 Write password security configuration sector information.

6 Find user identification file.

7 Issue any required messages.

8 Determine if user is authorized to reload a secured system and close the file.

9 Scan user identification file for user profile.

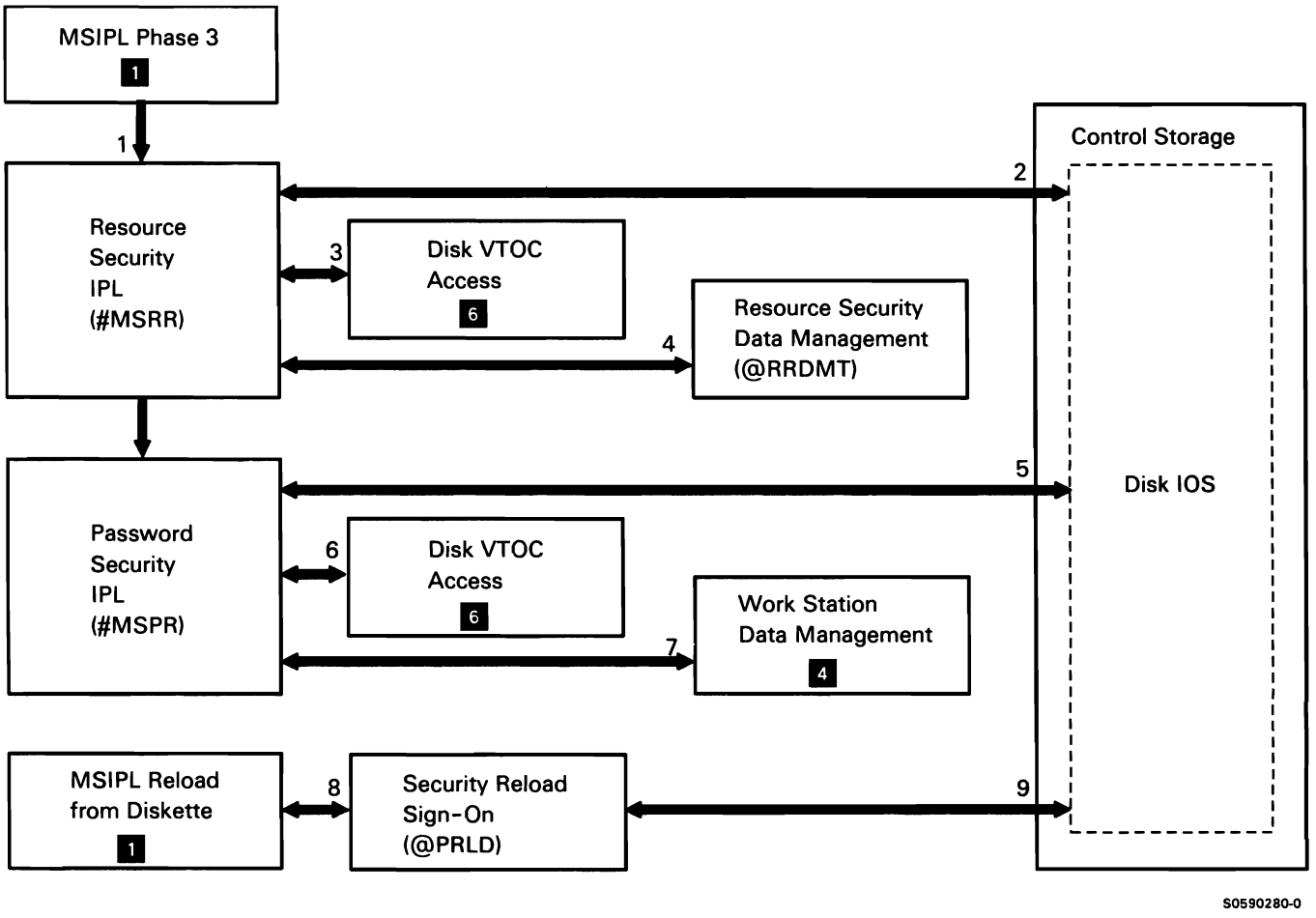


Chart 8.23.2.1 Security IPL Modules Control Flow

Change User Security Profile

Change user security profile alters the user default menu, default library, and/or the beginning help menu fields in the user's profile. It is evoked by command processor help, when the user presses command key 23 or 24.

The following change user security profile processes are shown in Chart 8.23.2.2:

1 Get security files interlock to ensure no other security module shares access to the file.

2 Read/write/scan user identification file records.

3 Release security files interlock.

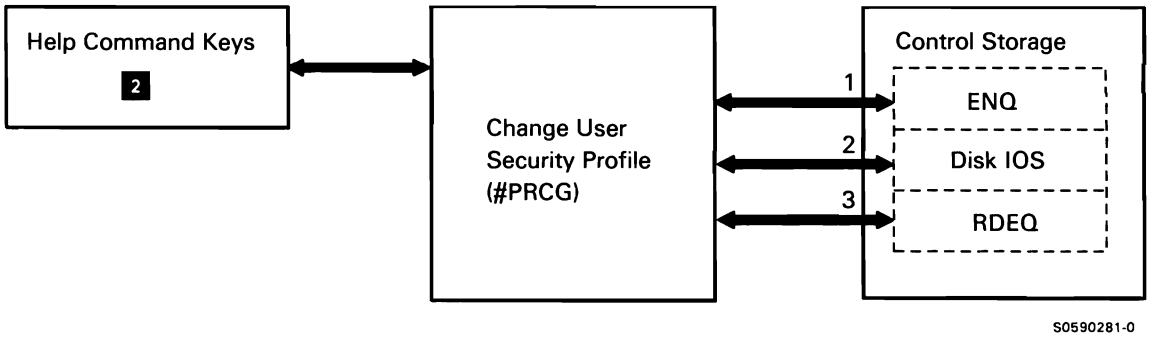


Chart 8.23.2.2 Change User Security Profile Control Flow

Password Sign-On

If password security is active, password sign-on determines if the user is authorized to sign on the system. Whether security is active or not, the sign-on transient passes the user profile back to the caller at sign-on time.

The following password sign-on processes are shown in Chart 8.23.2.3:

1 Get security files interlock to ensure no other security module shares access to the file.

2 Read/write/scan user identification file records.

3 Release security files interlock.

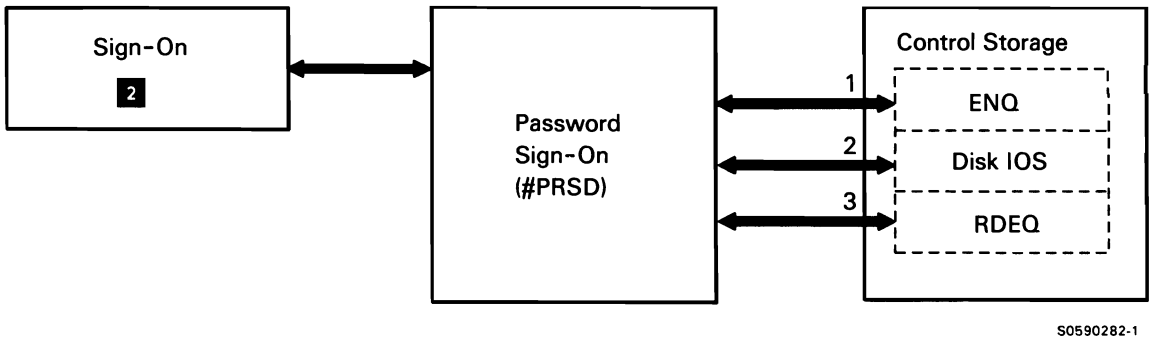


Chart 8.23.2.3 Password Sign-On Control Flow

Security Bind Processor

The security bind processor is called to verify the remote location's password when the APPC subsystem starts a session (if password security is active).

The following security bind processor processes are shown in Chart 8.23.2.4:

1 Get the user ID file interlock to ensure no other security programs can write or move the file.

2 Read/scan the user ID file for the appropriate location profile.

3 Encrypt the random seed using the location password as a key.

4 Issue any required information messages to the system console and log them to the history file.

5 Release the user ID file interlock.

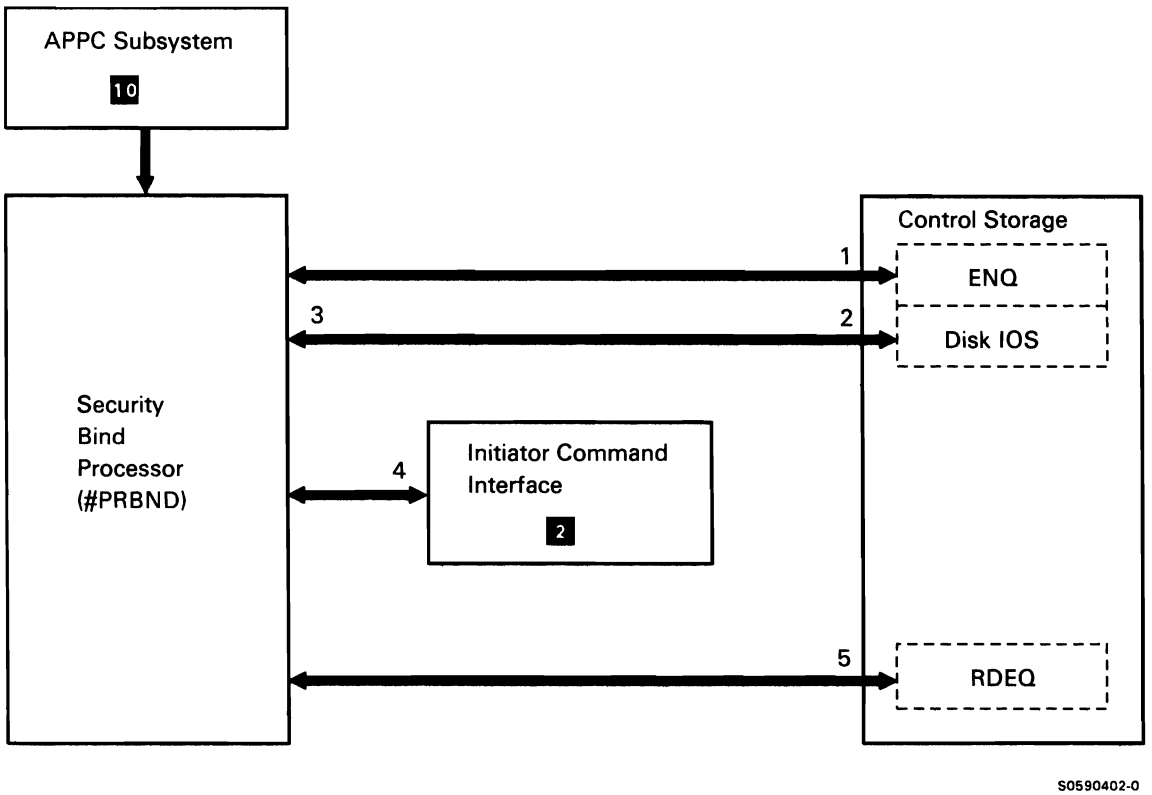


Chart 8.23.2.4 Security Bind Processor Control Flow

Resource Security Check One

Resource security check one verifies that the user is authorized to use a file, library, or folder. It is called when a user attempts to allocate a file or folder, or to find a library.

The following resource security check one processes are shown in Chart 8.23.2.5:

1 Get resource security file interlock to ensure no other security program moves the file.

2 Map to requester's DTF.

3 Free main storage security element.

4 Read/scan resource security file records.

5 Get the history file message.

6 Put message to history file.

7 Release security files interlock.

8 Issue any required messages.

9 Terminate this job step.

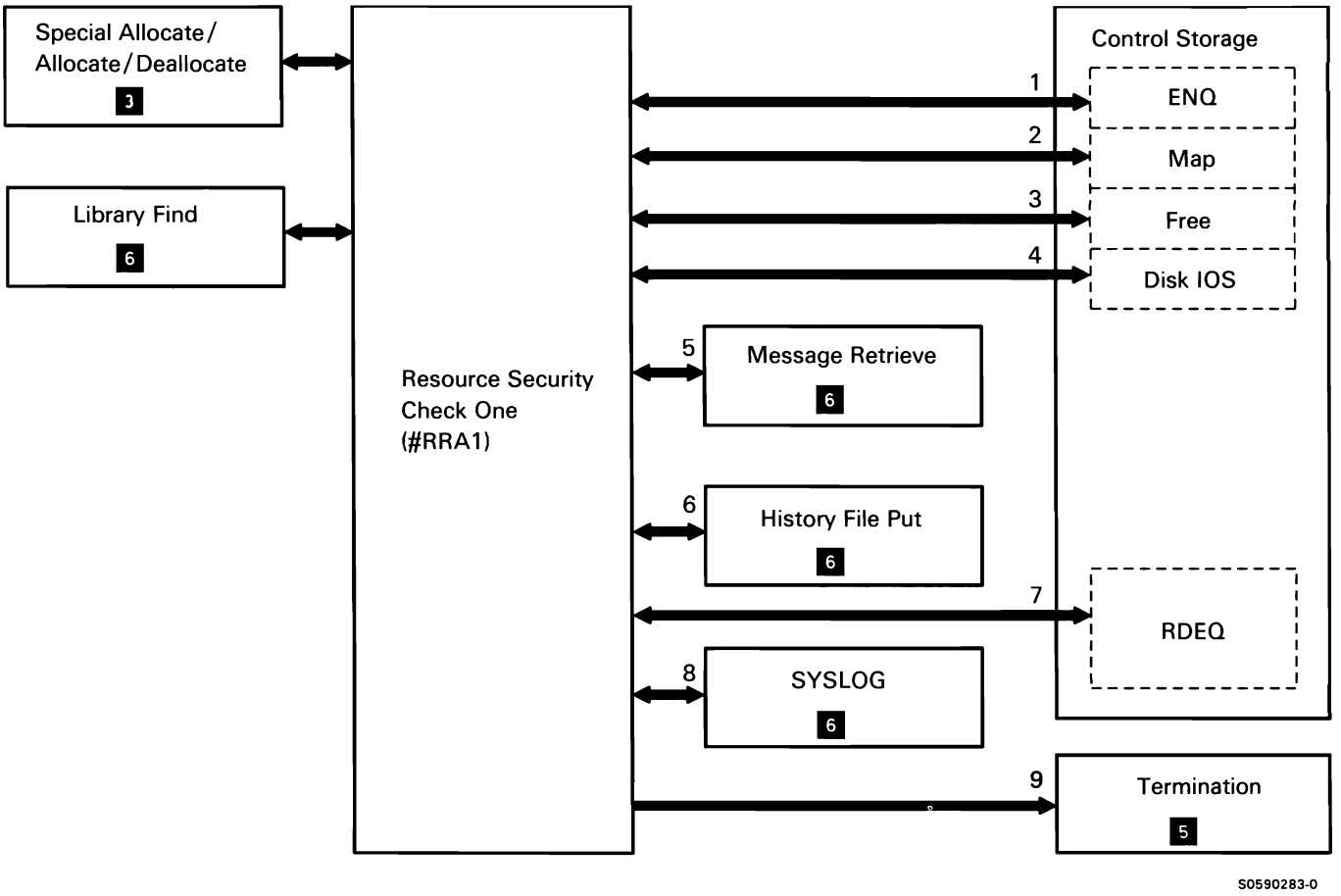


Chart 8.23.2.5 Resource Security Check One Control Flow

Resource Security Check Two

Resource security check two verifies that the user is authorized to attach a task that may have secured resources allocated.

The following resource security check two processes are shown in Chart 8.23.2.6:

1 Get resource security file interlock to ensure no other security program moves the file.

2 Read/scan user resource security file records.

3 Get the history file message.

4 Put message to history file.

5 Release security files interlock.

6 Issue any required messages.

7 Terminate this job step.

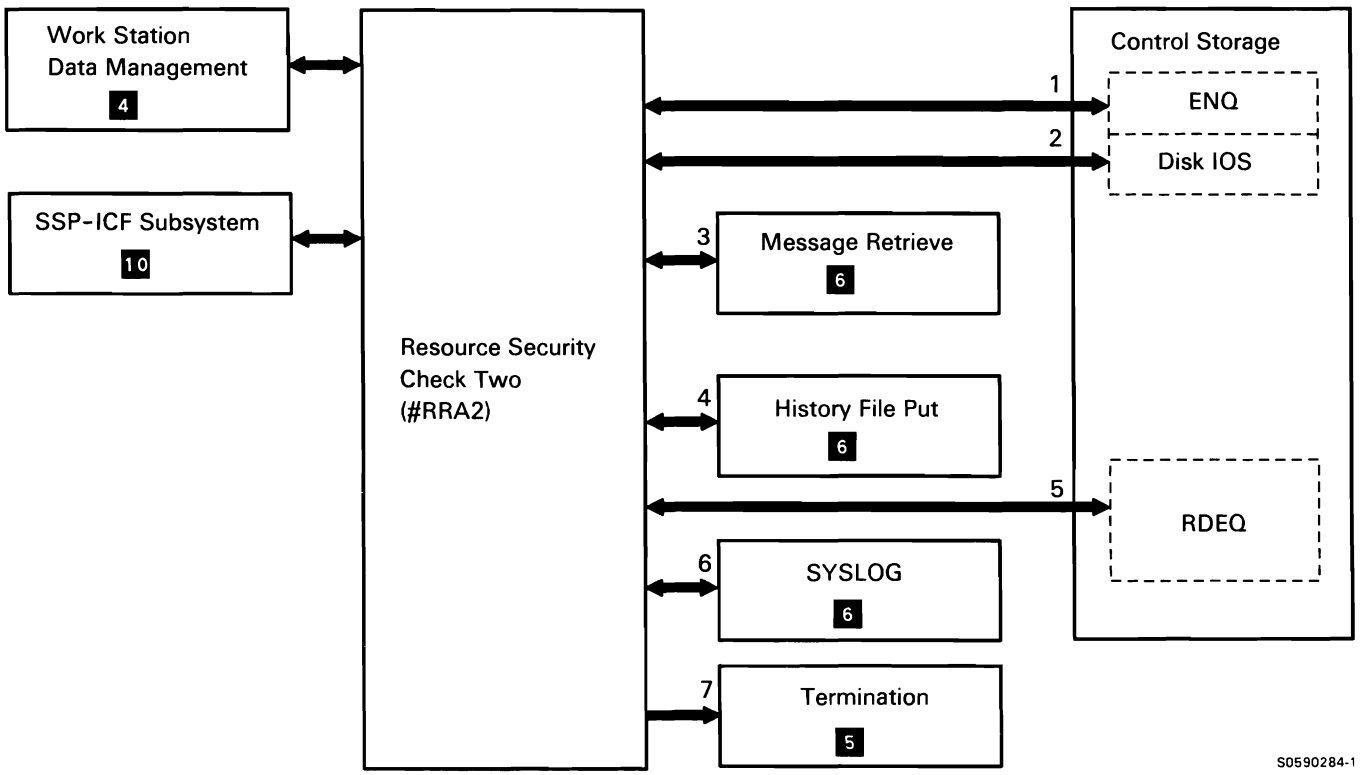


Chart 8.23.2.6 Resource Security Check Two Control Flow

Resource Security Check Three

Resource security check three verifies that the user is authorized to rename a secured file, library, or folder.

The following resource security check three processes are shown in Chart 8.23.2.7:

1 Get security file interlock to ensure no other security module moves the file.

2 Get assign/free area for IOB.

3 Read or scan resource security file records.

4 Read security information for the file, library, or folder.

5 Release security file interlock.

6 Issue any required messages and log messages to the history file.

7 Terminate this job step.

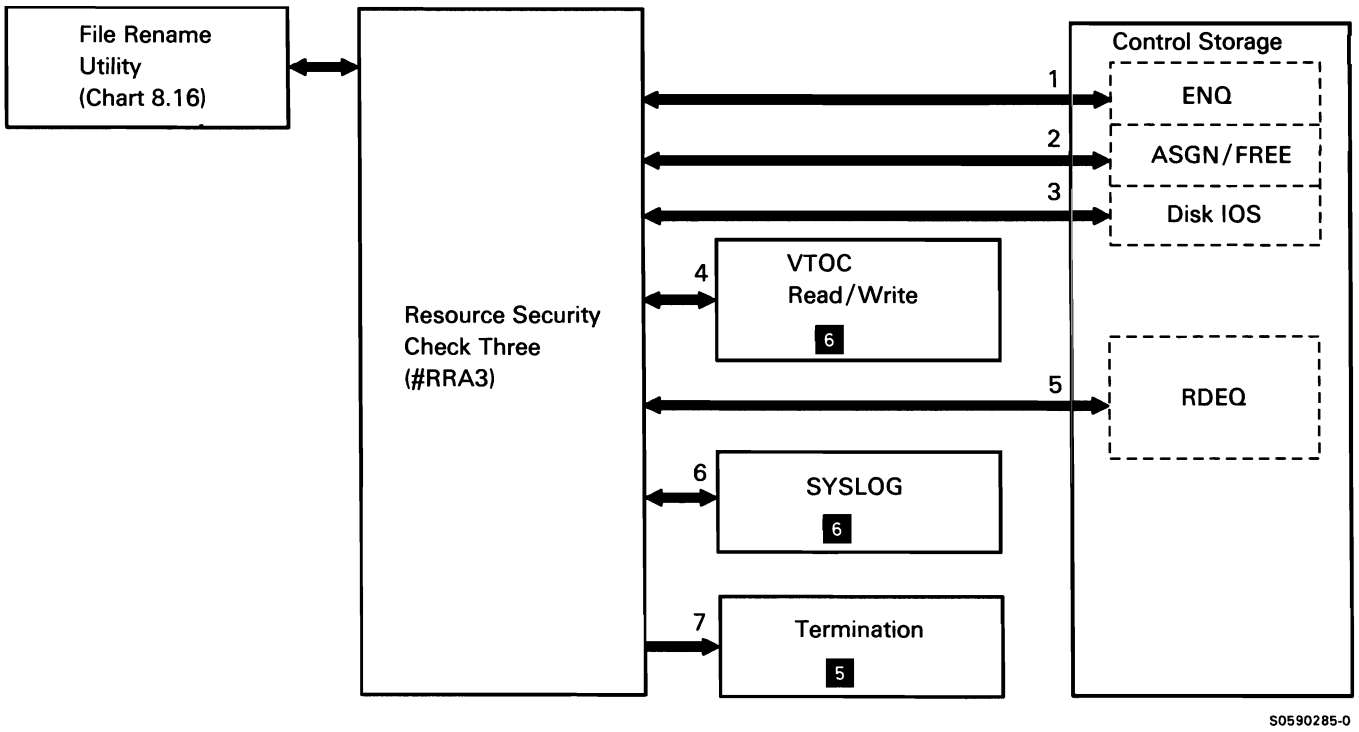


Chart 8.23.2.7 Resource Security Check Three Control Flow

Resource Security Check Four

Resource security check four verifies that the user is authorized to access a folder member. It also returns the user's access level to a file, library, or folder.

The following resource security check four processes are shown in Chart 8.23.2.8:

1 Get resource security file interlock to ensure no other security module moves the file.

2 Read/scan resource security file records.

3 Free the main storage security element.

4 Read security information for the file, library, or folder.

5 Release security file interlock.

6 Issue any required messages and log messages to the history file.

7 Terminate this job step.

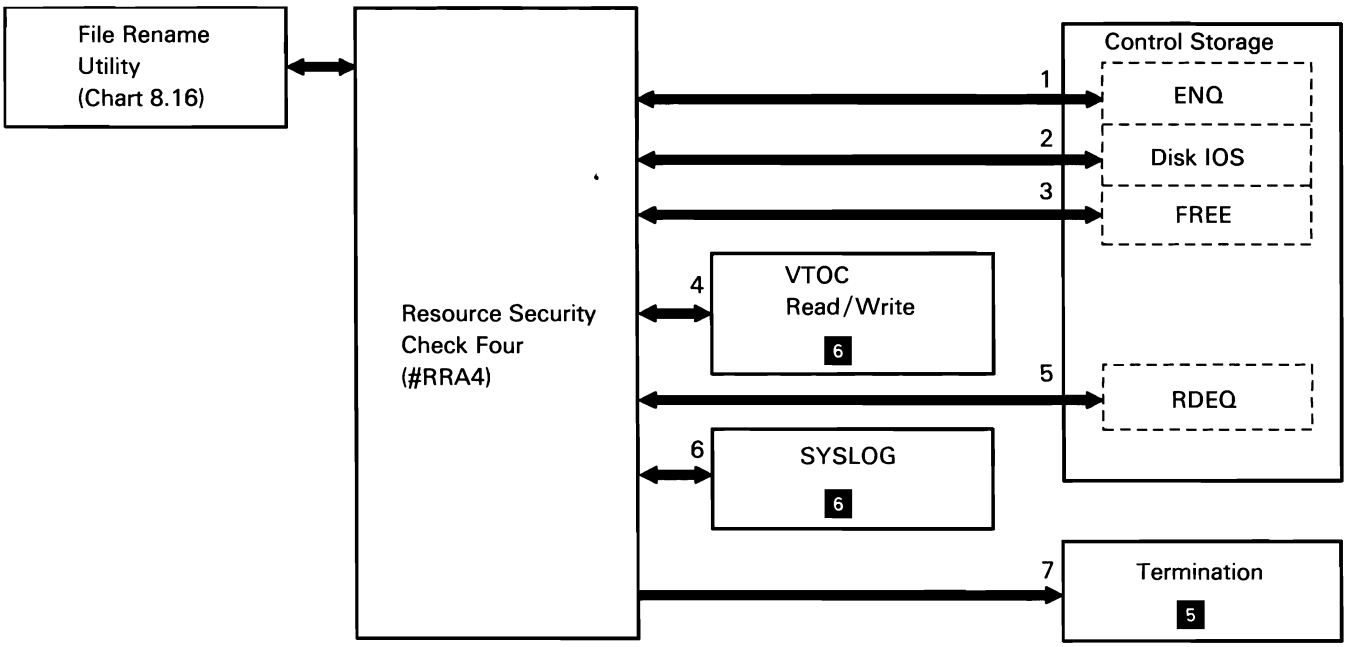


Chart 8.23.2.8 Resource Security Check Four Control Flow

This page is intentionally left blank.

FILE/LIBRARY/FOLDER RENAME UTILITY ($RENAM)

The file/library/folder rename utility changes disk file, user library, or folder names. The file, library, or folder name to be changed is specified by file/library/folder label. A file to be changed can also be specified by date; if no date is specified, the most recent file of the specified name is renamed. If the old file name is defined in the network resource directory (NRD), it is renamed on the remote system. The rename utility is invoked by the RENAME procedure or by user-supplied OCL and utility control statements.

The following file/library/folder rename utility processes are shown in Chart 8.24:

1 Check syntax of utility control information.

2 Lock disk format 1's.

3 Ensure that old file/library/folder exists and that new file/library/folder does not exist.

4 If the old file exists in the NRD, call DDM source rename file function.

5 Special allocate the file/library/folder.

6 If resource security is active, check user authorization to file/library/folder.

7 Write new file/library/folder name (with latest data indicator) to disk VTOC.

8 If old file was the latest file with multiple files of the same label, write the latest date indicator to disk VTOC.

9 Unlock disk format 1's.

10 Deallocate the file/library/folder.

11 Issue error messages if necessary.

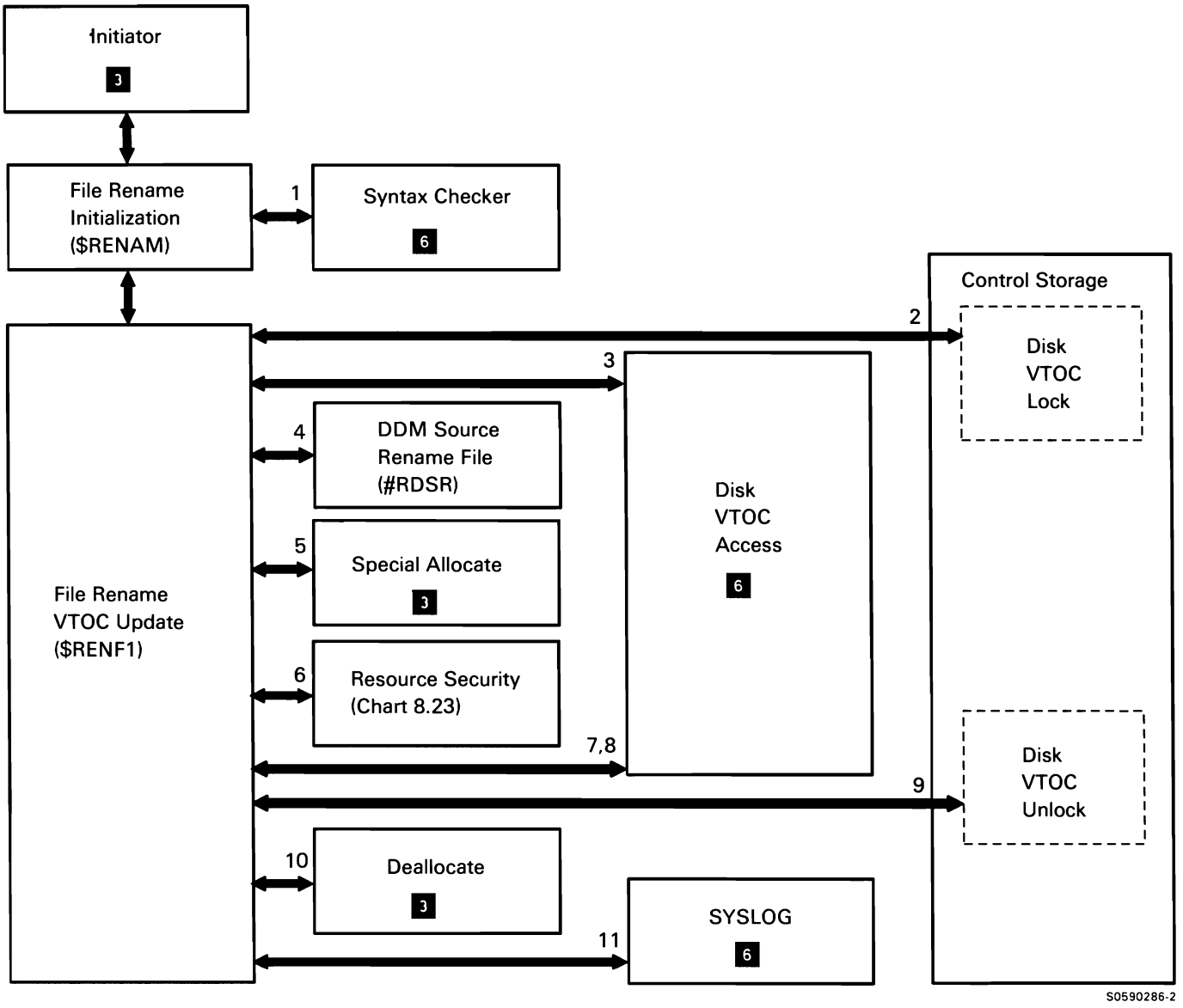


Chart 8.24 File/Library Rename Utility Control Flow

WORK STATION CONFIGURATION ($SETCF)

The work station configuration utility changes items in a display station's configuration record or in its communications configuration record. Each display station that is defined as a command display station during system configuration has an associated display station configuration record and communications configuration record. The communications and work station configuration records were originally built from the UDT and system configuration record. The environment items that can be changed by the work station configuration utility include:

- Display station
- BSC
- Local area network (LAN)
- SDLC
- Asynchronous
- User program BSC specifications

The $SETCF utility modifies only the configuration records associated with a work station. It can be evoked by the SET, PRINTKEY, and ALTERCOM procedure commands or by equivalent OCL and utility control statements.

The following $SETCF utility processes are shown in Chart 8.25:

1 Modify values in the associated terminal's work station configuration record or communications configuration record, routing control where necessary.

2 Read and syntax check utility control statements.

3 Issue any required messages.

4 When END statement is read, terminate this job step.

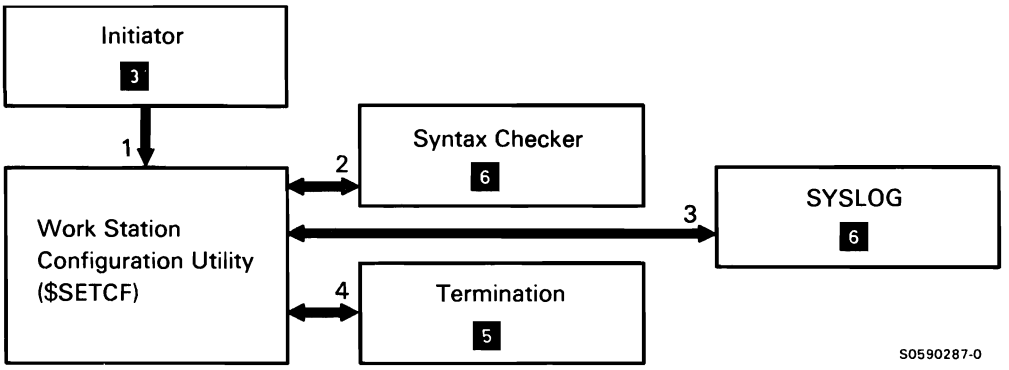


Chart 8.25 Work Station Configuration Utility Control Flow

COMMUNICATIONS CONFIGURATION UTILITY ($SETCP)

The communications configuration utility allows the user to alter certain communications configuration bits in the system configuration record on disk, after the system has been preset by IBM manufacturing or the customer engineer/customer service representative. The utility is evoked by the SETCOMM system procedure or equivalent OCL, and executes as a MRT program that can be executed only in a dedicated environment. The system IPL must be performed before the changes are effective.

The following communications configuration utility processes are shown in Chart 8.26:

1 Alter system communications configuration record according to user requests, routing control where necessary.

2 Syntax check utility control statements.

3 Read/write configuration record to disk.

4 Issue any required messages, including message stating that system IPL must be performed.

5 Terminate this job step.

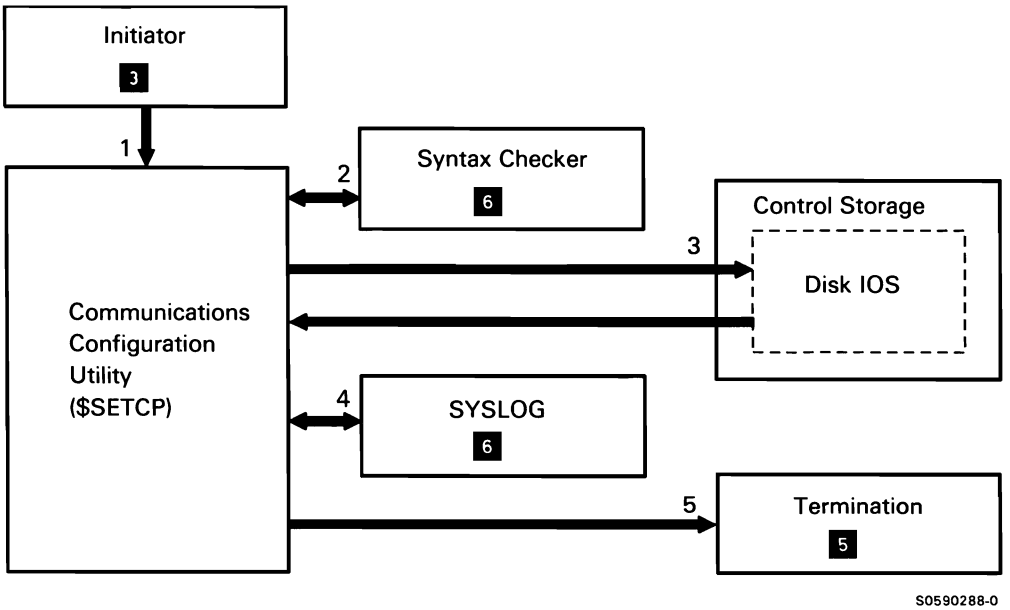


Chart 8.26 Communications Configuration Utility Control Flow

SCREEN FORMAT GENERATOR UTILITY ($SFGR)

The screen format generator utility processes user-generated or compiler-generated display format specifications and produces a display format(s) in a specified load member. It is evoked by the FORMAT procedure command, by program product procedures, or by equivalent OCL and utility control statements.

The following screen format generator utility processes are shown in Chart 8.27:

1 Read and syntax check utility control statements.

2 Process required messages to the operator.

3 Prepare work file to read source records.

4 Build data stream for formats processed.

5 If required, resequence out-of-order fields.

6 Perform load member maintenance:

- Calculate size of new or replaced load member.
- Write the format to the library format load member.
- Delete formats as required in this copy process.

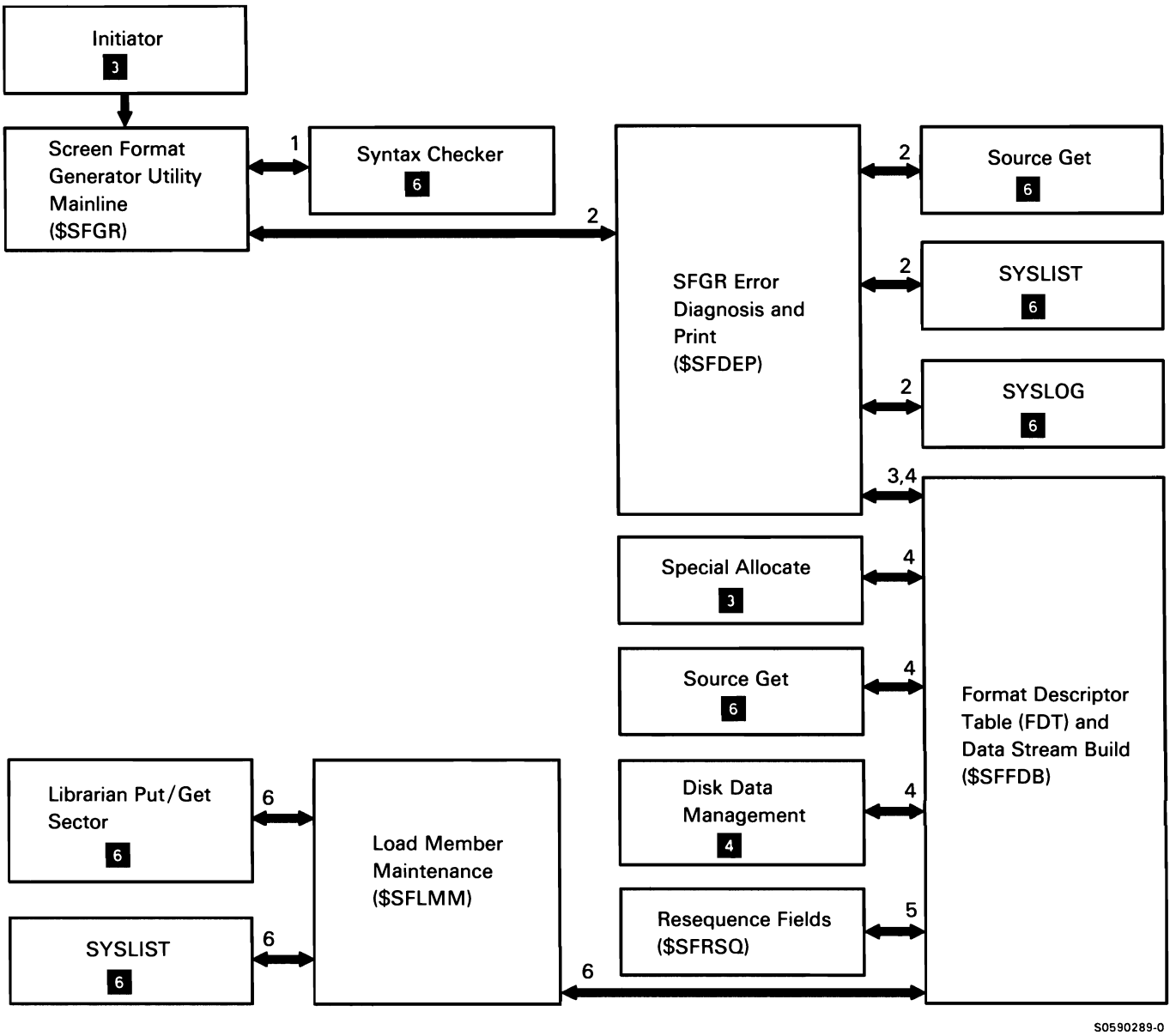


Chart 8.27 Screen Format Generator Utility Control Flow

DISPLAY SPOOL FILE ENTRIES UTILITY ($UASC)

The display spool file entries utility allows the user to inspect the spool file entries placed in a user file by the spool file utility ($UASF). In addition to viewing the output, the user can print a copy of the spool file information on the session printer. The user can view output in four different modes:

- Header mode
- Paging mode
- Record mode
- Full-line mode (132-character display only)

The utility is called by the COPYPRT system procedure or by equivalent OCL and utility control statements.

The following display spool file entries utility processes are shown in Chart 8.28:

1 Read format 1.

2 Allocate and open printer DTF and required files. If specified, allocate and open work station DTF.

3 Get records from disk.

4 If specified, print records.

5 If specified, display entries to display screen.

6 Close files.

7 Terminate this job step.

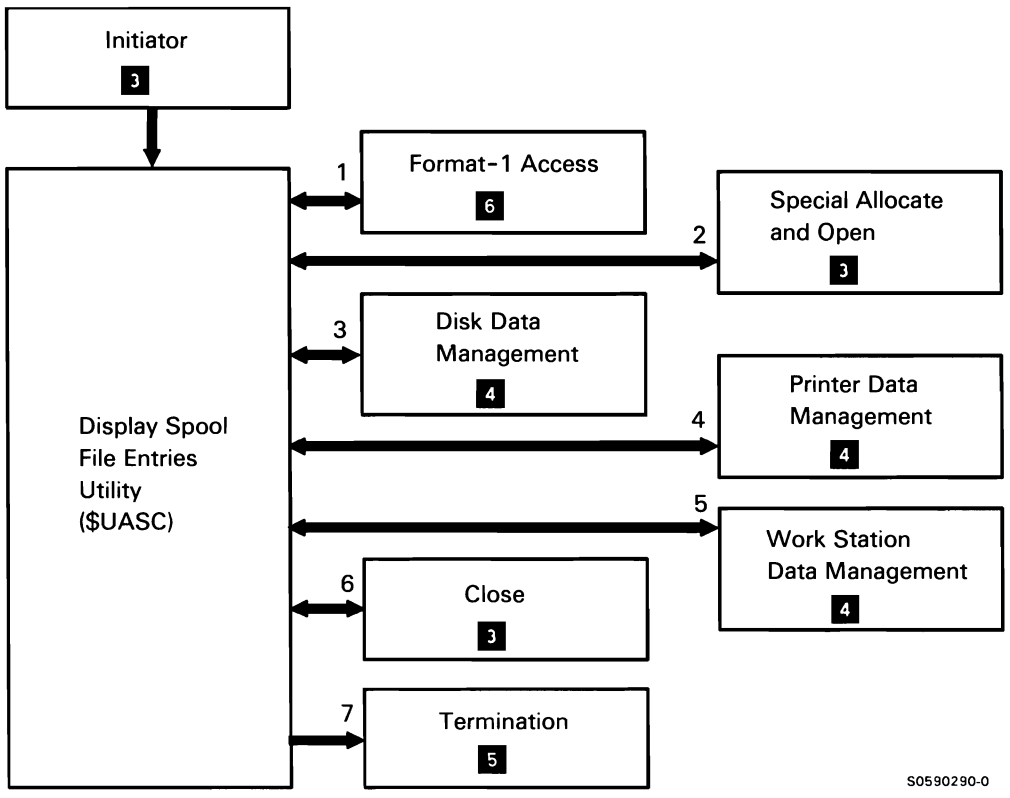


Chart 8.28 Display Spool File Entries Utility Control Flow

USER ACCESS TO SPOOL FILE UTILITY ($UASF)

The user access to spool file utility copies entries from the spool file to a user disk file in expanded format. The user invokes the utility through the COPYPRT procedure or appropriate OCL and utility control statements to do one of the following:

- Copy a single spool file entry
- Copy all of a user's entries with a specified forms number
- Copy all of a user's entries
- Copy the entire spool file (requester's security level must be appropriate)

The header and data records copied are 150 characters in length for 132-print-position printer; for print line lengths greater than 132 print positions, the data record length is 215 bytes.

Note: The COPYPRT procedures also invokes the display spool file entries utility, $UASC (Chart 8.28).

The following user access to spool file utility processes are shown in Chart 8.29:

1 Read and syntax check utility control statements.

2 Allocate file ($USAF calculates file size).

3 Write records to disk ($USAF expands from compressed format and removes SCS commands).

4 Close file.

5 Issue any required messages.

6 Terminate.

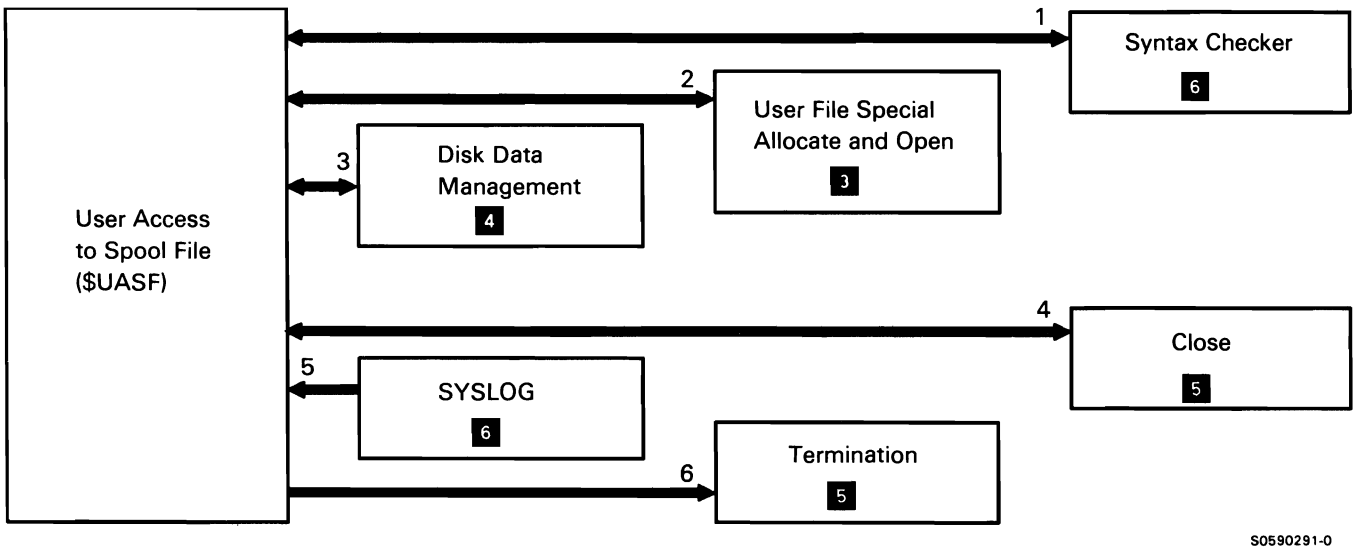


Chart 8.29 User Access to Spool File Utility Control Flow

TAPE INITIALIZATION UTILITY ($TINIT)

The tape initialization utility prepares magnetic tape for use. It is evoked via the TAPEINIT procedure or equivalent OCL and utility control statements. It performs the following functions:

- Check the first file on tape to ensure that it has not expired before writing a new volume label.
- Unconditionally initialize tape, ignoring file label and file expiration status.
- Erase the tape by writing blanks on the tape after label processing (data security erase).

The following tape initialization utility processes are shown in Chart 8.30:

1 Read and syntax check utility control statements.

2 Allocate tape drive.

3 Perform requested function.

4 Issue any required message.

5 Terminate this job step.

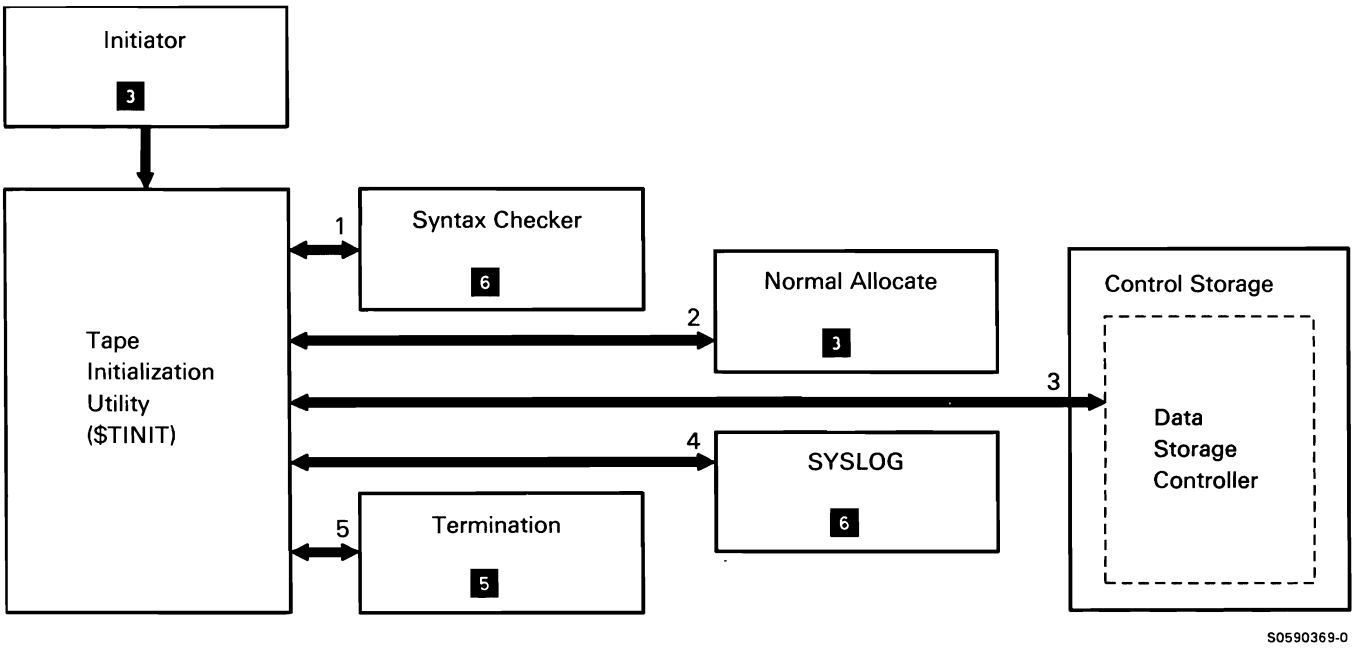


Chart 8.30 Tape Initialization Utility Control Flow

TAPE COPY UTILITY ($TCOPY)

The tape copy utility can be used to exchange data among tape, disk and the SYSLIST device.

The following tape copy processes are shown in Chart 8.31.0:

1 Read and syntax check utility control statements.

2 Process any required messages.

3 Get disk and/or tape AFAs based on contents of COPYIN and COPYO file statements.

4 If DISPLAY utility control statement, display tape file on SYSLIST device.

5 If TRANSFER utility control statement specifies COPYIN disk file, transfer or add a disk file to a tape file.

6 If TRANSFER utility control statement specifies COPYIN tape file, transfer or add a tape file to a disk file.

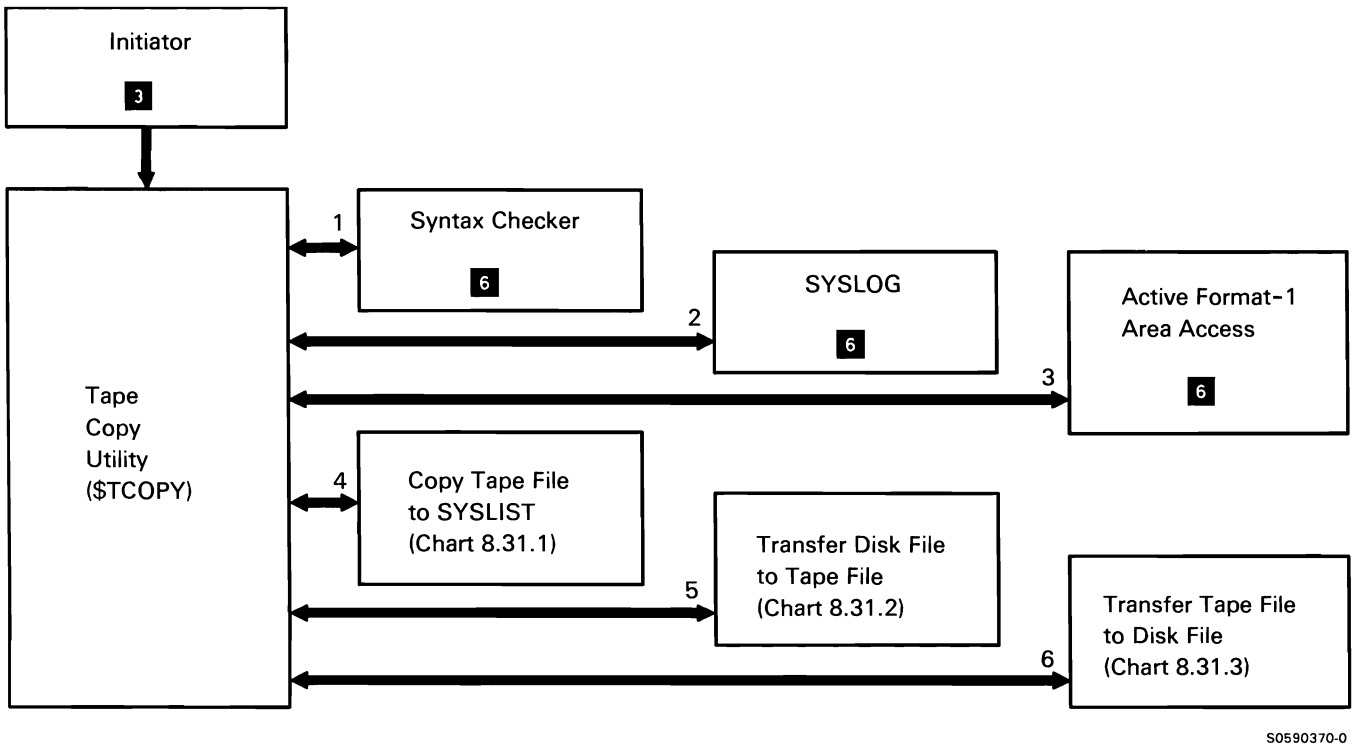


Chart 8.31.0 Tape Copy Utility Overview Control Flow

Tape File to SYSLIST Copy

Tape file to SYSLIST copy displays or prints the contents of a tape file at the SYSLIST device.

The following tape file to SYSLIST copy processes are shown in Chart 8.31.1:

1 Allocate tape COPYIN file and SYSLIST device.

2 Set up display or printer parameter list.

3 Open tape COPYIN file.

4 Read tape records.

5 If record's RRN is within limits, display or print record.

6 Close tape DTF.

7 Issue any required messages.

8 Terminate this job step.

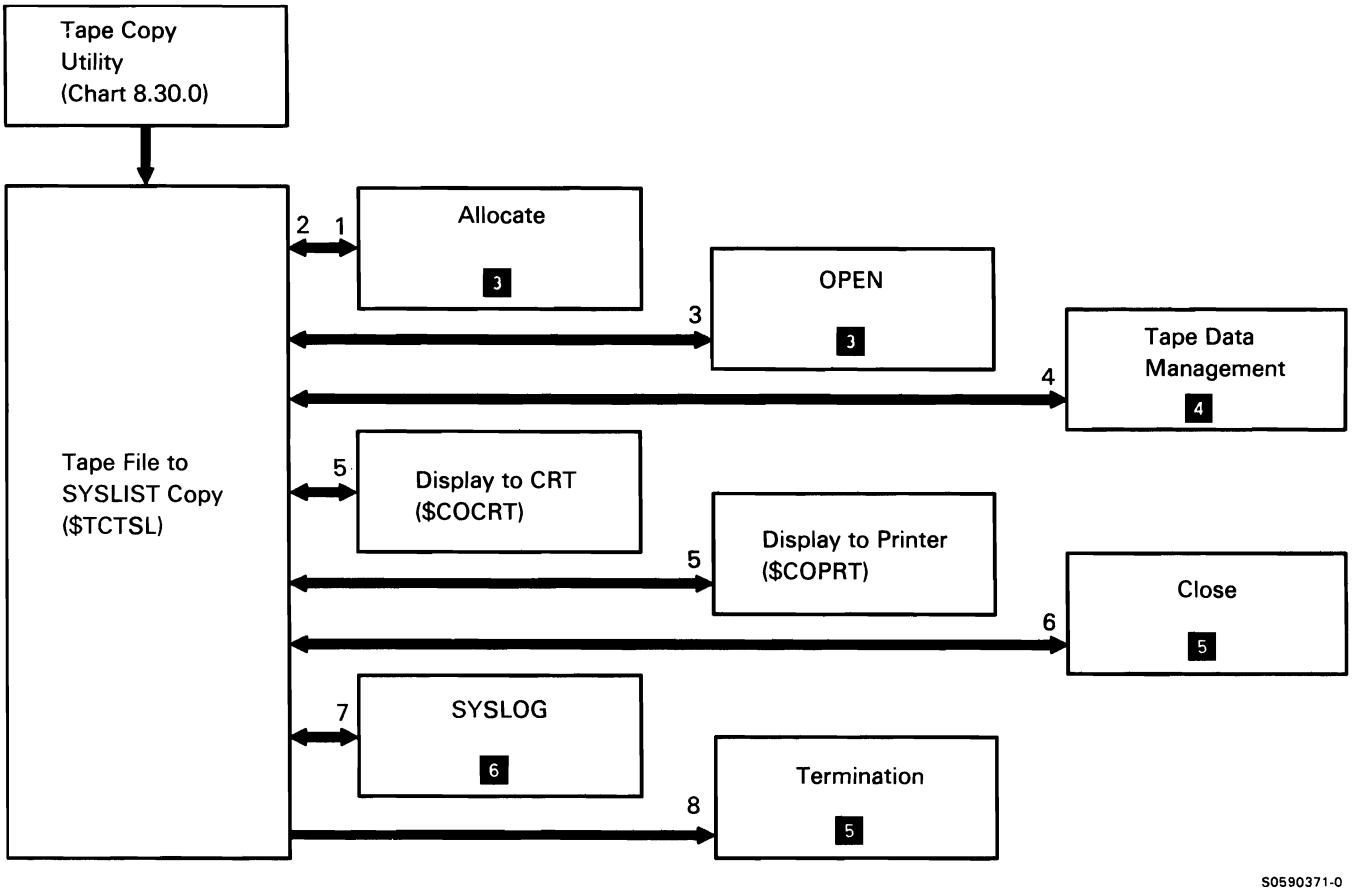


Chart 8.31.1 Tape File to SYSLIST Copy Control Flow

Disk File to Tape File Copy

Disk file to tape file copy transfers data from a sequential or indexed disk file to a tape file or adds data from a sequential disk file to a tape file.

The following disk file to tape file copy processes are shown in Chart 8.31.2:

1 Allocate COPYIN file, COPYOUT file, and tape device.

2 Diagnose request, set up required tape data management and tape buffers.

3 Open disk and tape DTFs.

4 Read disk records.

5 Write records to tape.

6 Close disk and tape DTFs.

7 Issue any required messages.

8 Terminate this job step.

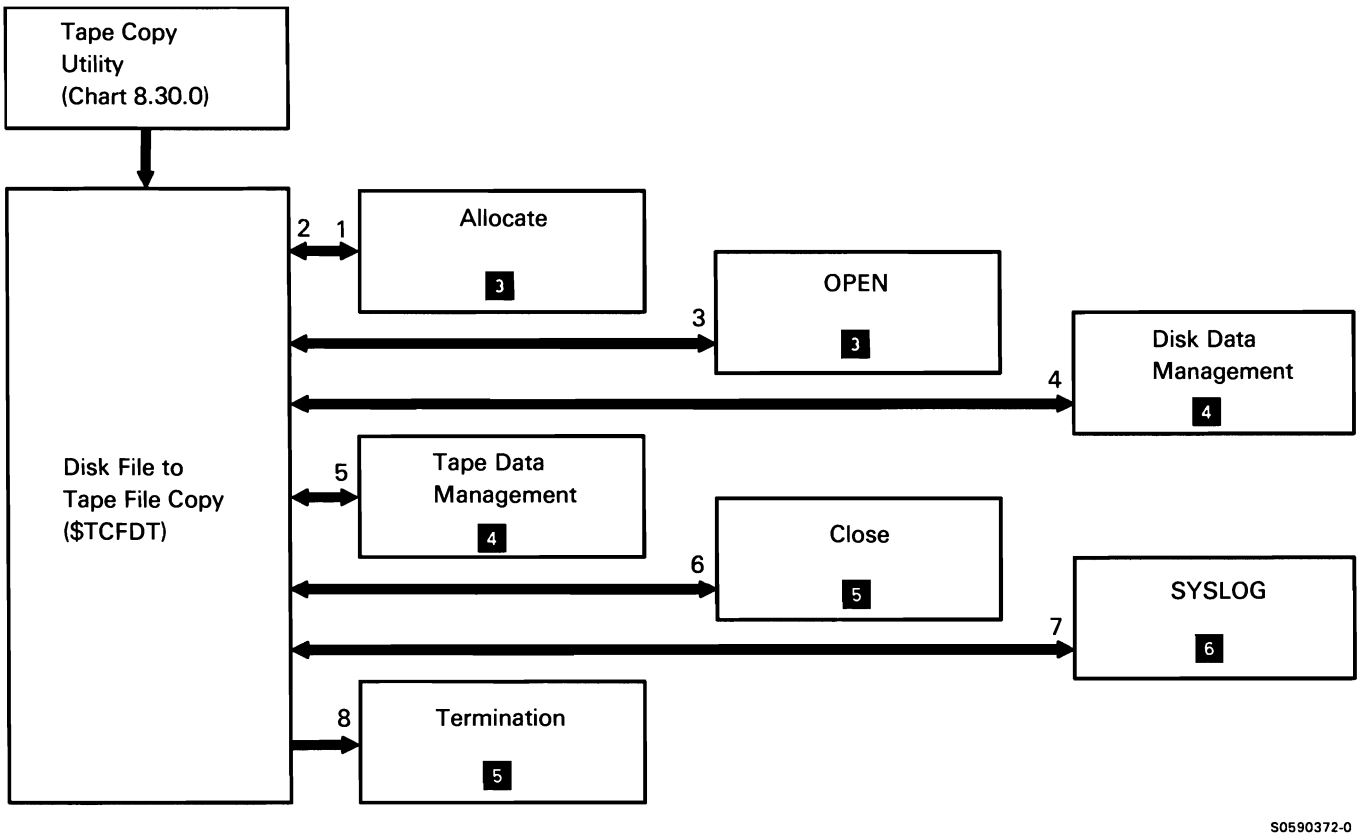


Chart 8.31.2 Disk File to Tape File Copy Control Flow

Tape File to Disk File Copy

Tape file to disk file copy transfers data from a tape file to a sequential or indexed disk file or adds data from a tape file to a sequential disk file.

The following tape file to disk file copy processes are shown in Chart 8.31.3:

1 Allocate COPYIN file, COPYOUT file, and tape device.

2 Diagnose request, set up required tape data management and tape buffers.

3 Open disk and tape DTFs.

4 Read tape records.

5 Write records to disk.

6 Close disk and tape DTFs.

7 Issue any required messages.

8 Terminate this job step.

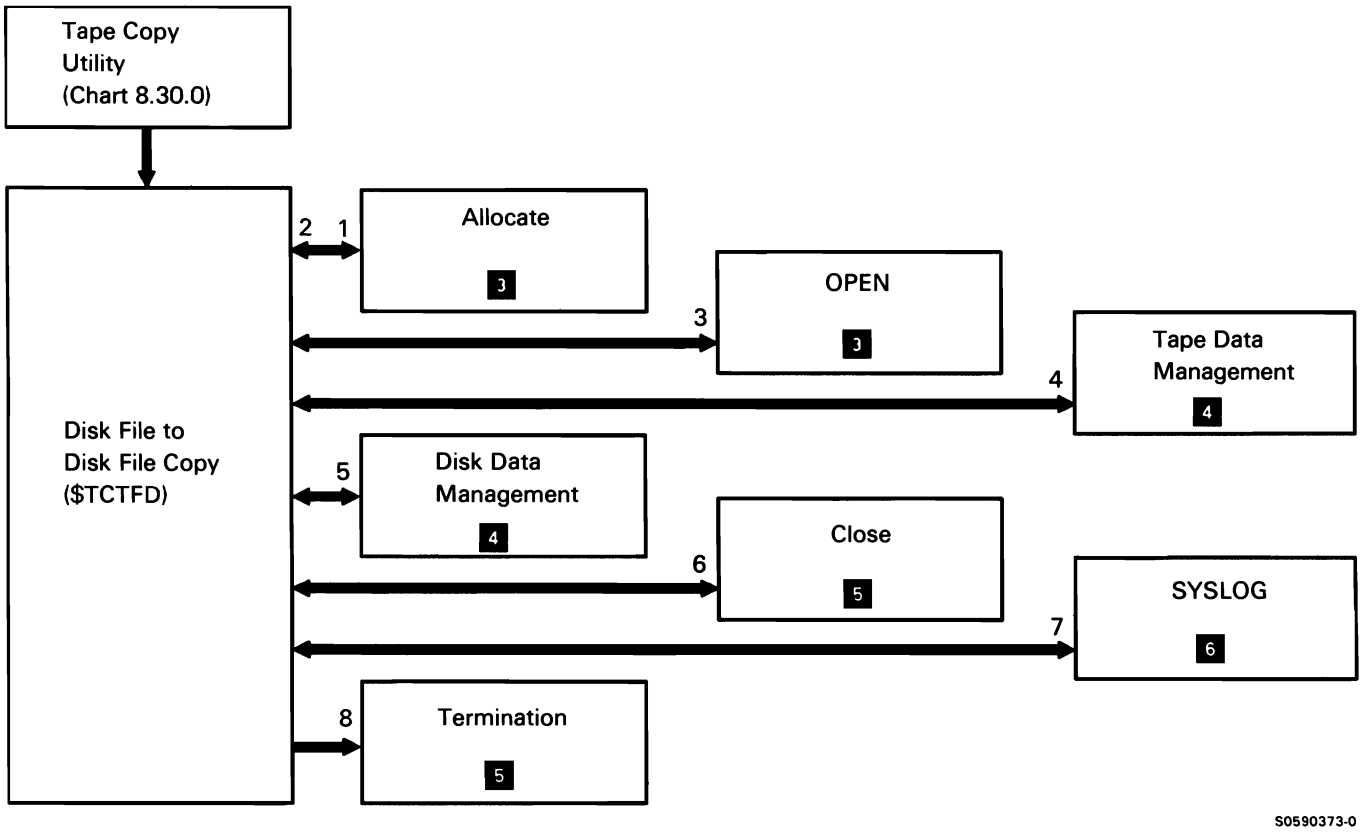


Chart 8.31.3 Tape File to Disk File Copy Control Flow

This page is intentionally left blank.

EXTENDED CHARACTER FILE RESTORE UTILITY ($XREST)

The extended character file restore utility allows the user to restore the extended character file from a diskette file. The user can selectively restore the IBM-supplied extended character file, user-defined characters, or a specified portion of the user-defined characters. The extended character file restore utility is evoked by the RESTEXTN system procedure or by equivalent user-supplied OCL and utility control statements.

The following extended character file restore utility processes are shown in Chart 8.32:

1 Find and check COPYIN and COPYO AFAs.

2 If diskette magazine drive, find correct diskette.

3 Allocate and open the diskette file to be restored to disk.

4 Allocate, open, and process the header disk file named #HDR1818 or #HDR2424.

5 Check syntax of utility control statements.

6 Determine if a new extended character file must be allocated.

7 Allocate the extended character disk file.

8 Open the extended character disk file.

9 Get records from diskette file.

Note: Diskette data management module #DRDI is used when restoring a file saved on System/36 via $XSAVE or an I-format file from any other system. Diskette data management module #DRSM is used when restoring a file saved on System/34.

10 Put records to the extended character file by relative record number.

11 Create the valid code point table if needed.

12 Update header disk file.

13 Close extended character file, header disk file, and diskette file.

14 Process any required messages.

15 Terminate this job step.

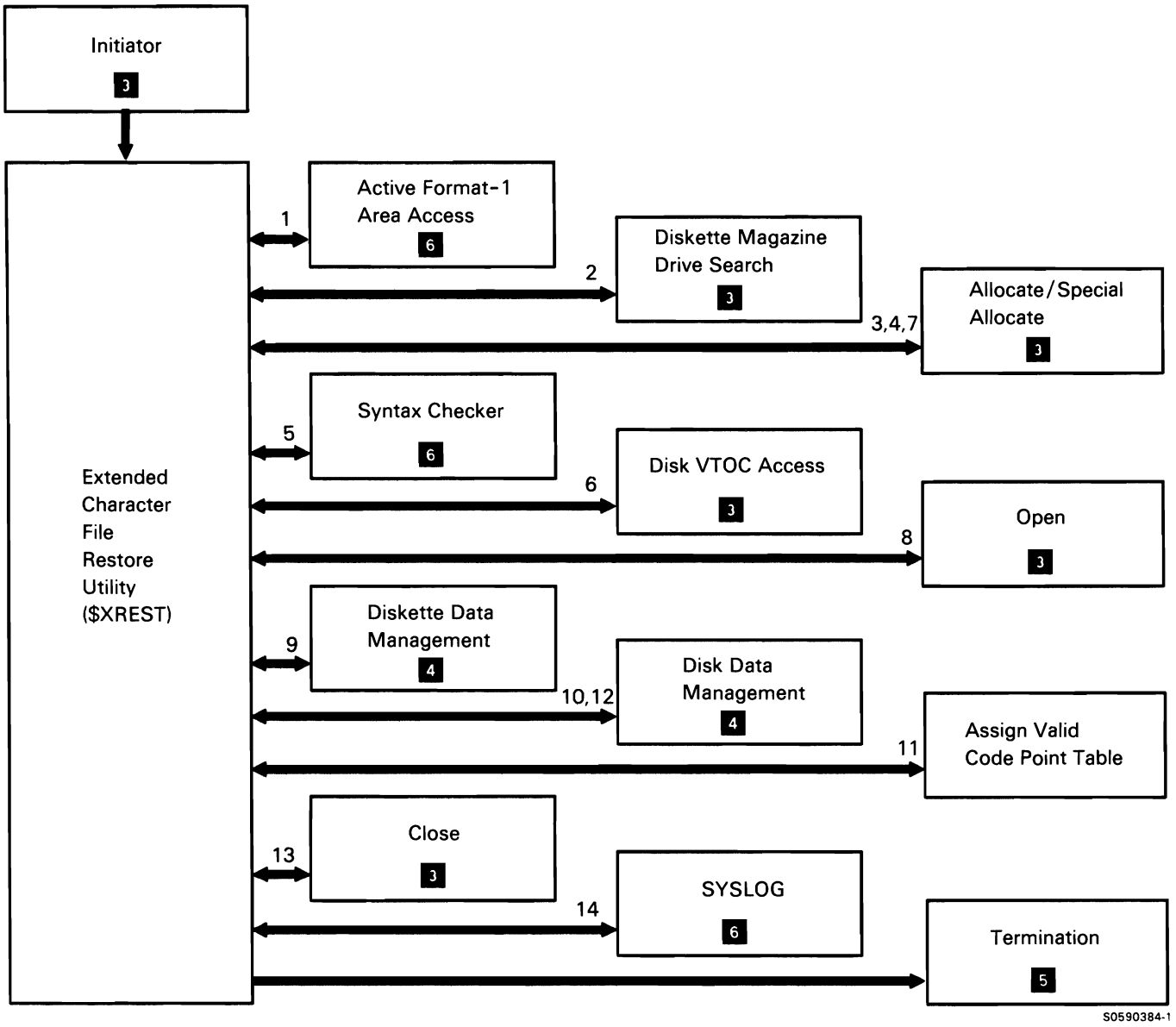


Chart 8.32 Extended Character File Restore Control Flow

EXTENDED CHARACTER FILE SAVE UTILITY ($XSAVE)

The extended character file save utility allows the user to save the extended character file in an I-format diskette file. The user can selectively save the IBM-supplied extended character file, user-defined characters, or a specified portion of the user-defined characters. The extended character file save utility is evoked by the SAVEEXTN system procedure or by equivalent user-supplied OCL and utility control statements.

The following extended character file save utility processes are shown in Chart 8.33:

1 Find and check COPYIN and COPYO AFAs.

2 If diskette magazine drive, find correct diskette.

3 Allocate the diskette file on which characters are to be saved.

4 Allocate, open, read, and process header disk file #HDR1818 or #HDR2424.

5 Close the header disk file.

6 Check syntax of utility control information.

7 Allocate the extended character disk file.

8 Open the extended character disk file.

9 Open the diskette file.

10 Write header records to the diskette file.

11 Get records, by relative record number, from the extended character file.

12 Put records to diskette file.

13 Close extended character disk file and diskette file.

14 Process any required messages.

15 Terminate this job step.

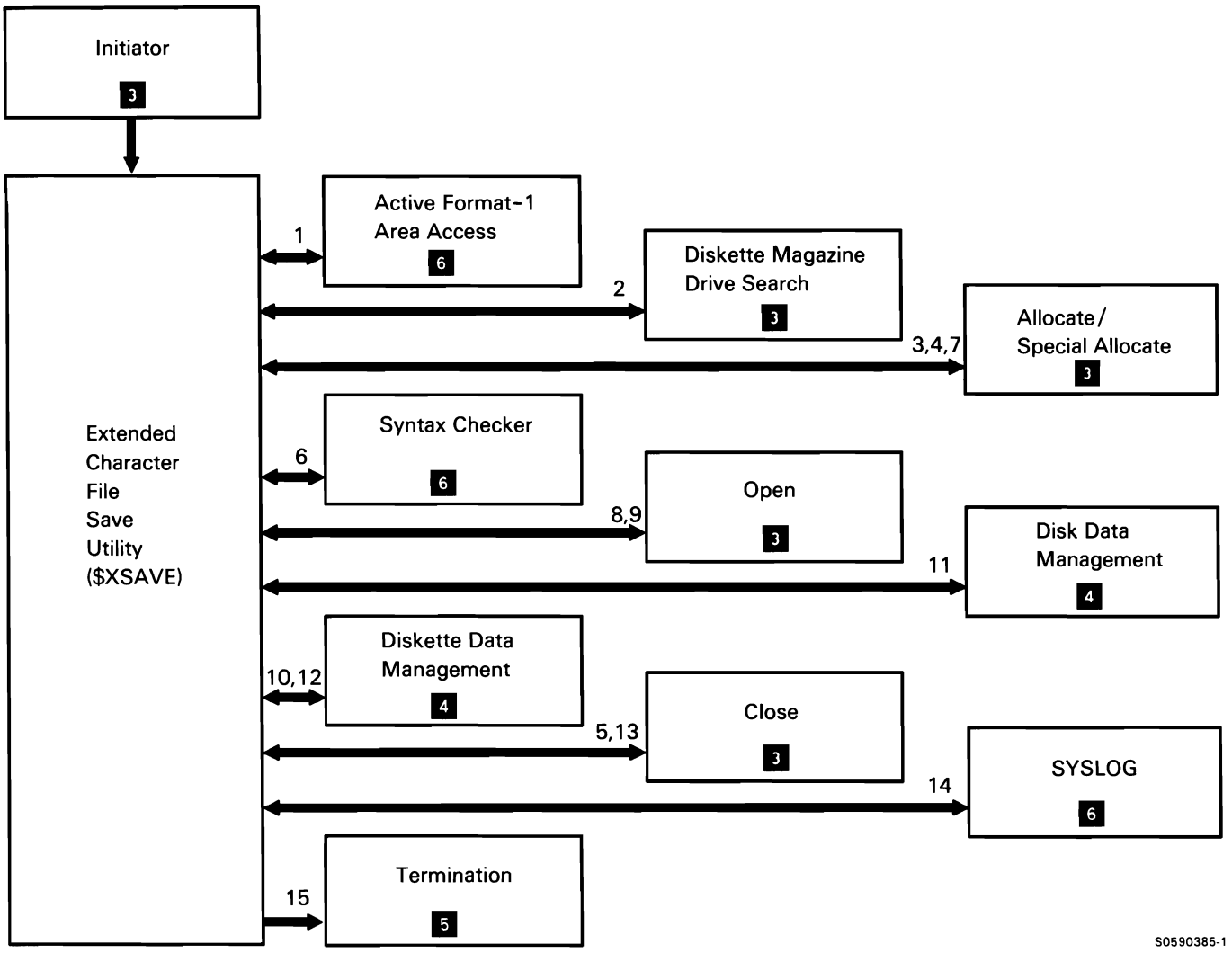


Chart 8.33 Extended Character File Save Control Flow

Network Resources Directory (NRD) Utilities

The network resources directory (NRD) utilities allow the user to edit, save, restore, or delete the NRD. The user invokes the NRD utilities via the appropriate procedures or equivalent OCL statements.

EDITNRD Utility

The EDITNRD utility allows the user to create the network resource directory (NRD) file and edit entries in the NRD file.

The following EDITNRD utility processes are shown in Chart 8.34:

1 Initialize for NRD utility and route control for requested function.

2 Perform initialization for interactive environment.

3 Determine if NRD file exists or not.

4 Allocate a new NRD file if it does not exist.

5 Perform requested edit functions on NRD file.

6 Issue any required messages and log messages to the history file.

7 Terminate this job step.

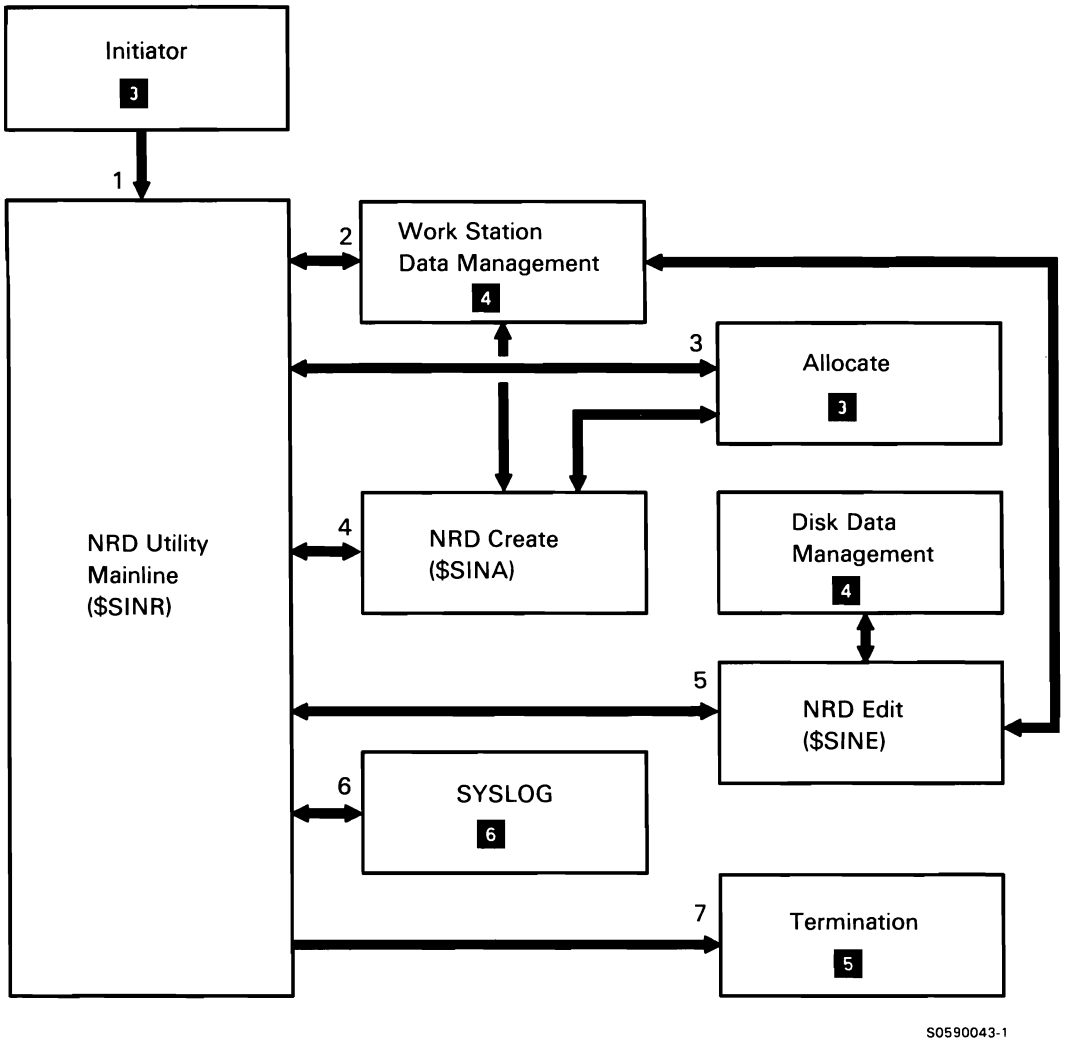


Chart 8.34 EDITNRD Utility Control Flow

The SAVENRD utility allows the user to save the network resource directory (NRD) file on diskette or tape.

The following SAVENRD utility processes are shown in Chart 8.34.1:

1 Copy NRD file to save medium.

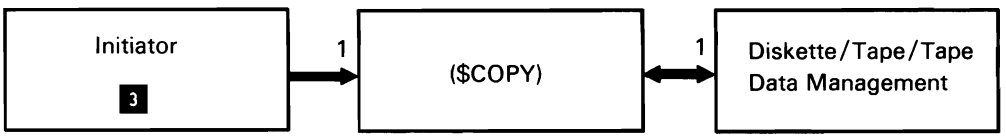


Chart 8.34.1 SAVENRD Utility Control Flow

RESTNRD Utility

The RESTNRD utility allows the user to restore a previously saved network resource directory (NRD) file to the system from diskette or tape.

The following RESTNRD utility processes are shown in Chart 8.34.2:

1 Copy NRD off save medium to temporary file.

2 Allocate NRD file.

3 Copy temporary file to NRD file.

4 Issue any required messages.

5 Terminate this job step.

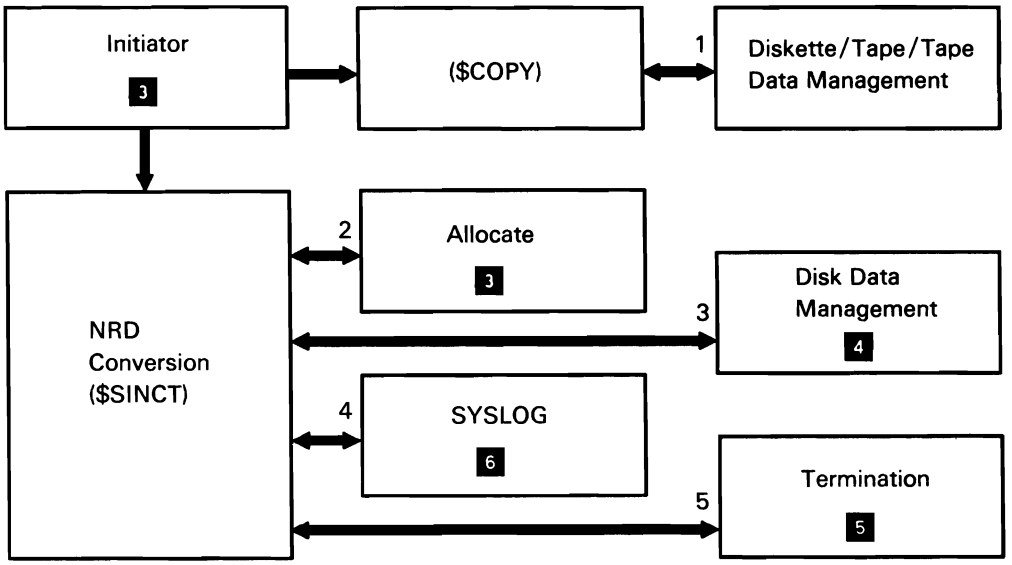


Chart 8.34.2 RESTNRD Utility Control Flow

The DELNRD utility allows the user to delete the network resource directory (NRD) file from the system.

The following DELNRD utility processes are shown in Chart 8.34.3:

1 Allocate file.

2 Delete file.

3 Issue any messages.

4 Terminate this job step.

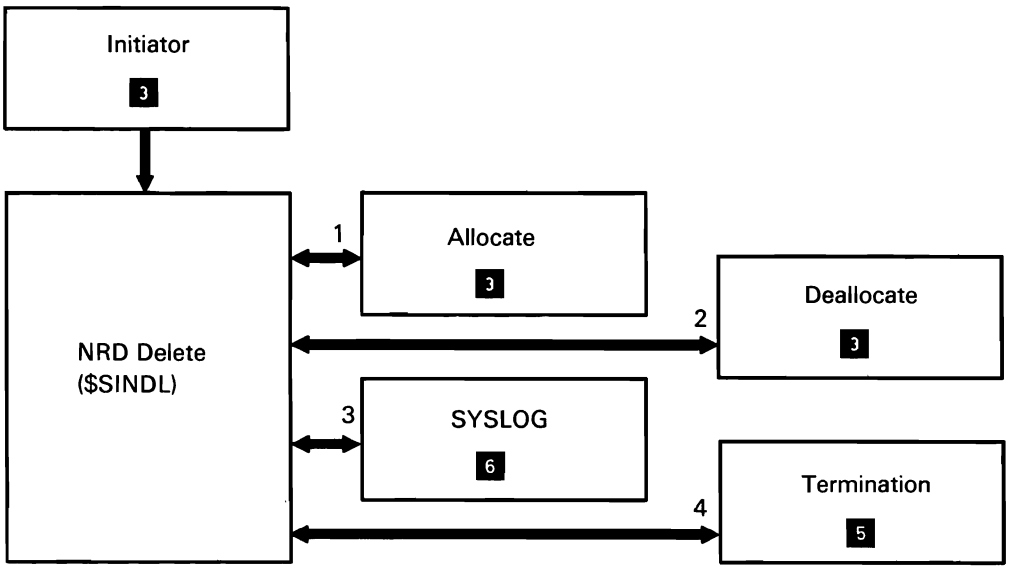


Chart 8.34.3 DELNRD Utility Control Flow

LISTNRD Utility

The LISTNRD utility allows the user to list the entries in the network resource directory (NRD) on the SYSLIST device.

The following LISTNRD utility processes are shown in Chart 8.34.4:

1 Read and syntax check control statements.

2 Allocate NRD file.

3 Use disk IOS to read NRD entries.

4 Display sorted and formatted entries.

5 Deallocate NRD.

6 Issue any required messages.

7 Terminate this job step.

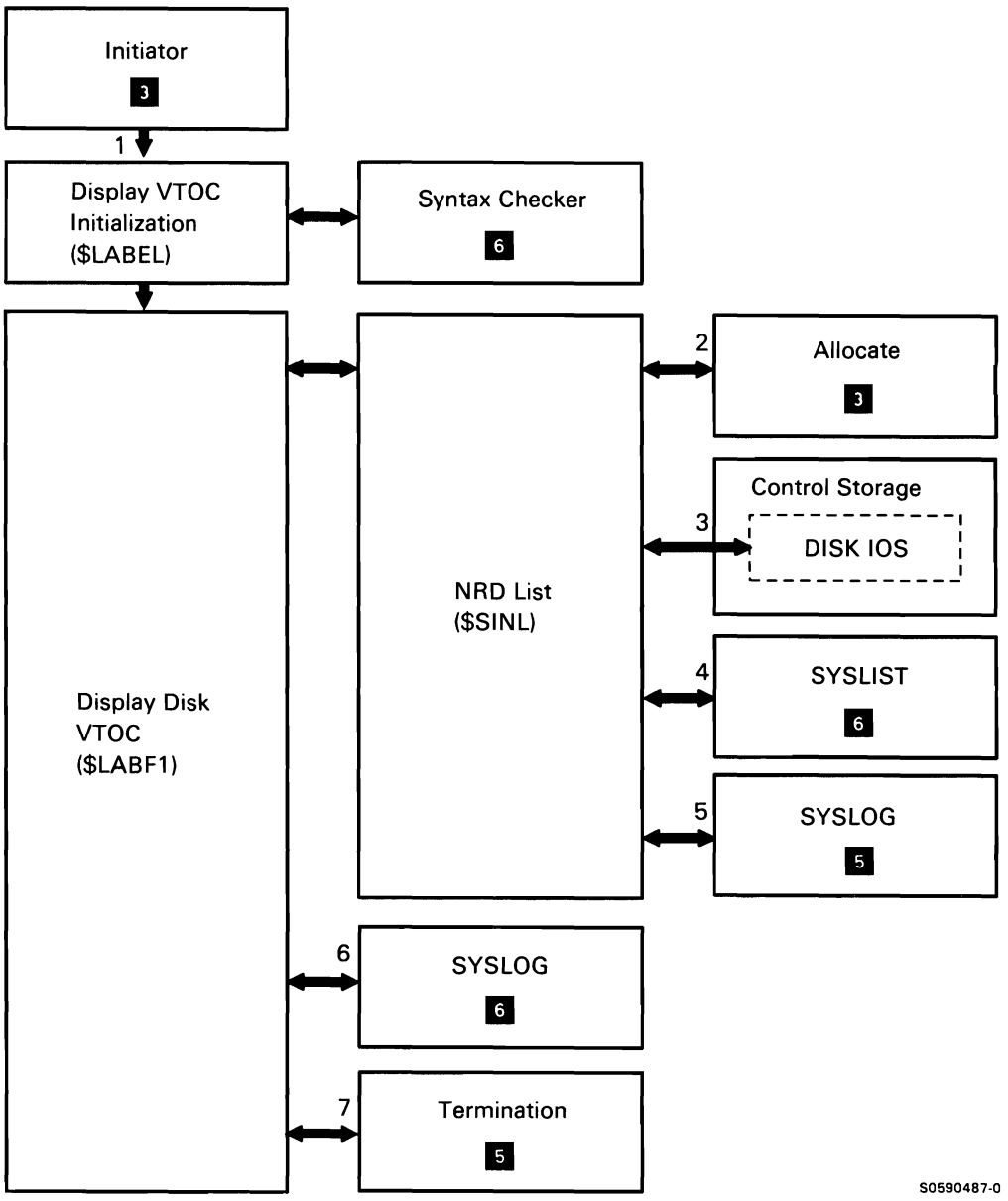


Chart 8.34.4 LISTNRD Utility Control Flow

Folder Management Services (FMS) Utility ($TMSERV)

The folder management services (FMS) utility can be used to perform any of the following functions:

- Save a single folder to a disk, diskette, or tape file.
- Save all folders on the system to diskette or tape files.
- Restore a saved folder from a disk, diskette, or tape file.
- Reorganize a folder.
- Save folder members to a disk, diskette, or tape file.
- Restore saved folder members from a disk, diskette, or tape file.
- List information about archived members from a diskette or tape file.
- Move a folder from one location to another on a disk.

The FMS utility is invoked by the following system procedures or equivalent utility control statements (UCLs): SAVEFLDR, RESTFLDR, ALOCFLDR, CONDENSE, ARCHIVE, RETRIEVE, LISTFILE, and MOVEFLDR.

The following FMS utility processes are shown in Chart 8.35:

1 Read and syntax check utility control statements.

2 Issue any required messages.

3 Process request to ARCHIVE folder member to disk, diskette, or tape file.

4 Process request to RETRIEVE folder member from disk, diskette, or tape file.

5 Process request to perform LISTFILE of archived folder member on a diskette or tape file.

6 Process request to perform ALOCFLDR or CONDENSE folder.

7 Process request to SAVEFLDR a folder to a diskette or tape file.

8 Process request to SAVEFLDR a folder to a disk file.

9 Process request to RESTFLDR a folder from a diskette or tape file.

10 Process request to RESTFLDR a folder from a disk file.

11 Process request to MOVEFLDR from one location on disk to another location.

12 Terminate this job step.

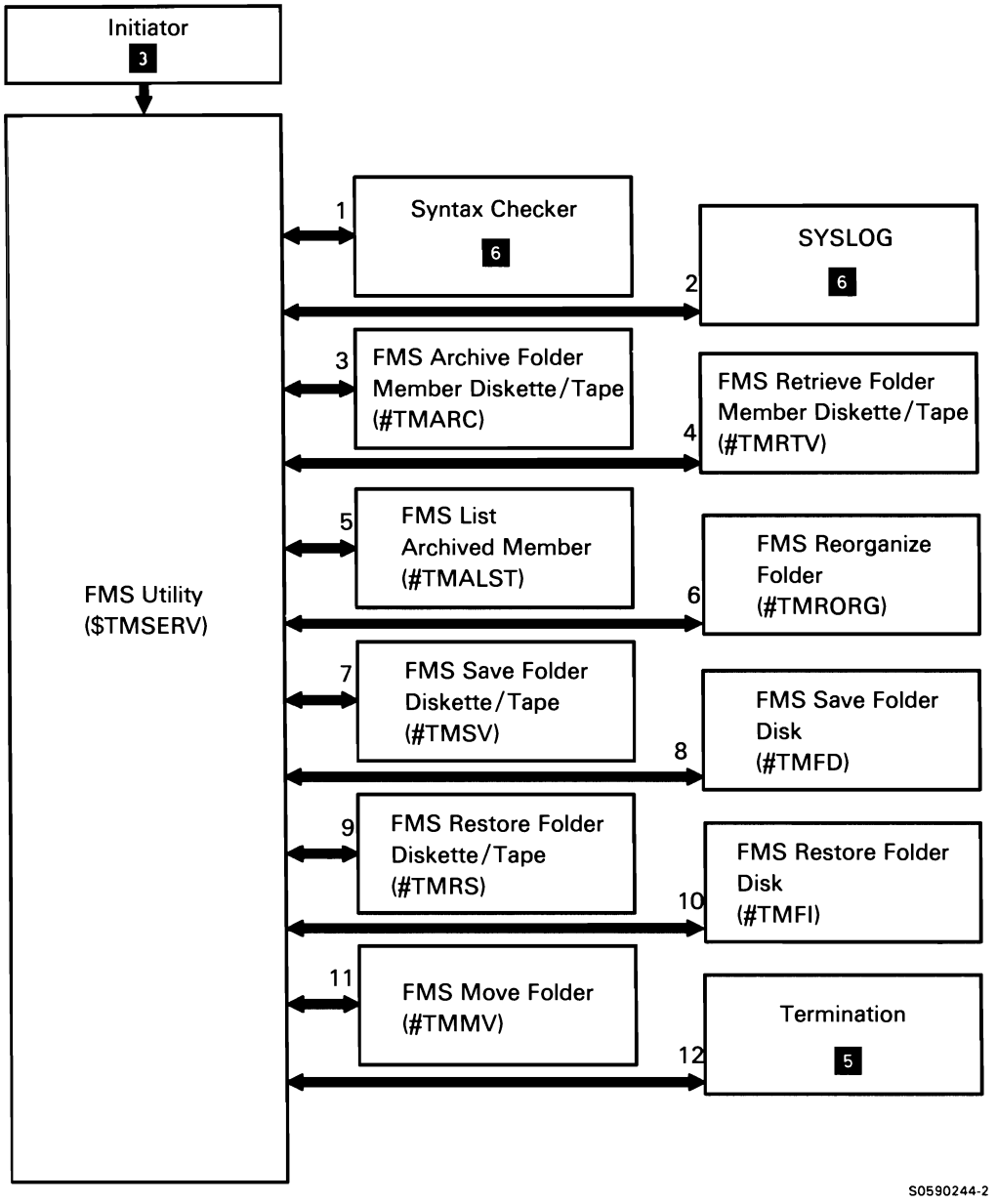


Chart 8.35 Folder Management Services (FMS) Utility Control Flow

Interactive Data Definition Utility (IDDU)

The interactive data definition utility (IDDU) allows users to define data dictionaries and the disk file definitions and communication file definitions they contain. Once defined, IDDU definitions are used by Query/36, the data entry facility, PC Support/36 file transfer, or the APPC subsystem.

IDDU provides for the following file and dictionary maintenance:

- Define data dictionary.
- Create, update, or delete disk file definitions.
- Link/unlink disk file definition and disk file label.
- Enter/update disk file data (via Query/36).
- Create, update, or delete communications file definitions.

IDDU generates internal source that is usable by Query/36 or DW/36, without compilation. The data definitions built by IDDU are contained in data dictionaries.

A data dictionary is a collection of all definitions related to a file, a group of related files, an application, a group of related applications, or the entire system. Entries in the data dictionary consist of field definitions grouped into format definitions, grouped into file definitions.

The following IDDU processes are shown in Chart 8.36.0:

1 Initialize and route control for IDDU processing:

   - Determine type of call: If procedure, process data in PPSA; if called by an SVC transfer instruction, process IDDU parameter list.
   - Store parameter information in IDDU global area and route control for required processing.

2 If selected, process for dictionary definition options.

3 If selected, process for dictionary options.

4 If selected, process for disk file option to be performed on file specified (by label).

5 If selected, perform interactive link or unlink disk file options.

6 If selected, perform batch printing.

7 If selected, perform dictionary rebuild.

8 If selected, perform disk file link/unlink.

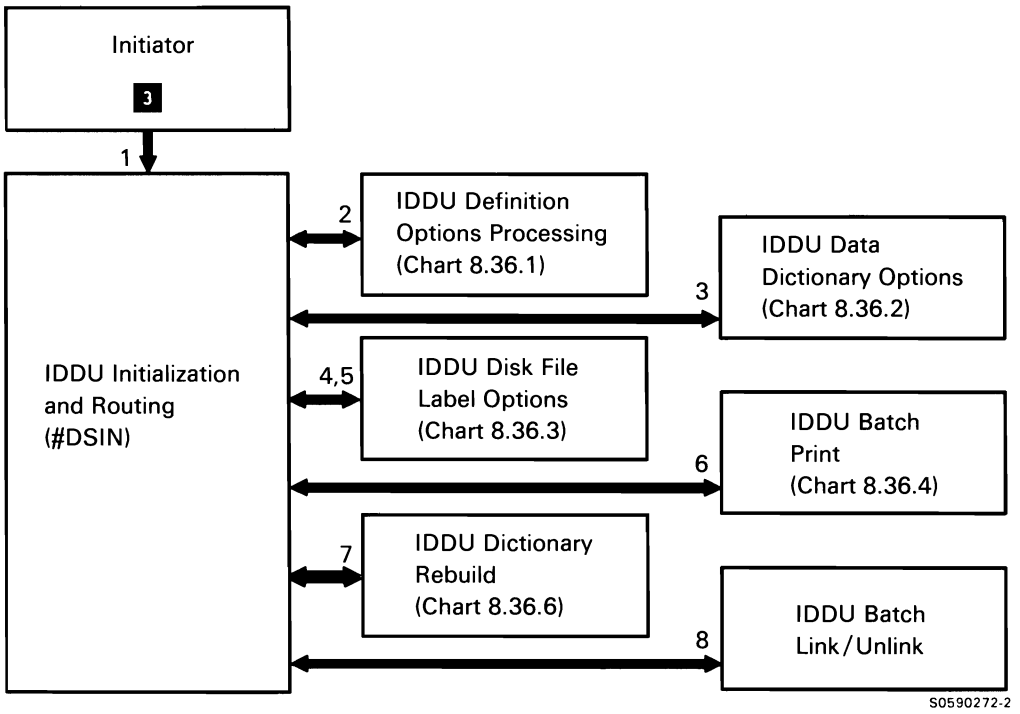


Chart 8.36.0 Interactive Data Definition Utility Overview Control Flow

Interactive Data Definition Utility (IDDU) Definition Options Processing

IDDU definition options processing allows users to create or change entries in the data dictionaries.

The following IDDU definition options processes are shown in Chart 8.36.1:

1 Prompt for, and process, definition change requests from user:

- Get parameter information from IDDU global area.
- Display and process definition-type display.
- Route control according to options selected.
- On return from specific option processor, display definition-type display.
- If #DSIN was loaded, route control to termination; otherwise, return to #DSIN.

2 Prompt for, and process, data dictionary change requests from user.

3 If selected, rebuild dictionary.

4 Prompt for, and process, user-specified field options for a definition.

5 Prompt for, and process, user-specified format options for a definition.

6 Prompt for, and process, user-specified file options for a definition.

7 Prompt user to select data dictionary name.

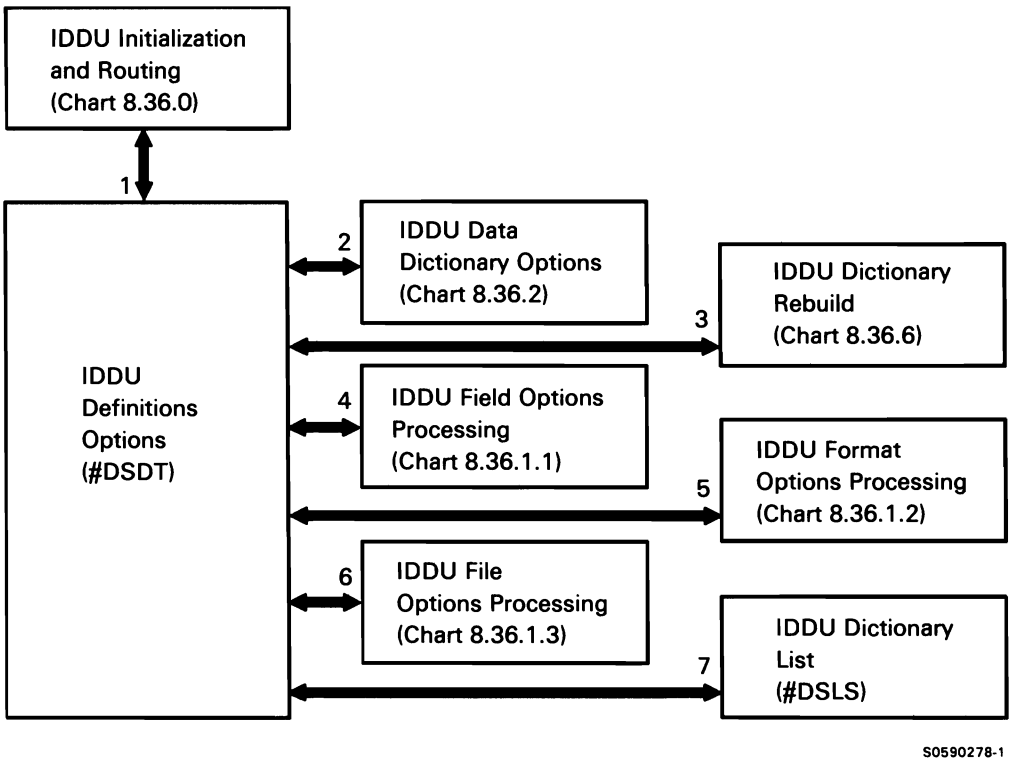


Chart 8.36.1 Interactive Data Definition Utility Definition Options Processing Control Flow

Interactive Data Definition Utility (IDDU) Field Options Processing

IDDU field options processing allows users to create or change field options for data dictionary entries.

The following IDDU field options processes are shown in Chart 8.36.1.1:

1 Perform mainline processing, routing control where necessary:

- Display and process options screen.
- Perform list and recovery processing as required.
- Verify field name to be used in processing.
- Process selected options.
- Return to caller.

2 Prompt for, and process, user-specified changes to a field definition.

3 Prompt for, and process, field attributes changes.

4 Prompt for, and process, changes to numeric field edit characteristics.

5 Prompt for, and process, long comment.

6 Prompt for, and process, save option and update data dictionary with field definition changes according to options selected.

7 Prompt for, and process, copy selection.

8 Prompt to confirm delete.

9 Prompt for, and process, print options to be performed on the specified definition.

10 Prompt for, and process, rename options for the specified field.

11 Prompt user to view list of items that were changed according to user request(s).

12 Prompt for creating multiple field definitions.

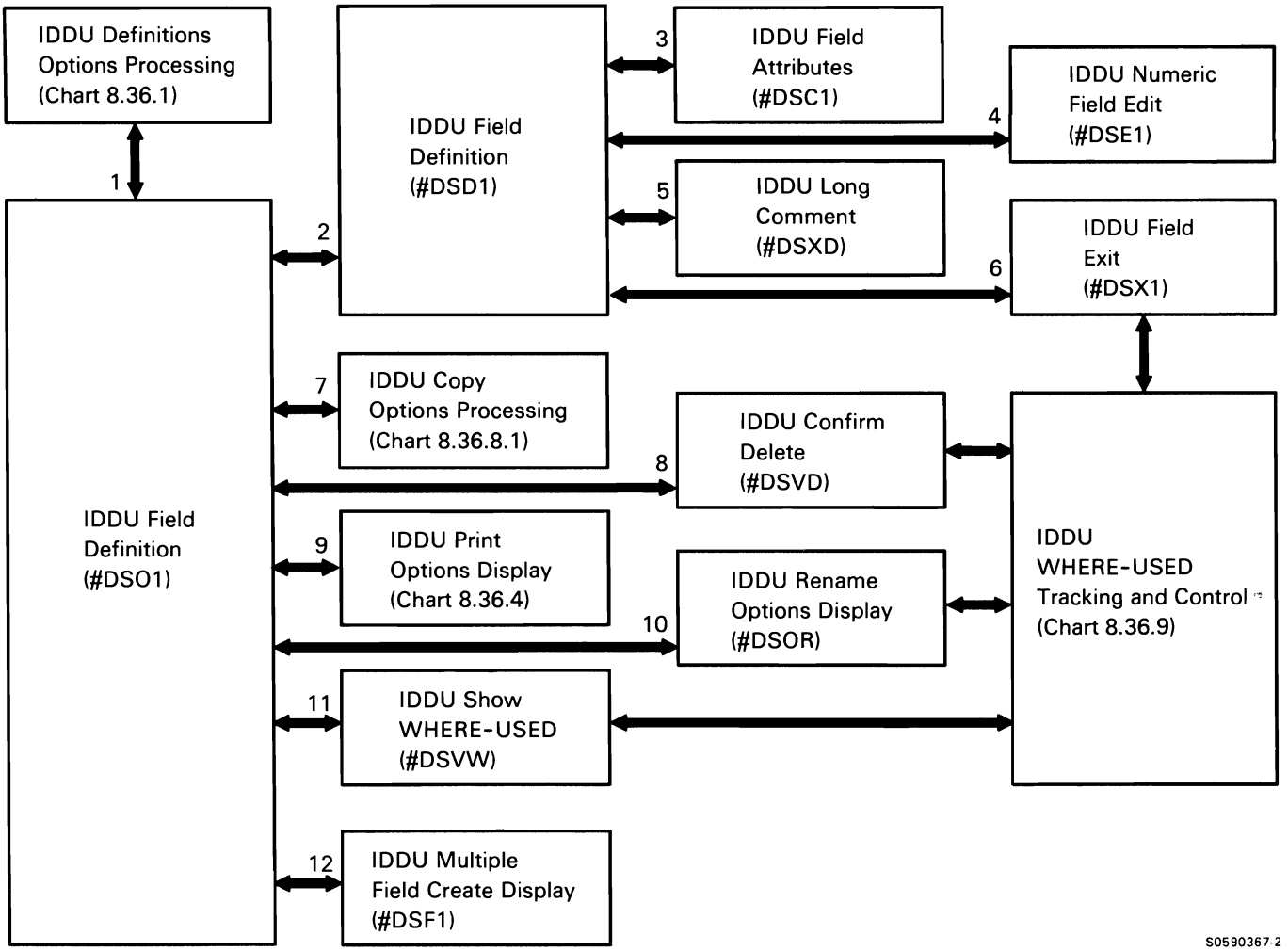


Chart 8.36.1.1 Interactive Data Definition Utility Field Options Processing Control Flow

Interactive Data Definition Utility (IDDU) Format Options Processing

IDDU format options processing allows users to create or change format options for data dictionary entries.

The following IDDU format options processes are shown in Chart 8.36.1.2:

1 Perform mainline processing, routing control where necessary:

- Display and process options screen.
- Perform list and recovery processing as required.
- Verify format name to be used in processing.
- Process selected options.
- Return to caller.

2 Prompt for, and process, user-specified changes to a format definition, routing control when necessary.

3 Prompt to verify WHAT-USED table entries for specified format definition.

4 Prompt for, and process, communications attributes changes.

5 Display requested record-ID codes.

6 Prompt for, and process, long comment.

7 Prompt for, and process, save option and update data dictionary with format definition changes according to options selected.

8 Prompt for, and process, copy selection.

9 Prompt to confirm delete.

10 Prompt for, and process, print options to be performed on the specified definition.

11 Prompt for, and process, rename options for the specified format.

12 Prompt user to view list of items that were changed according to user request(s).

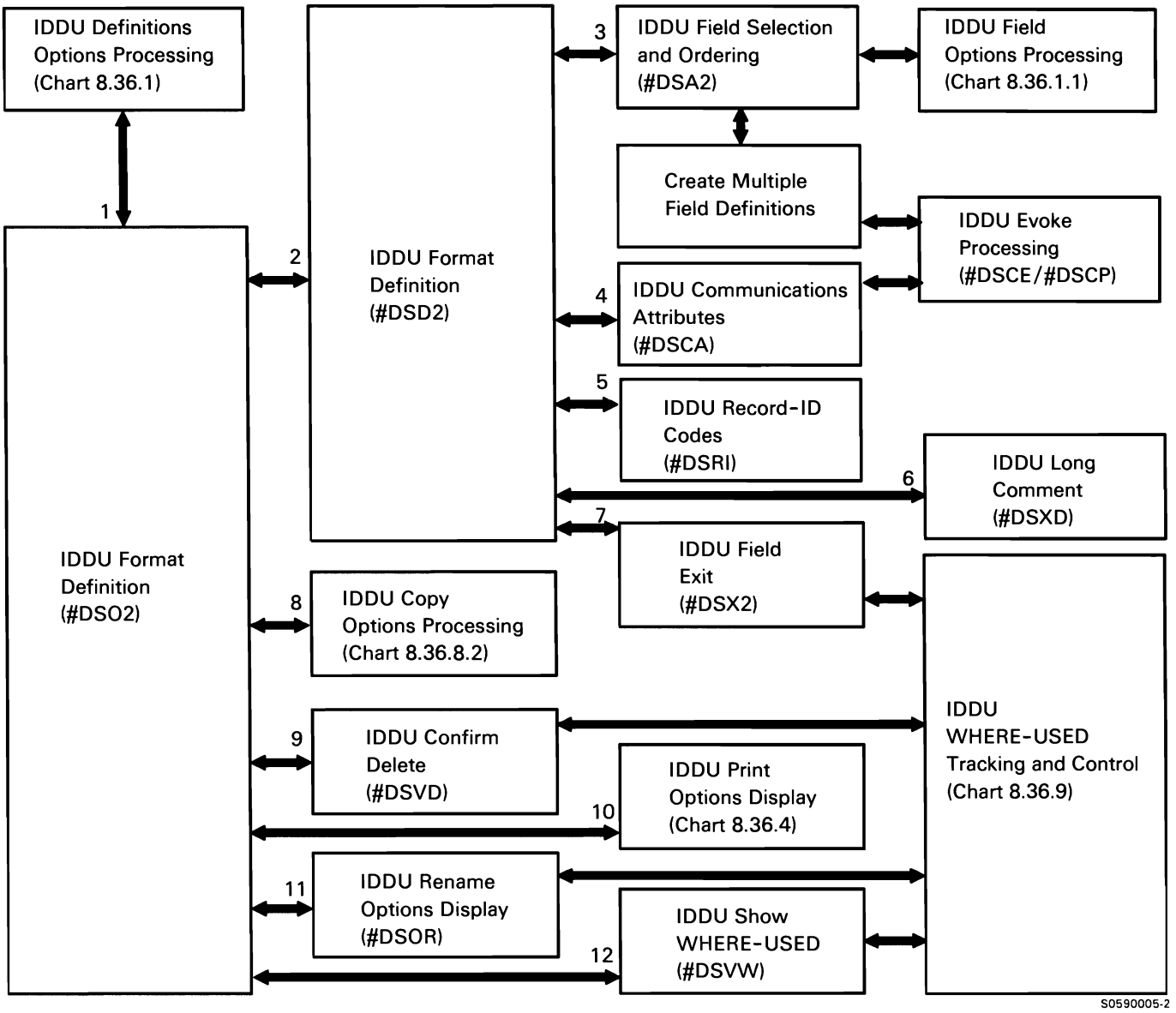


Chart 8.36.1.2 Interactive Data Definition Utility Format Options Processing Control Flow

Interactive Data Definition Utility (IDDU) File Definition Options Processing

IDDU file definition options processing allows users to create or change file options for data dictionary entries.

The following IDDU file options processes are shown in Chart 8.36.1.3:

1 Perform mainline processing, routing control where necessary:

- Display and process options screen.
- Perform list and recovery processing as required.
- Verify file name to be used in processing.
- Process selected options.
- Return to caller.

2 Prompt for, and process, user-specified changes to a file definition.

3 Prompt to verify WHAT-USED table entries for specified file definition.

4 Prompt for, and process, user specifications for long comment.

5 Prompt for, and process, save option and update data dictionary with file definition changes according to options selected.

6 Prompt for, and process, copy selection.

7 Prompt to confirm delete.

8 Prompt for, and process, print options to be performed on the specified definition.

9 Prompt for, and process, rename options for the specified file.

10 Prompt user to view list of items that were changed according to user request(s).

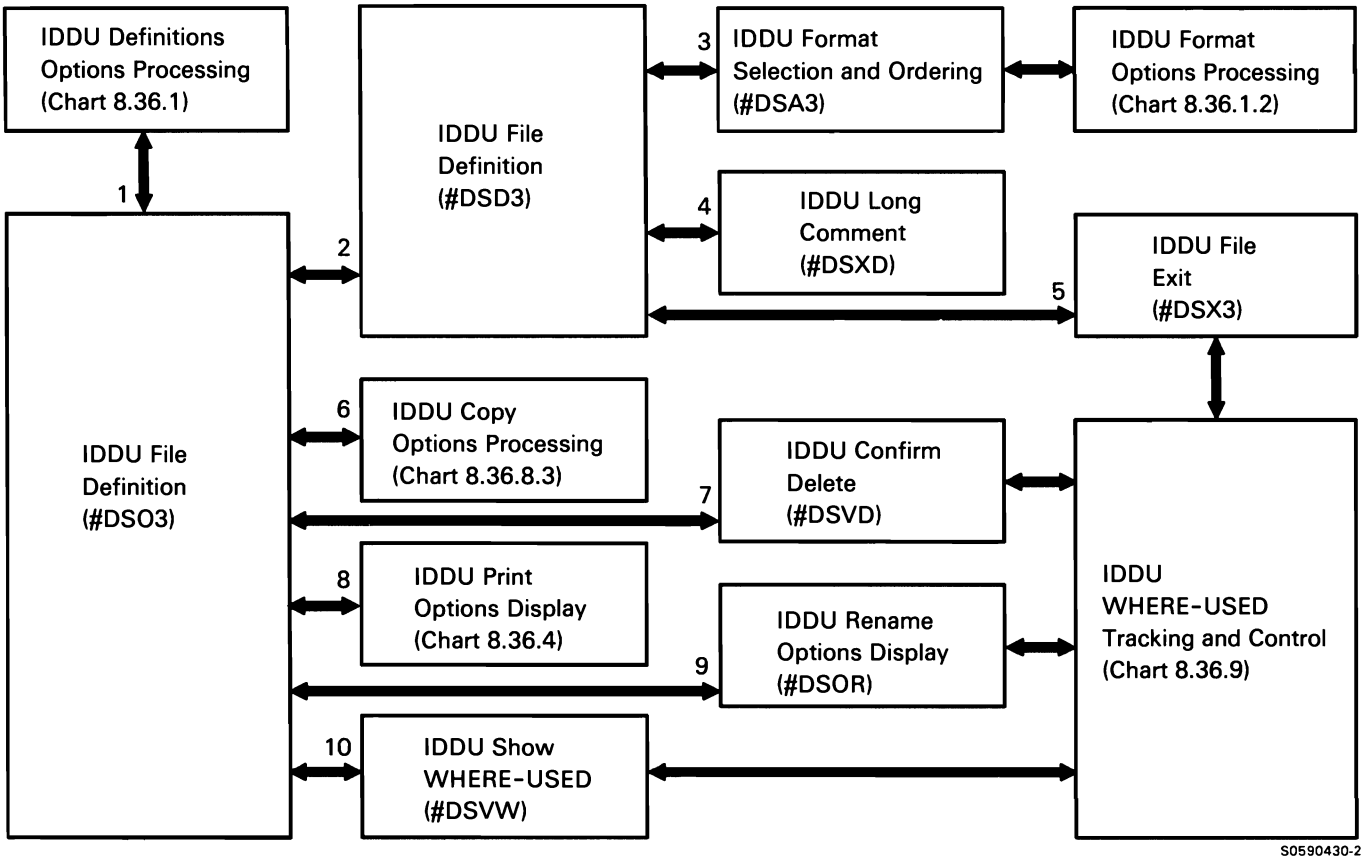


Chart 8.36.1.3 Interactive Data Definition Utility File Options Processing Control Flow

Interactive Data Definition Utility (IDDU) Data Dictionary Options Processing

Interactive data definition utility (IDDU) data dictionary options processing prompts for and processes the following options:

- Create data dictionary or security authorization.
- Revise data dictionary or security authorization.
- Delete data dictionary.
- Print data dictionary.
- Rename data dictionary.

The following IDDU data dictionary options processes are shown in Chart 8.36.2:

1 Perform mainline dictionary options processing:

- Verify dictionary name.
- Display and process dictionary options display, provide list, and route control according to options selected.
- Return to caller.

2 Create new data dictionary.

3 Change dictionary long/short comment and, if selected, dictionary security.

4 Assign or change dictionary security.

5 Prompt for, and process, long comment.

6 Prompt for confirmation of delete request.

7 Prompt for, and process, print options.

8 Prompt for, and process, rename options.

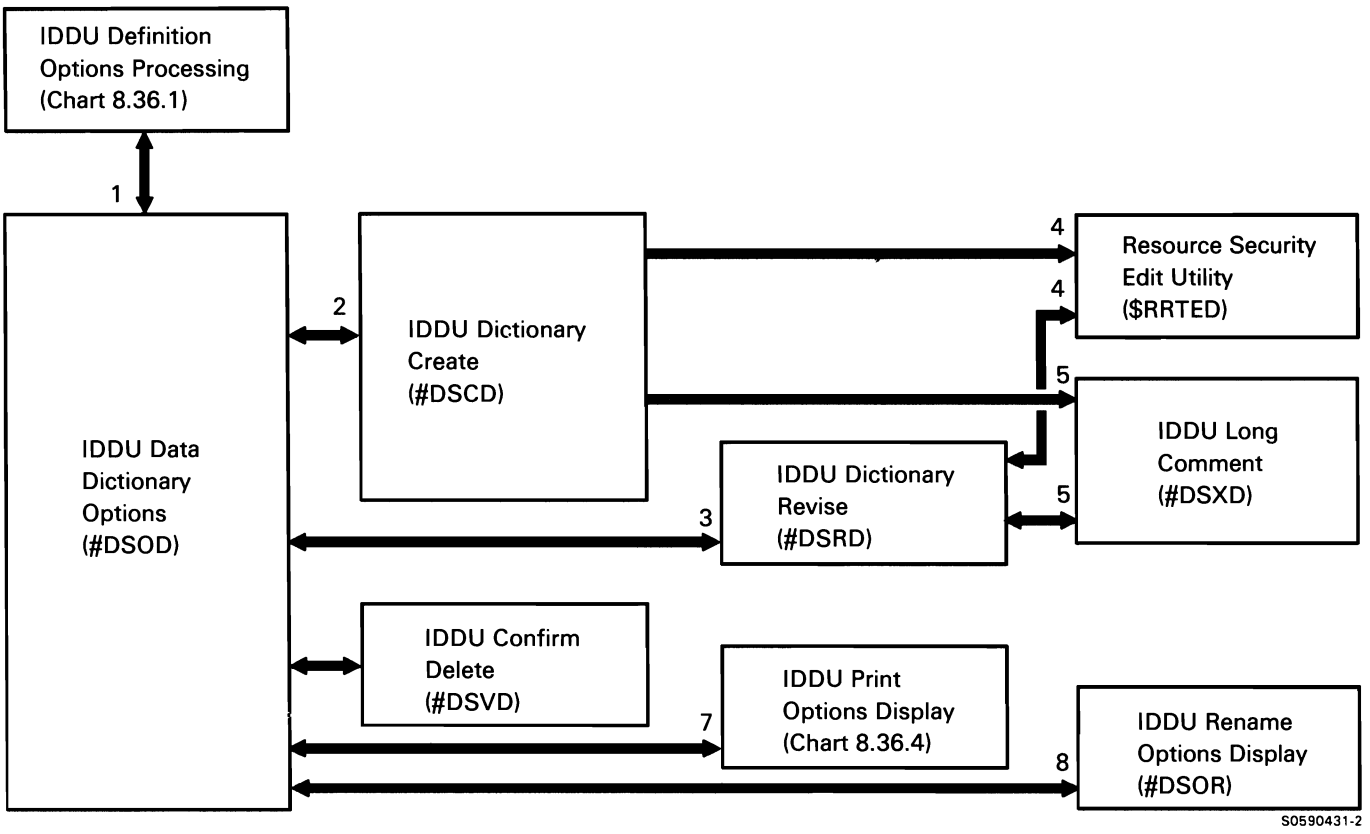


Chart 8.36.2 Interactive Data Definition Utility Data Dictionary Options Processing Control Flow

Interactive Data Definition Utility (IDDU) Disk File Label Options

The interactive data definition utility (IDDU) disk file label options processor prompts for the following disk file options to be performed on the file specified by the label passed in the IDDU global area:

- Create disk file.
- Enter/update data in disk file.
- Link disk file to file definition.
- Unlink disk file from file definition.

The following IDDU disk file label options processes are shown in Chart 8.36.3:

1 Perform mainline processing for disk file label options:

   - Verify dictionary name.
   - Display and process options display, provide list, and route control according to options selected.
   - Return to caller.

2 Read VTOC and list specified file labels.

3 Prompt for creation of a new disk file.

4 Perform requested data entry utility processing.

5 Prompt for, and process, user-specified options to link to a definition.

6 List data dictionaries and process user selection.

7 List file definitions and process user selection.

8 Go to function via command keys.

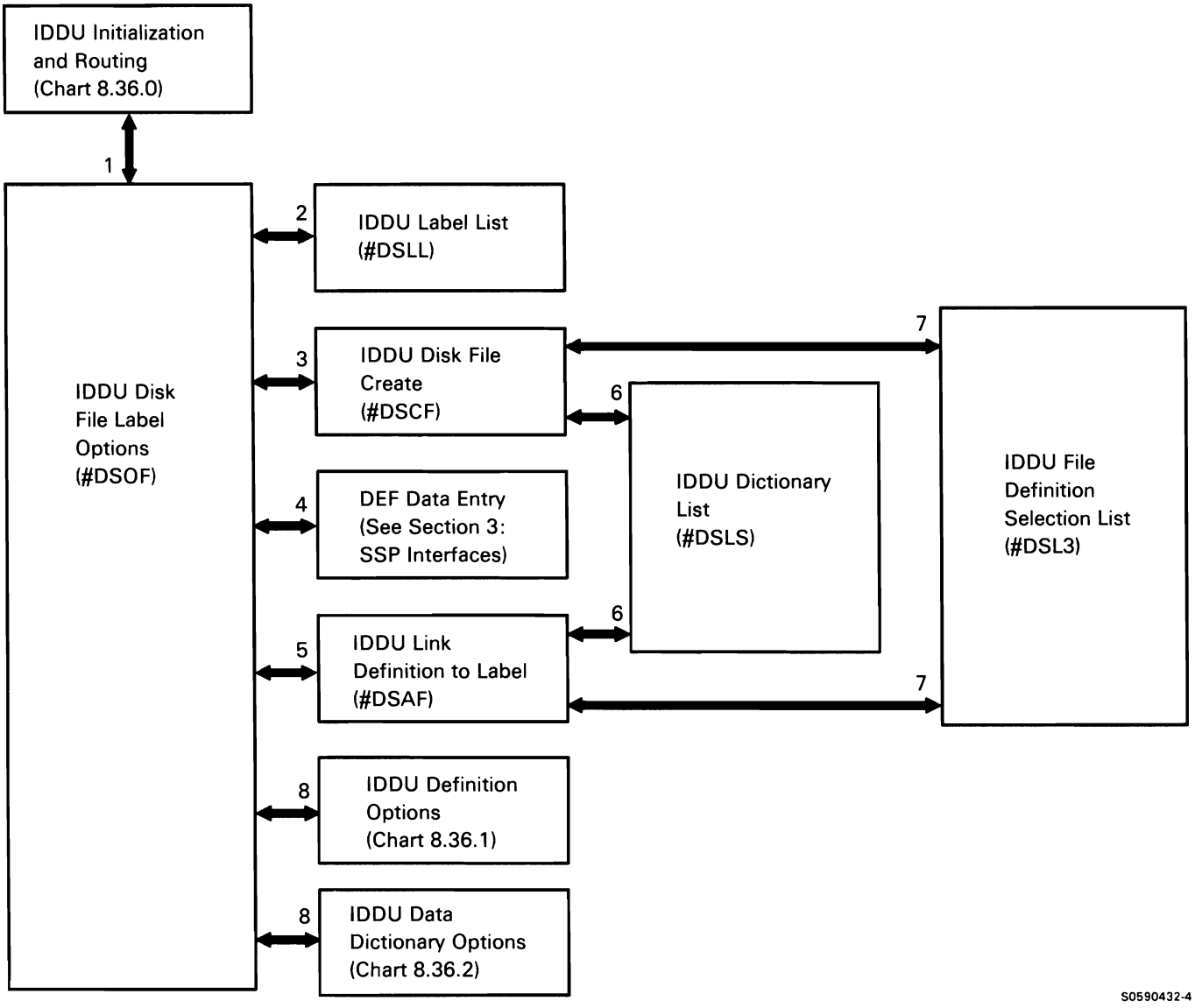


Chart 8.36.3 Interactive Data Definition Utility Disk File Label Options Processing Control Flow

Interactive Data Definition Utility (IDDU) Batch Print Processing

Interactive data definition utility (IDDU) batch print processing performs all printing specified by IDDU users.

The following IDDU utility batch print processes are shown in Chart 8.36.4:

1 If selected, set up for batch printing:

- Get DDA information.
- Allocate and open print file.
- Route control for required processing according to options selected:
  - Print file definitions.
  - Print format definitions.
  - Print field definitions.
- Return to caller.

2 Print requested field definitions and return to caller.

3 Print requested format definitions and return to caller.

4 Print requested file definitions and return to caller.

5 Retrieve WHERE-USED information on specified field, format, or file definition, then return to caller.

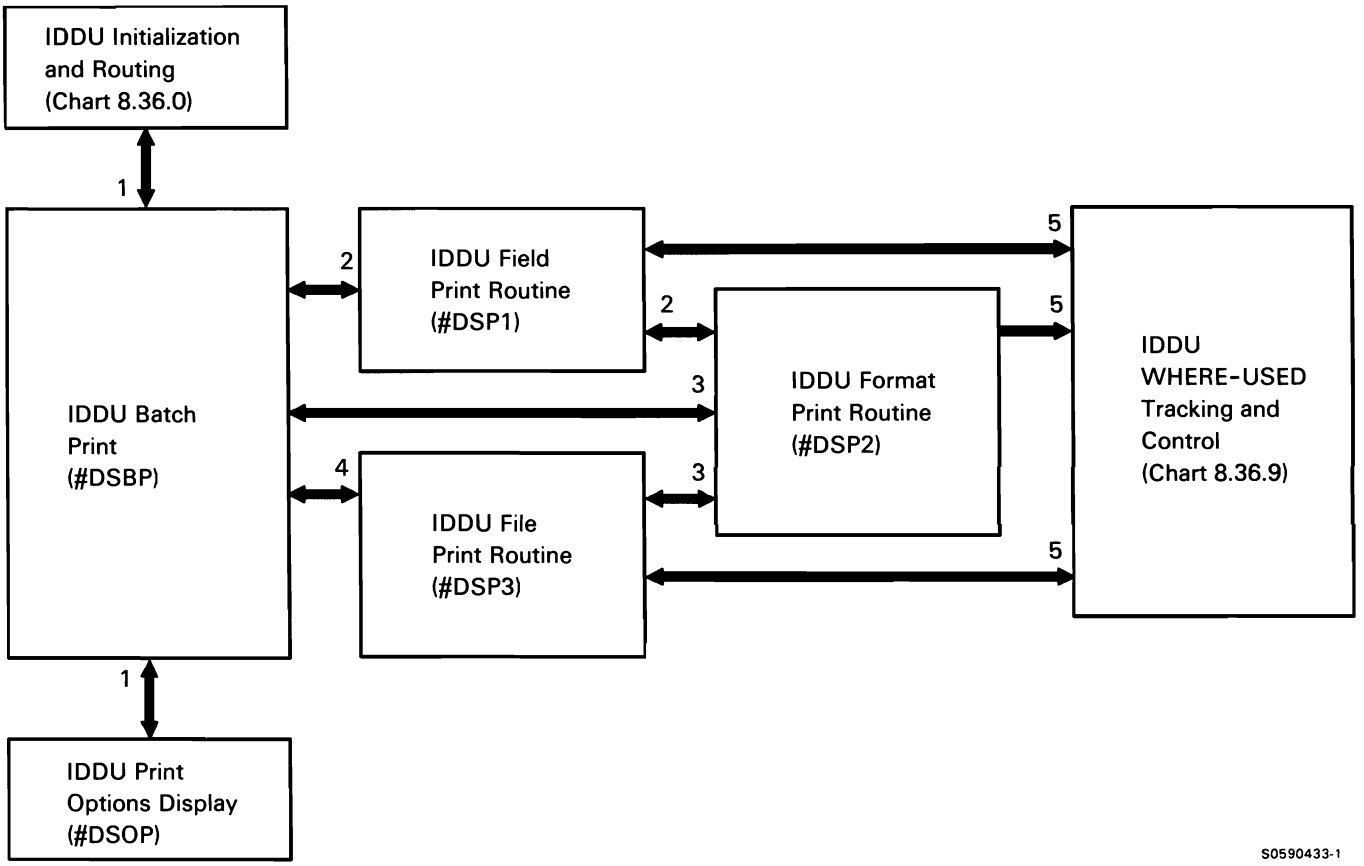


Chart 8.36.4 Interactive Data Definition Utility Batch Print Processing Control Flow

Interactive Data Definition Utility (IDDU) F and I Specification Conversion

Interactive data definition utility (IDDU) F and I specification conversion converts RPG II FILE and INPUT specifications into data dictionary definitions. It is invoked via the IDDUXLAT procedure.

The following IDDU F and I specification conversion processes are shown in Chart 8.36.5:

1 Initialize for IDDUXLAT processing:

- Allocate TWS.
- Build parameter lists in global data area.
- Get parameters passed with IDDUXLAT procedure from PPSA.
- Verify existence of data dictionary.
- Route control to source processor, #DSZSR.

2 Process F and I specifications:

- Map to global data area set up by #DSZIN.
- Initialize work area and parameter lists.
- Route control for required processing.

3 Find and get specified member.

4 Read source member and create work file.

5 Use global data area and scratch work file passed by #DSZSR to create data dictionary entries, routing control where necessary.

6 Convert F and I specifications to IDDU definitions, routing control where necessary.

7 Update data dictionary with field definition changes.

8 Update data dictionary with format definition changes.

9 Update data dictionary with file definition changes.

10 Terminate this job step.

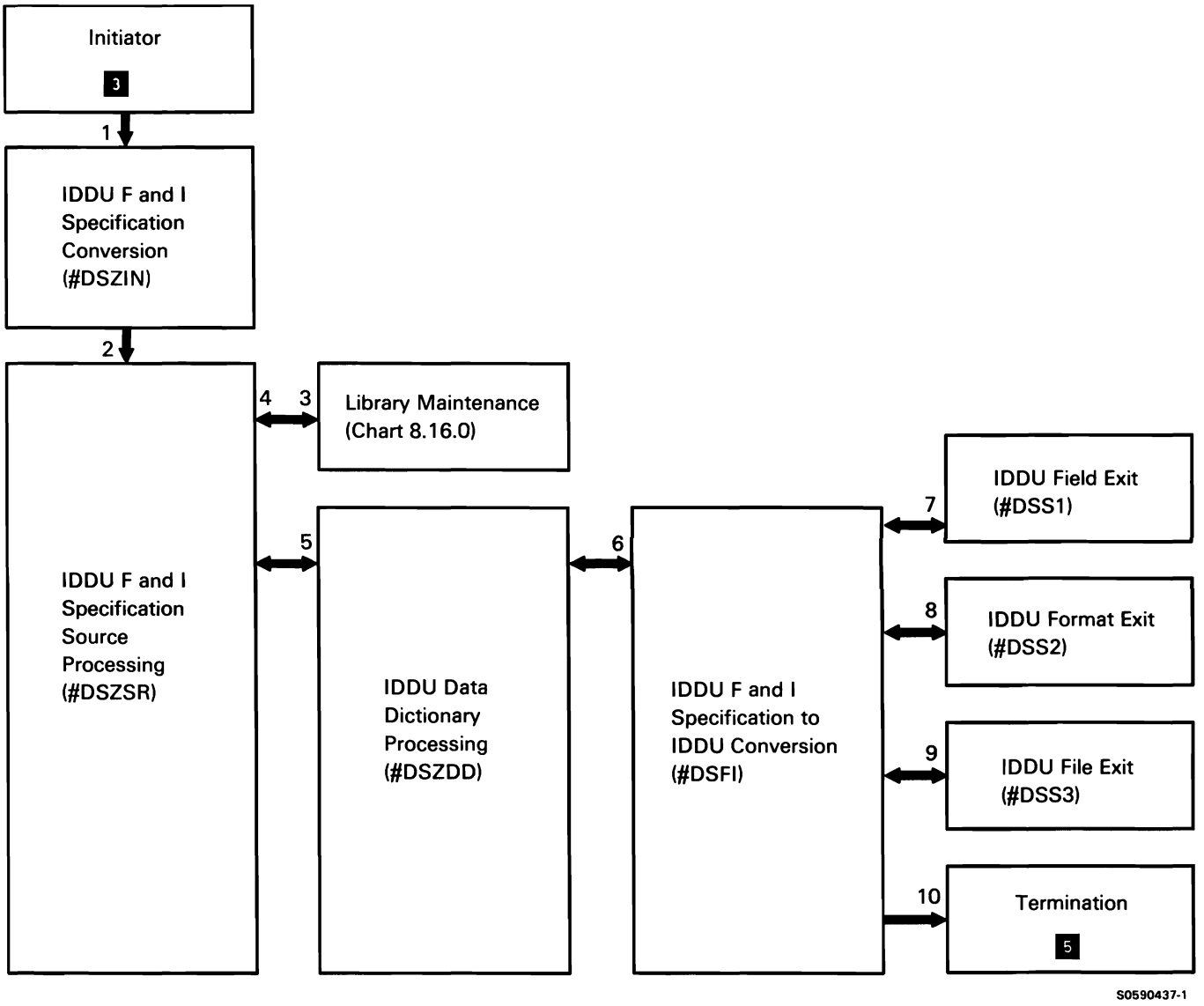


Chart 8.36.5 Interactive Data Definition Utility F and I Specification Conversion Processing Control Flow

Interactive Data Definition Utility (IDDU) Dictionary Rebuild

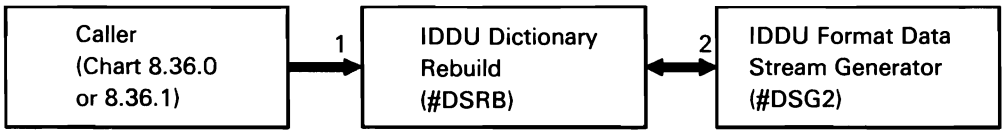
Interactive data definition utility (IDDU) dictionary rebuild rebuilds dictionary definitions.

The following IDDU dictionary rebuild processes are shown in Chart 8.36.6:

1 Perform dictionary rebuild processing, routing control where necessary:

- Get dictionary interlock in DDA.
- Check that all files in user's list are not in use.
- Set rebuild in progress flag.
- Rebuild dictionary definitions.

2 Generate record-ID code and communications format data streams.



S0590438-0

Chart 8.36.6 Interactive Data Definition Utility Dictionary Rebuild Control Flow

Interactive Data Definition Utility (IDDU) Common Open

Interactive data definition utility (IDDU) common open opens data dictionaries and file definitions within them. IDDU common open also generates required security for the dictionaries and definitions. Callers invoke common open with a transfer SVC instruction and a TMIO parameter list.

The following IDDU common open processes are shown in Chart 8.36.7:

1 Perform open requests for specified data dictionary and/or file definitions, routing control where necessary:

- Check TMIO parameter list to determine if caller requests read-only or read/write access.
- Create security key if data management run access is specified.
- Open specified data dictionary.
- Open specified file definition.
- Return to caller with specified file definition opened.

2 Process security requests for data dictionaries and their file definitions.

3 Open specified data dictionary.

4 Open specified file definition(s).

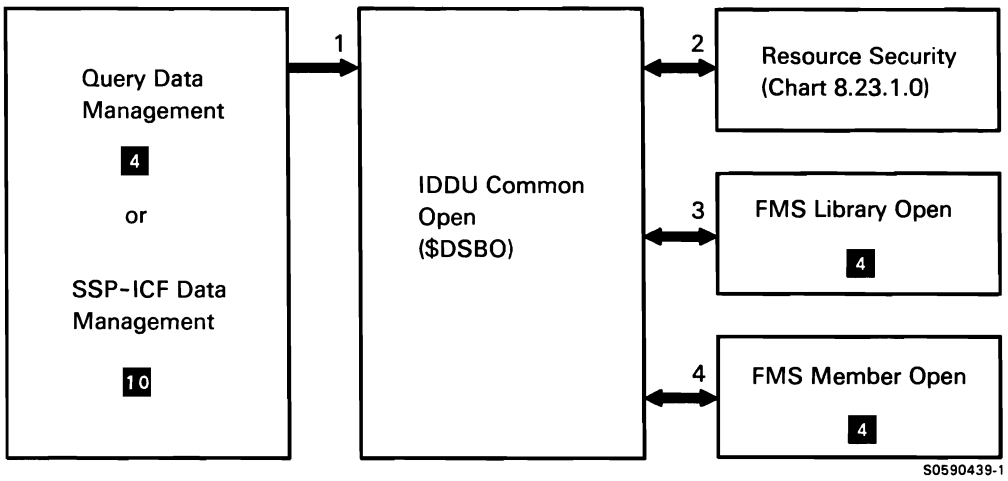


Chart 8.36.7 Interactive Data Definition Utility Common Open Control Flow

Interactive Data Definition Utility (IDDU) Copy Options Displays

Interactive data definition utility (IDDU) copy options displays prompt for, and process, copy options for the specified field, format, or file definitions. Copy options processing for field, format, and file definitions are shown in Charts 8.36.8.1, 8.36.8.2, and 8.36.8.3, respectively.

The following IDDU field definitions copy options processes are shown in Chart 8.36.8.1:

1 Prompt for, and process, copy-to selections, routing control where necessary:

- Prompt for, and process, copy-to dictionary name.
- Prompt for, and process, copy-to field definition.

2 Process copy request for definition being copied.

3 Prompt for, and process, dictionary list selections.

4 Prompt for, and process, field definition list selections.

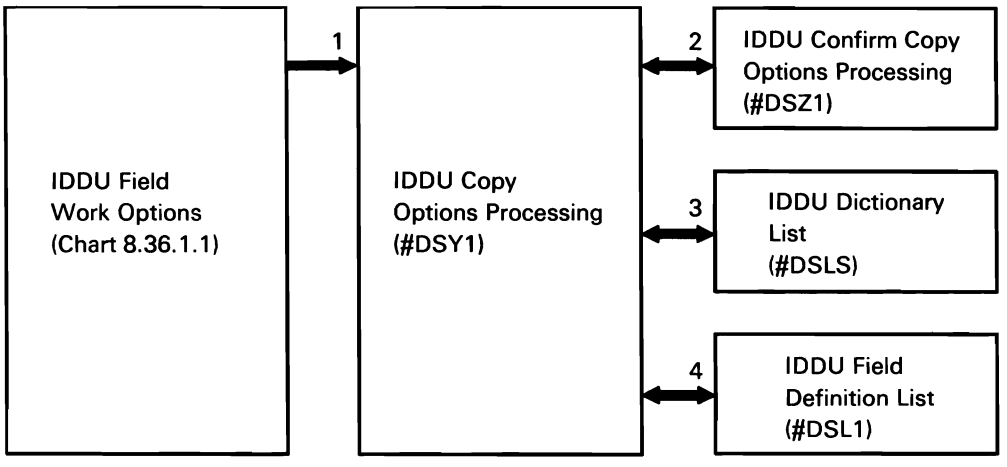


Chart 8.36.8.1 Interactive Data Definition Utility Field Definitions Copy Options Processing Control Flow

The following IDDU format definitions copy options processes are shown in Chart 8.36.8.2:

1 Prompt for, and process, copy-to selections, routing control where necessary:

   - Prompt for, and process, copy-to dictionary name.
   - Prompt for, and process, copy-to format definition.

2 Prompt for copy option and process copy request for format definition.

3 Prompt for, and process, dictionary list selections.

4 Prompt for, and process, format definition list selections.

5 Copy fields used by format definition.

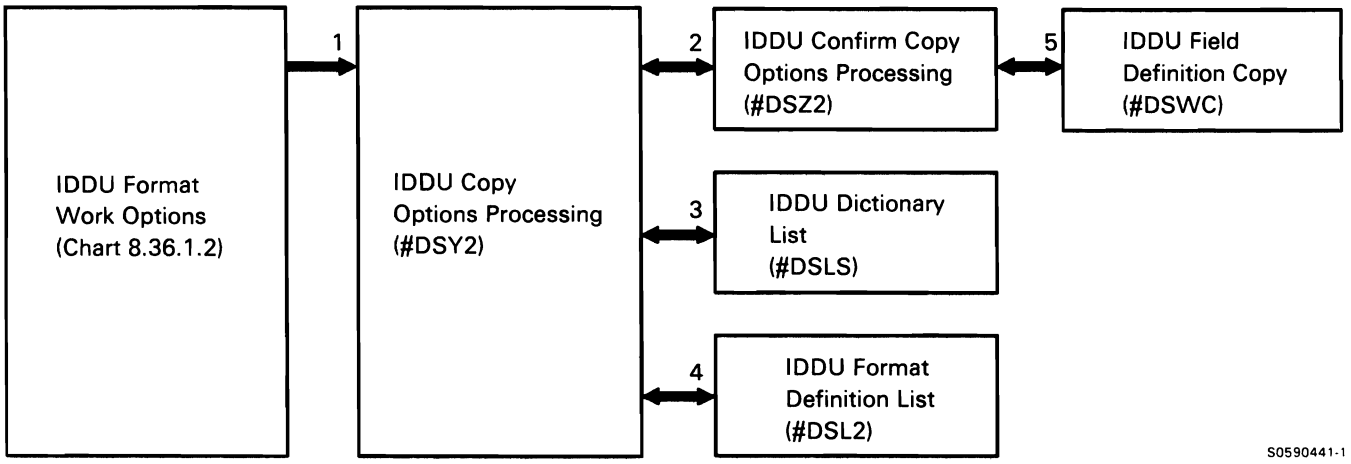


Chart 8.36.8.2 Interactive Data Definition Utility Format Definitions Copy Options Processing Control Flow

The following IDDU file definitions copy options processes are shown in Chart 8.36.8.3:

1 Prompt for, and process, copy-to selections, routing control where necessary:

   - Prompt for, and process, copy-to dictionary name.
   - Prompt for, and process, copy-to file definition.

2 Prompt for copy option and process copy request for file definition.

3 Prompt for, and process, dictionary list selections.

4 Prompt for, and process, file definition list selections.

5 Copy fields and formats used by file definition.

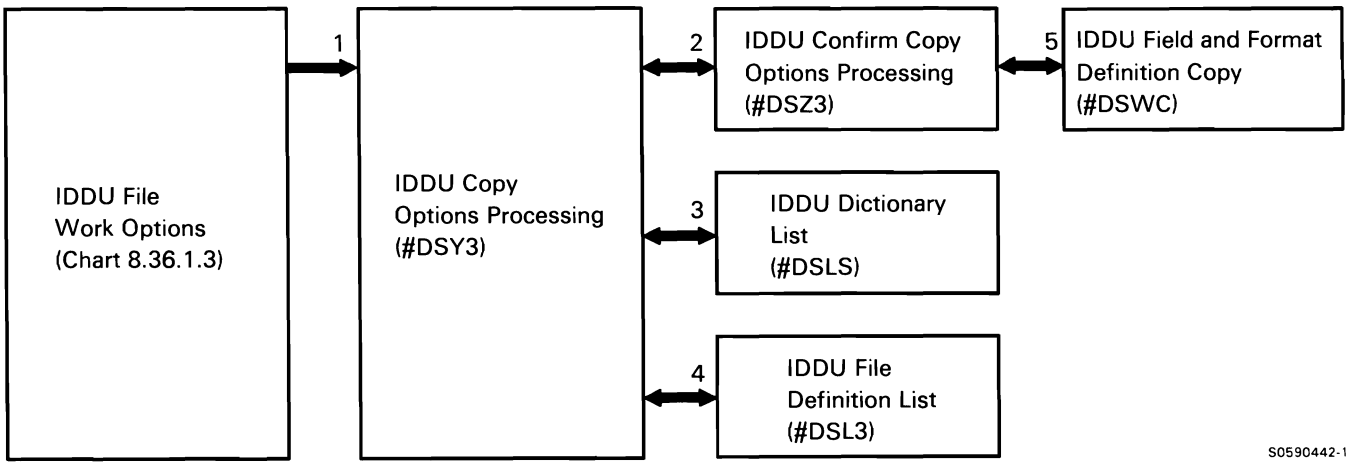


Chart 8.36.8.3 Interactive Data Definition Utility File Definitions Copy Options Processing Control Flow

Interactive Data Definition Utility (IDDU) WHERE-USED Tracking and Control

Interactive data definition utility (IDDU) WHERE-USED tracking and control maintains the WHERE-USED cross-reference directories for all dictionary definitions. When field, format, or file definitions are changed, #DSWU is called to confirm that the change will not cause errors in higher-level definitions and to make appropriate changes to those higher-level definitions.

The following IDDU WHERE-USED tracking and control processes are shown in Chart 8.36.9:

1 Process WHERE-USED options requested, routing control where necessary:

- Search for affected definitions and labels and maintain affected items records.
- Perform verification processing as requested.
- Perform mark operations as requested.

2 Retrieve list of affected labels.

3 Verify that changes made to specified definition will not cause errors in affected items.

4 Mark affected items, as requested.

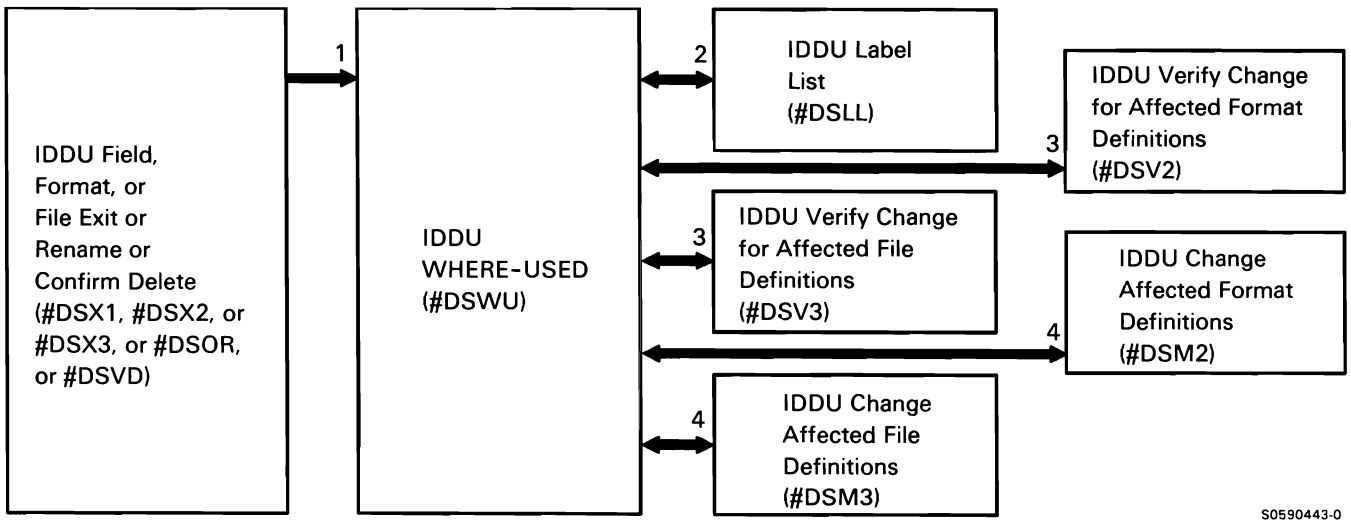


Chart 8.36.9 Interactive Data Definition Utility WHERE-USED Tracking and Control Processing Control Flow

Diagnostic Diskette Copy Utility

The diagnostic diskette copy utility copies all sectors (except sectors 3, 5, and 6 on cylinder 0, head 0) of diagnostic formatted diskettes.

The following diagnostic diskette copy utility processes are shown in Chart 8.37:

1 Perform or route for diskette copying requested.

2 Allocate diskette drive.

3 Prepare diskette for copy.

4 Copy all sectors from diskette to disk.

5 Copy all sectors from disk to output diskette.

6 Issue error messages.

7 Terminate program.

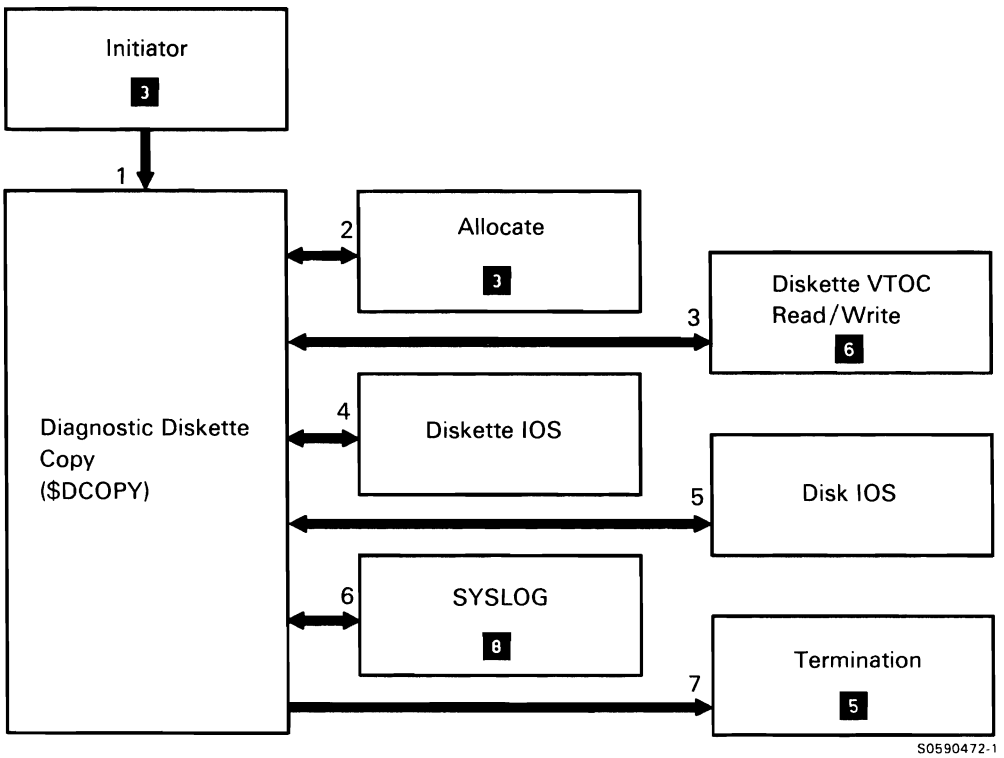


Chart 8.37 Diagnostic Diskette Copy Utility Control Flow

This page is intentionally left blank.

Data Communications Function 9

The data communications function provides the user with the communications line protocols and other detailed functions needed to use the System/36 SSP data communications higher-level programming features. It includes the following functions:

- Batch BSC — Chart 9.1.n
- SDLC — Chart 9.2.0
- MLCA/ELCA controller check handler — Chart 9.3.1
- X.21 — Chart 9.4.0
- Autocall — Chart 9.5
- Remote work station support — Chart 9.6.0
- Link station test — Chart 9.7
- X.25 — Chart 9.8.0
- Display Station Pass-Through — Chart 9.9.1
- Support of the local area network (LAN) — Chart 9.10.0

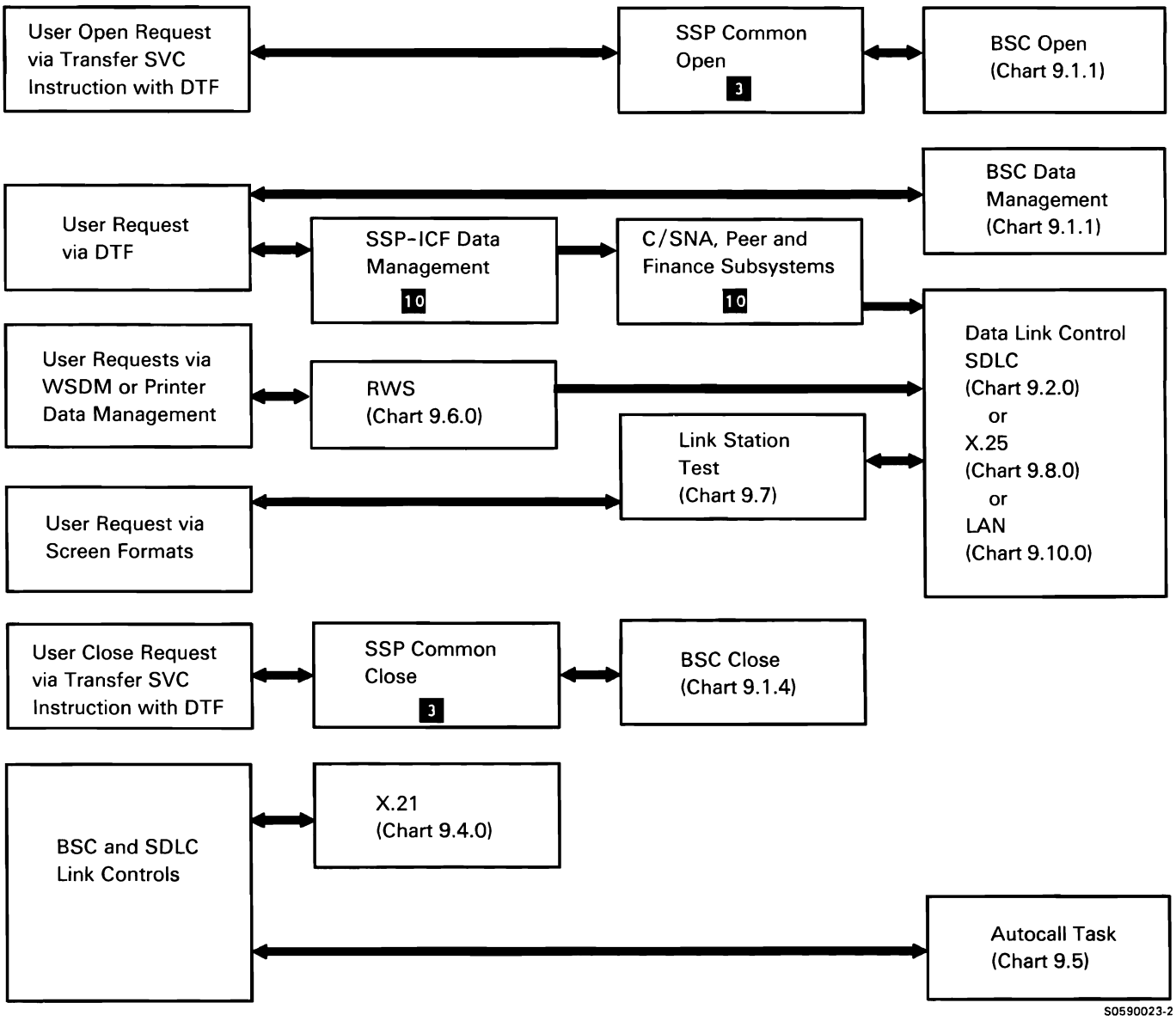


Chart 9.0 Data Communications Overview Control Flow

BATCH BSC SUBFUNCTION

The batch BSC subfunction provides the RPG II or assembler language user with the support necessary to use the communications adapter to transmit and receive data over communications lines to and from other systems or terminals. For information on batch BSC line protocols, refer to *Appendix A: Data Communications Line Protocols*.

Batch BSC consists of a data management module, a main storage interrupt handler module (running as a separate task), and several transients. The user program interfaces directly with the BSC data management module via the transfer SVC instruction. XR2 points to the user program DTF, which points to the logical buffer.

Other interface information needed by the BSC task is contained in the communications specifications block (CSB). The System/36 initiator (see *Command Processor* 2) builds the CSB according to the following input:

- Communications configuration record
- The commands associated with the BSC-invoking procedure
- The parameters passed in the BSC-invoking procedure

The batch BSC subfunction performs the following:

- Open and initialize to establish line control — Chart 9.1.1
- Put user data to line — Chart 9.1.2.1
- Get user data from line — Chart 9.1.2.2
- Process waits, posts, op ends, and error recovery — Chart 9.1.3
- Close and terminate — Chart 9.1.4

Open and initialize processing includes:

1 Open the BSC DTFs and alter them based on user-specified parameters in the ALTERCOM procedure.

2 Open the CSB (communications specification block) and assign the line buffers in SQS, or SWS if greater than 2K bytes and line work area (in SQS).

3 If an active DTF is in process, call BSC close.

4 Format the I/O area into IOBs, line buffers, and switched IDs.

5 Enable the communications adapter (by calling control storage BSC code to bring up data terminal ready and dataset ready).

6 Initialize the lines by sending or receiving enquiry.

7 Start communications on the BSC line(s).

8 Route for any required error messages.

Batch BSC Interrupt Handler Region Map

The following table shows the locations of key data areas associated with the batch BSC interrupt handler:

Dump Address (in Hex)	Contents
0800	Nucleus
1000	BSC transients
2000	Batch BSC interrupt handler (#BSCM)
8000	First line buffers (if greater than 2K bytes)
A000	Second line buffers (if greater than 2K bytes)
C000	Third line buffers (if greater than 2K bytes)
E000	Fourth line buffers (if greater than 2K bytes)

Batch BSC Data Management Region Map

The following table shows the locations of key data areas associated with batch BSC data management:

Dump Address (in Hex)	Contents
0800	Nucleus
1000	Batch BSC data management subroutines (#BSDB)
2000	User DTF or user buffer
8000	First line buffers (if greater than 2K bytes)
A000	Second line buffers (if greater than 2K bytes)
C000	Third line buffers (if greater than 2K bytes)
E000	Fourth line buffers (if greater than 2K bytes)

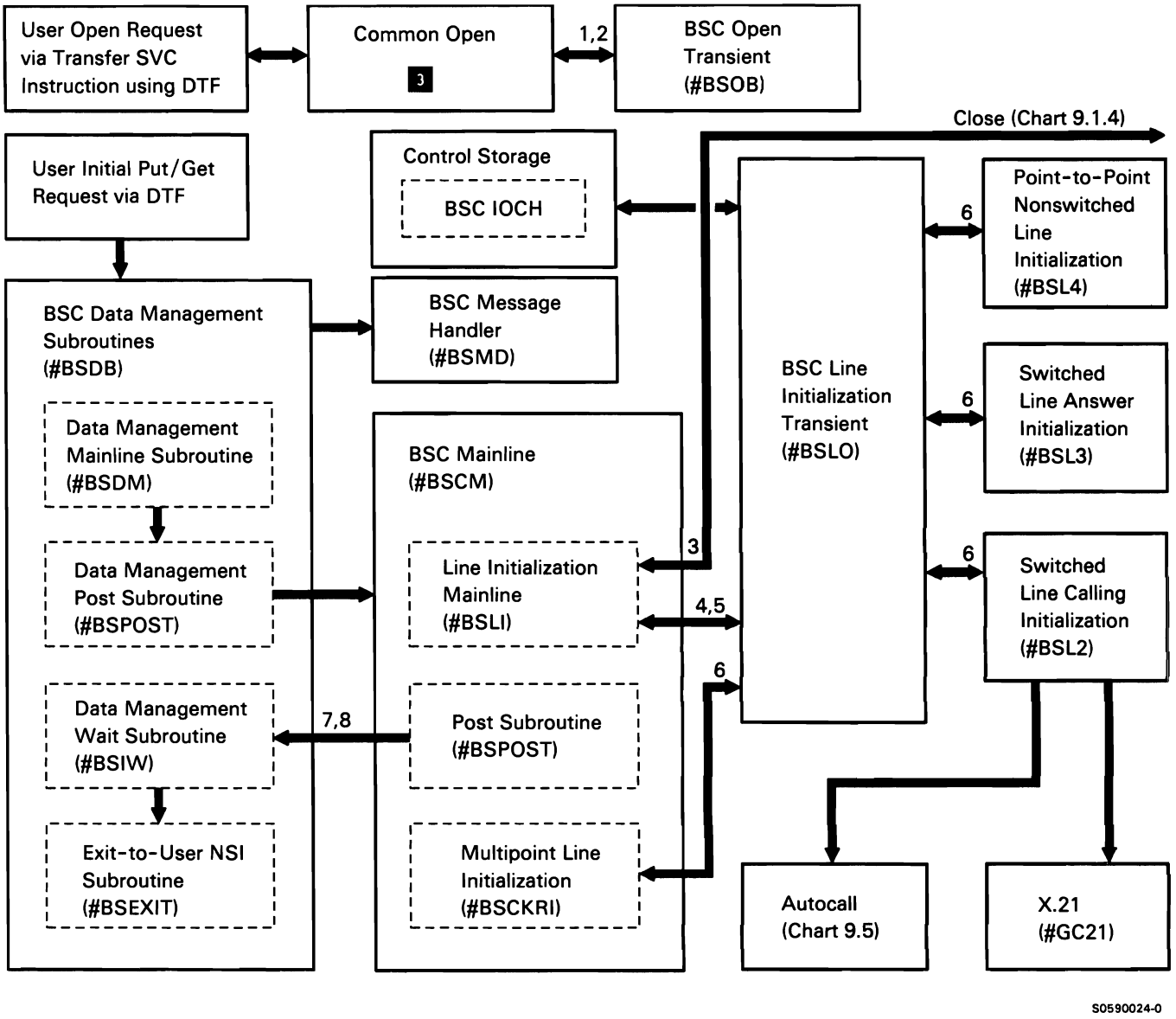


Chart 9.1.1 BSC Open and Initialize Control Flow

Put-user-data-to-line processing includes:

1 Wait until previous transfer completes, then check IOB completion code.

2 Determine from DTF the operation request and route for the following:

3 Move data, by record, from user's logical buffer to an available line buffer and format (block if specified) for line transmission.

4 Insert start of text (STX) and record separator (if specified) control characters and compress or truncate blanks if specified.

5 Insert end-of-text block (ETB) and end-of-text block (ETX) for last record line control characters.

6 Transmit data, by block, to the receiving station.

7 Exit to requesting task.

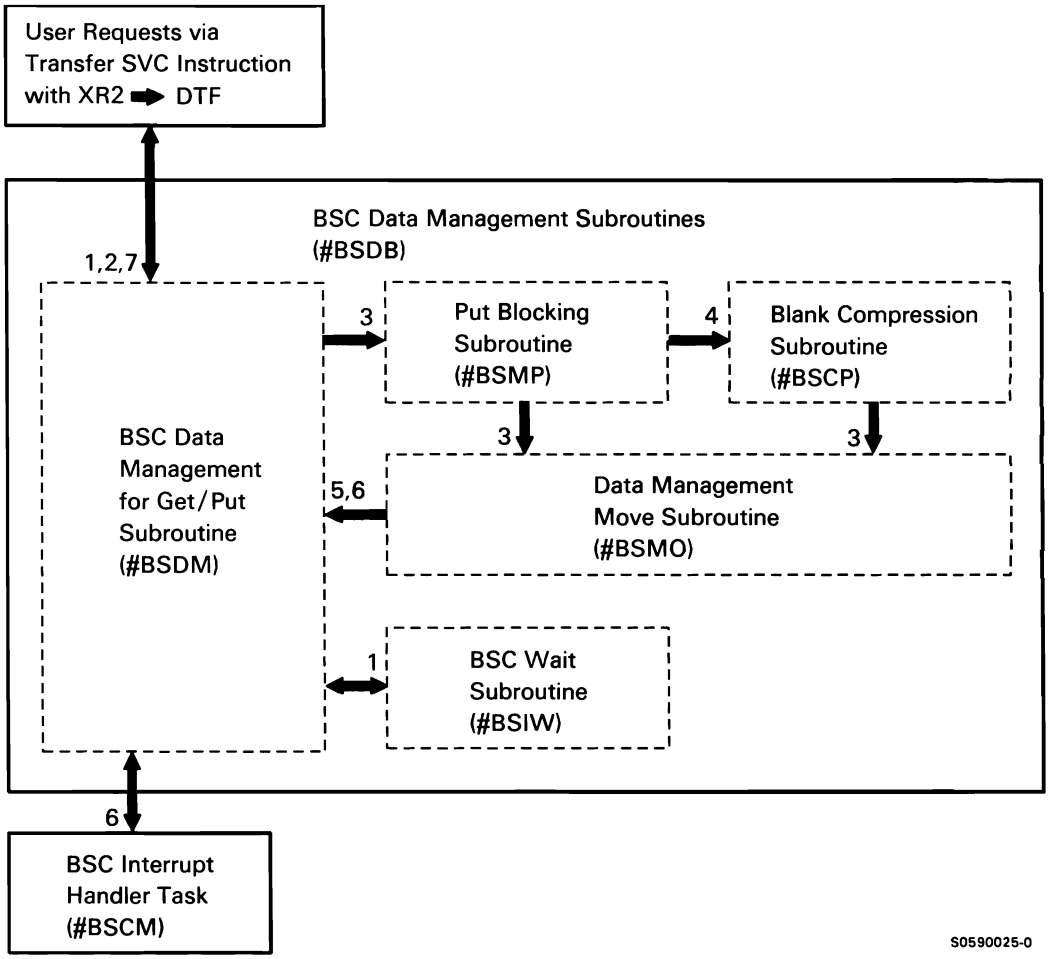


Chart 9.1.2.1 BSC Put-User-Data-to-Line Control Flow

Get-user-data-from-line processing includes:

1 Determine from DTF the operation request and route for the following:

2 Wait until data transfer is completed, then check IOB completion code.

3 Strip off control characters, move data (by deblocked record) from line buffer to user's logical buffer (inserting blanks), and format for receive.

4 Exit to requesting task.

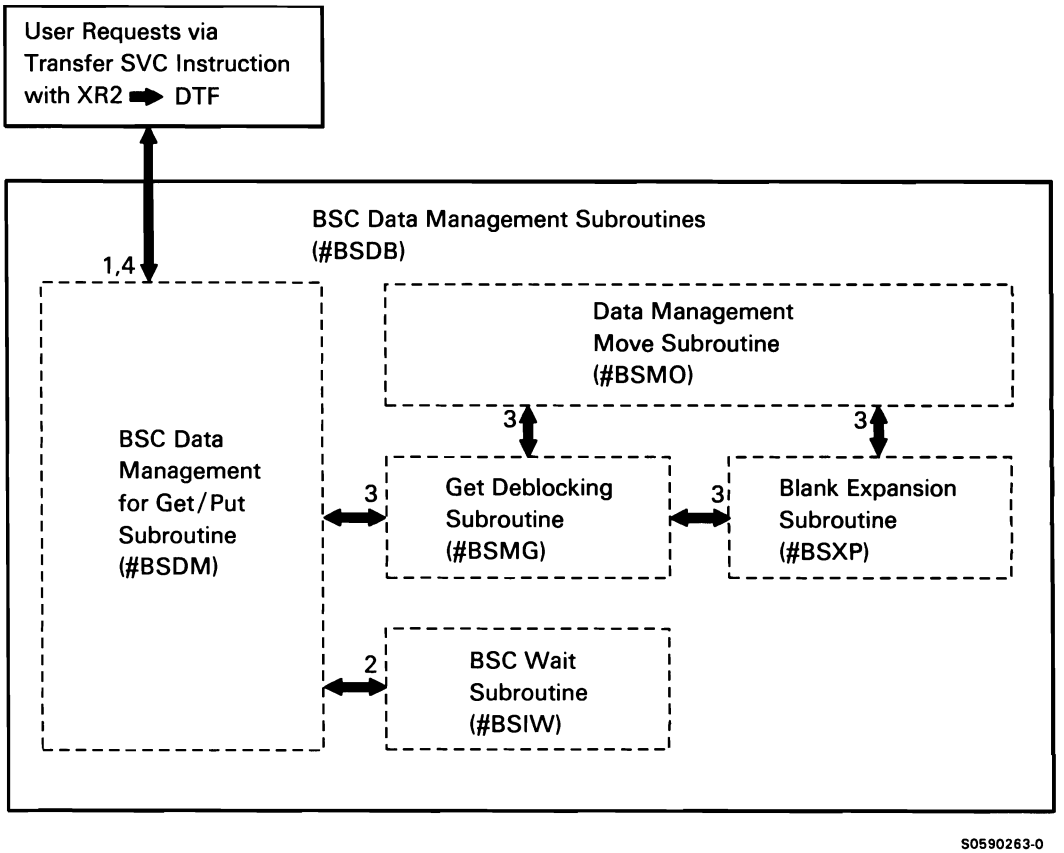


Chart 9.1.2.2 BSC Get-User-Data-from-Line Control Flow

Wait, post, op-end, and error recovery processing includes:

1 Wait for data transfer and check IOB completion code.

2 Check all BSC operations for errors and perform any required error routing; if unrecoverable (permanent) error occurs when user requests BSC close, disable the line by calling control storage BSC code to drop data terminal ready and dataset ready.

3 Check for line buffer ready-for-data-transfer, and call control storage BSC code as required.

4 If multipoint, respond negatively to host select or polls, while waiting for user.

5 If trace active, trace each op-end from control storage.

6 Issue required error messages.

7 Log permanent errors to system ERAP table.

8 If RPG II application program, return user-specified modification of return code. If error, set permanent error indicator.

9 If assembler language application program, issue return code in DTF.

10 Process user end-of-job or close request.

11 Post BSC data management task.

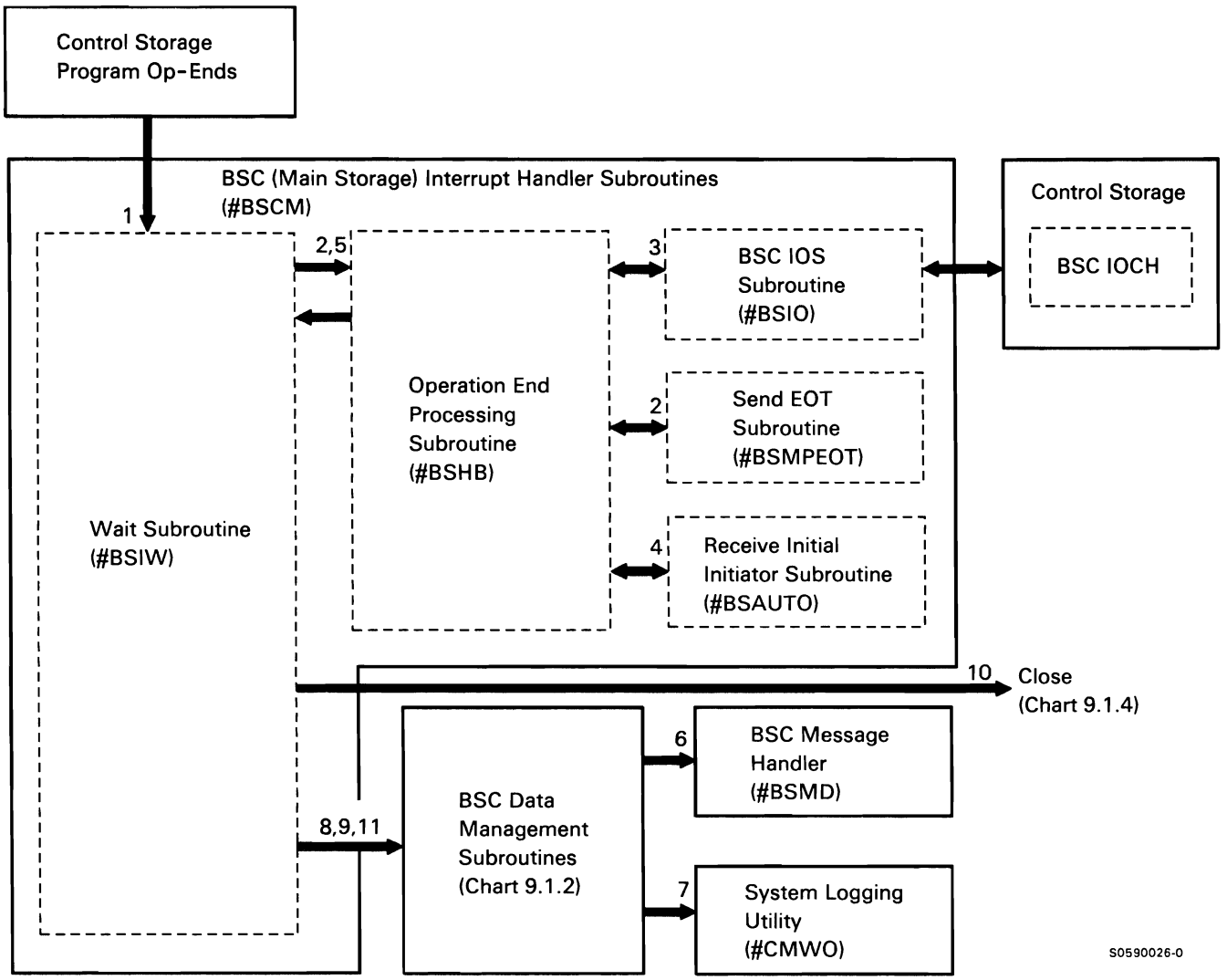


Chart 9.1.3 BSC Op-end, Wait, Post, and Error Recovery Control Flow

Close and terminate includes:

Note: Items 5 through 8, below, are performed by #BSCL if normal termination or #BSER if abnormal termination.

1 Route BSC close requests to BSC data management.

2 Post the interrupt handler with the close or terminate request.

3 Route close request to BSC close transient.

4 If close request, complete processing of any put-file data still in I/O buffers and reset open flags in DTF.

5 Disable communications line by calling control storage code to drop data terminal ready and dataset ready.

6 In ERAP, log text sent, text received, and error counters accumulated.

7 Free I/O areas and line work area.

8 If last BSC user, terminate task.

9 For abnormal termination, also log error and display messages to BSC users.

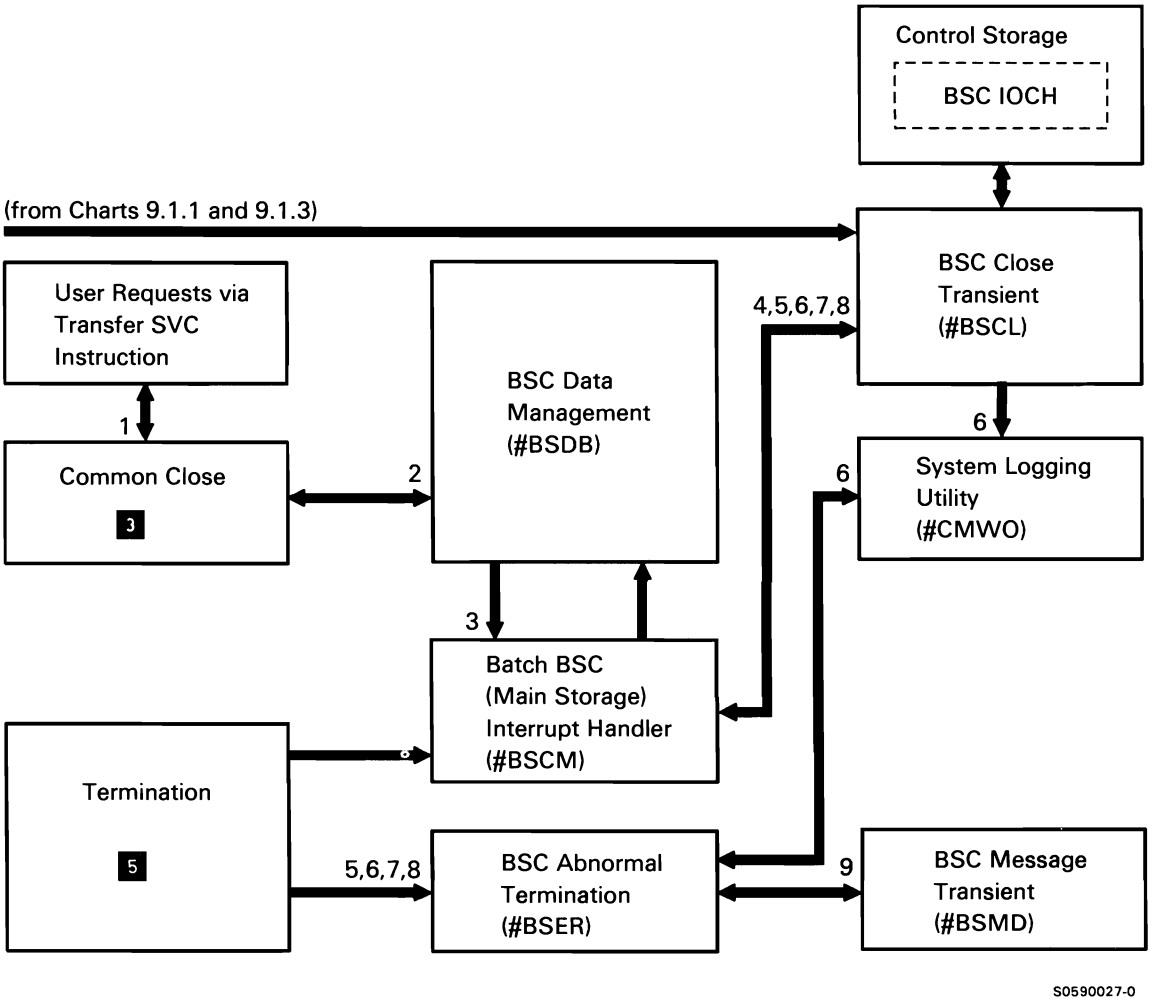


Chart 9.1.4 BSC Close and Terminate Control Flow

SYNCHRONOUS DATA LINK CONTROL (SDLC) SUBFUNCTION

Synchronous data link control (SDLC) is a communications protocol designed to manage information transfer between buffered stations, on a data transmission link that has a centralized control. The SDLC is the standard line control discipline for the systems network architecture (SNA).

System/36 can exist in an SDLC data link as both a primary and a secondary station (on separate lines). As a primary station, System/36 sends commands to, and receives responses from, its secondary stations. As a secondary station, System/36 responds to the commands sent from the primary station.

Unlike the BSC protocol which can stand alone given a user-program-DTF interface, the SDLC protocol operates one programming layer below, at the IOB interface level. This means the user application should be interfacing (at the DTF level) to the applicable SNA task, to printer data management or work station data management in the case of remote work stations, or to a user-provided, equivalent task.

All communication from SDLC tasks to control storage is done via IOBs on the line and transmit queues. The control storage SDLC microcode removes IOBs from the line queues and posts them to the SDLC tasks when the requested operations are completed.

The synchronous data link control subfunction also includes the application program interface to X.25 and support of the LAN. The SNA to data link control (DLC) resident interface (#SDRI) receives IOBs from application programs, determines whether the IOBs are intended for SDLC, X.25, or LAN, then routes them via post-task-by-ID SVC instructions to the appropriate data link control module. This control flow is not shown in Chart 9.2.0, but is shown in Chart 9.2.3.

Note: All charts describing the SDLC components use the general term 'Applicable SNA task' to refer to other SSP components running above SDLC.

The following SNA tasks are users of primary SDLC:

- Remote work station support (Chart 9.6.0)
- SSP-ICF Finance subsystem 10
- SSP-ICF Peer subsystem 10
- Link station test (Chart 9.7)
- C/SNA 10 (for the APPC subsystem)

The following SNA tasks are users of secondary SDLC:

- SSP-ICF Peer subsystem 10
- C/SNA 10 (handles SDLC requests for the MSRJE subsystem, the SNUF subsystem, the APPC subsystem, and the SNA 3270 device emulation subsystem)

Chart 9.2.0 shows the overview control flow for SDLC:

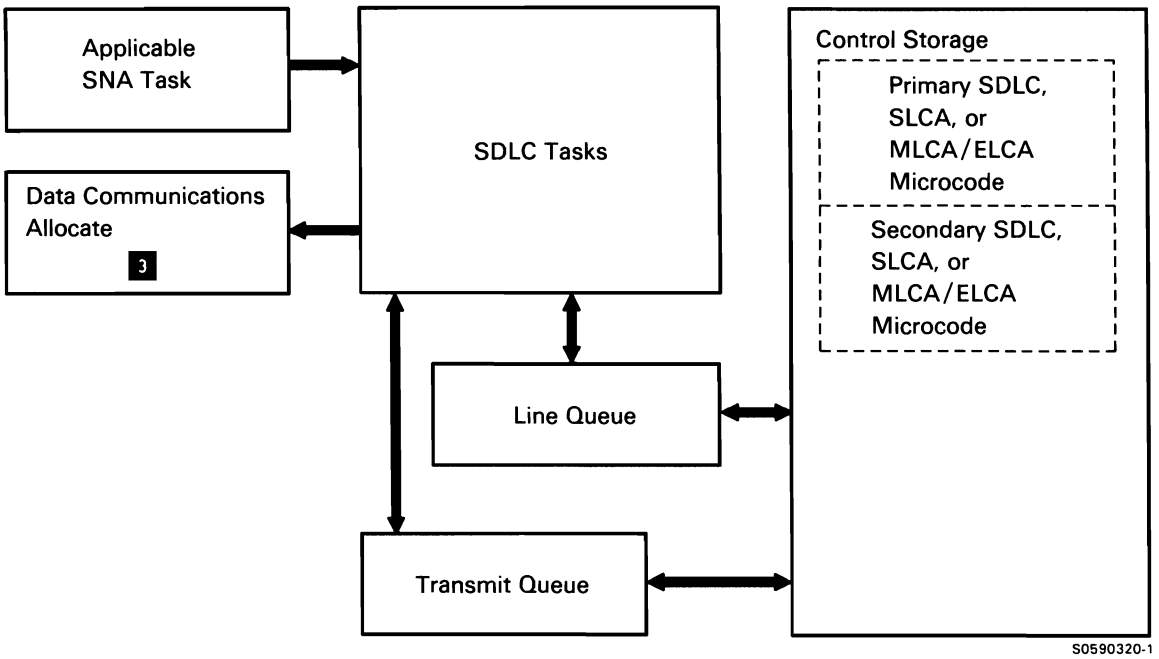


Chart 9.2.0 SDLC Overview Control Flow

SDLC uses four data areas in processing IOBs:

There is a *communications specification block (CSB)* for each line used by SDLC. Each CSB has a CSB extension called the common area that contains status information for the respective line.

The *poll list* contains entries for each of the stations with which SDLC is communicating. It is pointed to by the CSB. The poll list contains one *group* or control entry for each SNA using the line. The group entries are chained together. Following each group, is one or more station entries, each of which describes the status and contains the transmit queue for a remote station.

When an SNA task issues the line open operation, SDLC builds new group and station entries in the poll list. When mainline SDLC issues a start auto-poll or auto response command, the control storage SDLC microcode begins monitoring fields in the poll list that indicate whether to slow poll, normal poll, stop scanning the poll list (transmit IOB found), or continue monitoring the poll list.

The *transmit queue*, contained within the poll list, holds IOBs to be transmitted and IOBs that have been transmitted but not yet verified by Nr-Ns counts. There is one transmit queue chained to each station entry in the poll list. IOBs are moved to/from the transmit queue by the SNA-to-SDLC interface and by mainline SDLC.

The *line queue* contains all IOBs issued to the SDLC control storage microcode. The line queues (one per line) are contained in SCA and are pointed to by queue headers.

Note: An IOB on the transmit or line queue causes control flow to or from control storage, as shown in Chart 9.2.0. In subsequent SDLC control flow charts, these queues and the poll list (which contains the transmit queue) are shown receiving and passing control; this is done to simplify the control flow chart and should in no way imply that the queues contain free-standing, executable code.

SDLC consists of four basic parts:

• SDLC initialization and termination subtask	Charts 9.2.1.1 and 9.2.1.2
• Mainline SDLC subtask	Charts 9.2.2.1 and 9.2.2.2
• SNA-to-data link control interface	Chart 9.2.3
• SDLC abnormal termination routines	Chart 9.2.4

SDLC Initialization and Termination Subtask

The SDLC initialization/termination subtask performs SDLC initialization and termination processing, as well as message handling for SDLC. It is a swappable subtask that performs both SDLC primary and SDLC secondary processing.

The following SDLC intialization processes are shown in Chart 9.2.1.1:

1 Perform initial processing of open request:

- Get resident SNA-to-SDLC interface loaded (#SDRI contains code that loads the resident interface routine).
- Attach mainline SDLC task as a subtask.
- Build poll list group entry and post mainline task to add group to poll list.

2 Allocate line and CSB, and assign poll list and IOBs.

3 Return SDLC line open parameter list complete.

4 Process transmit IOB containing enable command.

5 If switched line, issue manual call or manual answer message.

6 Process enable command, returning data terminal ready and data set ready.

7 If data set ready not on, (data set not ready), issue message.

8 If autocall, establish autocall connection and post task.

9 Process enable post from initialization/termination by initializing fields in the poll list.

10 Return enabling IOB, indicating operation is complete.

11 Process receive IOBs and start-autopoll IOB (control storage processes these IOBs placed on line queue by #SDLC).

12 Process transmit IOB containing open station command (primary SDLC only) and initialize station entry in the poll list.

13 Process returned IOB containing open station command complete.

14 *At any time in above sequence*, process errors requiring abnormal termination.

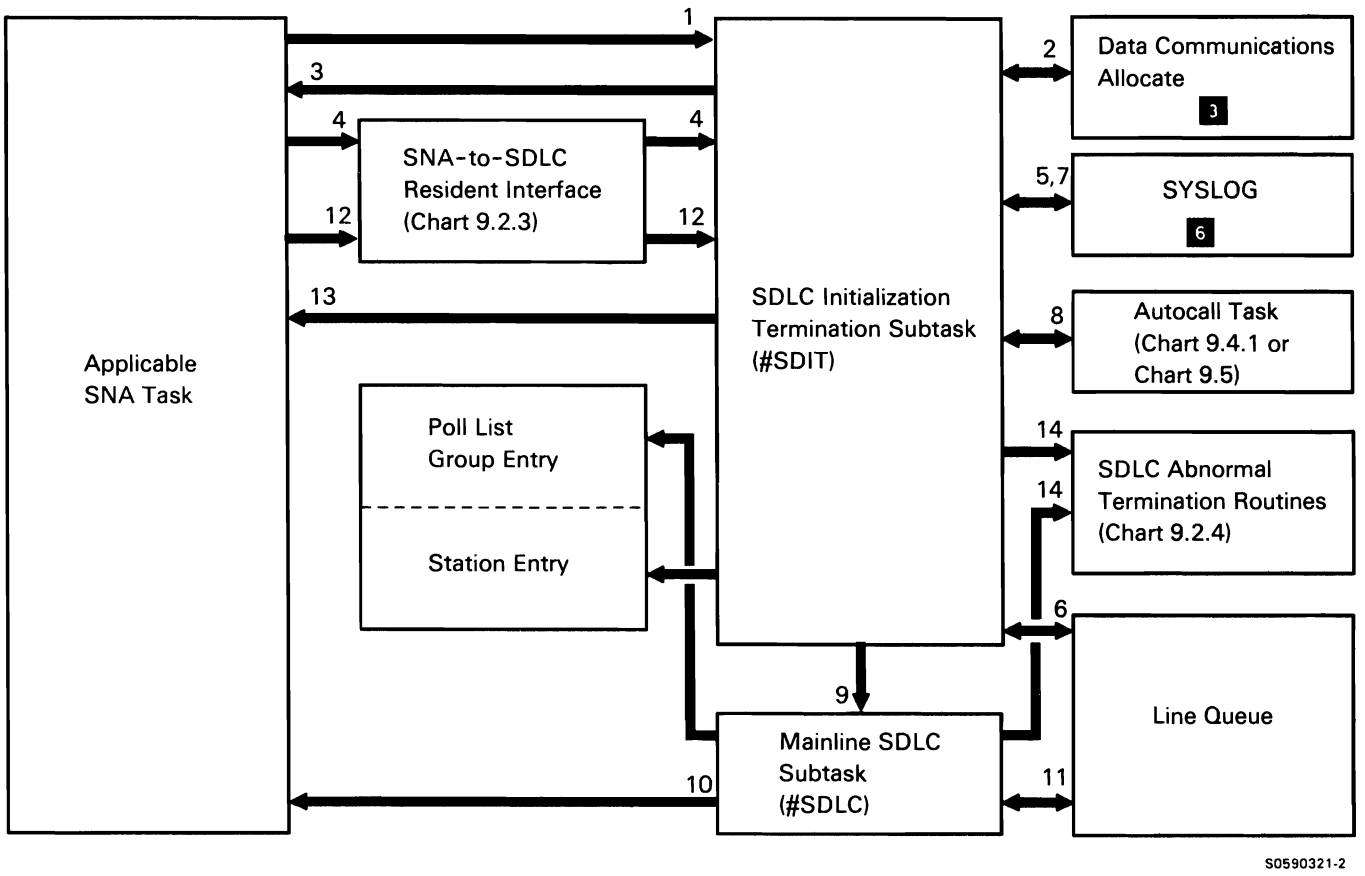


Chart 9.2.1.1 SDLC Initialization Control Flow

The following SDLC termination processes are shown in event sequence in Chart 9.2.1.2:

1 Process terminate-detach IOB command.

2 Stop auto-poll.

3 Stop auto-poll posted complete followed by start auto-poll complete.

4 Remove all transmit IOBs from the transmit queue and return them to the SNA task with completion codes of request ignored.

5 Remove poll list group from the queue and queue it to the terminate IOB.

6 Give terminate IOB to initialization/termination task.

7 Perform disable processing:

- Disable adapter.
- Clear receive IOBs from the line queue.
- If permanent handware error, put wrap test IOB on line queue and wait for its completion.

8 Where appropriate issue message based on wrap test results.

9 Process terminate/detach IOB returned from SDLC.

10 Perform any required error logging.

11 Deallocate line.

12 *At any time in above sequence*, process errors requiring abnormal termination.

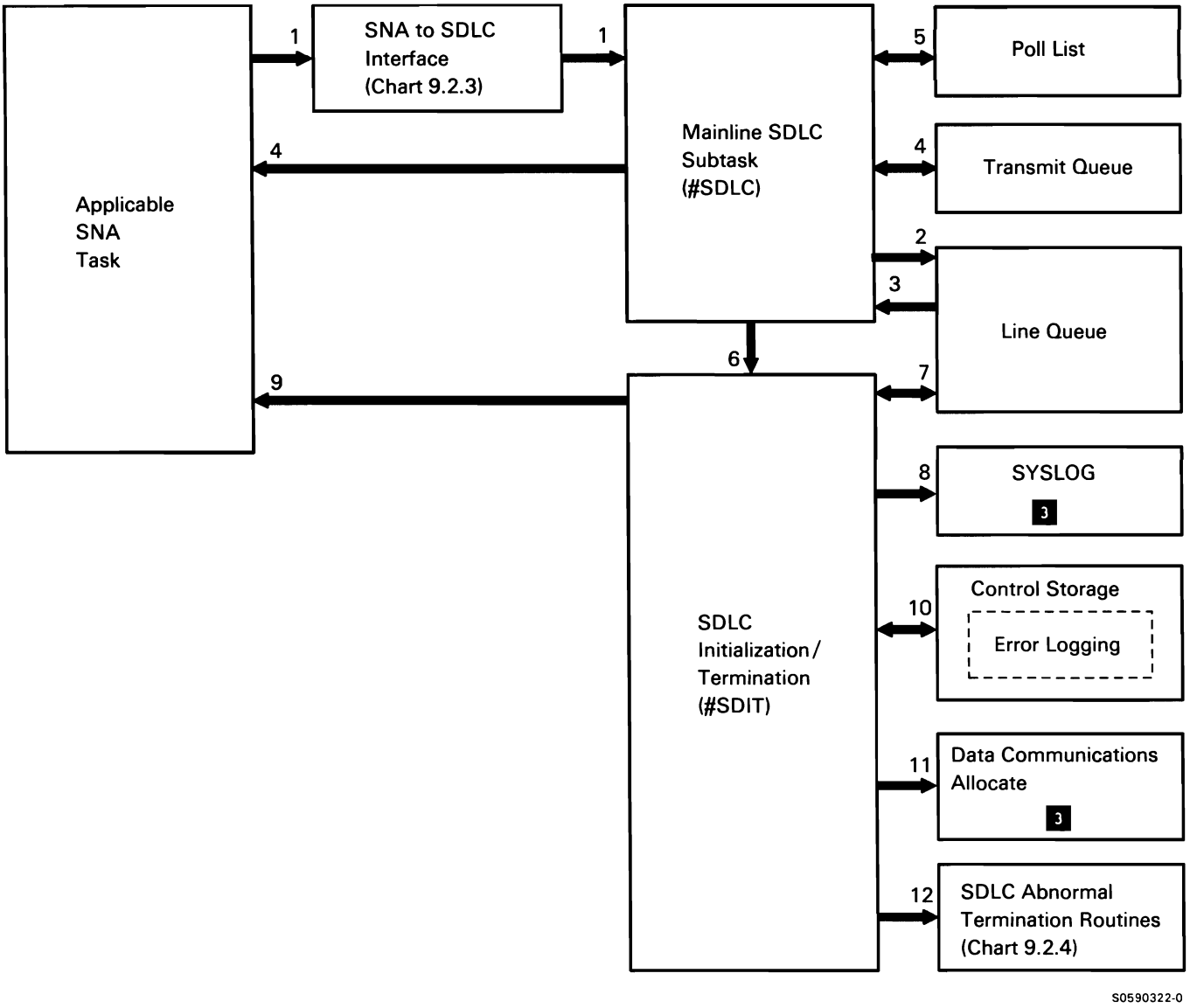


Chart 9.2.1.2 SDLC Termination Control Flow

Mainline SDLC Subtask

The mainline SDLC subtask performs transmit and receive operations for the requesting task. It is a resident nonswappable subtask that performs both SDLC primary and SDLC secondary processing.

The following mainline SDLC subtask receive processes are shown in event sequence in Chart 9.2.2.1:

1 Process receive IOB posted complete (possibly with an error) by control storage.

2 Post receive IOB with data (or with permanent hardware error).

3 Return receive IOB to line queue.

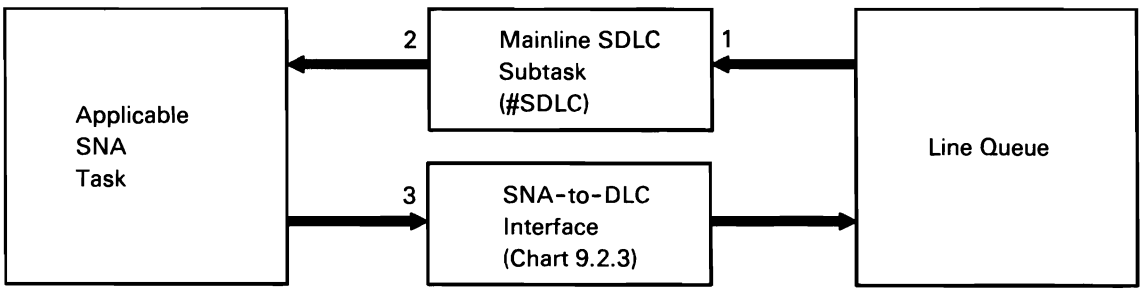


Chart 9.2.2.1 Mainline SDLC Subtask Receive Control Flow

The following mainline SDLC subtask transmit processes are shown in event sequence in Chart 9.2.2.2:

1 Queue transmit IOB to transmit queue.

2 Process start-auto poll complete post (internal IOB) from microcode.

3 Put transmit IOB on line queue.

4 After response is received from remote system or controller, process transmit IOB complete post from microcode by verifying, removing it from the transmit queue, and returning it to the SNA task.

5 If required, log hardware errors returned by SDLC microcode.

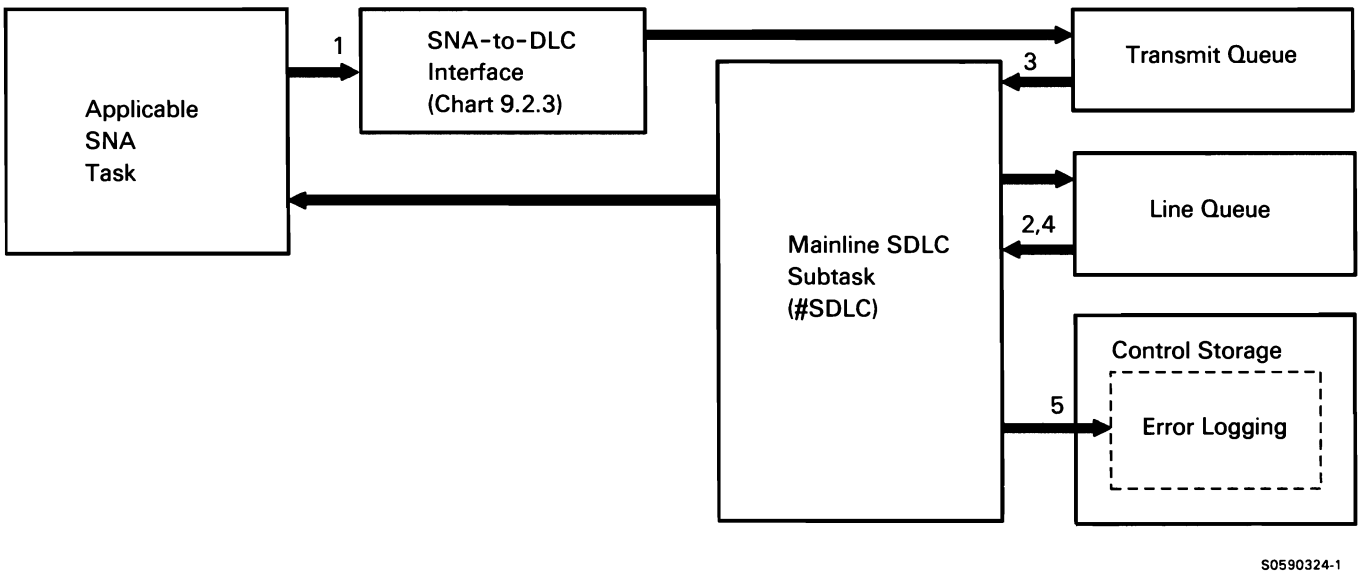


Chart 9.2.2.2 Mainline SDLC Subtask Control Flow

This page is intentionally left blank.

SNA-to-DLC Interface

The SNA-to-DLC interface routes all requests from applicable SNA tasks to one of the data link controls. It receives IOBs from SNA tasks, determines whether the IOBs are intended for SDLC, X.25, or support for the local area network (LAN), then routes them via post-task-by-ID, communications IOCH, IOCH for the LAN, or queue SVC instructions to the appropriate data link control module. Because it routes all X.25 IOBs directly to X.25 mainline (#XTML) regardless of the function requested, this topic describes control flow for SDLC and support for the LAN only.

SNA initiates communication with SDLC or support for the LAN by issuing an SDLC line open parameter list. The parameter list is posted to the initialization/ termination task, #SDIT or #LNIT, which calls SNA-to-DLC interface, #SDRI. #SDRI loads a section of code called the *Resident Interface* which remains in main storage as long as SDLC or support for the LAN is active. All IOBs issued by the SNA are given to the resident interface. The resident interface routes the IOBs to the appropriate function via post-task-by-ID, queue, communications IOCH, or IOCH for the LAN SVC instructions.

SNA-to-SDLC Interface Posting: The SNA-to-DLC interface (#SDRI) posts various components dependent on the type of IOB issued and the conditions under which the IOB was issued. The following chart summarizes the posting variations:

IOB, IOB Operation Code, or Completion Code	SNA Task Posted	#SDLC Posted	#SDIT Posted
Unrecognized operation	X		
Invalid write		X	
No poll list entry found	X		
SNRM ERROR RESET		X	
DISC		X	
TERMINATE or TERMINATE/ DETACH		X	
No poll list segment found	X		
Internal IOBs		X	
ENABLE and OPEN station			X
Station not operational		X	
Receive IOBs with permanent error		X	
Receive IOBs without line enabled		X	

SNA-to-LAN Interface Posting: The SNA-to-DLC interface (#SDRI) posts various components dependent on the type of IOB issued and the conditions under which the IOB was issued. The following chart summarizes the posting variations:

IOB, IOB Operation Code, or Completion Code	SNA Task Posted	IOB Queued to Line Queue	#LNIT Posted
Unrecognized operation	X		
No SNA list found	X		
ENABLE is in process			X
Permanent link (network) error			X
ENABLE IOB issued by SNA			X
TERMINATE or TERMINATE/ DETACH			X
OPEN station			X
Read IOB (ordinary)		X	
Read IOB (internal)			X (queued to CSB)
SNRM, DISC/ REQDISC, SNRM ERROR RESET		X	
XID and Test			X
Ordinary transmit		X	

The following SNA-to-SDLC interface processes are shown in event sequence in Chart 9.2.3:

1 Process all IOBs from SNA:

- If invalid request, return it immediately.
- If X.25 request, post it to X.25 mainline task.
- If SDLC request, route appropriately.

2 Process posted internal IOBs and the following posted requests:

- Receive IOBs with errors.
- Disconnect/Request Disconnect.
- Terminate/Detach.
- SNRM Error Reset.

3 Process all X.25 IOBs.

4 Process normal transmit IOBs (SNRM, XID, TEST, or WRITE opcodes) queued by #SDRI.

5 Process normal receive IOBs placed on queue by #SDRI via the communications IOCH SVC instruction.

6 Process enable and open station SNA IOB opcodes.

7 Process the following:

- Enable and open station SNA IOB opcodes for LAN.
- Disconnect/Request Disconnect.
- Terminate.
- Terminate/Detach.
- Error return code from IOCH.
- Pended XIDs from IOCH.

8 Process normal transmit IOBs for LAN.

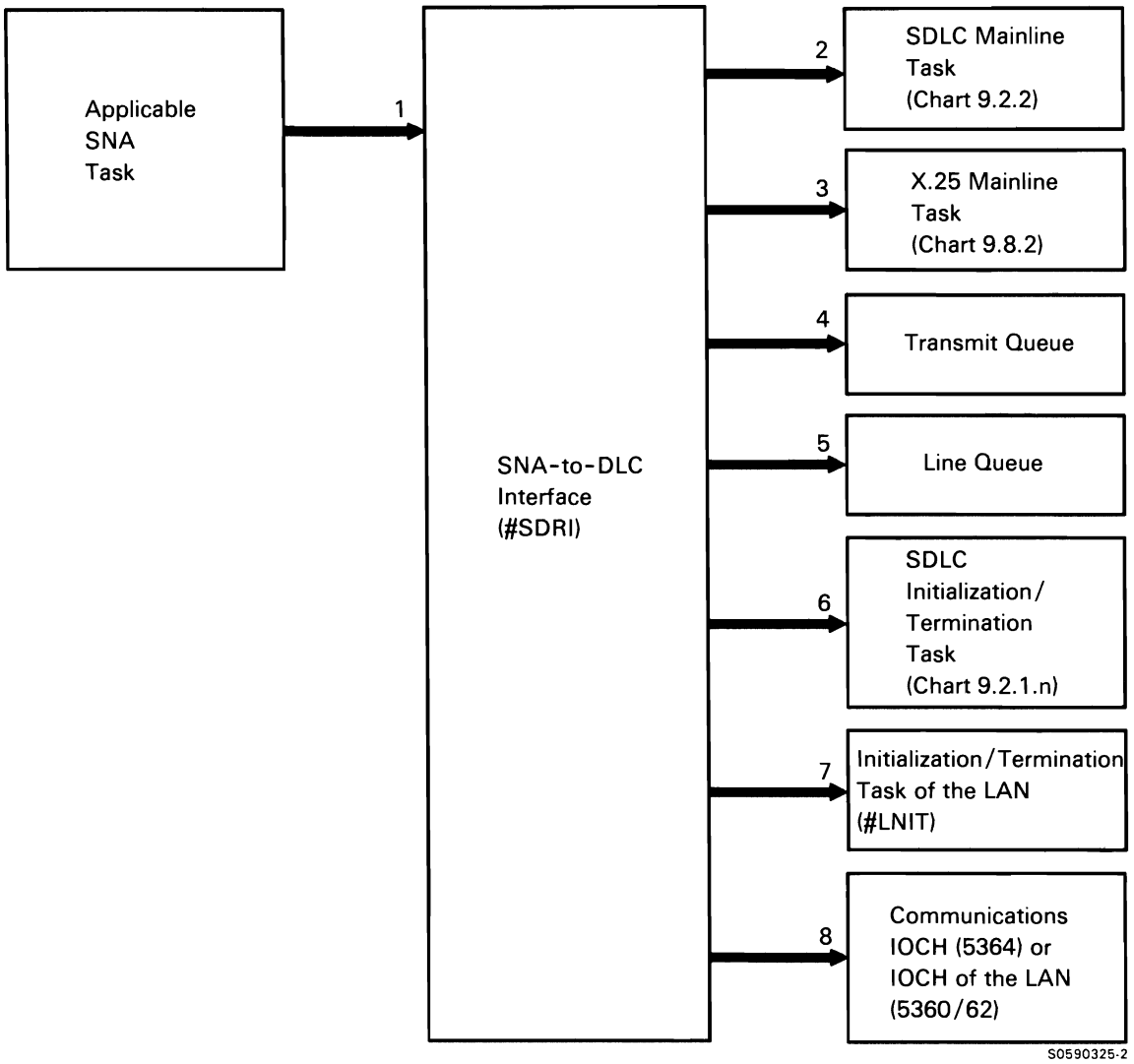


Chart 9.2.3 SNA-to-DLC Interface Control Flow

SDLC Abnormal Termination

SDLC abnormal termination receives control for the following errors:

- MLCA/ELCA controller check
- 2K-page failure
- SDLC task abnormal termination

Abnormal termination can be evoked from SDLC initialization/termination or from SDLC mainline. If SDLC mainline (#SDLC) encounters or detects an error, control storage termination loads abnormal termination for mainline (#SDAM) over SDLC mainline. Similarly, if SDLC initialization/termination (#SDIT) encounters or detects an error, control storage termination loads abnormal termination for initialization/termination (#SDAT). Whichever abnormal termination routine gets control initially, it notifies the other, normally running task, and that task forces (via control storage termination) the loading of its respective abnormal termination routine. In either case, #SDAM has replaced #SDLC and #SDAT has replaced #SDIT before abnormal termination processing begins.

The following abnormal termination processes are shown in Chart 9.2.4:

1 Disable adapter(s) and clear line queue(s).

2 Return any transmit IOBs with abnormal completion codes. If no transmit IOBs exist, post the SNA with an internal IOB indicating abnormal termination condition.

3 Post terminate/detach IOB to #SDAM.

4 Post terminate/detach IOB to #SDAT.

5 Process for terminate/detach.

6 Return terminate/detach IOB and go to end of job.

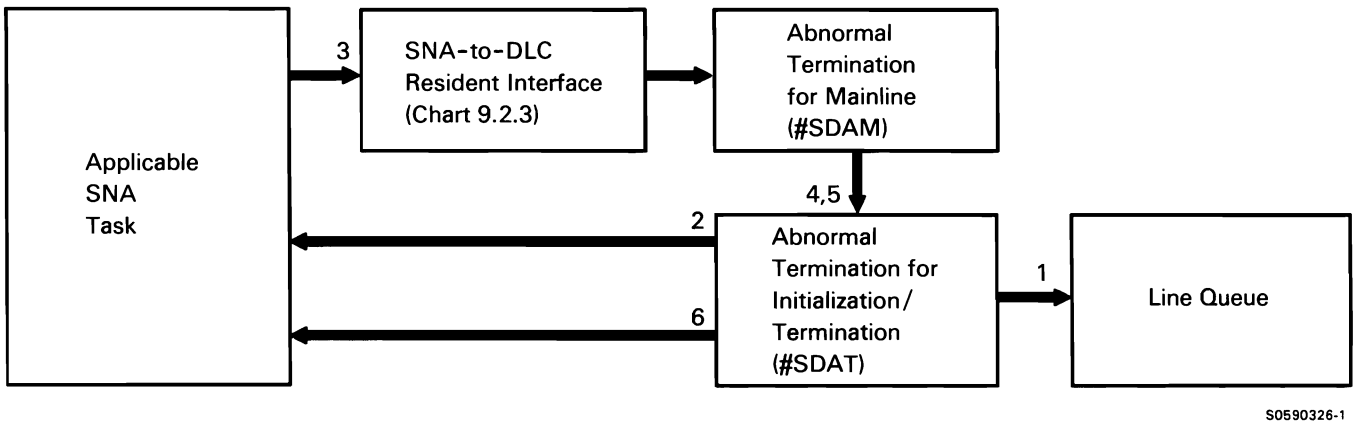


Chart 9.2.4 SDLC Abnormal Termination Transient Control Flow

MULTILINE COMMUNICATIONS ATTACHMENT/ EIGHT LINE COMMUNICATIONS ATTACHMENT (MLCA/ELCA) SUPPORT SUBFUNCTION

Most of the main storage multiline and eight line communications attachment support is coded into the system programs using the MLCA/ELCA. The support makes the detail programming required in maintaining multiline communications transparent to the user. User applications need only specify the line number to use the MLCA/ELCA.

The only SSP modules totally dedicated to the MLCA/ELCA are the MLCA/ELCA controller recovery module and the MLCA/ELCA controller check handler. The controller check handler executes a wrap test to determine the nature and extent of the adapter or microcode problem, then it reports the severity of the problem to all active communications tasks.

The following MLCA/ELCA controller check handler processes are shown in Chart 9.3.1:

1 Attach controller check handler.

2 Load MLCA/ELCA control storage wrap test.

3 Issue wrap test IOB and display message containing system reference code to system console.

4 Post the wrap test results to all data communications link controller tasks that have an active IOB.

5 Reload the MLCA/ELCA control storage I/O control handler programs.

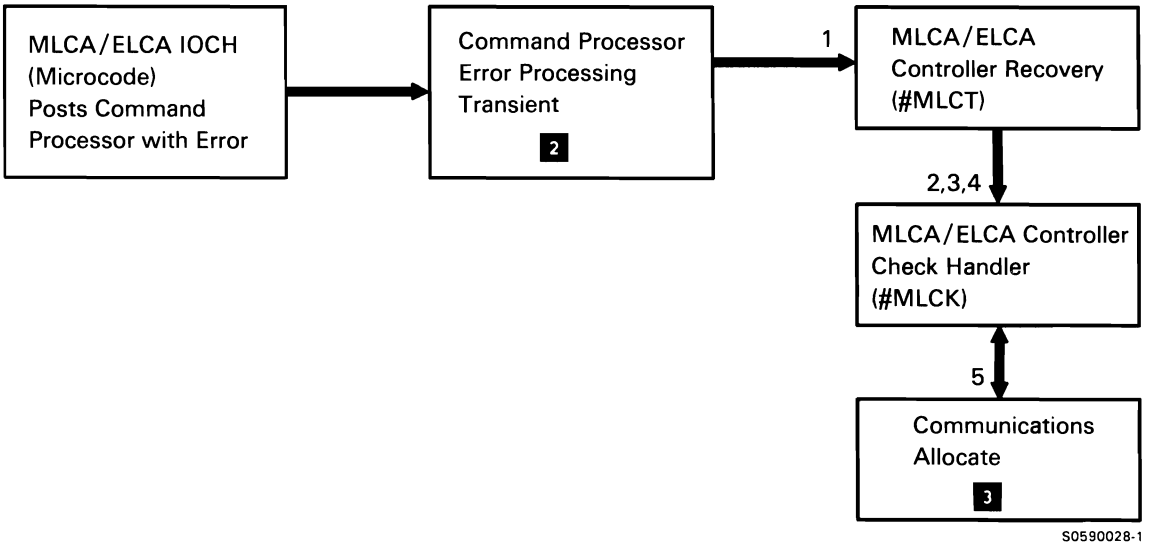


Chart 9.3.1 MLCA/ELCA Controller Check Handling Control Flow

X.21 SUPPORT SUBFUNCTION

X.21 support consists of the SSP data communications programming that permits System/36 to be attached to an X.21 Public Data Network.

The following capabilities are provided:

- Automatic calling of remote DTEs that have subscribed to the X.21 network and status retrieval of each call (via the X.21 network call progress signal).
- Registration for, and cancellation of, user facilities offered by the X.21 Public Data Network.
- Creation of X.21 connection lists that contain the public data network addresses of the remote DTEs to be called.

X.21 support consists of an autocall program (#GC21), an error processing transient (#GCAT), and two utilities (#GCFR and $XNLM) evoked by the REQUESTX and DEFINX21 procedures.

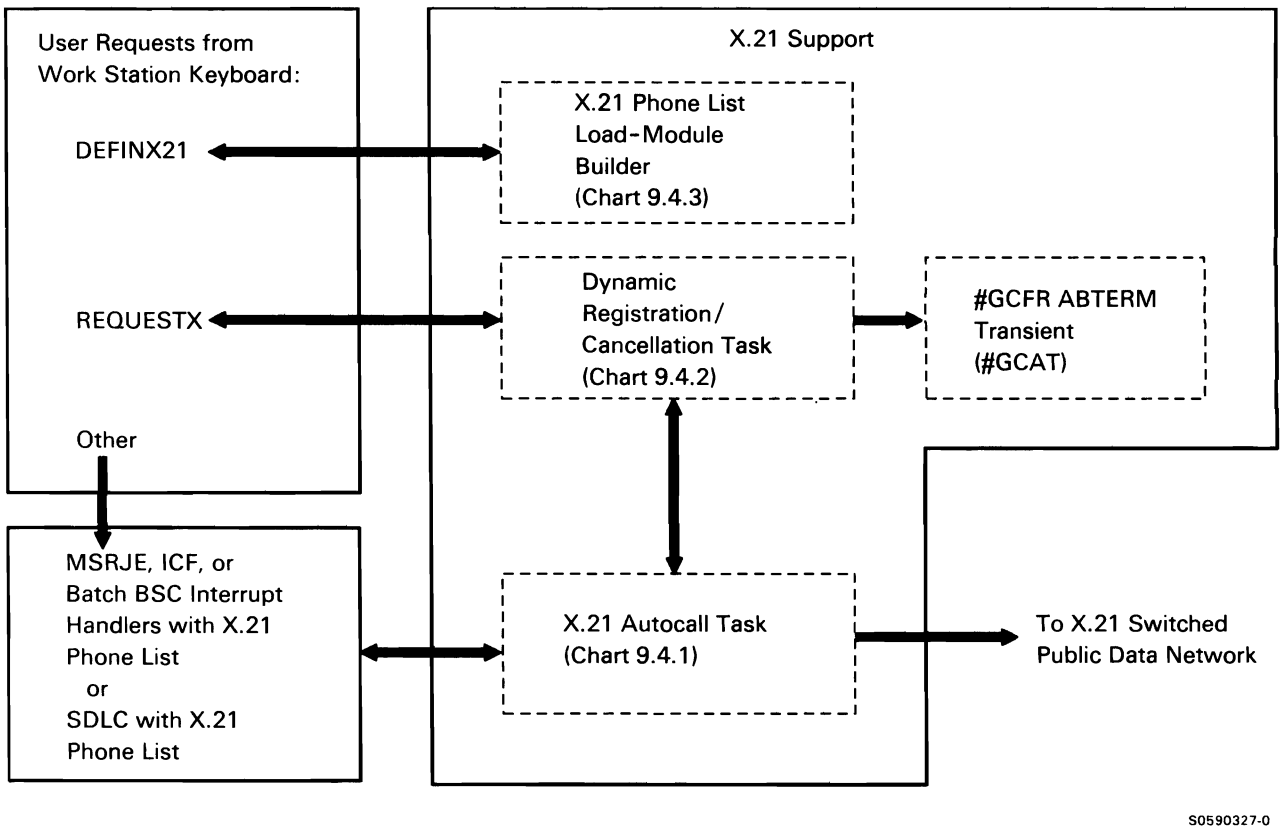


Chart 9.4.0 X.21 Support Overview Control Flow

X.21 Autocall

The X.21 autocall task is the interface between the communications microcode and the protocol link control modules (which function as main storage interrupt handlers) attempting to establish communications in an X.21 circuit-switched Public Data Network. In addition to placing the call, the autocall task also monitors and displays the status of each call. After the data link is established, the X.21 autocall task is swapped to disk until the next call request is issued.

The X.21 autocall task supports up to eight X.21 circuit-switched lines. The link control modules (which function as main storage interrupt handlers) invoke the autocall task by issuing post task with ID SVC instructions with XR1 pointed at the IOB (the IOB points to the connection list).

The following X.21 autocall task processes are shown in Chart 9.4.1:

1 Initialize X.21 communications at IPL.

2 Perform dial requests received from protocol link control modules and maintain status of connection list.

3 Post the caller successful or unsuccessful.

4 Process reset requests by resetting the status of the call and posting the request complete.

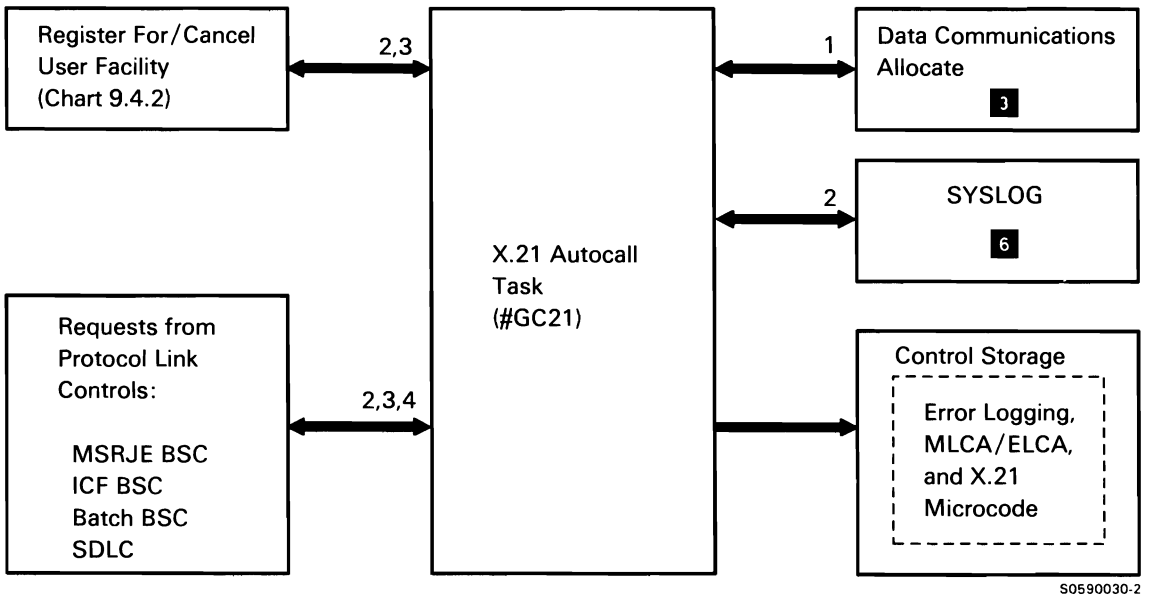


Chart 9.4.1 X.21 Autocall Control Flow

X.21 REQUESTX Utility

The X.21 REQUESTX utility allows an X.21 user to dynamically register for and cancel network facilities provided by the Public Data Network. Depending on the network requirements, some facilities may be registered for at subscription time, while others may be registered for or canceled at the time the service is actually needed. This utility was originally provided for subscribers in Japan using the DDX circuit-switched-network Closed User Group facility. It is expected that this utility will be usable in other countries whose X.21 networks are still in various stages of development.

After the REQUESTX utility is invoked, registration/cancellation sequences can be initiated from a work station or a source member. The sequences are contained in source records that specify the line number on which the requests are to be transmitted. On receiving the request, the REQUESTX utility points XR1 at its IOB and posts the X.21 autocall task to transmit the request. The IOB points to the X.21 connection list containing the facility sequence.

The following REQUESTX utility processes are shown in Chart 9.4.2:

1 Read utility control statements.

2 Initialize for utility:

- Read communications configuration record.
- Assign data areas.
- Print output or issue message.
- Initialize X.21 connection list.

3 Perform source facility requests.

- Read source record from source member or work station device.
- Validate information.
- Allocate and enable line.
- Set up IOB and X.21 connection list.
- Post X.21 autocall task.
- Print output or issue message.

4 Terminate for utility:

- Deallocate and disable line.
- Free utility-assigned data areas.

5 Perform inquiry and cancel functions:

- Deallocate and disable X.21 line(s).
- Free X.21 active data areas.

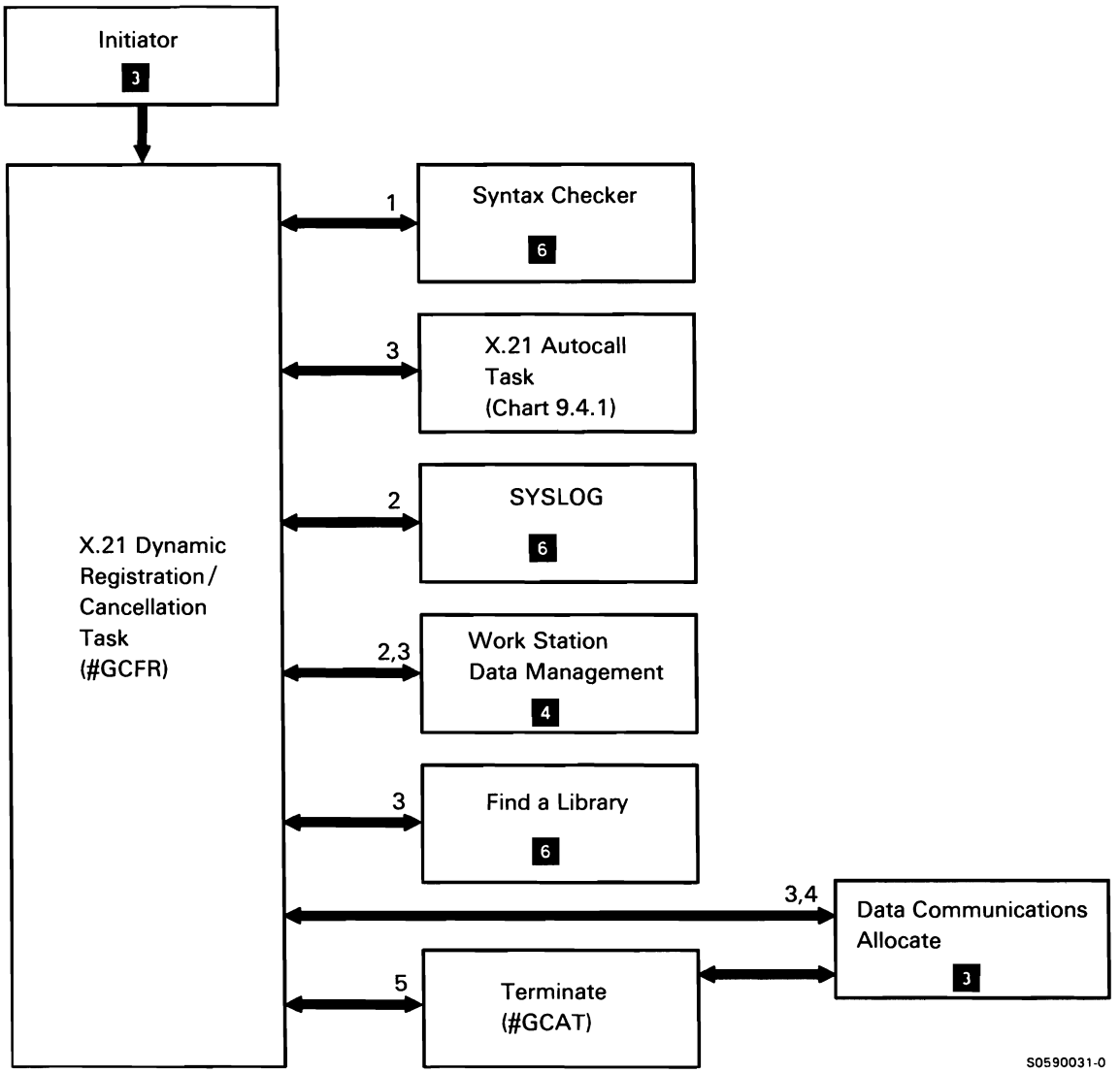


Chart 9.4.2 X.21 REQUESTX Control Flow

X.21 DEFINX21 Utility

The X.21 DEFINX21 utility allows an X.21 user to create and alter connection lists of X.21 Public Data Network numbers or X.21 short hold mode (SHM) line configurations. DEFINX21 is an interactive utility that takes data from the work station and places it in a library as an object member.

The following DEFINX21 utility processes to create or alter an X.21 connection list are shown in Chart 9.4.3:

1 Display prompts and read input.

2 Verify data and find specified library.

3 Build phone list load module on disk.

4 Terminate for utility.

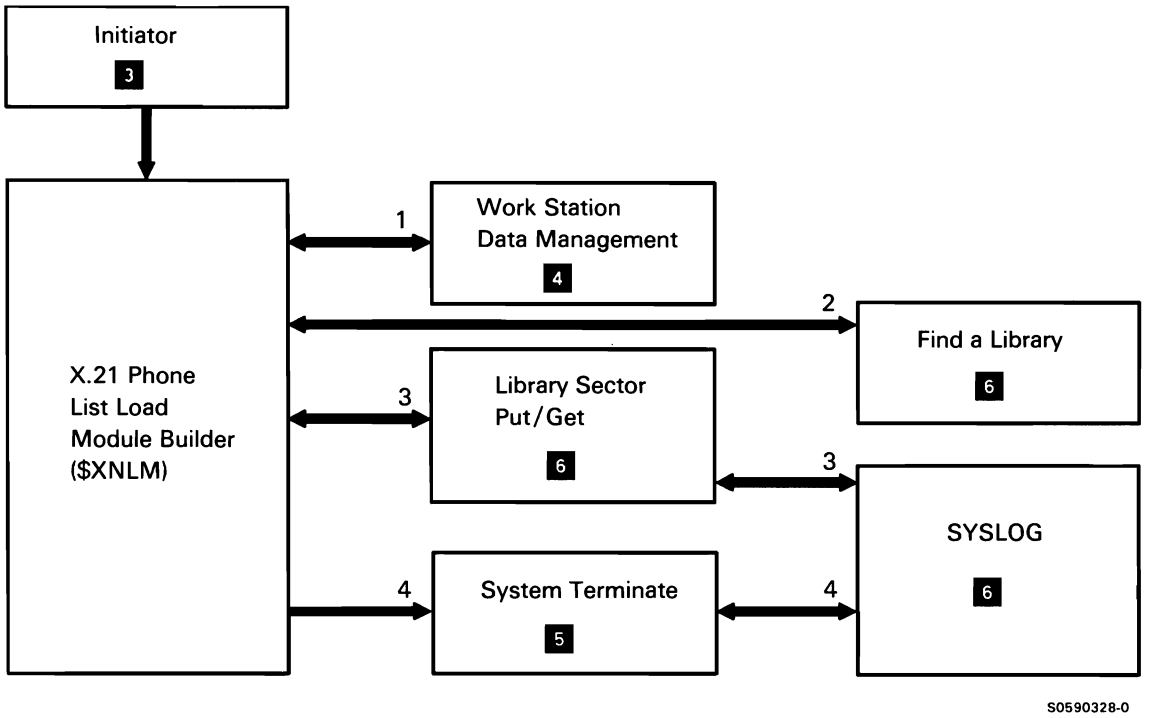


Chart 9.4.3 DEFINX21 Control Flow

X.21 DEFINX21 SHM Line Configuration Utility

The following DEFINX21 utility processes, that create or alter an X.21 SHM line configuration, are shown in Chart 9.4.4:

1 Display prompts and read input.

2 Verify data and find library and member, protect member, and read member.

3 Build X.21 SHM line configuration load module on disk.

4 Print X.21 SHM line configuration.

5 Terminate for utility.

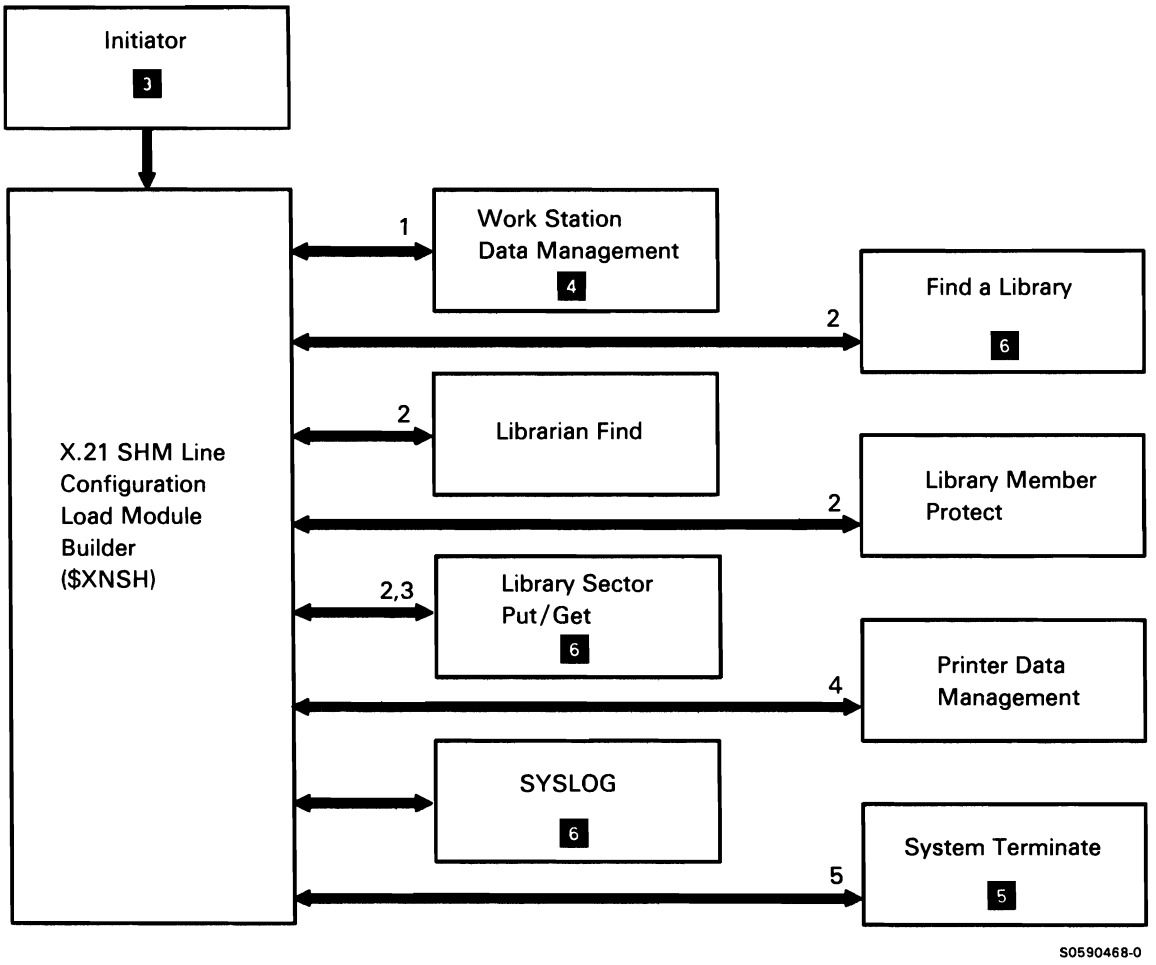


Chart 9.4.4 DEFINX21 SHM Line Configuration Control Flow

This page is intentionally left blank.

AUTOCALL SUPPORT SUBFUNCTION

The autocall feature is an attachment to MLCA/ELCA that allows the user to place calls on a switched line without operator intervention. The numbers to be called are contained in a phone list built by the DEFINEPN procedure. In addition to placing the calls, autocall monitors and updates the status of calls in the phone list.

The following autocall processes are shown in Chart 9.5:

1 Process IOB call request, using IOB-provided phone list pointer, task ID, device code and communications line number.

2 Display results of call (successful or unsuccessful).

3 Post the caller successful or unsuccessful.

4 Process reset request by resetting status of call and posting request to caller.

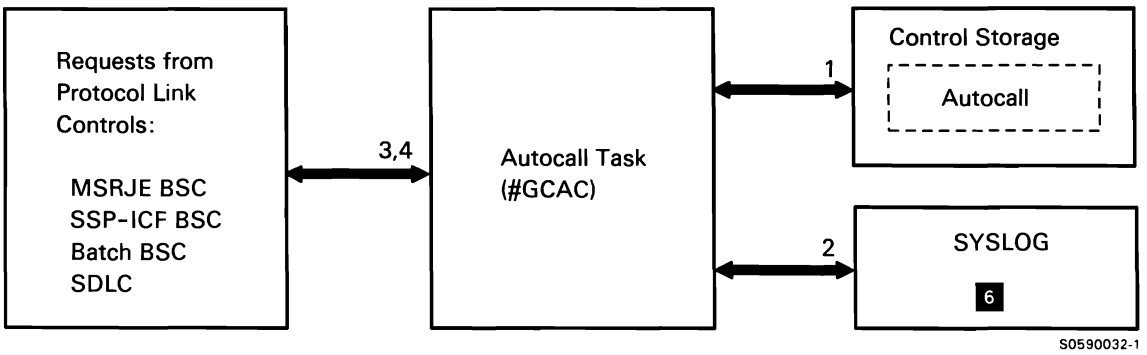


Chart 9.5 Autocall Support Control Flow

REMOTE WORK STATION (RWS) SUPPORT SUBFUNCTION

System/36 remote work station (RWS) support makes work stations attached to the System/36 via SNA/SDLC or X.25 appear the same as locally attached work stations. It allows the user to access remote work stations without having any knowledge of data communications protocols. (For information on RWS protocols, refer to *Appendix A: Data Communications Line Protocols.*)

The RWS support contains its own SNA support and operates as an independent task. The RWS mainline task ensures that proper SNA protocols are followed. Initialization, termination, and error recovery execute as overlays; they receive control via transfer supervisor call instructions. The overlays have addressability to system areas and the first 4K bytes (RWS work area and common subroutines) of the mainline task.

Note: The term *data link control (DLC)* is used throughout this description of the remote work station subfunction to mean either the SDLC or the X.25 data link control. Remote work station support accesses both data link controls via the SNA-to-data-link-control interface (#SDRI).

The following RWS overview processes are shown in Chart 9.6.0:

1 Process RWS Vary-On and Vary-Off commands from system console.

2 Process requests for work at a remote station.

3 Attach or detach SNA/SDLC task.

4 Process DLC IOBs.

5 Attach 3270 remote attach tasks for each 3274 controller.

6 Translate between 3270 and 5250 data stream. Detach 3270 remote attach tasks for each 3274 controller.

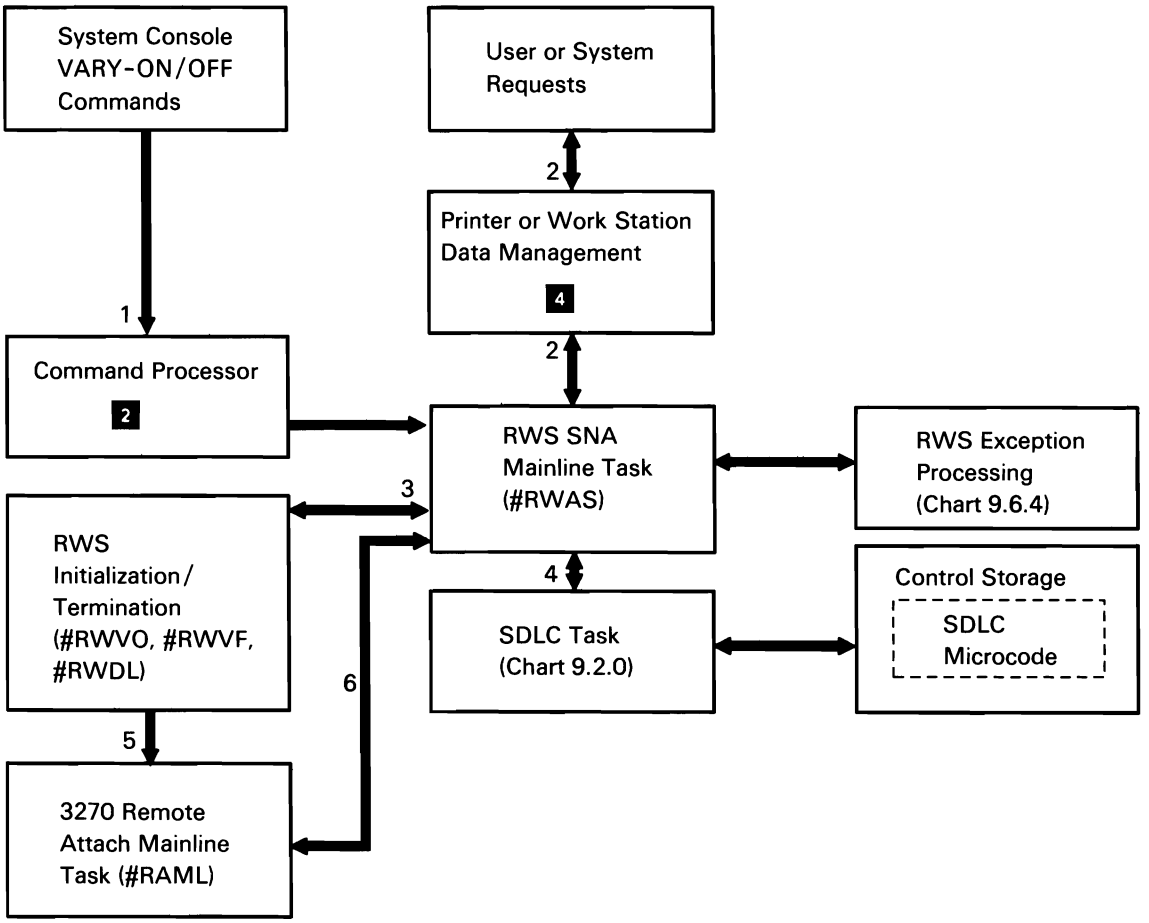


Chart 9.6.0 Remote Work Station (RWS) Support Overview Control Flow

RWS consists of four basic parts:

- RWS vary on — Chart 9.6.1.n
- RWS put/get data — Chart 9.6.2.n
- RWS vary off — Chart 9.6.3
- RWS exception processing — Chart 9.6.4

Remote Work Station (RWS) Vary On

RWS support can be activated by IPL (if any remote work stations are configured for auto-vary online), or by the system operator via the Vary-On command. The following table summarizes the sequence of events that occur during vary on. The columns list the request type, the module (and, in some cases, the routine) that builds and/or posts the request and its associated control blocks, the module that initially processes the request, and the module to which a successful return code is routed.

Vary On for 5250 Devices				
Sequence	**Request**	**Built/Posted By**	**Initially Processed By**	**Successful Return Code Routed To**
1	OPEN line	#RWVO	#SDIT	#RWVO via #RWAS
2	ENABLE line	#RWVO	#SDIT	#RWVO via #RWAS
3	OPEN station	#RWVO	#SDRI	#RWVO via #RWAS
4	XID	#RWVO	#SDRI	#RWVO via #RWAS
5	SNRM	#RWVO	#SDRI	#RWVO via #RWAS
6	ACTLU	#RWAS	#SDRI	#RWAS
VARY ON SUCCESSFUL message is issued at this point.				
7	BIND	#RWAS	#SDRI	#RWAS
If BIND indicates the station is a 5294 controller, the controller is loaded downline in the following two sequences:				
8	LOAD QUERY COMMAND	#RWDL	#SDRI	#RWDL via #RWAS
9	LOAD REQUEST REPLY	#RWDL	#SDRI	#RWDL via #RWAS
When controller is completely loaded, #RWAS sends power-on aid to command processor for display stations attached to the controller and, if printer spooling is active, attaches spool writer for printers attached to controller.				

Vary On for 3270 Devices				
Sequence	**Request**	**Built/Posted By**	**Initially Processed By**	**Successful Return Code Routed To**
1	OPEN line	#RWVO	#SDIT	#RWVO
2	ENABLE line	#RWVO	#SDIT	#RWVO via #RWAS
3	OPEN station	#RWVO	#SDRI	#RWVO via #RWAS
4	XID (for switched lines only)	#RWVO	#SDRI	#RWVO via #RWAS
5	SNRM	#RWVO	#SDRI	#RWVO via #RWAS
6	ACTPU	#RWAS	#SDRI	#RWVO via #RWAS
When ACTPU completes, #RMVO attaches the 3270 remote attach task for the controller.				
7	ACTLU	#RWAS	#SDRI	#RWAS
VARY ON SUCCESSFUL message is issued at this point.				
8	BIND	#RWAS	#SDRI	#RWAS
9	SDT	#RWAS	#SDRI	#RWAS
10	BID	#RWAS	#SDRI	#RWAS
When the BID completes, #RWAS sends power-on aid to command processor for display stations or, if print spooling is active, attaches spool writer for printers (for non-IPDS devices).				

(IPDS only)

11	INIT-IPDS	#RWAS	#SDRI	#RWAS
When INIT-IPDS completes and if print spooling is active, #RWAS attaches the spool writer for the printer.				

The following vary-on processes for 5250 devices are shown in Chart 9.6.1.1:

1 If this is the first vary on, build all remote line information blocks (RWLBs), remote work station alternate line blocks (RWALTs) for switched lines, and controller unit blocks (CUBs).

  Build and initialize device unit block (DUB) and DUB extension for device, and post RWS task for vary on.

2 Route RWLB, CUB, and/or DUB to vary-on processor, depending on vary-on command.

3 Perform any required DLC/controller initialization:

  - If line is not open, call DLC initialization/ termination to open it.
  - If control unit is not initialized, do the following:
    - Build an IOB with an open station command; post the IOB to DLC.
    - After open station IOB returned, build an IOB with an XID command and post it to DLC.
    - After IOB with an XID response is returned, build an IOB with an SNRM command and post it to DLC.
    - Return IOB with SNRM command to #RWAS.

4 Perform SNA session initialization:

  - Build IOB with ACTLU command and post it to DLC.
  - If ACTLU response indicates that device is powered on, build an IOB with a BIND command, and post it to DLC.

5 If BIND indicates the controller is capable of being downloaded, do the following:

  - Send load query command to controller via #RWAS and data link control.
  - Process load request command from controller.

6 Pass load query reply from controller to #RWDL. (If the load query reply indicates there are more requests, repeat the above download sequence.

7 Process transmit IOBs.

8 Complete vary on by doing the following:

  - If bind is for a display station, post the command processor with a power-on aid, which allows the command processor to send the sign-on screen.
  - If bind is for a printer and spooling is being used, attach the spool writer.

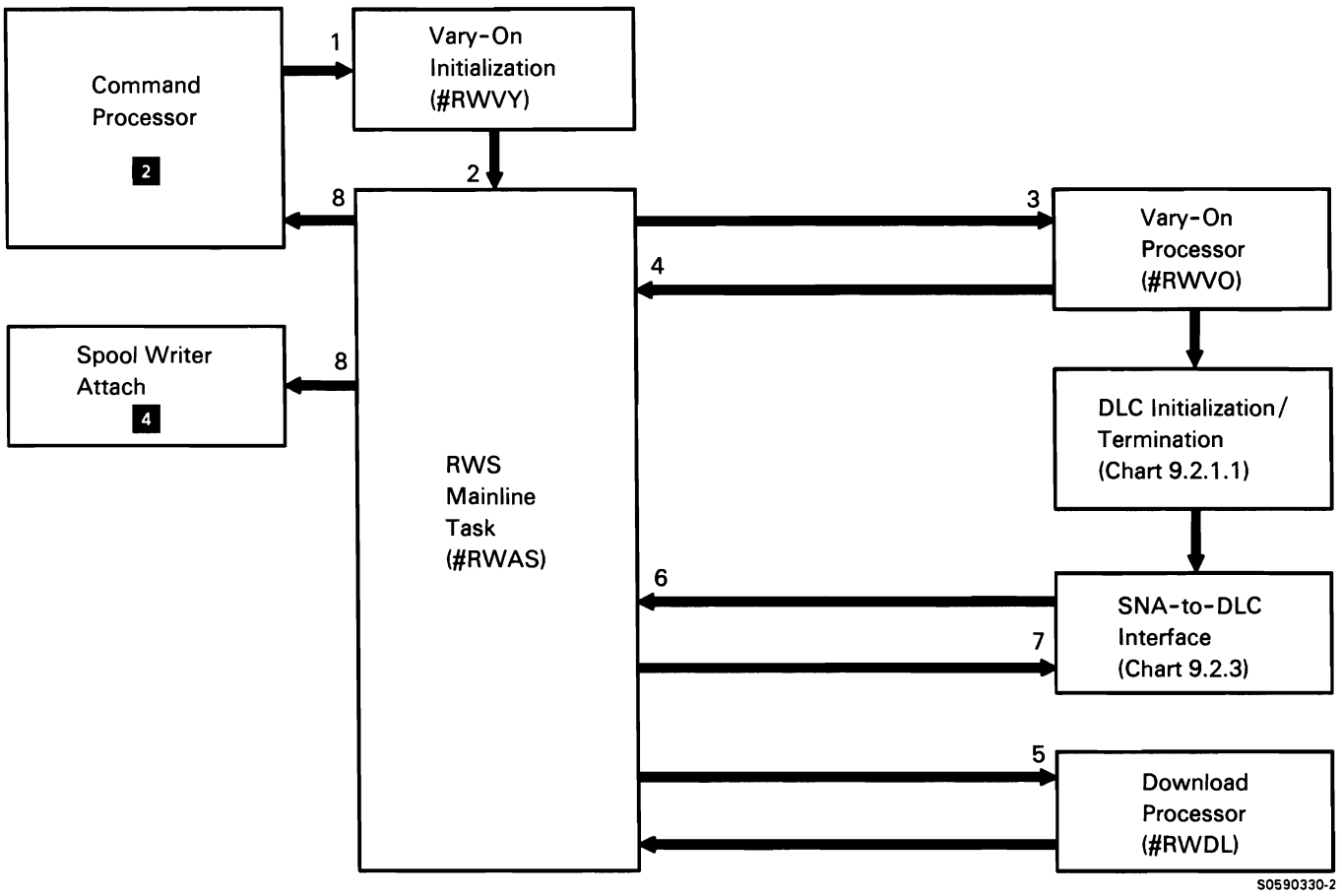


Chart 9.6.1.1 Remote Work Station Vary-On Control Flow for 5250 Devices

The following vary-on processes for 3270 devices are shown in Chart 9.6.1.2:

1 If this is the first vary on, build all remote line information blocks (RWLBs), remote work station alternate line blocks (RWALTs) for switched lines, and controller unit blocks (CUBs). Build and initialize device unit block (DUB) and DUB extension for device and post RWS task for vary on.

2 Route RWLB, CUB, and/or DUB to vary-on processor, depending on vary-on command.

3 Perform any required DLC/controller initialization:

- If line is not open, call DLC initialization/ termination to open it.
- If control unit is not initialized, perform the following:
  - Build an IOB with an open station command and post the IOB to DLC.
  - After open station IOB returned if the controller is configured on switched lines, build an IOB with an XID command and post it to DLC.
  - After XID IOB response received, or if controller is configured on leased lines, build an IOB with an SNRM command and post it to DLC.
  - After SNRM IOB returned, build an ACTPU command and post it to DLC.
  - Wait for positive response from ACTPU.

4 When positive response to ACTPU is received, attach the 3270 remote attach mainline task. Return to #RWAS.

5 Perform SNA session initialization:

- Build IOB with ACTLU command and post it to DLC.
- If ACTLU response indicates device is available, build an IOB with a BIND command and post it to DLC.
- After a BIND response is received, build an IOB with an SDT command and post it to DLC.
- After SDT response is received, build an IOB with a BID command and post it to DLC.
- After BID response is received if device is an IPDS printer, build an IOB with an INIT-IPDS command and post it to DLC.

6 After BID response or INIT-IPDS response (for an IPDS printer) is received, perform the following:

- If the response is for a display station, post the command processor with a power-on aid, which allows the command processor to send a sign-on screen.
- If response is for a printer and spooling is being used, attach the spool writer.

7 Process transmit IOBs.

8 Process receive IOBs.

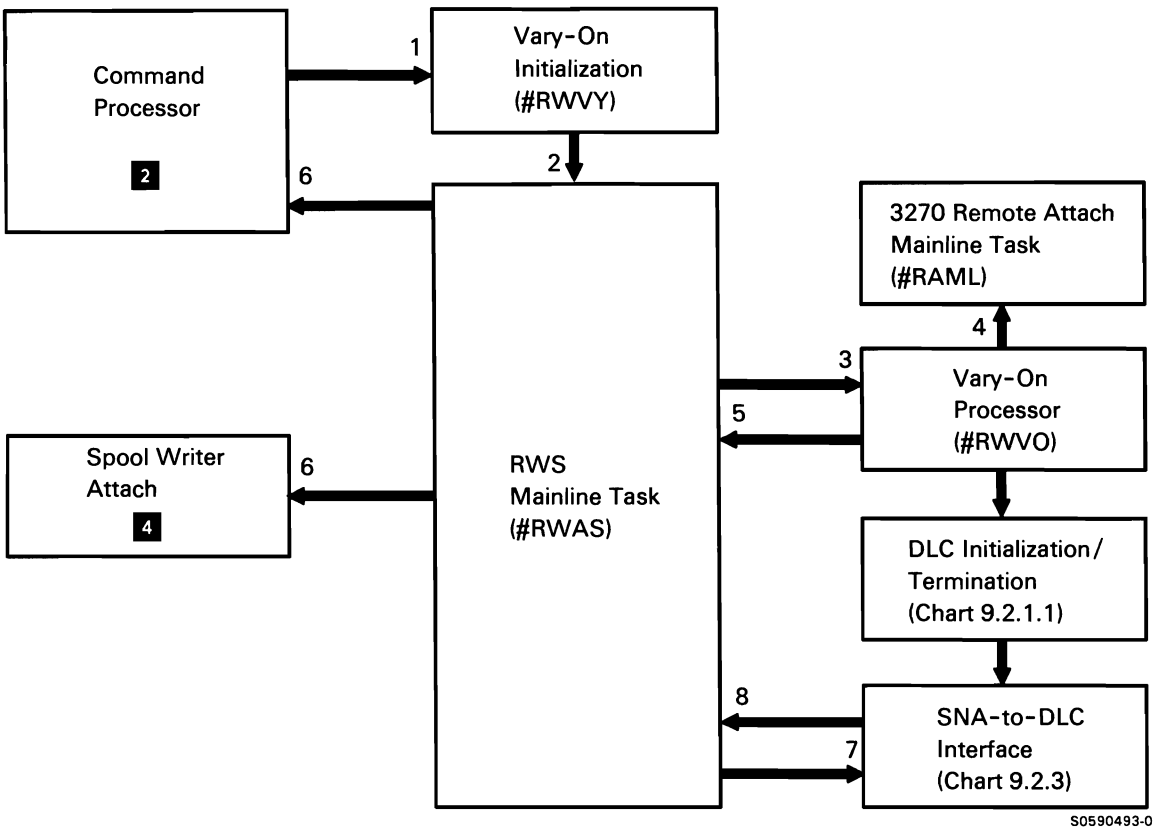


Chart 9.6.1.2 Remote Work Station Vary-On Control Flow for 3270 Devices

This page is intentionally left blank.

Remote Work Station (RWS) Get Data/Save Screen for 5250 Display Stations

RWS get data/save screen in initiated by a post from work station data management with a TUB marked with a get data or save screen request. On a save screen request, the TUB process bytes (attribute bytes indicating current status of display) are also saved as part of the data.

The following get data/save screen processes for 5250 display stations are shown in Chart 9.6.2.1:

1 Do the following:

- Build an IOB containing a read command.
- Set up SNA transmission header (TH) and request header (RH) for transmit.

2 Post DLC with DLC transmit IOB containing read command.

3 Process IOB and 256 bytes (maximum) of get/save data from DLC:

- For a get, assign WSQS for received data. For a save, use TWS assigned by work station data management for received data.
- If more than 256 bytes of get or save data, continue to process IOBs posted by DLC.

4 If not enough available WSQS, write data to TWA on disk.

5 If invalid data amount, handle the exception.

6 Post work station data management with updated TUB after all get/save data has been received.

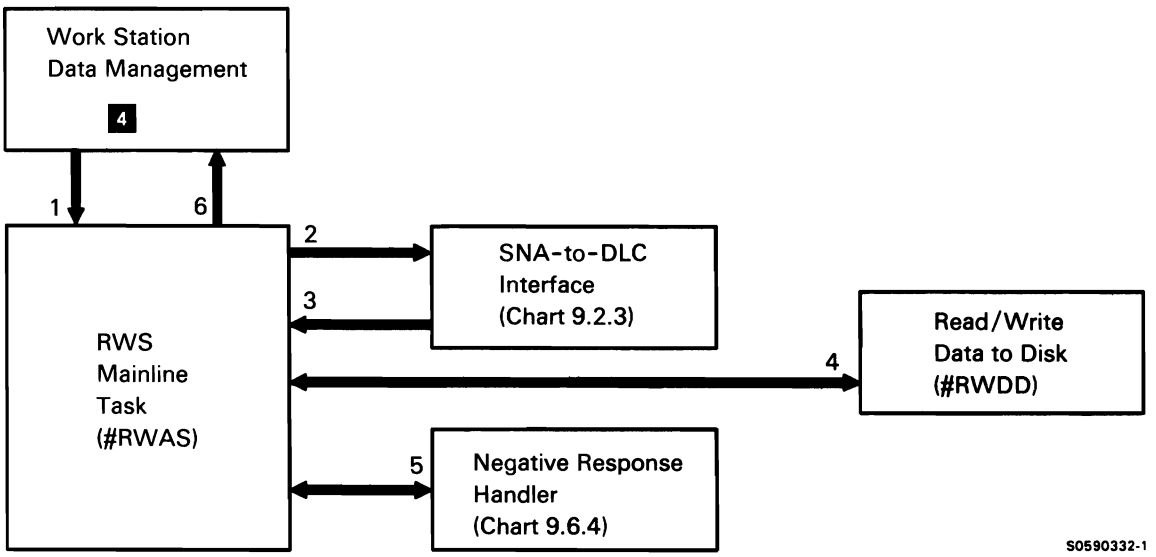


Chart 9.6.2.1 Remote Work Station Get Data/Save Screen Control Flow for 5250 Display Stations

Remote Work Station (RWS) Get Data/Save Screen for 3270 Display Stations

RWS get data/save screen is initiated by a post from work station data management with a TUB marked with a get data or a save screen request. On a save screen request, the TUB process bytes (attribute bytes indicating current status of display) are also saved as part of the data.

The following get data/save screen processes for 3270 display stations are shown in Chart 9.6.2.2:

1 Perform the following:

- If request is a get data read input fields or a get data read screen, go to process 4.
- Build an IOB containing a read command.
- Set up SNA transmission header (TH) and request header (RH) for transmit.

2 Post DLC with DLC transmit IOB containing read command.

3 Process IOB and 256 bytes (maximum) of get/save data from DLC:

- Move data from IOB to TWS assigned by 3270 remote attach.
- If more than 256 bytes of data, continue to process IOBs posted by DLC.

4 Post 3270 remote attach mainline to translate data or command.

5 Perform one of the following:

- If the command is read screen or read input fields, assign WSQS and move the latest screen data (saved in an internal buffer called the shadow buffer) to the WSQS.
- If the command is save screen, use TWS assigned by work station data management for received data.
- For other get data commands, translate the 3270 data to 5250 data, assign WSQS, and move the translated data to the WSQS.

6 Post work station data management with updated TUB after all get/save data has been received. If invalid data amount, handle the exception.

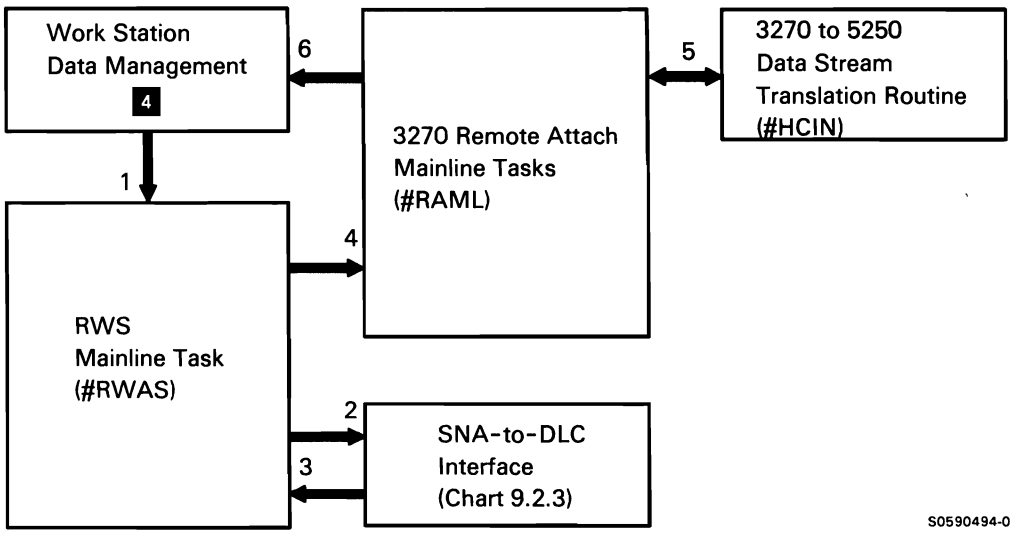


Chart 9.6.2.2 Remote Work Station Get Data/Save Screen Control Flow for 3270 Display Stations

This page is intentionally left blank.

Remote Work Station (RWS) Put Data/Restore Screen for 5250 Display Stations

RWS put data/restore screen is initiated by a post from work station data management with a TUB marked with a put data or restore screen request. On a restore screen request, the TUB process bytes (attribute bytes indicating saved status of display) are restored from part of the data.

The following put data/restore screen processes for 5250 display stations are shown in Chart 9.6.2.2:

1 Do the following:

- Build an SDLC IOB with a maximum of 256 bytes of put data.
- Set up SNA transmission header (TH) and request header (RH) for a transmit IOB.
- If more than 256 bytes of data, continue building IOBs.
- If restore request:
  - Assign WSQS.
  - If data is stored on disk, read data from disk and move the data from the TWA on disk to the WSQS buffer.
  - Build DLC IOB.

2 Post DLC with DLC transmit IOB(s) containing write command and data.

3 Wait for response from remote controller and verify response.

4 If a positive response is received, post work station data management.

5 If a negative response is received, call the negative response handler.

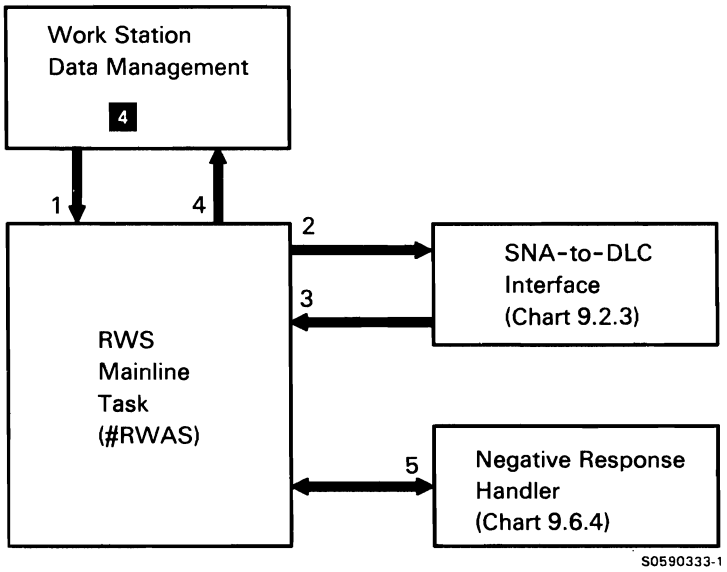


Chart 9.6.2.3 Remote Work Station Put Data/Restore Control Flow for 5250 Display Stations

Remote Work Station (RWS) Put Data/Restore Screen for 3270 Display Stations

RWS put data/restore screen is initiated by a post from work station data management with a TUB marked with a put data or restore screen request. On a restore screen request, the TUB process bytes (attribute bytes indicating saved status of the display) are restored from part of the data.

The following put data/restore screen processes for 3270 display stations are shown in Chart 9.6.2.4:

1 Post RWS with a TUB marked put data or restore screen.

2 Post the 3270 remote attach mainline task with the TUB.

3 Call the 5250 to 3270 data stream translation routine, which does the following:

- If restore request, move saved data from disk to the pre-assigned TWS.
- Otherwise, translate the data to 3270 data stream and put the translated data in the pre-assigned TWS.
- Update the internal screen buffer (shadow buffer) to have the latest data on the display station's screen.

4 Post RWS to do the following:

- Build an SDLC IOB with a maximum of 256 bytes of put data.
- Set up SNA transmission header (TH) and request header (RH) for a transmit IOB.
- If more than 256 bytes of data, continue building IOBs.

5 Post DLC with DLC transmit IOB(s) containing write command and data.

6 Wait for response from controller.

7 If a positive response is received, post work station data management.

8 If a negative response is received, call the 3270 remote attach exception translator (#RWNA).

9 Call one of the RWS exception handlers (#RWNR, #RWER).

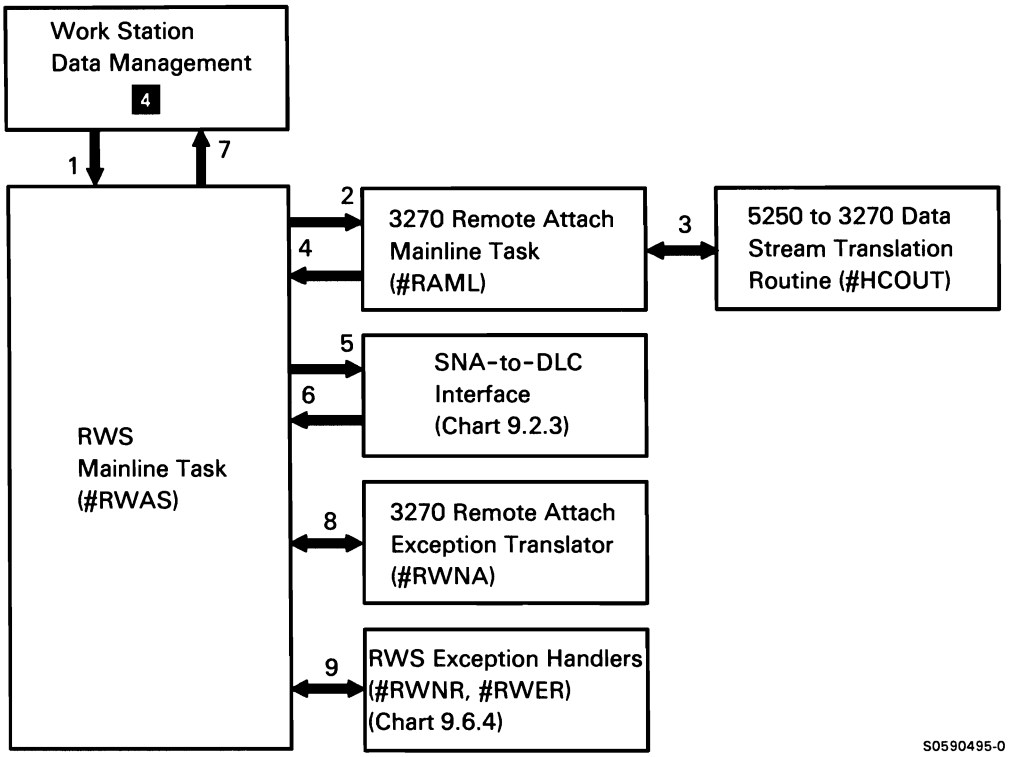


Chart 9.6.2.4 Remote Work Station Put Data/Restore Control Flow for 3270 Display Stations

Remote Work Station (RWS) Printer Put/Get

RWS printer operations are initiated by a posted printer unit block (PUB) from command processor error recovery or by a posted printer IOB from other tasks.

The following printer put/get operation processes are shown in Chart 9.6.2.5:

1 Process the PUB or printer IOB post by finding the first printer IOB queued for the associated printer. If the printer is an IPDS printer or a 5250 SCS printer, go to process 5.

2 Move the printer IOB data to pre-assigned TWS for translation and post the 3270 remote attach task with the associated PUB.

3 Translate the data from 5250 SCS data to 3270 SCS data.

4 Post the RWS mainline back with the PUB.

5 Process the printer operation:

- If the printer is IPDS, build a DLC IOB and send the put or get command with the appropriate 5250 or 3270 transmission header (TH) and request header (RH).
- If the printer is a 5250 SCS printer, build a DLC IOB and send the put data with transmission header (TH) and request header (RH).
- If the printer is a 3270 SCS printer, build as many DLC IOBs as necessary to handle all of the translated data in TWS, and set up the transmission headers (TH) and request headers (RH).

6 Process the receive DLC IOB:

- If the DLC IOB contains printer status for an IPDS printer, move the data into the associated printer IOB storage and go to process 9.
- If the DLC IOB contains a positive response to a non-spooled put, go to process 9.
- If the DLC IOB contains a positive response to a spooled put, find the related "dummy" IOB and go to process 9.

7 If the DLC IOB contains a 3270 exception (negative response, LUSTAT, or signal), call #RWNA to translate it to the appropriate 5250 exception.

8 Call one of the RWS exception handlers to process the exception.

9 Post the printer IOB complete.

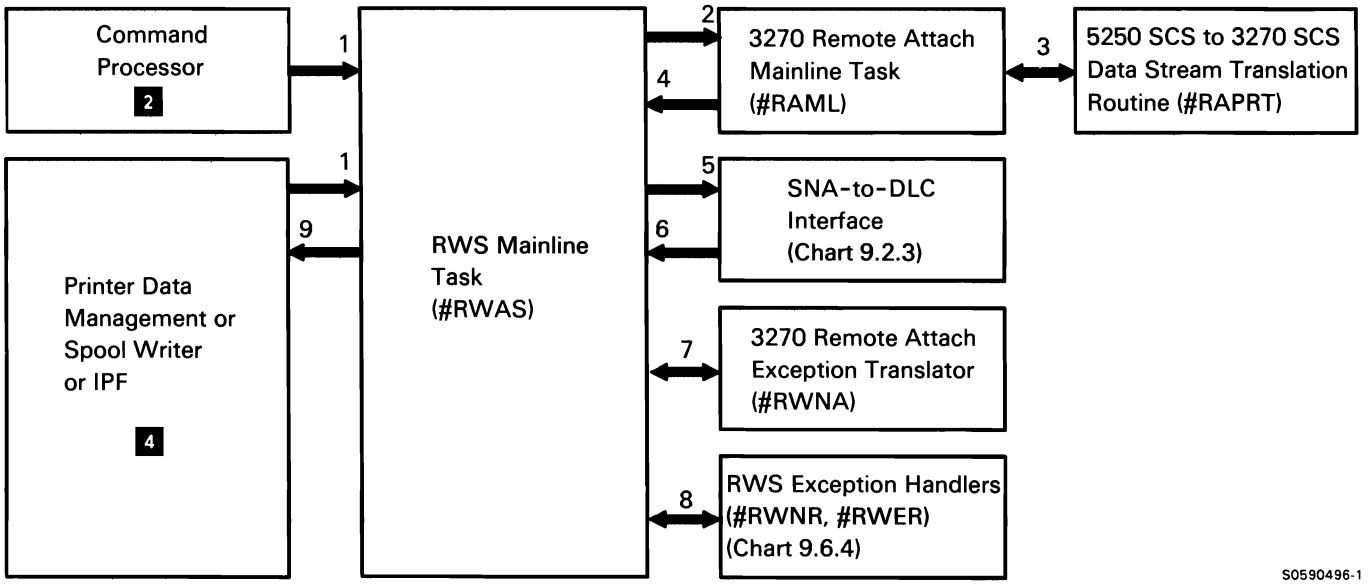


Chart 9.6.2.5 Remote Work Station Printer Put/Get Control Flow

This page is intentionally left blank.

Remote Work Station (RWS) Vary Off

The system operator deactivates RWS support with the Vary-Off command. The following table summarizes the sequence of events that occur during vary off. The columns list the IOB operation code or command, the module that builds and/or posts the IOB and its associated control blocks, the module that initially processes the request, and the module to which a successful return code is routed.

Vary Off for 5250 Devices				
Sequence	**IOB Operation Code or Command**	**Built and Posted By**	**Initially Processed By**	**Successful Return Code Routed To**
1	UNBIND	#RWVF	#SDRI	#RWVF via #RWAS
2	DACTLU	#RWVF	#SDRI	#RWVF via #RWAS
VARY OFF SUCCESSFUL message is issued at this point.				
3	REQMS (Sent) and RECFMS (Received)	#RWAS	#SDRI	#RWAS
4	DISCONNECT	#RWVF	DLC mainline	#RWVF via #RWAS
5	TERMINATE	#RWVF	DLC mainline	#RWVF via #RWAS

Vary Off for 3270 Devices				
Sequence	**IOB Operation Code or Command**	**Built and Posted By**	**Initially Processed By**	**Successful Return Code Routed To**
1	UNBIND	#RWVF	#SDRI	#RWVF via #RWAS
2	DACTLU	#RWVF	#SDRI	#RWVF via #RWAS
VARY OFF SUCCESSFUL message is issued at this point.				
3	DACTPU	#RWAS	#SDRI	#RWAS
3270 remote attach task is detached at this point.				
4	DISCONNECT	#RWVF	DLC mainline	#RWVF via #RWAS
5	TERMINATE	#RWVF	DLC mainline	#RWVF via #RWAS

The following vary-off processes are shown in Chart 9.6.3:

1 If vary-off command, process by posting RWS task with DUB or RWALT to be varied off.

2 Route DUB or RWALT to vary-off processor (#RWVF).

3 Perform necessary SNA session termination:

- Build IOB containing an UNBIND command and post it to DLC.
- Build IOB containing a DACTLU command and post it to DLC.
- If last DUB on control unit, send REQMS to get maintenance statistics, or send DACTPU if control unit is a 3274.
- If there are no possible lines, free the DUB.
- Perform necessary DLC station termination by building DISCONNECT IOB and posting it to DLC.
- Perform necessary DLC line termination by building TERMINATE IOB and posting it to DLC.
- When no remote DUBs are left on any line, issue EOJ for RWS.

4 Route complete IOBs, positive responses from controller, and received RECFMS commands from data link control

5 Route UNBIND and DACTLU IOB responses to #RWVF.

6 Detach 3270 remote attach task when response to DACTPU is received.

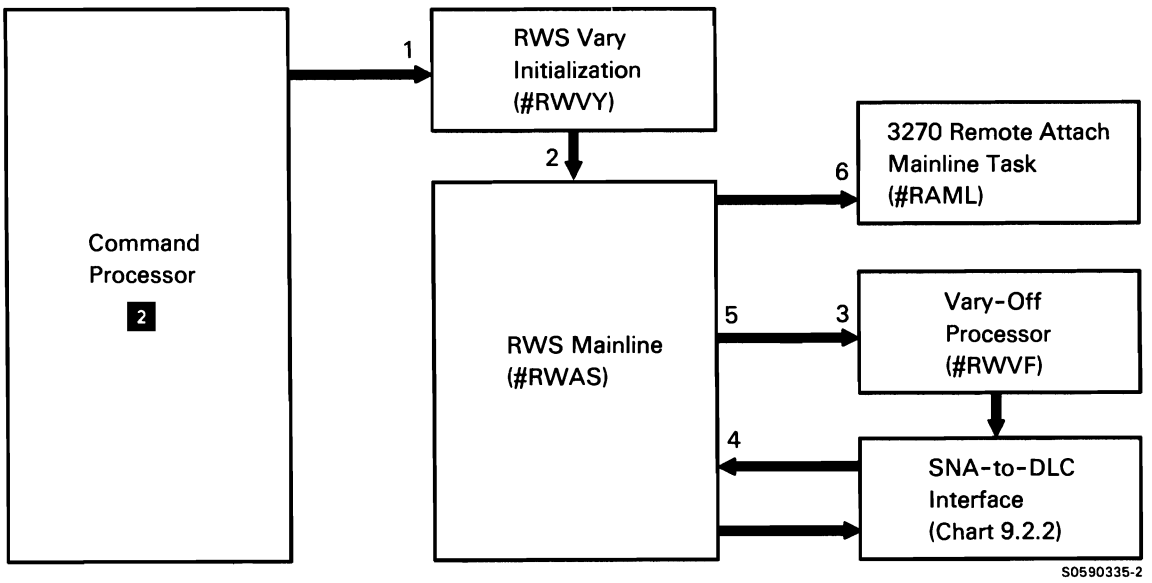


Chart 9.6.3 Remote Work Station Vary-Off Control Flow

This page is intentionally left blank.

Remote Work Station (RWS) Exception Processing

Remote work station (RWS) exception processing receives its input from the DLC task. In addition to handling DLC link errors and device errors, it also processes commands to/from the System/36 SNA data flow control (DCF) command handler.

The RWS negative response handler, the RWS DFC command handler, and the link error handler, all set up dump requests if their processing indicates there might be a problem in IBM-supplied programming; they also call the control storage printer/display error router ($NUPD) to route the error to the proper component. Other processes performed by RWS exception processing, shown in Chart 9.6.4, are:

1 Route for the following:

- Error IOBs.
- Received IOBs containing SNA error status.
- IOBs containing SNA data flow control (DFC) commands.

If link error, route to link error handler.

If a LUSTAT from a remote controller requests the downloading of a GAIJI character matrix to a 5555 display station, post RWS DFC command handler.

2 If negative response IOB, pass the IOB and control to #RWNR. If link error, route to #RWLK; otherwise, pass IOB and control to #RWER.

3 Process the negative response:

- If error recovery for this device is already in progress, ignore the error and return to mainline.
- Set DUB sense bytes to indicate error type.
- If device is varying on or off, indicate that vary failed.
- If negative response was for a command reject, set up to send change direction indicator.
- Return to mainline.

4 If a command reject was processed, build an IOB for change direction indicator and pass it to the DLC task.

5 Process the data flow control (DFC) commands (signals, LUSTATs, RSHUTD):

- If this device is already busy, pend the request and return to mainline.
- Set DUB sense bytes to indicate error type.
- If command was shutdown request (RSHUTD) indicate that UNBIND should be sent to device.
- Return to mainline.

6 Build a response to the DFC command and pass it to the DLC task.

If DFC command is in response to a request for the downloading of a GAIJI character matrix, mark TUB with appropriate AID and post the command processor.

7 Process GAIJI AID request by posting the GAIJI task for character matrix download.

8 Find requested character matrix, and send it to the 5555 Display Station by posting a TUB (with PUT request) to the RWS task.

9 Process put of GAIGI character string to 5555 Display Station.

10 Process link error IOB:

- If open station IOB has failed, build an RWS message block and post it.
- Perform the following for each device in error:
  - Get DUB for device.
  - If error processing is already in effect for this device, pend this error.
  - Set DUB sense bytes to indicate error type.
  - If device vary-on failure, indicate it.
  - If auto-reconnect is configured, indicate that device is to be reconnnected and do the following:
    - If line error or switched line, send terminate/reenable.
    - If station error, build disconnect.
  - Return to mainline.

11 Process RWS message block, and any other processing required by #RWLK. Route terminate and disconnect responses to #RWVF.

12 If terminate or disconnect request was received for auto-reconnect line(s), reset all error indicators in controller unit blocks (CUBs) and device unit blocks (DUBs) being reconnected, and post each CUB to #RWAS.

13 Route CUBs from #RWVF to #RWVO for completion of reconnect sequence.

14 Issue OPEN STATION, issue message indicating vary on in progress, and continue with normal vary-on sequence (Chart 9.6.1).

15 Post data link control with transmit IOB containing terminate and disconnect commands.

16 Post data link control with transmit IOBs containing SNA responses.

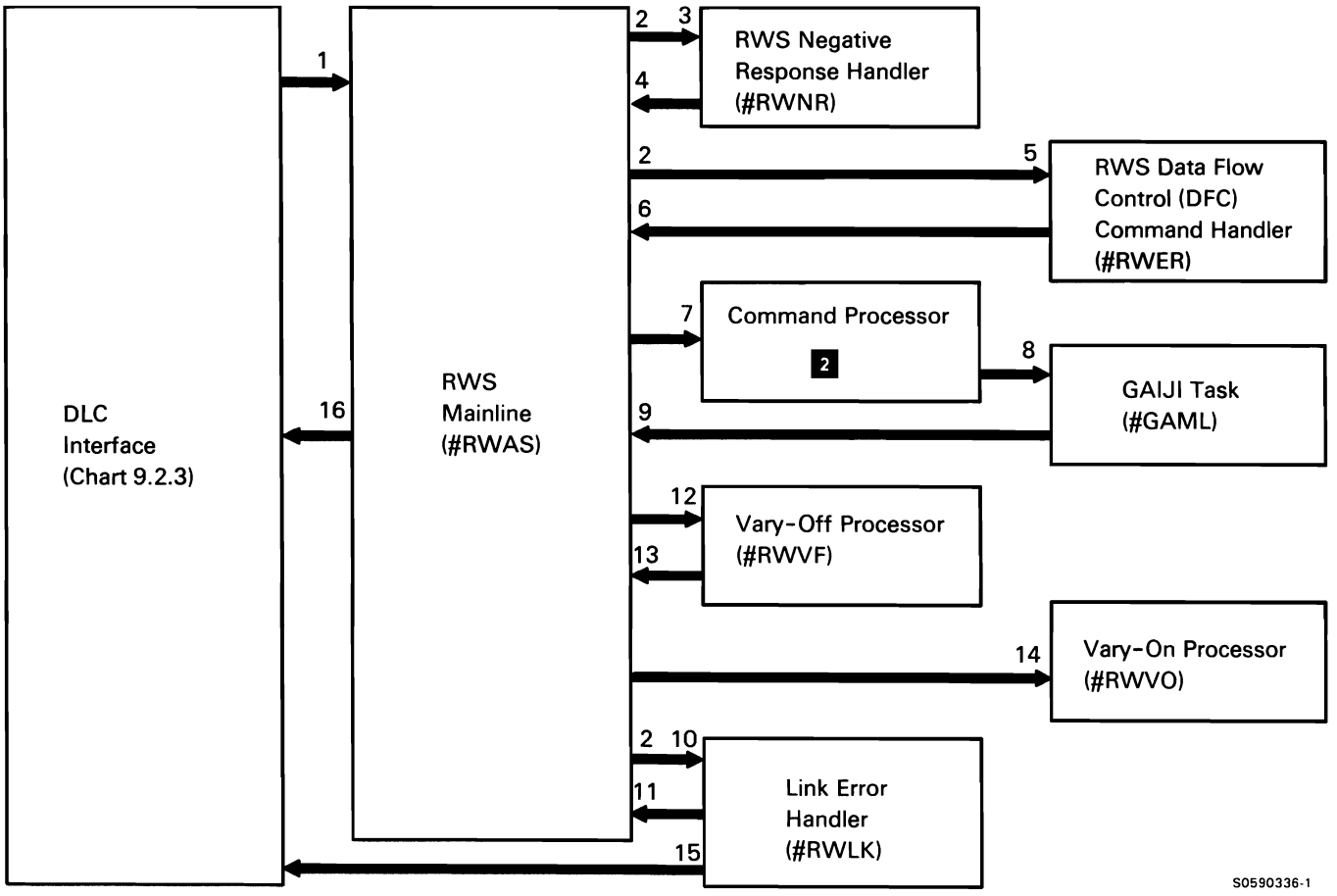


Chart 9.6.4 Remote Work Station Exception Processing Control Flow

LINK STATION TEST

The link station test subsystem allows a user to send a test command to a secondary SDLC station address. The test verifies the link between the system and the terminal or other system. Link station test can be executed while other primary SNA subsystems are executing, but cannot be run to a station being used by another SNA/SDLC function. The link station test is invoked by the STATEST command.

The following link station test processes are shown in Chart 9.7:

1 Perform the following initialization processing:

- Initialize data areas.
- Route control to SDLC initialization to open communications line.
- Route control to open and allocate work station DTF.
- Use work station data management to prompt for and process user responses.
- If necessary, route control to abnormally terminate.

2 Perform the following test processing:

- Build IOB and transmit test command to each operational station on the station list, using SNA-to-SDLC interface.
- Process responses.
- Use work station data management to display test status to user.
- Route for error recovery when necessary.
- Route for abnormal termination on user request or timer expiration.

3 Perform error recovery:

- Process test error completion codes from SDLC.
- Process abnormal response from station.
- Process SDLC failure.
- If the user selects SYSLOG option 3, pass control to abnormally terminate.

4 Display any required error messages.

5 Perform the following termination processing:

- Close the line.
- Route control to SDLC termination to detach SDLC.
- Call SSP termination to terminate link station test.

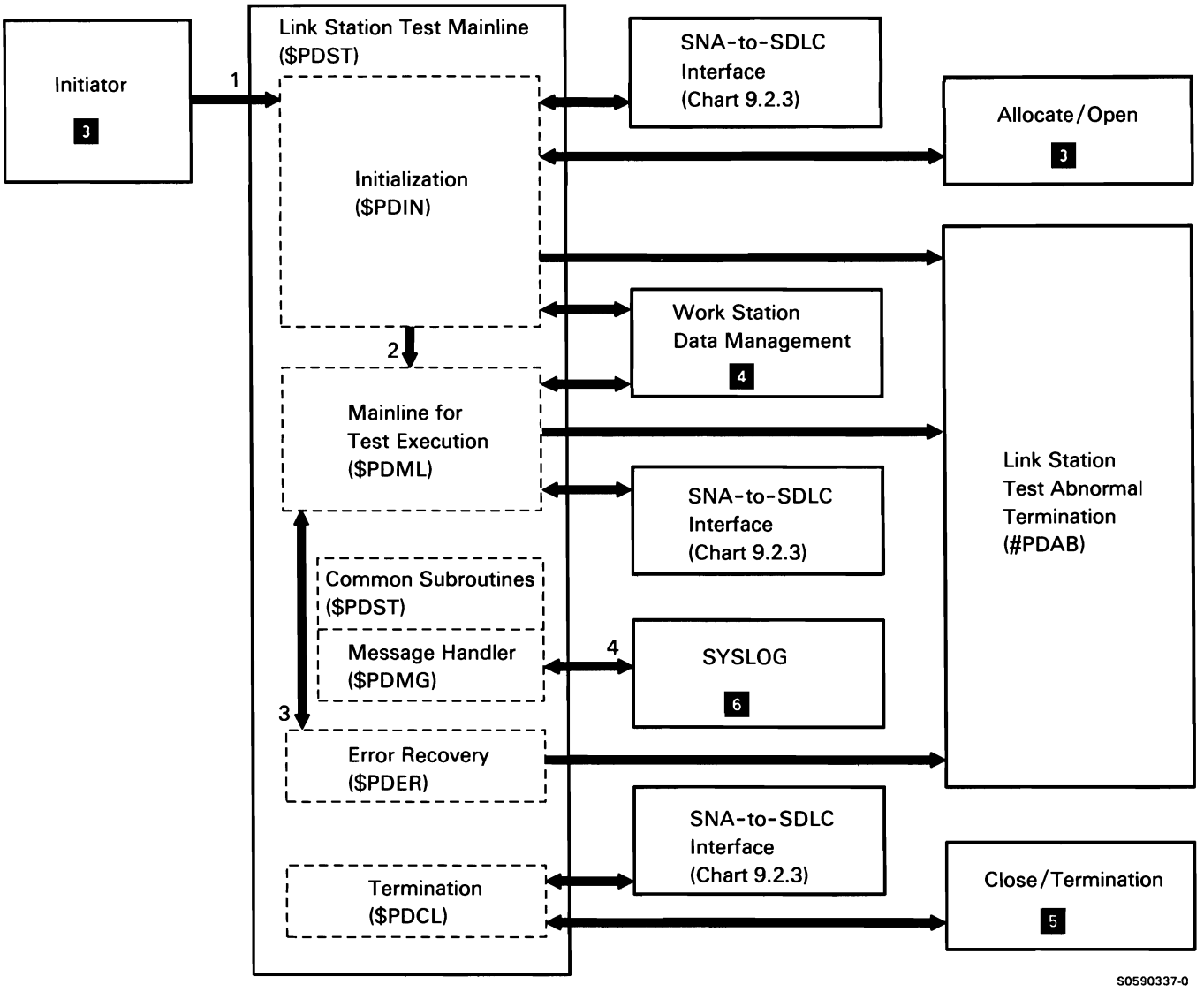


Chart 9.7 Link Station Test Control Flow

X.25 SUPPORT SUBFUNCTION

IBM-supplied X.25 programming for System/36 consists of packet switching interface support for the SSP. The System/36 Packet Switching Interface support in conjunction with the System/36 X.25 feature enables an application program running under the System/36 SSP to communicate with remote locations through an X.25 packet switching network.

System/36 X.25 support performs the following tasks:

- Manage X.25 communications protocols.
- Provide interface to a host system using the SNA Upline Facility (SNUF), SNA 3270 support, SNA MSRJE, or APPC.
- Provide interface to remote 3600 and 4700 controllers using the SNA Finance subsystem.
- Provide interface to remote 5251 Model 12 Display Stations using remote work station support (RWS).
- Provide interface to a System/34 or another System/36 using SNA Peer support.
- Provide interface to a remote station using the station test subsystem.
- Provide interface to a remote asynchronous work station using the asynchronous subsystem.
- Provide support for up to 16 active X.25 logical channels on each of up to three X.25 lines.
- Provide utility programs to define the X.25 network interface characteristics; to temporarily alter some of the X.25 network interface characteristics; to monitor the status of the X.25 links; to create network address lists; and to accumulate, display, reset, and print tariff-related statistics.
- Provide the capability to automatically connect and disconnect X.25 virtual circuits when a remote station is varied on/enabled and varied off/disabled.
- Support the association of stations on a switched line to switched virtual circuits, and the association of stations on a nonswitched line to switched or permanent virtual circuits.
- Provide for error recovery.

System/36 X.25 support runs under the System/36 SSP; it requires a system unit with a minimum of 256K bytes of main storage. The X.25 mainline task requires 32K bytes of nonswappable main storage, and the initialization/termination task requires 22K bytes of swappable main storage. The X.25 maintenance utility, the X.25 configuration utility, and the X.25 phone list utility each require 24K bytes of swappable main storage when they are loaded.

All X.25 main storage to control storage communications is done between X.25 main storage programs and the X.25 control storage programs for the particular line, via the control storage MLCA/ELCA and MLCA/ELCA IOCH processors.

The System/36 X.25 support programs execute as independent tasks. Application programs use them via SSP-ICF and remote work station (RWS) support to establish, maintain, and terminate communications sessions with remote stations. X.25 support consists of the following programs:

- Initialization/termination task
- Mainline task
- Abnormal termination tasks
- Utilities for X.25 configuration, maintenance, and phone list definition

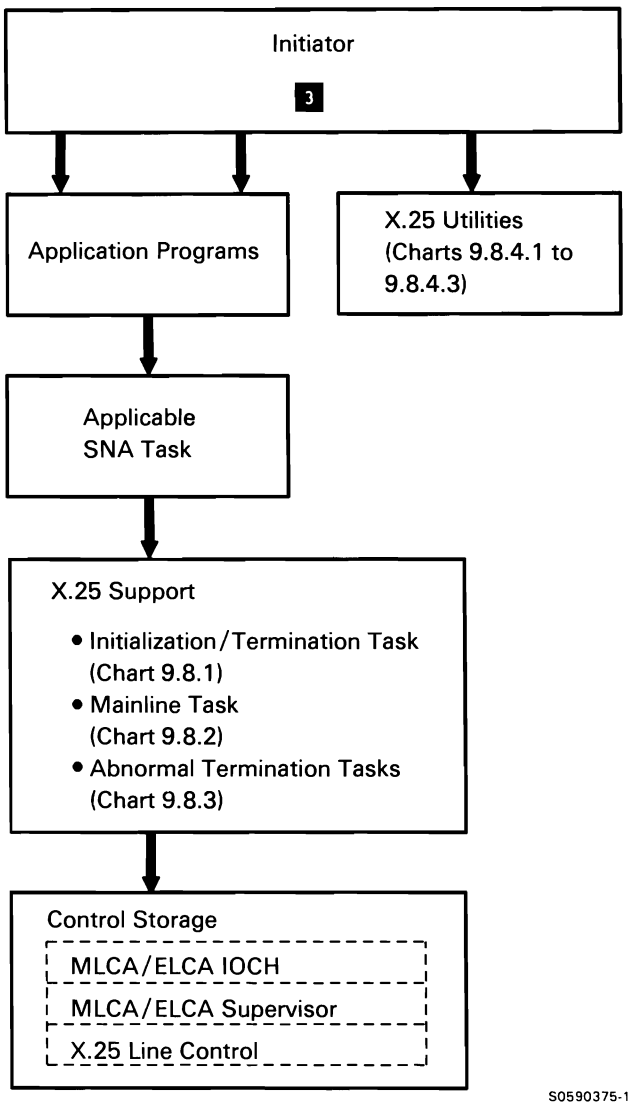


Chart 9.8.0 X.25 Overview Control Flow

X.25 Initialization/Termination Task

The X.25 initialization/termination task performs all operations necessary to initiate and terminate the X.25 links. It uses the configuration data to establish the links to the network.

The X.25 initialization/termination task posts IOBs to the control storage MLCA/ELCA IOCH processor for enable, disable, and open link operations.

The following X.25 initialization/termination task processes are shown in Chart 9.8.1:

1 Initialize and terminate X.25 tasks:

- Enable X.25 link (load SNA to DLC interface, #SDRI).
- Open/terminate communications line for X.25 communications.
- Enable/disable adapter.

2 Do the following:

- Issue console messages.
- Log error counter table.
- Log X.25 tariff information.
- Log IOBs and data in system trace area.
- Determine if normal or abnormal termination.

3 Perform normal termination of the X.25 tasks.

4 Perform abnormal termination on X.25 mainline or MLCA/ELCA processor check.

5 Process IOBs for enable, disable, and open link operations.

Note: MLCA/ELCA IOCH routes requests to the MLCA/ELCA processor. The MLCA/ELCA supervisor then passes the requests to the control storage X.25 code for the specified line.

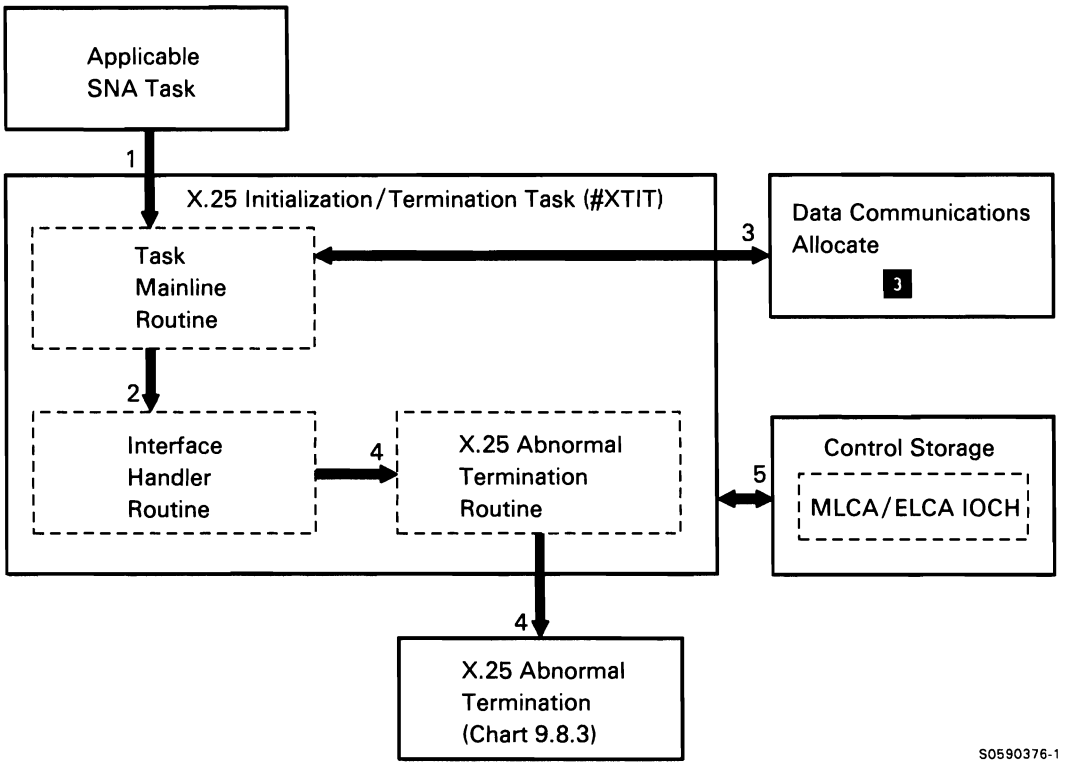


Chart 9.8.1 X.25 Initialization/Termination Task Control Flow

X.25 Mainline Task

The mainline task performs the control operations necessary for X.25 support. It routes control information to and from the X.25 feature. It also routes data between the X.25 feature and remote work station support or an SSP-ICF subsystem. The mainline task establishes and terminates virtual circuits.

The following X.25 mainline task processes are shown in Chart 9.8.2:

1 Process IOBs from the applicable SNA task for X.25 functions.

2 Route requests to the X.25 microcode for the following, based on the SNA or RWS operation codes:

- Open virtual circuit on incoming call:
  - Validate incoming call packet's facility field.
  - Validate incoming call packet's call user data.
  - Build an OPEN SVC IN IOB and send it to X.25 microcode.
- Open virtual circuit for outgoing call:
  - Call specified VCB location.
  - Allocate a free logical channel for call.
  - Build an OPEN SVC OUT or OPEN PVC IOB, and send it to X.25 microcode.
- Build call user data for a SVC OUT IOB.
- Build facility field for an SVC IN or SVC OUT IOB.
- Close virtual circuit.
- Transmit/receive.

3 Perform the following initialization interface functions:

- Put task attached by #XTIT in wait state.
- Enable VCB group.
- Add VCB group.
- Process message response.

4 Provide interface for X.25 microcode requests:

- Process completion of requests to microcode.
- Process hardware or software error by transferring to #XTEH for error history table logging and error recovery.

5 Log IOBs and data in system trace data area.

Note: Opening and closing virtual circuits, and enabling and terminating VCB groups is done by transferring to #XTOC.

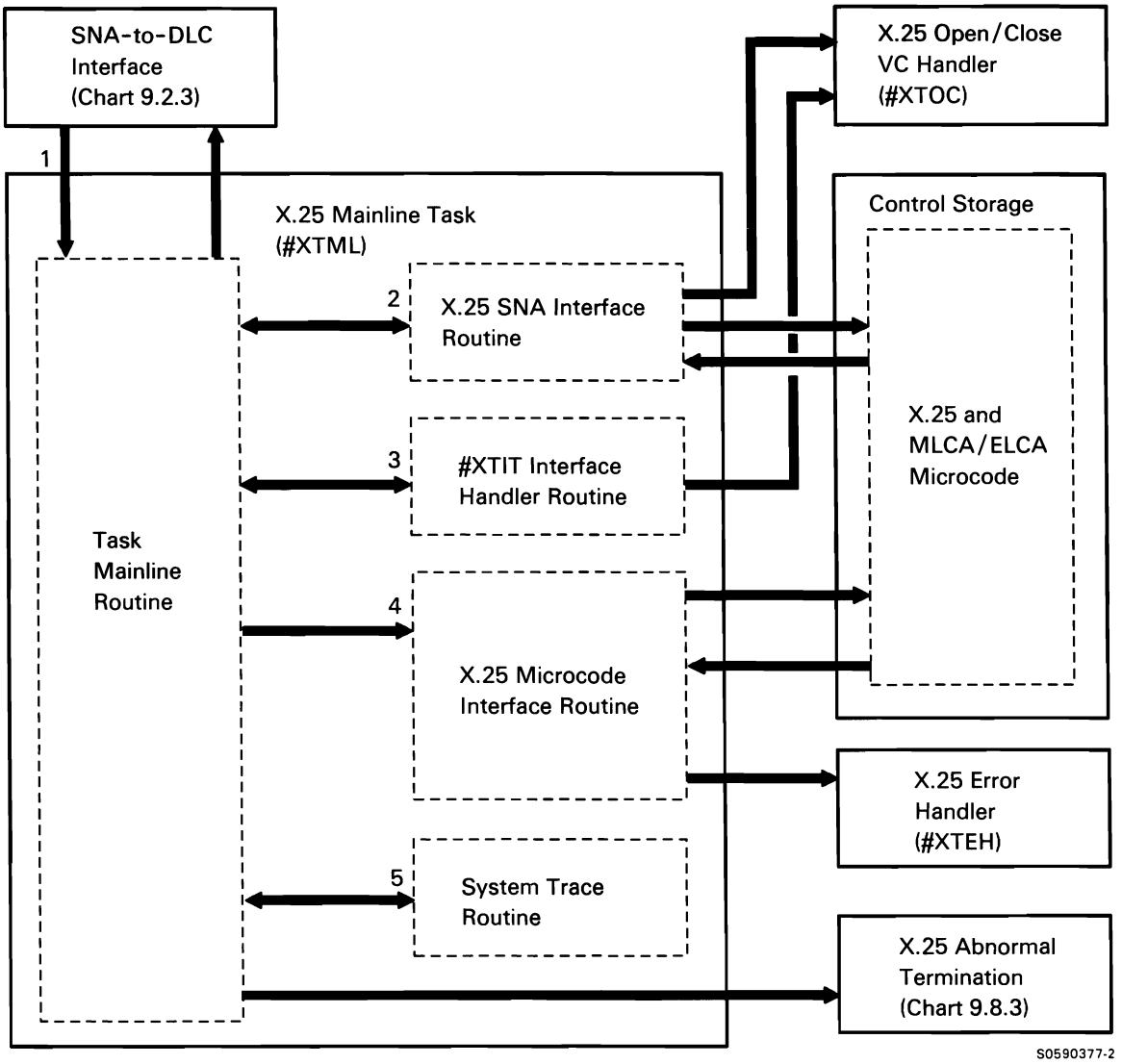


Chart 9.8.2 X.25 Mainline Task Control Flow

X.25 Abnormal Termination Task

The abnormal termination task consists of two transients that handle abnormal termination of the X.25 mainline task or the MLCA/ELCA.

The following X.25 abnormal termination task processes are shown in Chart 9.8.3:

1 Do the following:

- Notify #XTAT of processor check in #XTML or MLCA microcode processor check.
- Post #XTAT to acknowledge receipt of control for #XTIT processor check abnormal termination processing.
- Process all IOBs normally posted to #XTML; post terminate/detach IOBs (from SNA) to #XTAT.

2 Do the following:

- Acknowledge post notifying of #XTML or MLCA microcode error.
- Process SNA terminate/detach IOB.

3 Begin error processing when posted out of wait for #XTIT error processing.

4 Do the following mainline functions:

5 Disable the communications adapter.

6 Clear the lines by waiting for and processing terminate/detach or line open detach.

7 Clear the line queues and requeue internal IOBs onto network control blocks (NCB).

8 Clear the transmit queues of all virtual circuit blocks (VCBs) on the X.25 links.

9 Post internal IOBs to SNA to notify SNA of error.

10 Deallocate lines and log job counters.

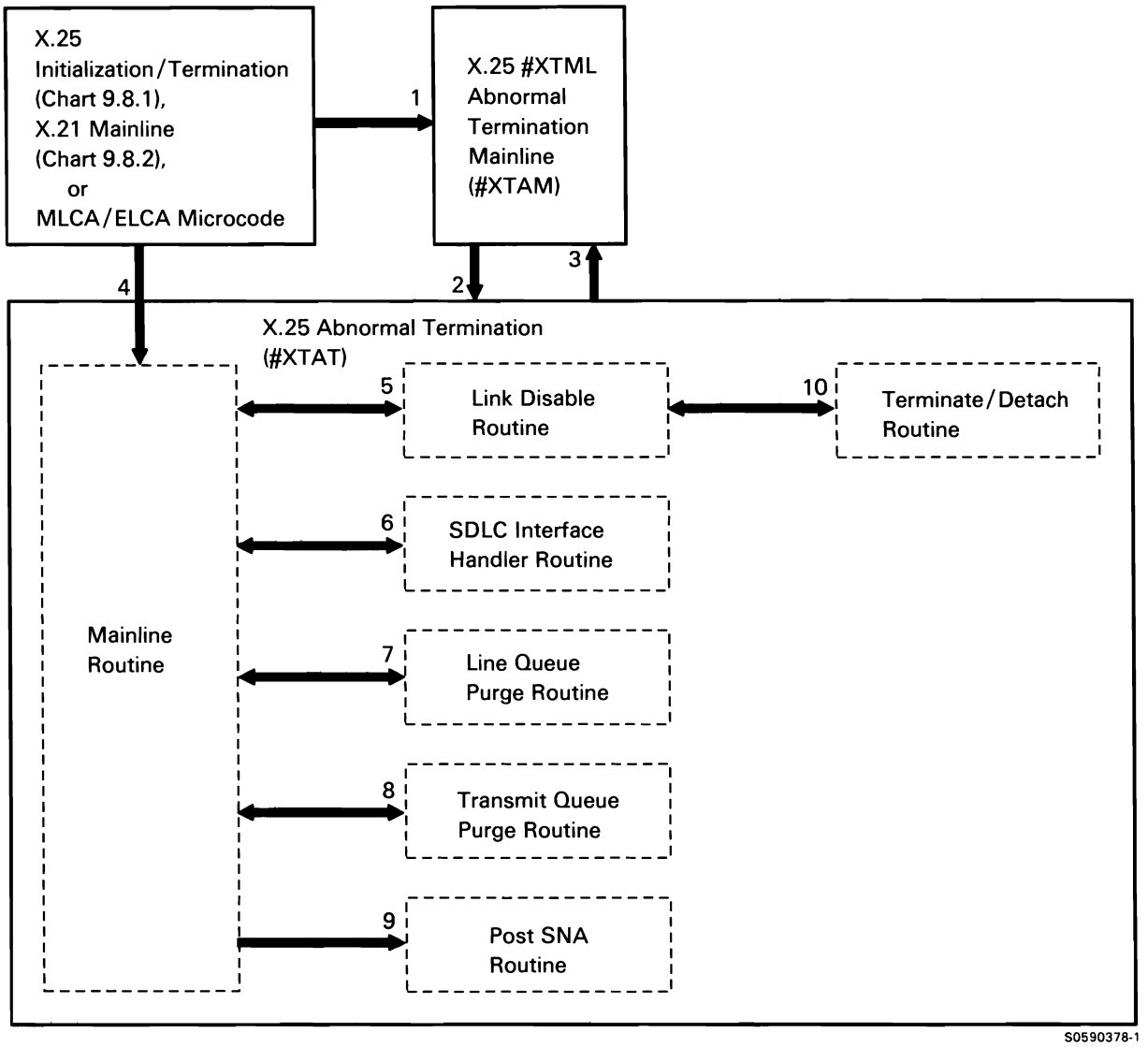


Chart 9.8.3 X.25 Abnormal Termination Control Flow

X.25 Utilities

X.25 Configuration Utility

The X.25 configuration utility allows a user to define the network configuration, the logical channel configurations, and the virtual circuit configurations.

The X.25 configuration utility is invoked by the CNFIGX25 procedure or equivalent OCL and utility control statements. If necessary, the CNFIGX25 procedure builds the #X25LIB library using $MAINT.

The following X.25 configuration utility processes are shown in Chart 9.8.4.1:

1 Perform mainline routing for utility.

2 Prompt operator for configuration option:

- Get option entered by user from main menu.
- Display errors or return valid data.

3 Route to network and logical channel configuration overlays:

- Display network configuration option menu, errors, and valid data.

4 Edit/update a network configuration.

5 Delete a network configuration.

6 Print a network configuration.

7 Route to virtual circuit configuration overlays.

8 Display library/member selection screen, errors, and valid data.

9 Edit and update a virtual circuit configuration.

10 Delete a virtual circuit configuration.

11 Format and print a virtual circuit configuration.

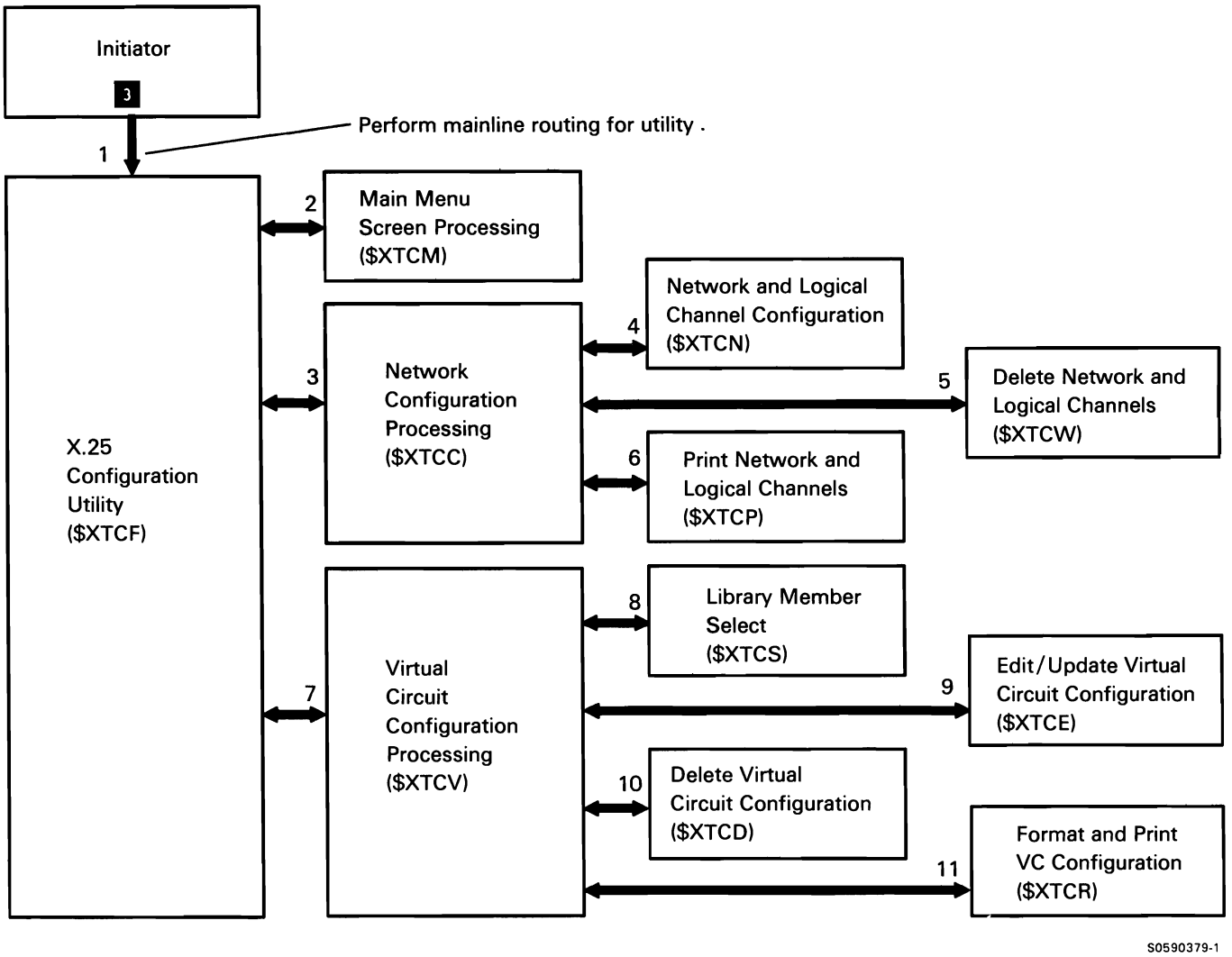


Chart 9.8.4.1 X.25 Configuration Utility Control Flow

X.25 Maintenance Utility

The X.25 maintenance utility allows the user to display status information about the X.25 links, and to temporarily alter some of the virtual circuit configuration parameters.

The maintenance utility is invoked by the MAINTX25 system procedure or equivalent OCL and utility control statements.

The following X.25 maintenance utility processes are shown in Chart 9.8.4.2:

1 Display X.25 maintenance utility menu and perform mainline routing for operator responses.

2 Control configuration member with tariff totals selection:

- Display list of virtual circuit members with tariff totals.
- If requested, display virtual circuit tariff totals for a virtual circuit member.
- If requested, print (and reset) virtual circuit tariff totals for a virtual circuit member.

3 Get and display logical channel status; if requested, update logical channel status display screen.

4 Display and control active virtual circuit configuration selection:

- Display list of active virtual circuit members.
- If requested, display status of virtual circuits in an active virtual circuit member.
- If requested, update active-virtual-circuit-member-list display screen.
- If requested, edit virtual circuit parameters of a station in an active virtual circuit member.

5 If requested, process call wait queue status/edit option:

- Display list of stations on the call-wait queue.
- If requested, update stations on the call-wait queue display screen.
- If requested, edit virtual circuit parameters of a station on the call-wait queue.

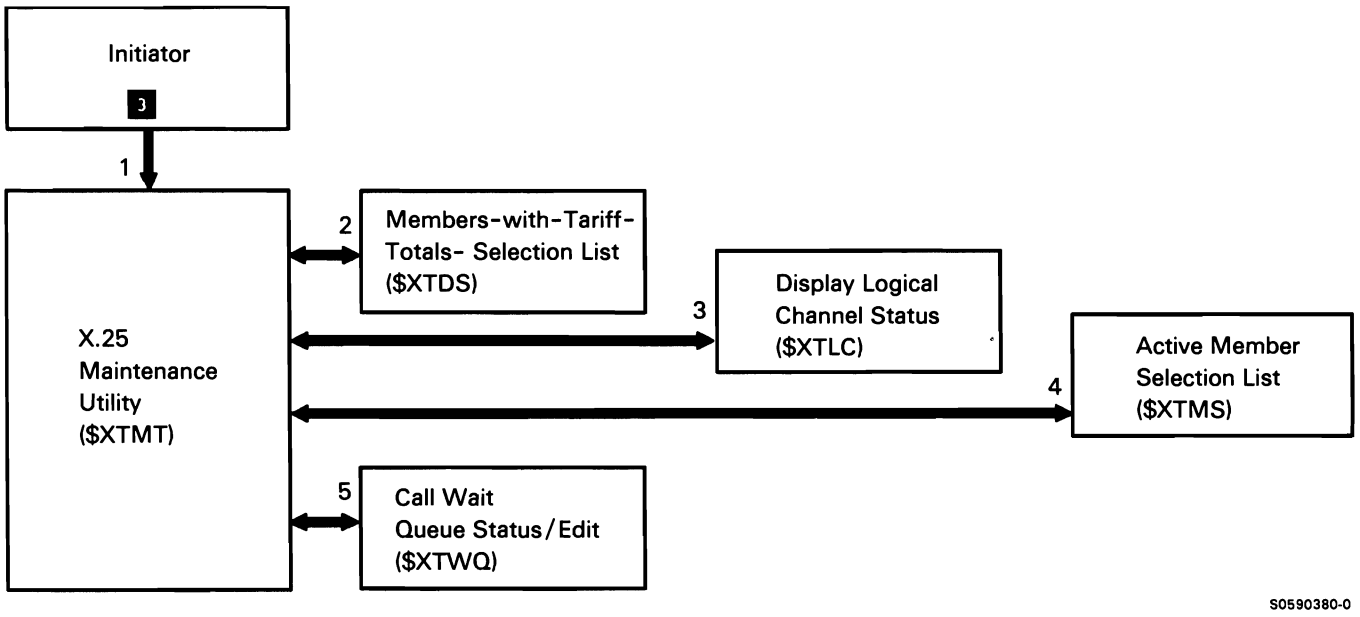


Chart 9.8.4.2 X.25 Maintenance Utility Control Flow

X.25 Phone List Utility

The X.25 phone list utility creates a new X.25 phone list load member, or alters, removes, or prints an existing X.25 phone list load member. The X.25 phone list utility is invoked by the DEFINX25 system procedure or equivalent OCL.

The following X.25 phone list utility processes are shown in Chart 9.8.4.3:

1 Do the following:

- Perform the operation requested.
- Route for the following:

2 Syntax check utility control statements.

3 Issue read/write requests for phone list load members.

4 Issue any required messages.

5 Terminate this job step.

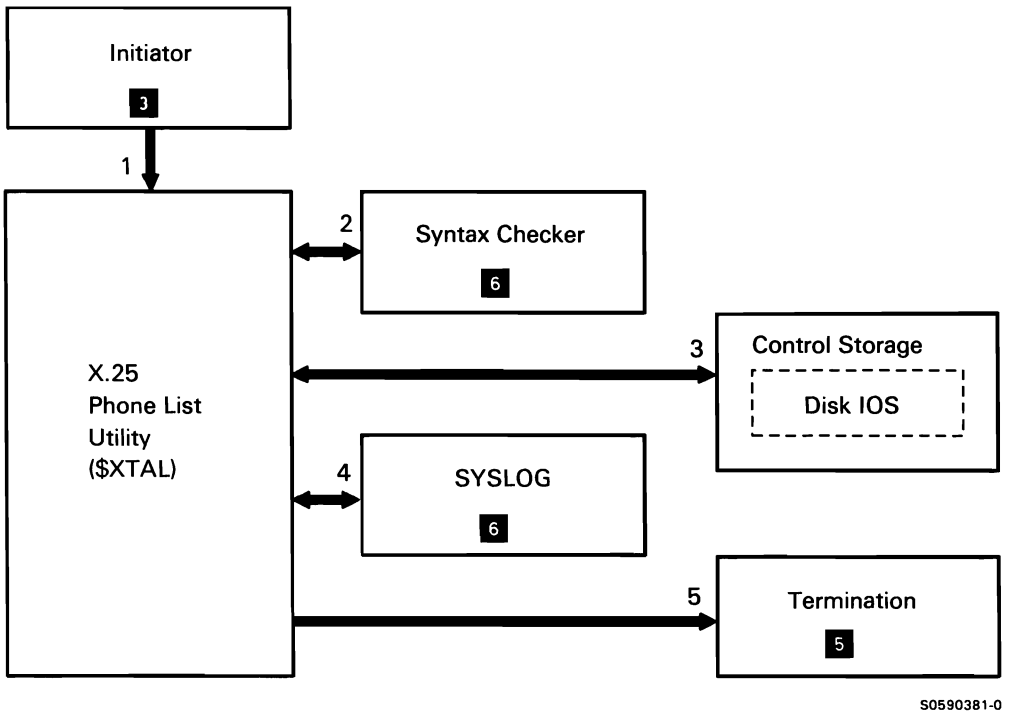


Chart 9.8.4.3 X.25 Phone List Utility Control Flow

This page is intentionally left blank.

DISPLAY STATION PASS-THROUGH SUPPORT

The Display Station Pass-Through support allows a user working at a display station attached to a System/36 or System/38 to sign on to a remote System/36 or System/38 and run applications on the remote system. There are two tasks associated with each display station pass-through session. The System/36 source task is an SRT and runs on the System/36 that the display station is locally attached to. The System/36 target task is an MRT and runs on the remote System/36.

Source Display Station Pass-Through Support

The System/36 Source Display Station Pass-Through support consists of initialization of the display station pass-through session on the source system and the operation request processing.

The operation request processing consists of accepting and decoding requests from the target task and issuing the appropriate WSDM operation. Upon completion of WSDM, a reply is sent to the target task with input data if there is any.

The following source display station pass-through overview processes are shown in Chart 9.9.1:

1 Receive control from the initiator and perform the initial setup.

2 Evoke the target display station pass-through program.

3 Receive operation requests from the target display station pass-through subfunction.

4 Decode the operation request and issue the appropriate operation to WSDM.

5 For message light requests, issue the request to WSIOCH if it is a local display station.

6 For message light requests, issue the request to RWS if it is a remote display station.

7 Send operation reply to the target display station pass-through subfunction upon completion of the operation.

8 For error messages, use SYSLOG.

9 When end-of-transaction is received, go to end of job.

10 Perform source cleanup for source termination.

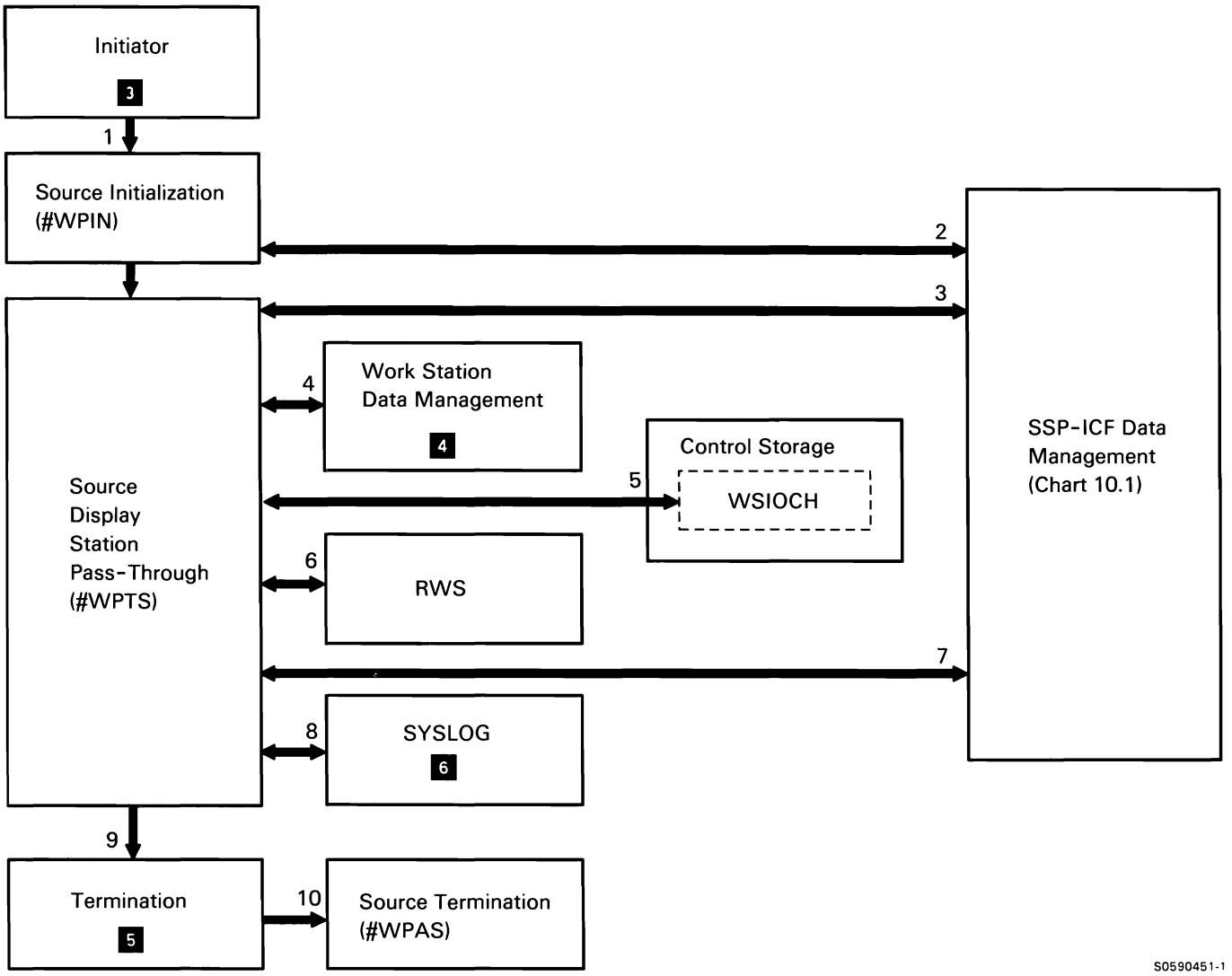


Chart 9.9.1 Source Display Station Pass-Through Support Control Flow

Target Display Station Pass-Through Support

The System/36 Target Display Station Pass-Through support allows display station requests from applications to be directed to displays attached to remote systems.

The Target Display Station Pass-Through support runs as a MRT that is loaded by the display station pass-through target procedure (X'7B30F0F0F5').

This task builds the necessary display station pass-through resources needed on the target system (vary on of a display station pass-through TUB) and accepts operation requests directed to the display station pass-through TUBs.

1 Receive control from the initiator and perform initial setup.

2 Accept any new requesters as they are attached to the target MRT.

3 For new requesters, vary on a display station pass-through TUB.

4 Accept posts from WSDM to do work on a display station pass-through TUB.

5 Send an operation request to the source display station pass-through subfunction.

6 Receive reply from source display station pass-through for operation completion and process operation completion.

7 Handle command processor operation requests such as message light requests.

8 For System/38 attached display station errors, map the negative response data to the TUB sense bytes and call $NUPD so user program gets a permanent error return code.

9 For display station errors, call $NUPD so user program gets a permanent error return code.

10 For vary off, call $NUPD to sign off the display station pass-through TUB.

11 For vary off, stop outstanding invites on display station pass-through TUBs.

12 End of task if no display station pass-through TUBs remain.

13 For abnormal termination, vary off all display station pass-through TUBs.

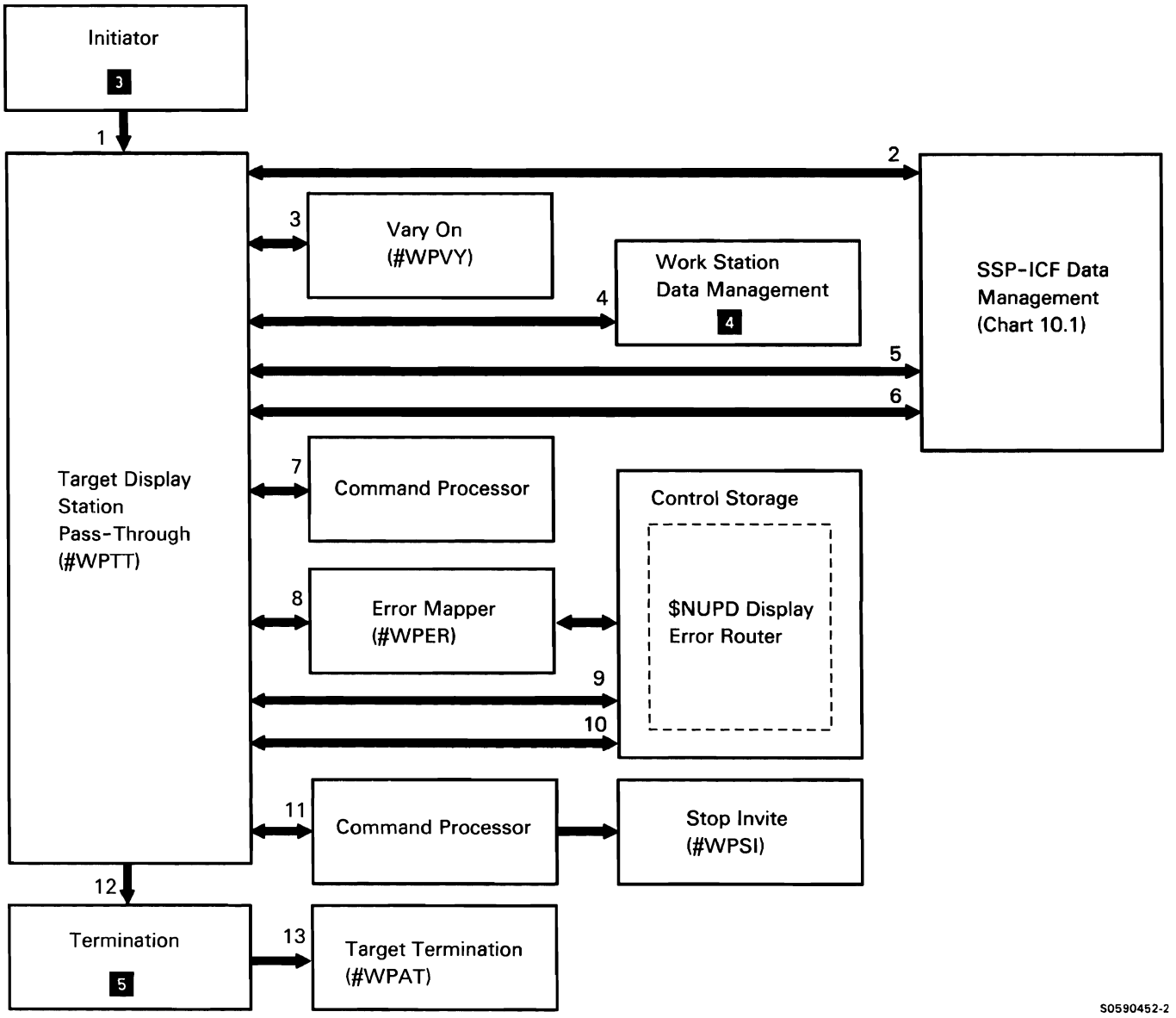


Chart 9.9.2 Target Display Station Pass-Through Support Control Flow

SUPPORT SUBFUNCTION FOR THE LAN

The Communications SSP feature for the local area network (LAN) for System/36 enables an application program running under the System/36 SSP to communicate with work stations and other systems through an IBM Token-Ring Network. The programming support works in conjunction with up to two local area network adapter cards located in either the console IBM PC (5364) or a channel-attached IBM Personal Computer AT® controller (5360/62). When used to communicate over the IBM Token-Ring Network, the programming for the LAN conforms to IEEE 802.2 and IEEE 802.5.

System/36 can communicate with up to 50 link stations (work stations and other systems) on each adapter card. The communicating link stations can be a mixture of primary and negotiable link stations.

The user application does not directly access the programming of the LAN. Instead, the programming of the LAN is accessed via an applicable SNA task. The SNA tasks access the programming of the LAN via the DLC resident interface (#SDRI), and all communication from the SNA tasks to the resident interface and the programming of the LAN is done via communications IOBs. (See Chart 9.2.3.)

The following SNA tasks are users of the programming subfunction for the LAN:

- Remote work station support (Chart 9.6.0)

  Note: Remote work station support is available only when the link station emulates a 327X terminal.

- Link station test (Chart 9.7)

- C/SNA (for APPC, MSRJE, SNUF, and SNA 3270 device emulation subsystems)

®Personal Computer AT is a registered trademark of IBM.

Chart 9.10.0 shows the overview control flow for the LAN:

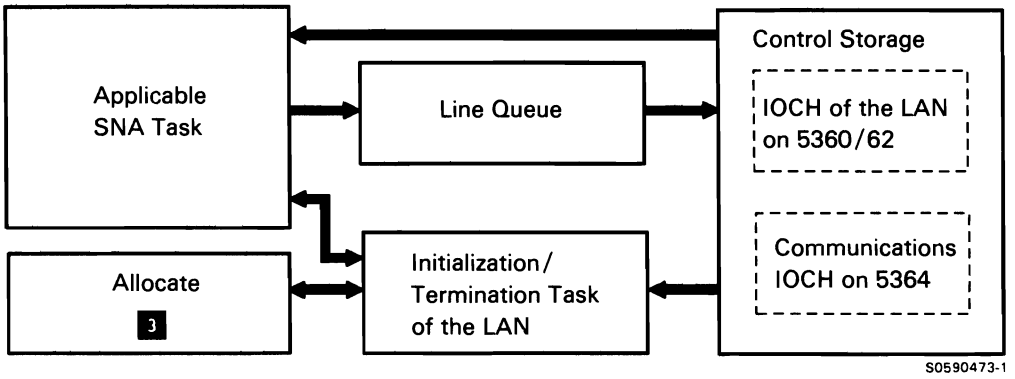


Chart 9.10.0 Overview Control Flow of the LAN

This page is intentionally left blank.

The LAN uses three data areas in processing IOBs:

There is a *communications specification block (CSB)* for each line used by the LAN. Each CSB has a CSB extension called the common area that contains status information for the respective line.

The *SNA list* contains entries for each of the SNAs that is communicating. It is pointed to by the CSB. The SNA list contains a status entry for each SNA using the line and possibly a list of IOBs that are awaiting completion of the enable. These entries are chained together and contain status information for the SNA such as whether or not its line enable has completed.

The *line queue* contains all IOBs issued to the control storage microcode of the LAN. The line queues (one per line) are contained in SCA and are pointed to by queue headers.

Note: An IOB on the transmit or line queue causes control flow to or from control storage as shown in Chart 9.10.0. In this and subsequent control flow charts for the LAN, these queues are shown receiving and passing control. This is done to simplify the control flow chart and should in no way imply that the queues contain free-standing, executable code.

The LAN consists of three basic parts:

- Initialization/termination task of the LAN (Charts 9.10.1.1 and 9.10.1.2)
- SNA-to-DLC resident interface (Chart 9.2.3)
- Abnormal termination routines of the LAN (Chart 9.10.3)

Initialization/Termination Task of the LAN

The initialization/termination task of the LAN performs initialization and termination processing for the LAN as well as message handling for the LAN. It is a swappable task.

The following initialization processes of the LAN are shown in event sequence in Chart 9.10.1.1:

1 Perform initial processing of line open request; get SNA-to-DLC resident interface loaded. (#SDRI contains code that loads the resident interface routine.)

2 Allocate the line and CSB and assign read and internal IOBs.

3 Return line open parameter list complete.

4 Process transmit IOB containing enable command.

5 If necessary, queue IOBs FIFO to the SNA list.

6 Issue download IOB to download personal computer controller.

7 Post download IOB complete to initialization/ termination task of the LAN.

8 If any of the following conditions exist, issue a message:

   - Initial diagnostic failure
   - Personal computer not powered on

9 Issue reserved read IOB and process enable command.

10 Post enable IOB complete to initialization/ termination task of the LAN.

11 If any of the following conditions exist, issue a message:

- Personal computer adapter malfunction
- Local name conflict
- Configuration of the LAN invalid
- Controller resource not available
- Unusual network condition

12 Post enable IOB complete to SNA.

13 If this is the first enable, issue read IOBs.

14 If necessary, remove IOBs from the SNA list and process.

15 Accept transmit IOB containing open station command if it is not already queued to the SNA list.

16 Process open station command.

17 Post open station IOB complete to initialization/ termination task of the LAN.

18 If any of the following conditions exist, issue a message:

- Network session limit reached
- Unusual network condition
- Controller resource not available

19 Post open station IOB complete to SNA.

20 Accept transmit IOB containing XID command.

21 Process XID command.

22 Accept pended XID if necessary.

23 Accept request for return of XID command if necessary.

24 Reprocess XID command.

25 *At any time in above sequence*, process errors requiring abnormal termination.

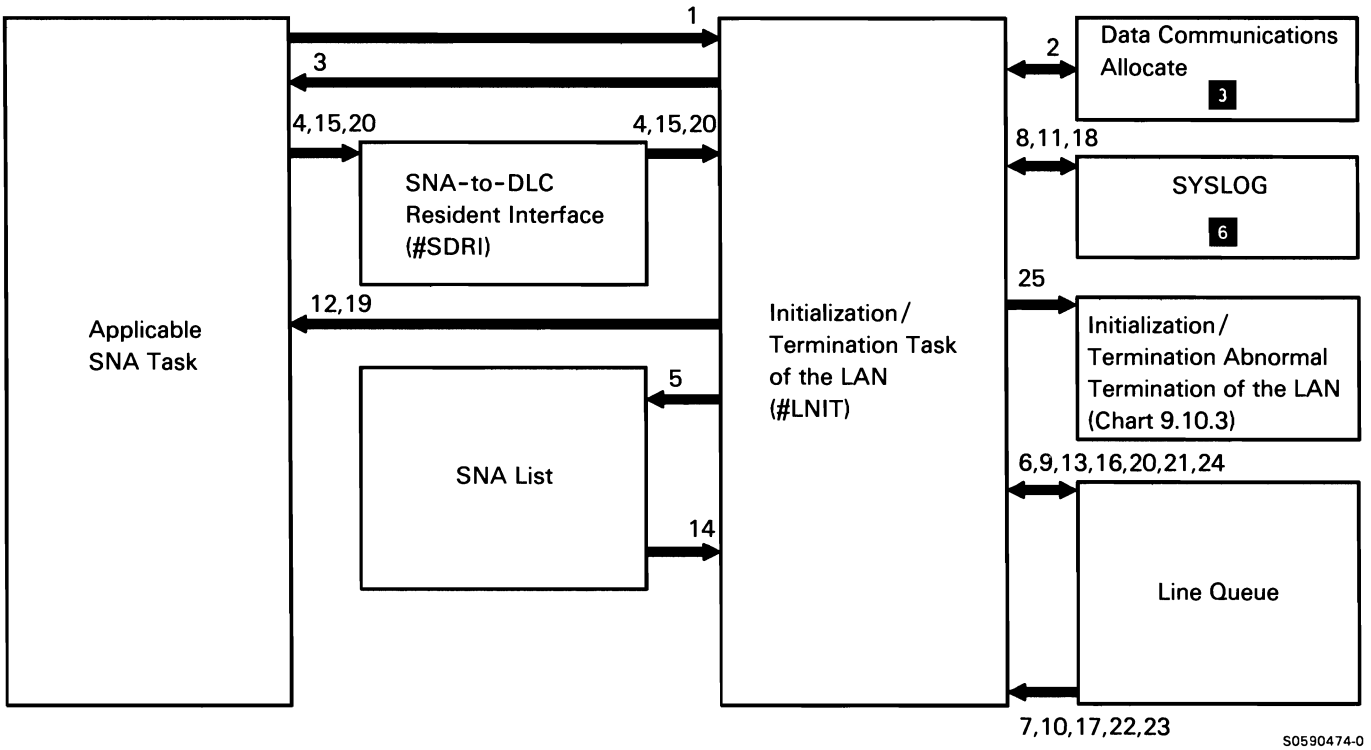


Chart 9.10.1.1 Initialization of the LAN Control Flow

The following termination processes of the LAN are shown in event sequence in Chart 9.10.1.2:

1 Process transmit IOB containing DISCONNECT command.

2 Post SNA with all previously outstanding transmit IOBs for disconnected station with request ignored return code.

3 Post disconnect IOB complete to initialization/ termination task of the LAN.

4 Post any pended IOBs complete to the SNA with request ignored completion.

5 Post disconnect IOB complete to SNA.

6 Process terminate IOB.

7 Post SNA with all previously outstanding transmit IOBs with request ignored return code.

8 Post terminate IOB complete to initialization/ termination task of the LAN.

9 Post any pended IOBs complete to the SNA with request ignored completion.

10 Post terminate IOB complete to SNA.

11 Process terminate/detach IOB.

12 Post SNA with all previously outstanding transmit IOBs with request ignored return code.

13 Post terminate/detach IOB complete to initialization/ termination task of the LAN.

14 Post any pended IOBs complete to the SNA with request ignored completion.

15 Post terminate/detach IOB complete to SNA.

16 If this was the last SNA task using the line, issue disable IOB.

17 Deallocate line.

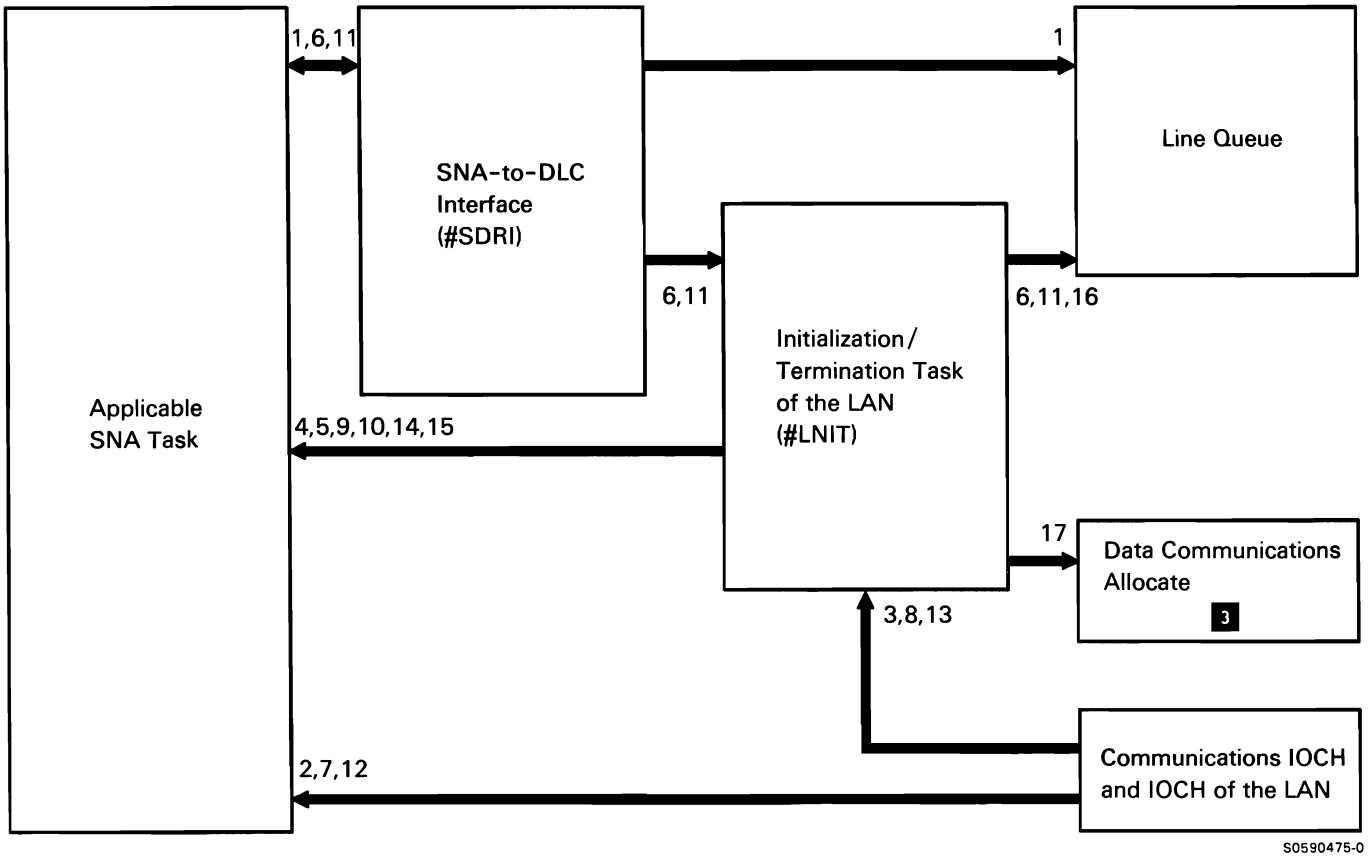


Chart 9.10.1.2 Termination of the LAN Control Flow

Controller Check of the LAN

A controller check of the LAN occurs when initialization/termination of the LAN receives an IOB from IOCH of the LAN or communications IOCH with a completion code of:

- Controller check of the LAN (5360, 5362, or 5364)
- Cable parity error (5360 or 5362)
- Adapter check of the LAN (5360 or 5362)

The SRC in the IOB further identifies the cause of the error.

A controller check is a nonrecoverable error for every active line of the LAN in the personal computer.

The following controller check processes of the LAN are shown in Chart 9.10.2:

1 IOCH informs initialization/termination of the LAN of a controller check condition.

2 Post controller check completion to SNA.

3 Issue disable IOCH.

4 Accept controller check completions in additional IOBs.

5 If necessary, post IOBs to SNA with controller check completion.

6 Queue IOBs to SNA list FIFO.

7 Issue controller check message.

8 Accept disable IOCH complete.

9 SNA issues terminate/detach.

10 Dequeue pended IOBs from SNA list.

11 Post IOBs to SNA with controller check completion.

12 Deallocate line of the LAN.

13 Allocate line for diagnostic wrap test of the system adapter of the LAN.

14 Issue wrap test message.

15 Issue wrap test for 5360 and 5362.

16 Accept wrap test complete.

17 Deallocate line for diagnostic wrap test.

18 Issue wrap test results message and go to end of job.

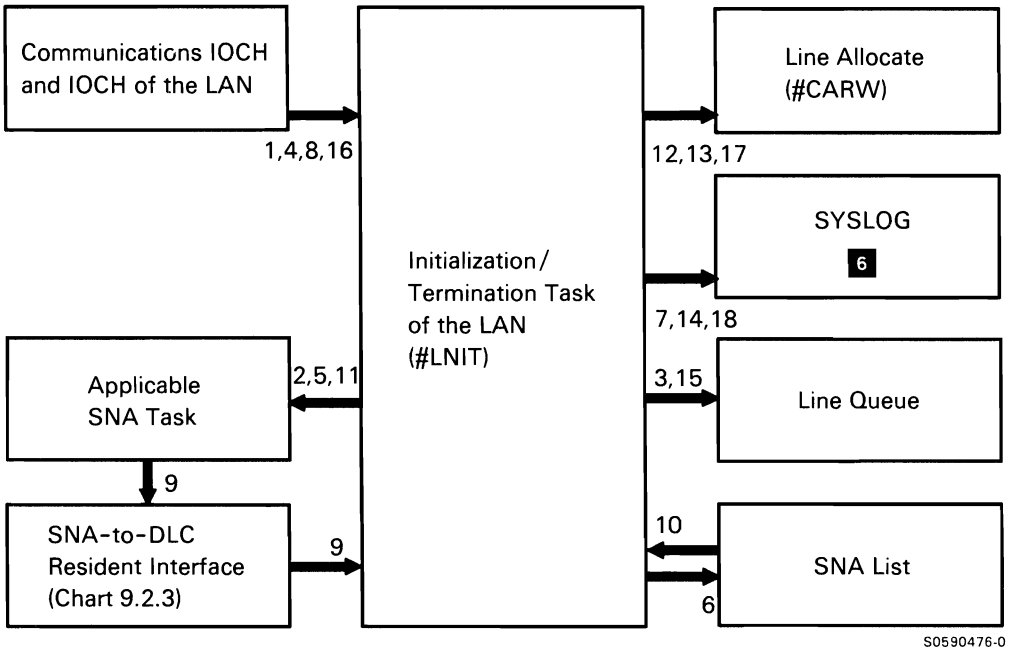


Chart 9.10.2 Controller Check of the LAN Control Flow

Initialization/Termination Abnormal Termination Task of the LAN

The initialization/termination abnormal termination task of the LAN will receive control if:

- Initialization/termination of the LAN received an IOB with an invalid completion code from the IOCH of the LAN or communications IOCH.
- Initialization/termination of the LAN received an IOB from IOCH with an operation code that it was not prepared to handle.
- Initialization/termination of the LAN received an invalid IOB from SNA.
- A 2K-page failure occurred.
- Initialization/termination of the LAN initiated abnormal termination because of some situation (other than above) that should never occur.

Abnormal termination can be evoked from initialization/ termination of the LAN when it recognizes a severe error. Control storage termination can also recognize an abnormal termination error. In both cases, control storage termination will load #LNAT in place of #LNIT before abnormal termination processing begins.

The following initialization/termination abnormal termination processes of the LAN are shown in Chart 9.10.3:

1 Issue abnormal termination message.

2 Issue disable line IOB for all lines (networks) and wait for them to complete.

3 Post all IOBs from line queue.

4 Post SNA IOBs with abnormal termination completion.

5 Dequeue SNA IOBs from SNA list.

6 Post SNA IOBs with abnormal termination completion.

7 Receive terminate/detach from SNA.

8 Process for terminate/detach.

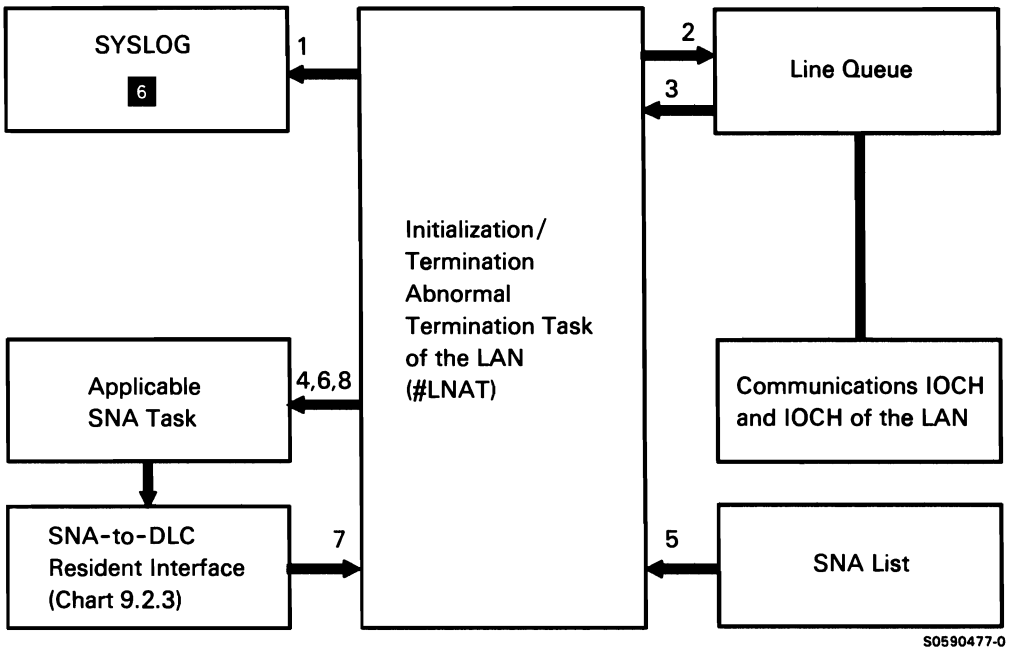


Chart 9.10.3 Initialization/Termination Abnormal Termination Task of the LAN Control Flow

SSP-ICF Function 10

Note: The descriptions of several separately orderable features are included in this function. The content and organization of this function are based solely on IBM software ease-of-service considerations. In this and in other parts of this manual, the term *SSP-ICF* is used in describing program components of the SSP-ICF feature as well as components of other program features closely related to SSP-ICF. The reader should infer nothing about the content or packaging of any IBM Sales Features from the content and organization of the SSP-ICF function, as presented in this manual.

The SSP-ICF function provides user interfaces that allow application programs on System/36 to communicate with another System/36, System/38, System/34, System/3, System/370, and other systems that use compatible BSC or SNA/SDLC protocols. It also allows two or more programs on the same System/36 to simulate remote interactive communications.

The SSP-ICF function consists of the SSP-ICF (SSP Interactive Communications Feature) and other System/36 features.

Note: SSP-ICF subfunctions that are used by more than one component contain control flow blocks that refer to the *appropriate subsystem* or to the *appropriate interrupt handler*. When you need more information on one of these blocks, refer to the following subfunction list to find the chart reference for the appropriate subsystem or interrupt handler. The term *interrupt handler*, when used in this section, refers only to *main storage* SSP-ICF interrupt handlers.

This function includes the following SSP-ICF subfunctions:

- SSP-ICF data management subfunction — Chart 10.1
- SSP-ICF control subfunction — Chart 10.2
- SSP-ICF enable/disable subfunction — Chart 10.3.1
- C/SNA subfunction — Chart 10.4.0
- SSP-ICF user aids subfunction: — Chart 10.5.0
  - SSP-ICF debug — Chart 10.5.1
  - SSP-ICF verify — Chart 10.5.2
- SSP-ICF program-to-program subsystems subfunction:
  - SSP-ICF BSC link control — Chart 10.6.1
  - Intra — Chart 10.6.2
  - IMS/IRSS — Chart 10.6.3
  - CICS — Chart 10.6.4
  - BSC CCP — Chart 10.6.5
  - BSCEL — Chart 10.6.6
  - SNA Upline Facility (SNUF) — Chart 10.6.7
  - Finance — Chart 10.6.8
  - Peer — Chart 10.6.9
  - APPC — Chart 10.6.10
  - APPN — Chart 10.6.11.n
  - Asynchronous communications — Chart 10.6.12
- MSRJE support subfunction — Chart 10.7.0
  - MSRJE BSC interrupt handler (main storage) — Chart 10.7.1
  - MSRJE BSC subsystem — Chart 10.7.2
  - MSRJE SNA subsystem — Chart 10.7.3
  - Reader/console task — Chart 10.7.4
  - Printer/punch — Chart 10.7.5
  - MSRJE forms control table utility — Chart 10.7.6
  - MSRJE disk file utility — Chart 10.7.7
- 3270 support subfunction: — Chart 10.8.0
  - BSC 3270 interrupt handler (main storage) — Chart 10.8.1
  - BSC 3270 subsystem — Chart 10.8.2
  - BSC 3270 display emulation — Chart 10.8.3
  - BSC 3270 printer emulation — Chart 10.8.4
  - SNA 3270 subsystem — Chart 10.8.5
  - SNA 3270 display emulation — Chart 10.8.6
  - SNA 3270 printer emulation — Chart 10.8.7
  - SNA 3270 personal computer support — Chart 10.8.8
- Communications and Systems Management (C & SM):
  - Change management — Chart 10.9.1
  - Problem management — Chart 10.9.2.n
- Personal computer support subsystem — Chart 10.10.n

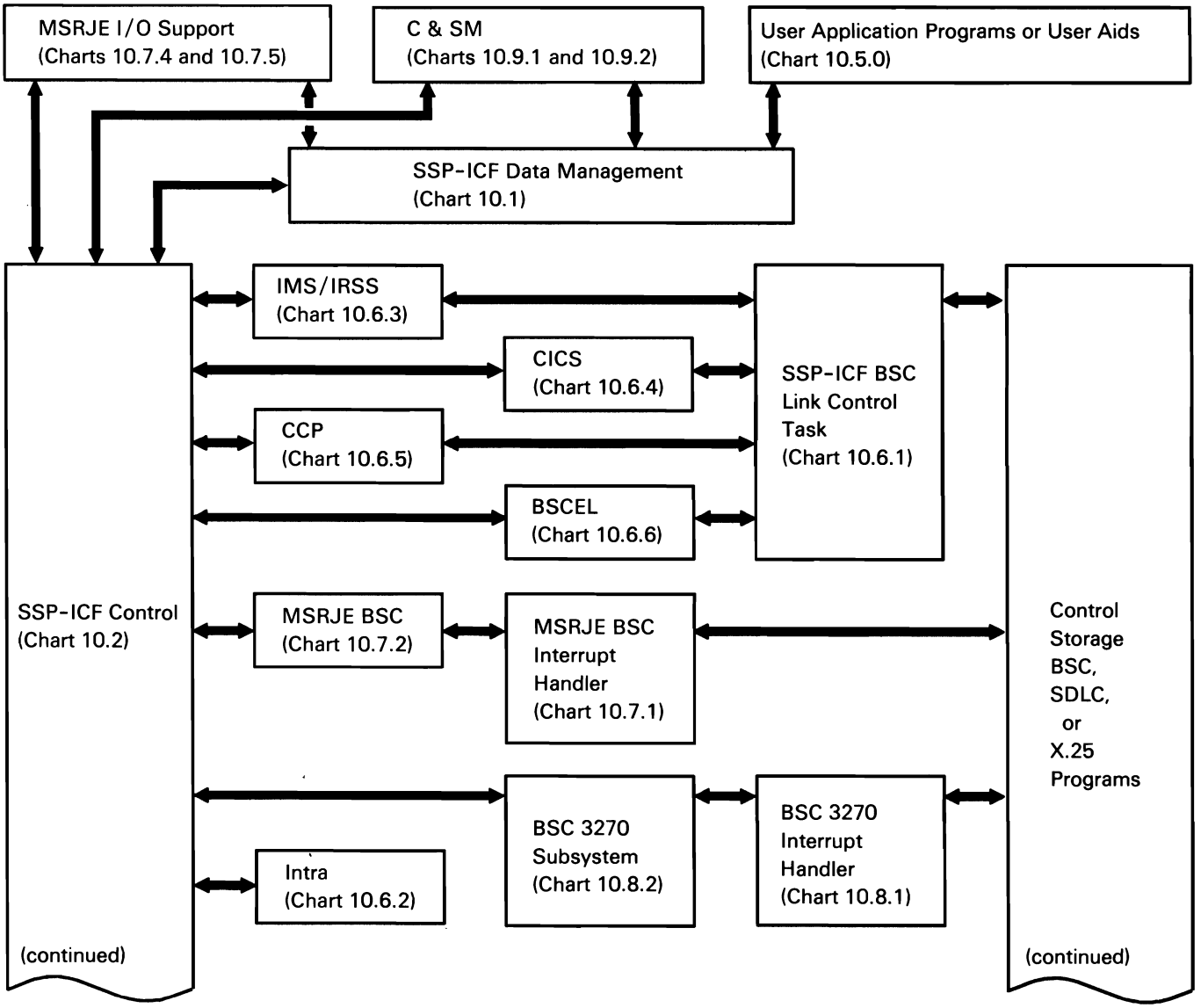


Chart 10.0 (Part 1 of 2) SSP-ICF Function Overview

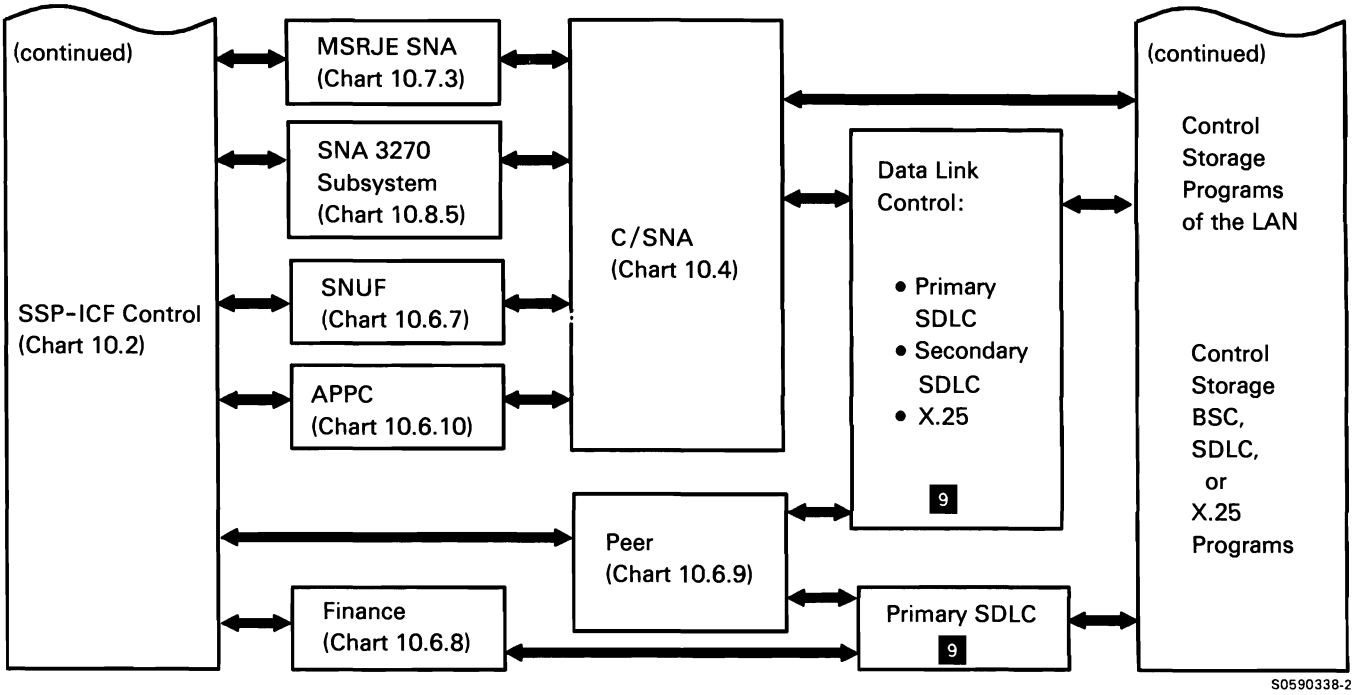


Chart 10.0 (Part 2 of 2) SSP-ICF Function Overview

SSP-ICF DATA MANAGEMENT SUBFUNCTION

SSP-ICF data management provides the interface between application programs and SSP-ICF.

Note: Although SSP-ICF data management is shown in this manual as a separate program module, it actually executes as a system transient running under the application program task. It passes function requests and data from the application program to the applicable SSP-ICF subsystem and passes data and data-related information from the applicable SSP-ICF subsystem to the application program.

Requests from the application program to SSP-ICF are issued to the work station data management router (#DWDM) in the form of a transfer SVC instruction with XR2 pointing to a DTF, which points to the user's logical buffer. When #DWDM determines that the request is valid and is for an SSP-ICF subsystem, it builds a work station parameter list (WSPL) from information contained in the DTF. (For privileged application programs, the request may already be in the form of a WSPL). #DWDM passes the WSPL to SSP-ICF data management via the fast transfer SVC instruction. SSP-ICF data management converts the request in the WSPL to a request in an SUB (built at acquire or evoke time) and posts the request to the appropriate subsystem via SSP-ICF control. If the request was for a no wait-type operation, SSP-ICF data management exits to the application program with the WSPL or DTF (depending upon which was originally passed to the work station data management router). If the request was for a wait-type operation, SSP-ICF data management waits for a post from the subsystem before returning the WSPL or DTF to the application program.

The following processes are shown in Chart 10.1:

1 Process user program request for a subsystem by building a work station parameter list (WSPL) based on information contained in the DTF and fast transferring to SSP-ICF data management.

2 Process application program request for a subsystem:

- Check session ID.
- If required, convert high-level-language request to specific subsystem request.
- Check for valid operation code.
- Convert request in WSPL to equivalent request in SUB.
- Route acquire and get attributes to #ICDC.
- Route the remaining operations to #ICDN.

Continue processing application program request for a subsystem:

- Route operations other than put, get, and evoke to appropriate processors.

Perform additional required processing for put, get, and evoke operations (calling subsystem subroutines when necessary):

- Assign required SQS.
- Map user area and TWS.
- Process output (put or evoke) requests by changing the request to the format required by the subsystem and assigning and moving data from the application program to a TWS and/or SQS buffer area.
- Post the subsystem.
- On return from subsystem for input (get) request, move data from the subsystem TWS buffer to the application program buffer.

When request is processed, exit to application program with appropriate completion code.

3 If IDDU request:

- Check session ID.
- Convert IDDU communications formats to a valid WSPL operation code.
- For evoke:
  - Assign required SQS and map to user area and TWS.
  - Change request to the format required by the subsystem; assign and move data from the application program to a TWS and/or SQS buffer area.

4 Process return code and any data returned.

5 Process INPUT, OUTPUT, or EVOKE requests.

6 If DDM SUB, do not process unless WDMDL bit is on or EOJ is requested. Otherwise:

Post user and subsystem for session termination for the following requests:

- Release.
- End of step.
- End of session.
- End of transaction.

7 Perform session cleanup after termination.

8 Process the message when an operation is posted with an error return code, by mapping the return code to the applicable MIC and performing any user-required error recovery.

9 Build SUB, post subsystem for acquire operation, and wait for subsystem response.

10 Process APPC subsystem get-attributes and get-status operations.

11 Retrieve information for get-attributes and get-status operations.

12 Process APPN implicit enable.

When SSP-ICF data management posts the subsystem via SSP-ICF control, XR1 contains the wait list displacement and XR2 points to the SUB.

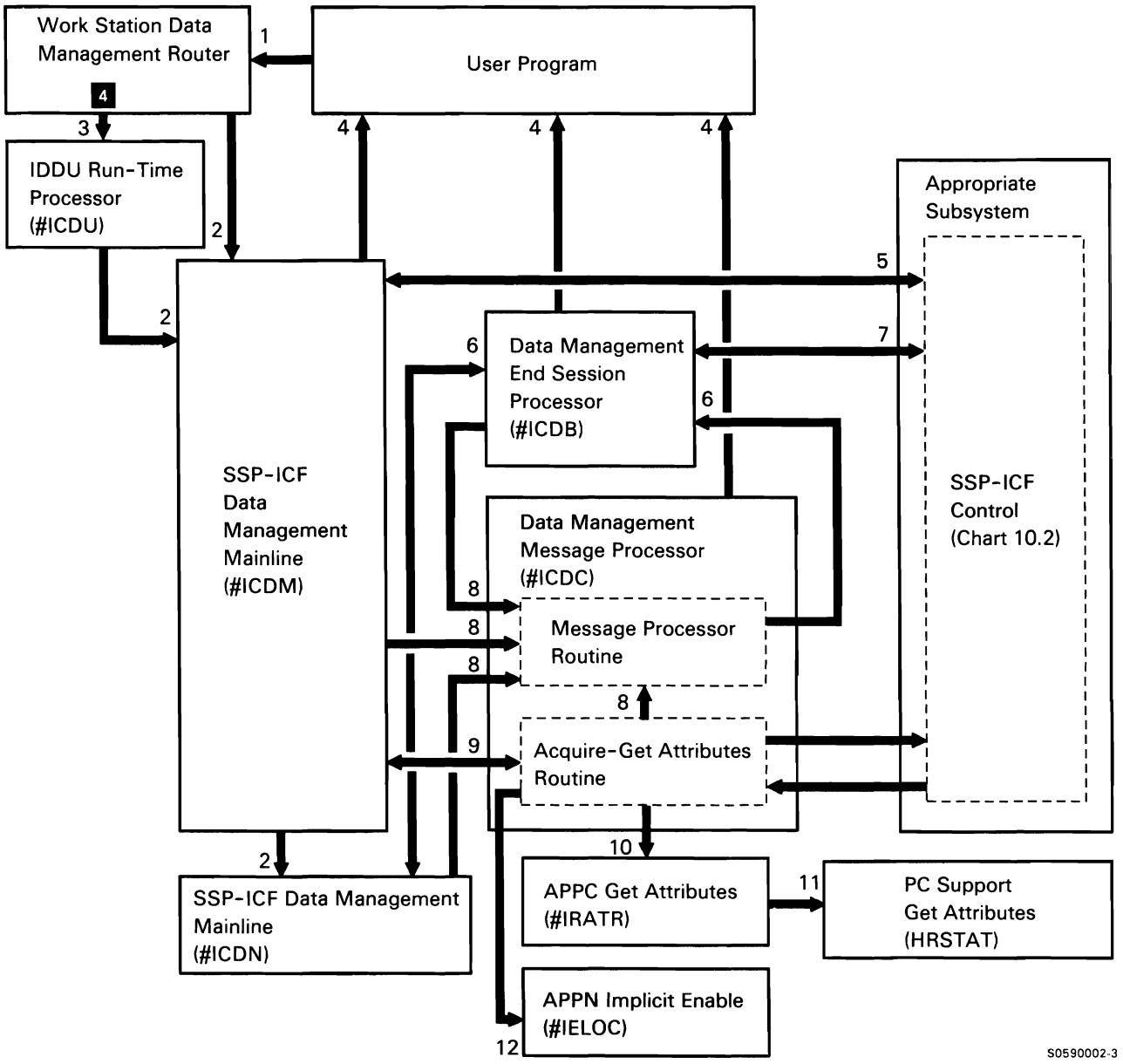


Chart 10.1 SSP-ICF Data Management Control Flow

This page is intentionally left blank.

SSP-ICF CONTROL SUBFUNCTION

SSP-ICF control contains subroutines that allow SSP-ICF subsystems to communicate with SSP-ICF data management, the data communications (main storage) interrupt handlers, and other SSP and SSP-ICF components.

Note: Although SSP-ICF control is shown in this manual as a separate program module, it actually executes as a part of the applicable subsystem task.

Essentially, SSP-ICF control is a wait list that is posted by SSP-ICF data management on behalf of the local application program or by the appropriate interrupt handler on behalf of the remote subsystem, or equivalent remote program. These programs post SSP-ICF control with XR1 containing the wait list displacement and XR2 pointing to the appropriate control block.

When posted and when all pending operations are completed, the applicable subsystem task performs the operation it is directed to by SSP-ICF control.

SSP-ICF control resides at logical hex address 1000 in the subsystem task, with an optional 2K-byte transient area following it at hex 2800. It performs the following functions:

1 Route the subsystem work request to the requested SSP-ICF control routine. When subsystem is idle, route to wait routine for work.

2 Route request posted from SSP-ICF enable/disable or SSP-ICF data management to the proper subsystem function. Give control to the subsystem at the address specified in the wait list for the post type received.

3 Manage task work space (TWS) and system queue space (SQS).

4 Start required programs for subsystems by scheduling an SVC command request to command processor program start. The request, accompanied by as many as 508 bytes of data, is attached to either an active MRT or to the initiator, which processes a procedure call.

5 Route for error recovery for subsystems.

6 Log trace entries for subsystems.

7 Manage subsystem's 2K-byte transient area.

8 Move data from real to translated storage.

9 Move XSUB to SUB.

10 Move SUB to XSUB.

11 Post SSP-ICF data management (post users) for subsystem, then return to subsystem.

12 Post the data link control, then return to subsystem.

When SSP-ICF control passes control to the applicable subsystem for a new XSUB request, XR2 points to the XSUB. The XSUB points to the user data in the TWS buffer and the BUB (which points to the line buffer) for most BSC subsystems, or the SNUB (which points to the intermediate buffer) for SDLC subsystems.

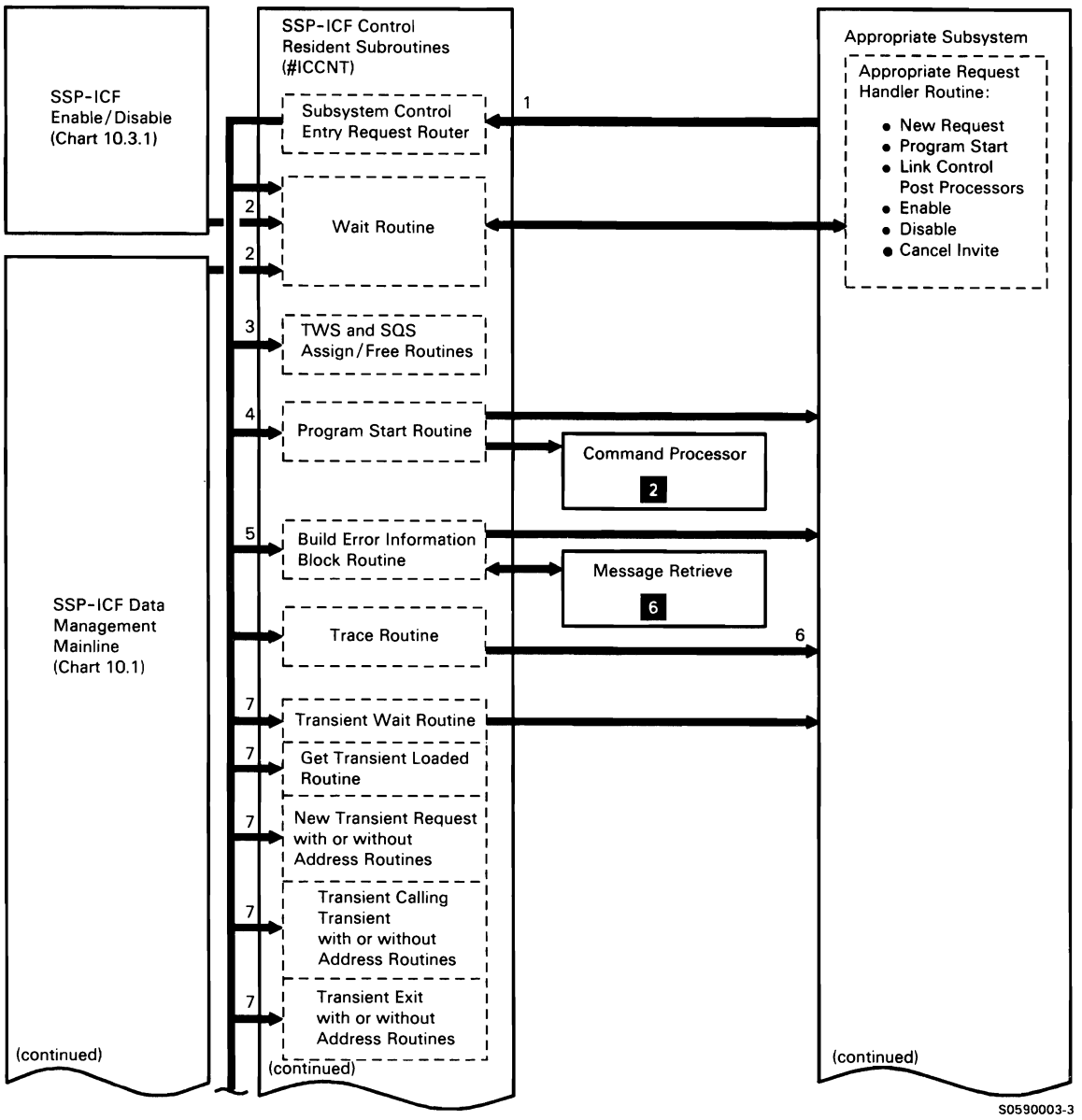


Chart 10.2 (Part 1 of 2) SSP-ICF Control Control Flow

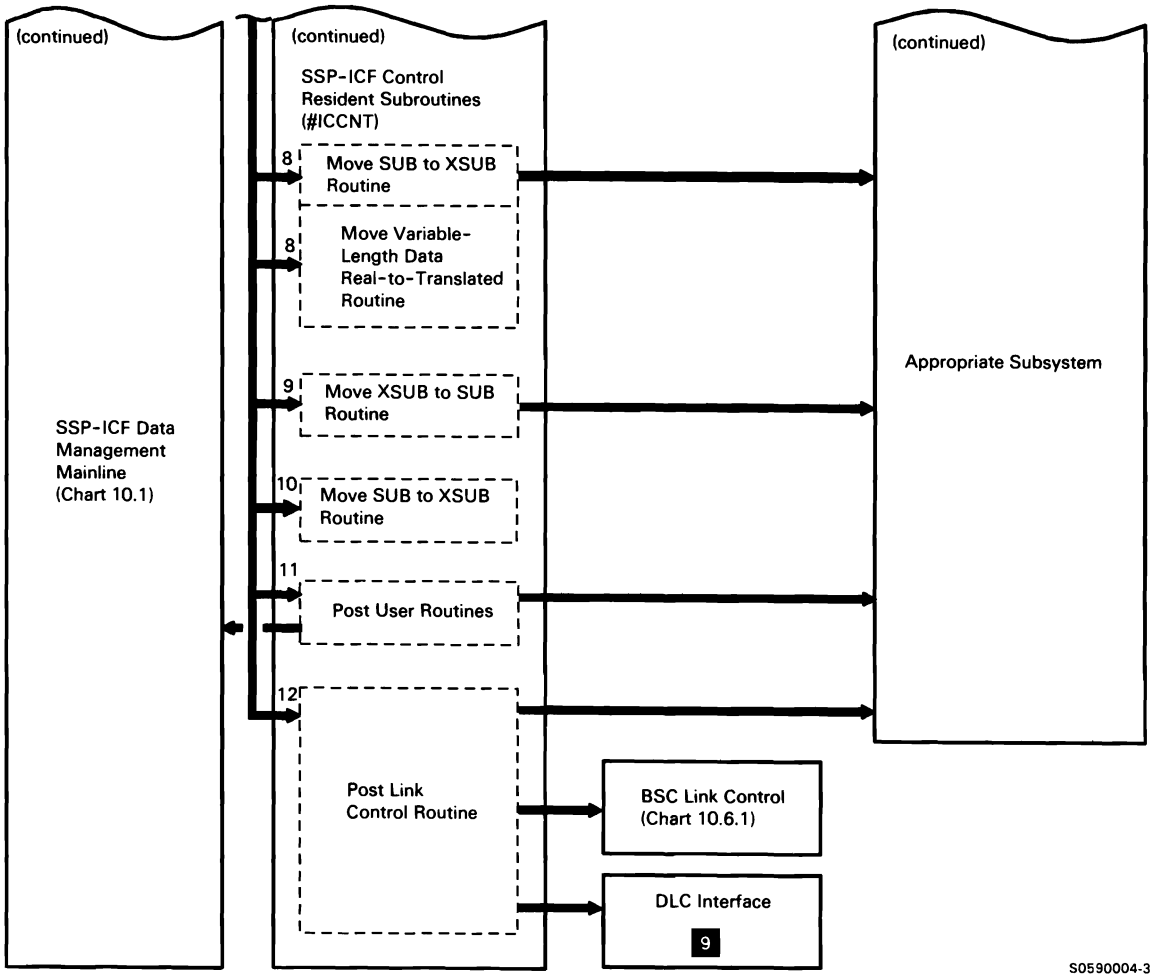


Chart 10.2 (Part 2 of 2) SSP-ICF Control Control Flow

SSP-ICF ENABLE/DISABLE SUBFUNCTION

Enable

SSP-ICF enable activates an SSP-ICF subsystem configuration and, if appropriate, attaches the subsystem and associated communications tasks. Enable builds the required subsystem control tables (SCTs) and location name elements (LOCs) to activate the subsystem. Enable is invoked by the ENABLE procedure.

The following enable processes are shown in Chart 10.3.1:

1 Check the syntax of parameters passed in the ENABLE procedure.

2 If SHOW is specified, review configured subsystem parameters by displaying the subsystem configuration member.

3 Read the line member and subsystem member. Ensure that members are at current levels and that the data matches the data communications features in the display station configuration member.

4 Do the following:

- Assign an SCT and add it to the SCT chain.
- If a single-location subsystem (BSC) is being enabled, assign an LOC for the location.
- Set up a TEB to return control to the appropriate subsystem module in the event of abnormal termination.
- For the first enable after IPL, assign the SSP-ICF communications area and save its address in the SCA.
- If trace is active, assign and initialize a trace area.

5 On subsequent enables, scan for the active SCT and subsystem task ID. Ensure that the line member and the subsystem member are the same. Post the subsystem.

6 If subsystem that allows multiple locations, build an LOC for each. If APPC subsystem, assign associated SGB(s).

7 On the first enable of a subsystem, load and attach the appropriate link control task and allocate the line.

8 On the first enable of a subsystem, load the subsystem code and transfer control to SSP-ICF control to begin subsystem processing.

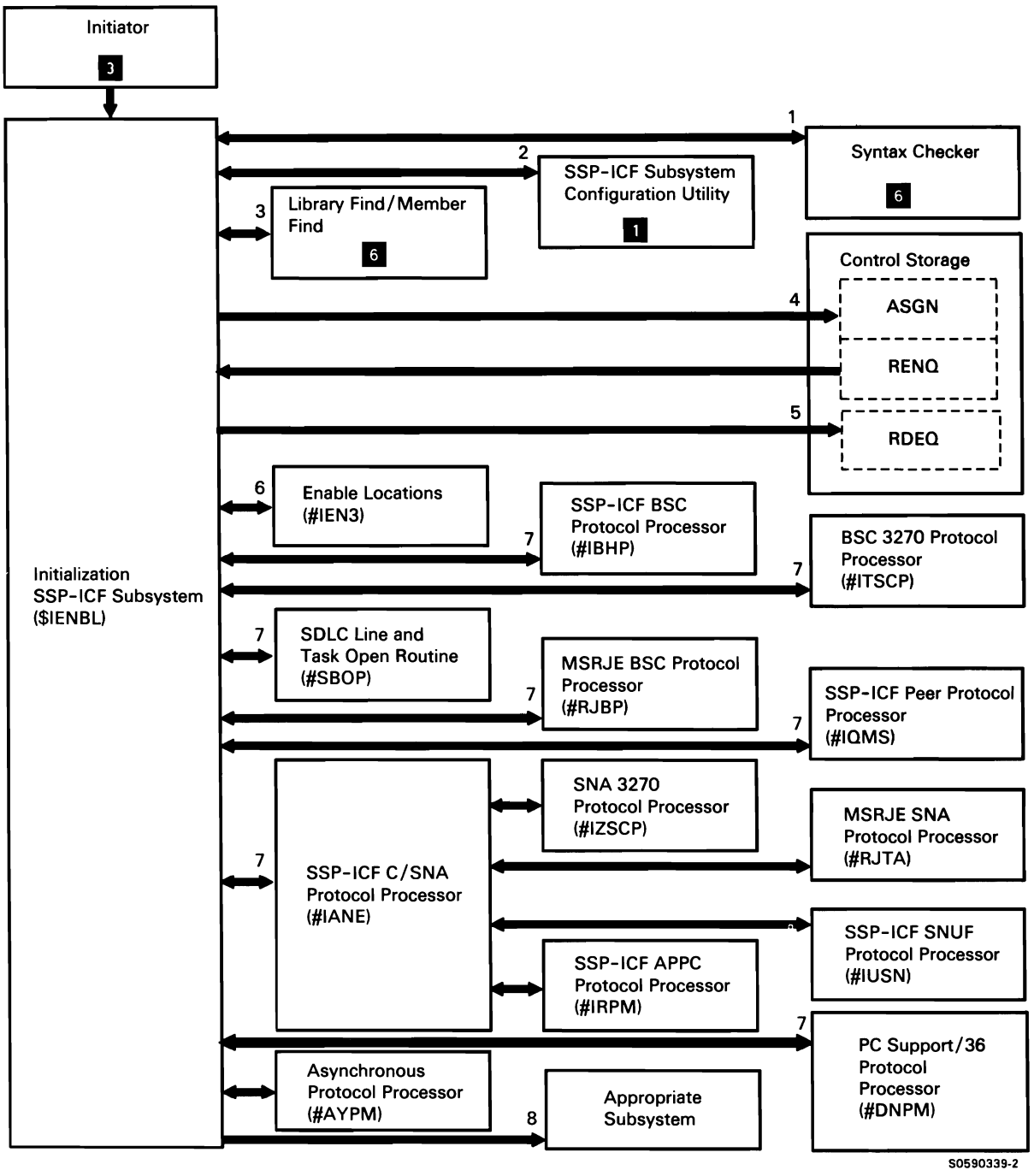


Chart 10.3.1 SSP-ICF Enable Control Flow

Disable

SSP-ICF disable deactivates an SSP-ICF subsystem configuration and, if appropriate, detaches the subsystem and communications tasks. Disable removes the subsystem control tables (SCTs) and location name elements (LOCs) to terminate the subsystem and associated communications tasks. Disable can be invoked by the DISABLE procedure; subsystems can also call disable to perform selected disable functions.

The following disable processes are shown in Chart 10.3.2:

1 Check the syntax of parameters passed in the DISABLE procedure.

2 Enqueue the SCT chain; search chain to ensure that subsystem is enabled.

3 Process request for the disabling of a single location in the subsystem:

- Ensure that location is enabled and that disable is not in progress.
- For APPN, call IRACTS (active sessions routine) to determine if sessions exist for location.
- If APPN line disable request, call IRACTS to determine if sessions are active for any location enabled on the line.
- If disable request and active sessions exist, issue halt and prompt for pend disable, retry disable, disable kill, or terminate disable.
- Flag LOC(s) with disable indication.
- Issue any required diagnostics.

4 Post the subsystem to perform additional disable processing.

5 Free LOC:

- Flag SUBs and post users for the location.
- Dequeue LOC chain entry.
- Free LOC.
- If APPC subsystem, free associated SGB(s).

6 If subsystem request is for complete subsystem disable, perform the following initial disable processing:

- If disable request and active sessions exist, issue halt and prompt for pend disable, retry disable, disable kill, or terminate disable.
- Flag SCT with disable indication.
- Flag corresponding SUBs if necessary.
- Dequeue SCT chain entry.

7 Post the subsystem to perform additional disable processing.

8 Do the following for the requesting subsystem:

- Free phone list.
- Dequeue and free subsystem LOCs.
- If APPC subsystem, free associated SGB(s).
- Dequeue and free SCT.
- If abnormal termination of a completely enabled subsystem, process any other subsystem SCTs for the same subsystem type.
- If last subsystem of this type, deactivate TEB and terminate TB.

9 If other lines are still enabled for this subsystem, continue processing.

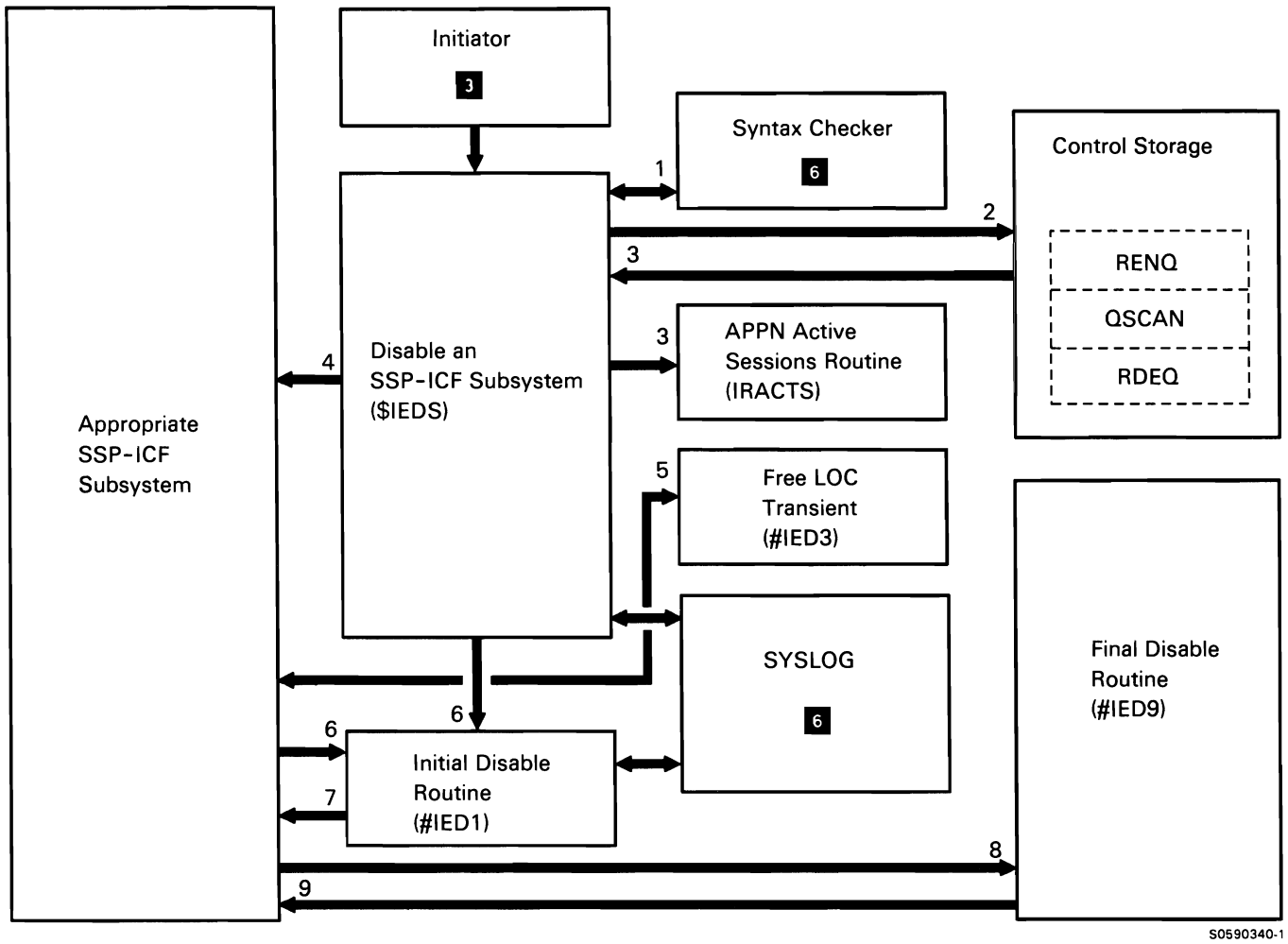


Chart 10.3.2 SSP-ICF Disable Control Flow

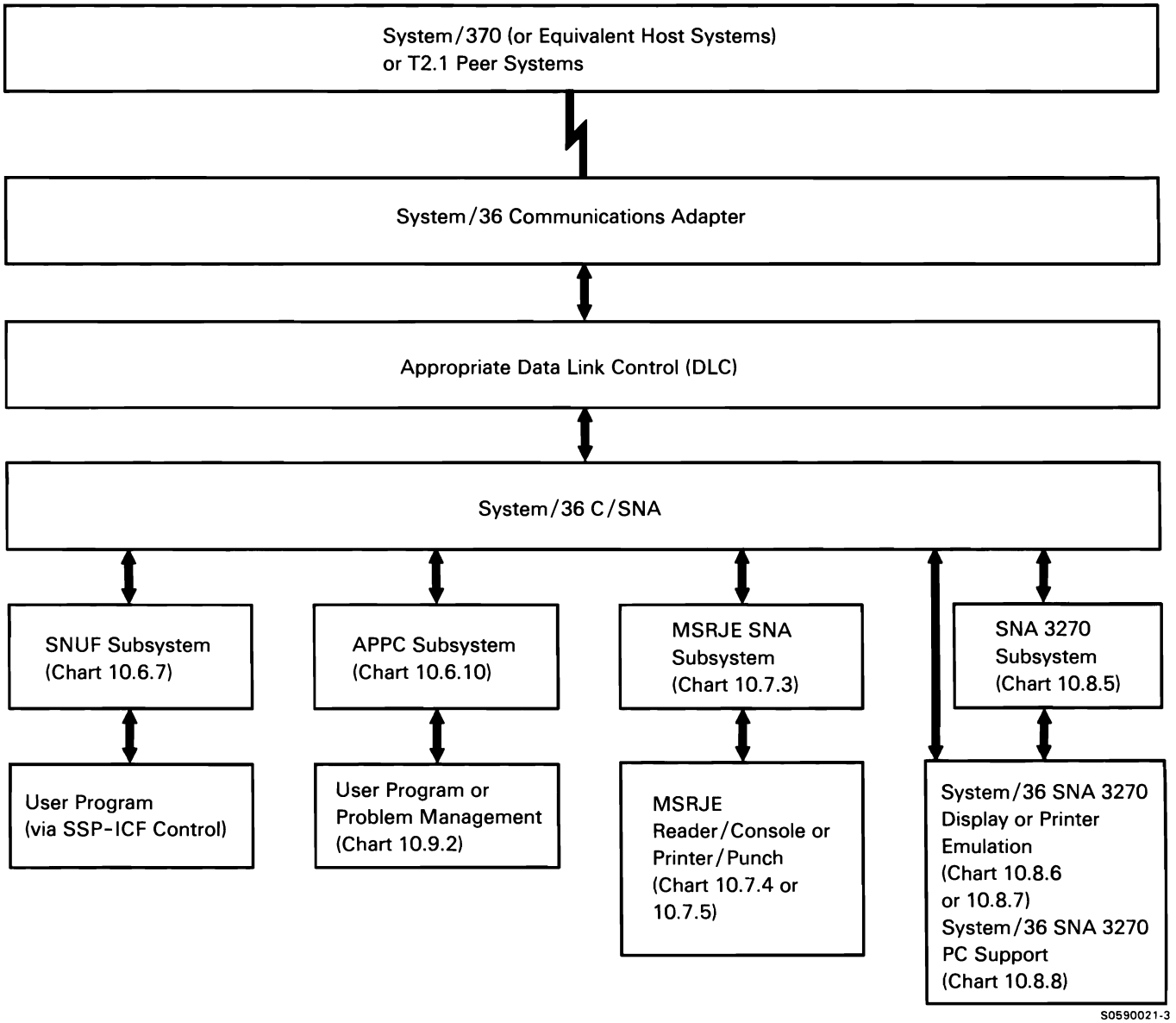
ICF C/SNA SUBFUNCTION

Systems Network Architecture (SNA) is a standard by which IBM designs and builds data communications products. Systems Network Architecture specifies the protocols (commands and responses) that communications products use to transmit a user's data from one location to another.

C/SNA is one of the subsets of SNA used by System/36. It is designed to allow user programs to send and receive data to and from remote locations in much the same way they would send data to or receive data from any I/O device on the system. This spares the user the task of programming complex data communications systems.

C/SNA allows System/36 SNUF, the APPC subsystem, SNA 3270 emulation, SNA 3270 PC support, or MSRJE SNA to initiate, communicate with, and terminate sessions with a System/370 or equivalent system that uses TSP3/FMP3, TSP4/FMP4, or TSP7/FMP19 protocols. The interface between the C/SNA system task and the user subsystem is the SNUB. C/SNA interfaces to DLC tasks via IOBs.

A System/36 C/SNA overview is shown below.



System/36 C/SNA Introduction

System/36 C/SNA consists of the following functional SNA layers:

- Path control (PC)

- Transmission control:
  - Transmission control/session control (TCSC)
  - Connection point manager (CPM)
  - Common session control (CSC)

- Data flow control (DFC)

- NAU (network addressable units) services:
  - LU services manager
  - PU services manager

C/SNA Mainline

Mainline processing uses the task complete queue to determine the work to be done. The scheduler/dispatcher executes the requests on the queue by routing control to the necessary C/SNA components.

The following C/SNA mainline processes are shown in Chart 10.4.0:

1 Process user requests (SNUBs) on the task complete queue by routing for appropriate processing.

2 Perform normal data receive/transmit operations.

3 Process the following:

- Configuration services requests (ACTPU/DACTPU).
- Activate/deactivate session.
- Reset PU to SSCP session.
- NOTIFY command.
- TERM-SELF command.

4 Process the following C/SNA-detected conditions:

- Close DLC request.
- DLC error completion.

5 Process the following C/SNA-detected conditions:

- SYSLOG requests via C/SNA.
- Timer completion.
- SYSLOG completion.

6 Process IOBs.

7 Process IOB completions.

8 Process return code (and data) from C/SNA.

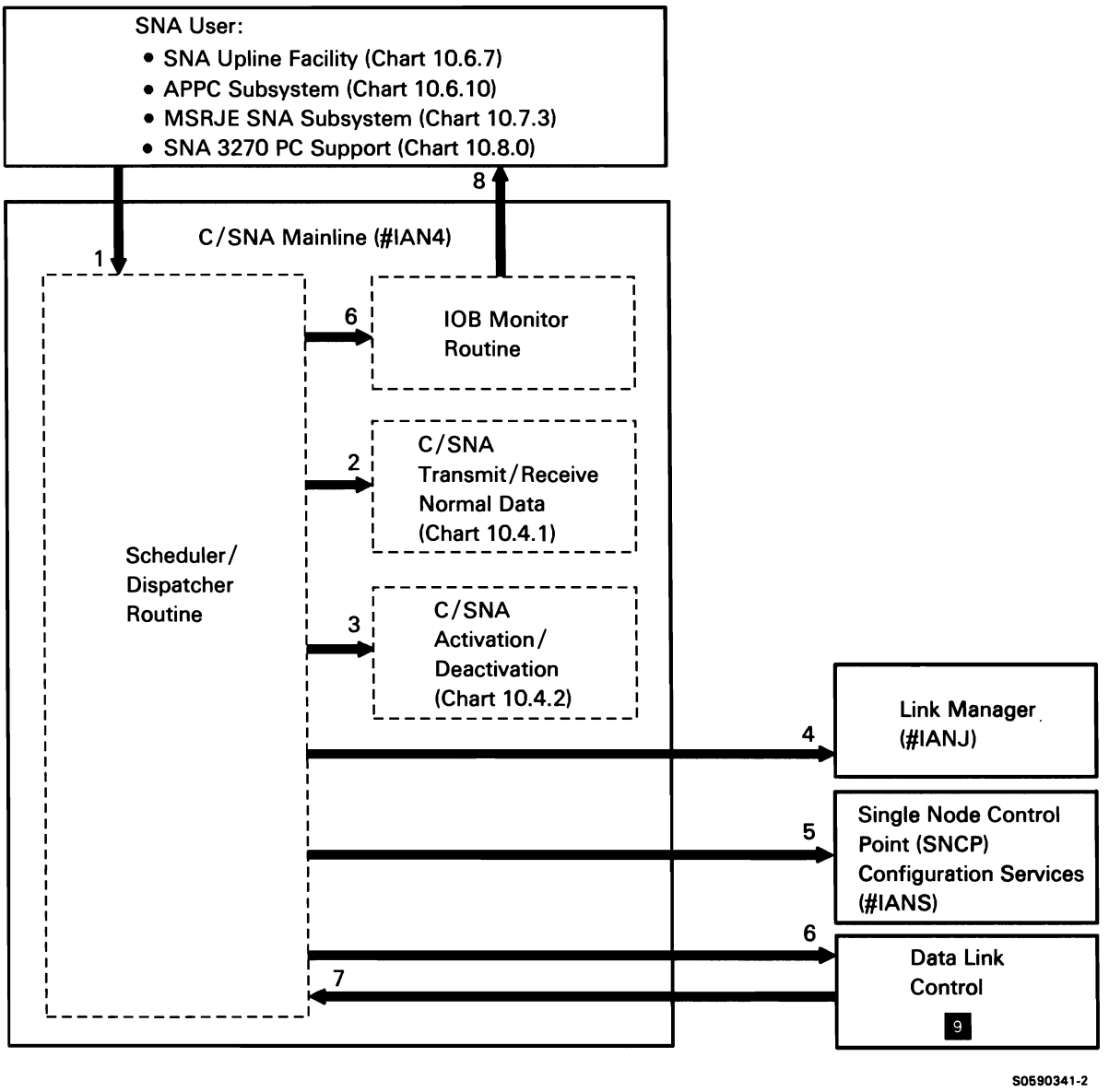


Chart 10.4.0 SSP-ICF C/SNA Mainline Control Flow

C/SNA Transmit/Receive Normal Flow

C/SNA transmit/receive normal flow processes data passed to/from devices (NAUs) in the System/36 portion of the SNA network.

The following C/SNA transmit/receive normal data processes are shown in Chart 10.4.1:

1 Perform normal data transmit/receive operations:

- Build request/response units (RUs) for transmit request.
- Interpret request/response units (RUs) for receive request.
- Post applicable subsystem with completion code and data.

2 Build basic information unit (BIU) by adding request/response header (RH) to transmit RU received from LU services manager.

3 Perform connection point routing for outbound requests; process pacing input.

4 Build BIU by adding RH to session control (pacing) RU received from LU services manager.

5 Build path information unit (PIU) by adding transmission header (TH) to the BIU received from data flow control send.

6 Build basic link unit (BLU) by adding link header (LH) and link trailer (LT) to BIU; send response basic link unit (BLU).

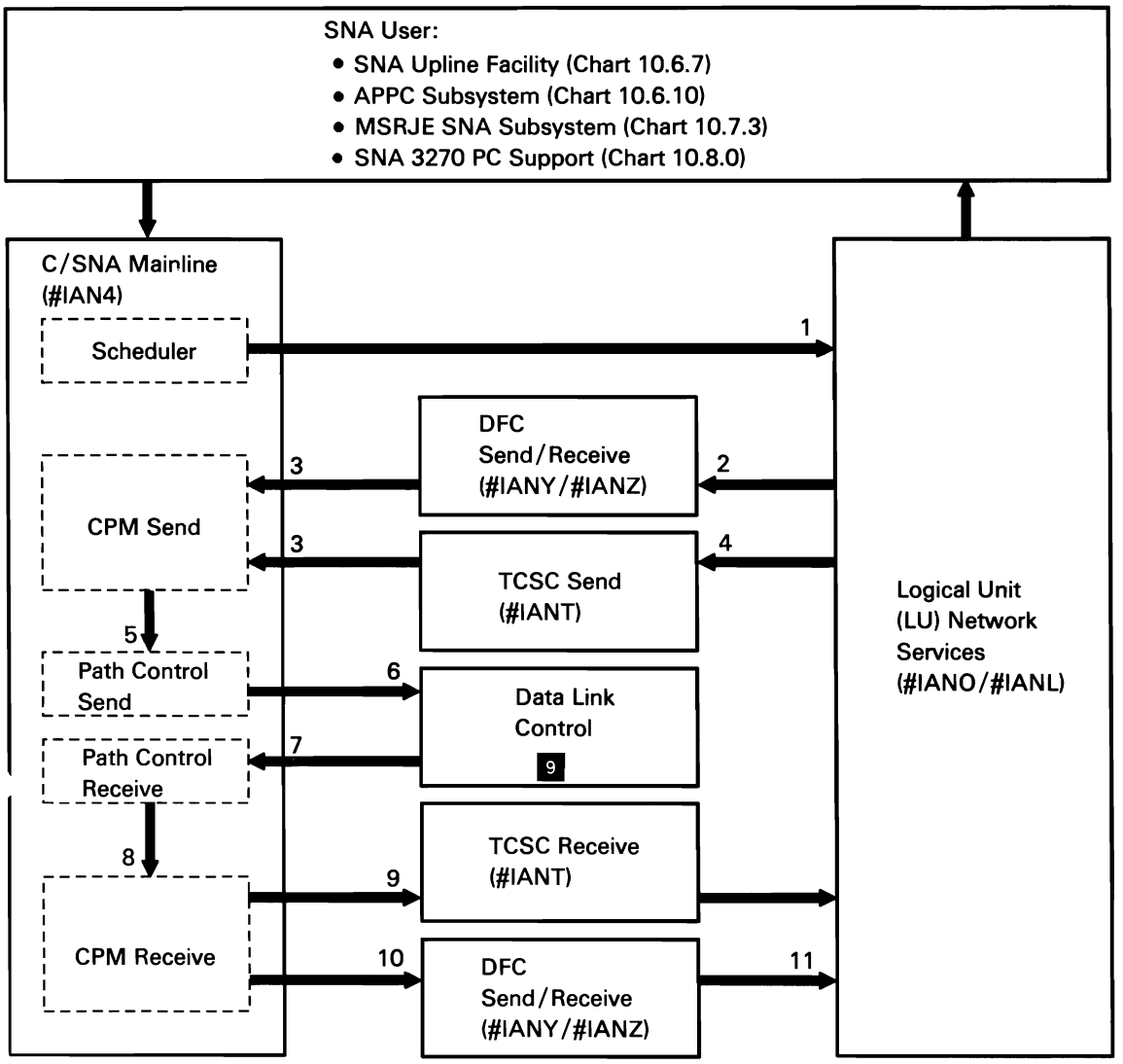
7 Process path information unit (PIU) from DLC by interpreting transmission header (TH).

8 Perform connection point routing for inbound data.

9 Process session control (pacing) BIU from connection point manager receive.

10 Process BIU from connection point manager receive by interpreting RH.

11 Process request/response unit (RU) from data flow control.



Note: Where two module names are given, the second-listed module performs all processing required for APPC subsystem (LU 6.2) support.

S0590342-2

Chart 10.4.1 SSP-ICF Transmit/Receive Normal Data Control Flow

C/SNA Activate/Deactivate

C/SNA activate/deactivate processes commands that alter the status of devices (NAUs) in the System/36 portion of the SNA network.

The following C/SNA activate/deactivate processes are shown in Chart 10.4.2:

1 Process activate/deactivate commands or build request/response unit (RU) for any required responses.

2 If required, build basic information unit (BIU) by adding request/response header (RH) for response to RU received from LU/PU services manager.

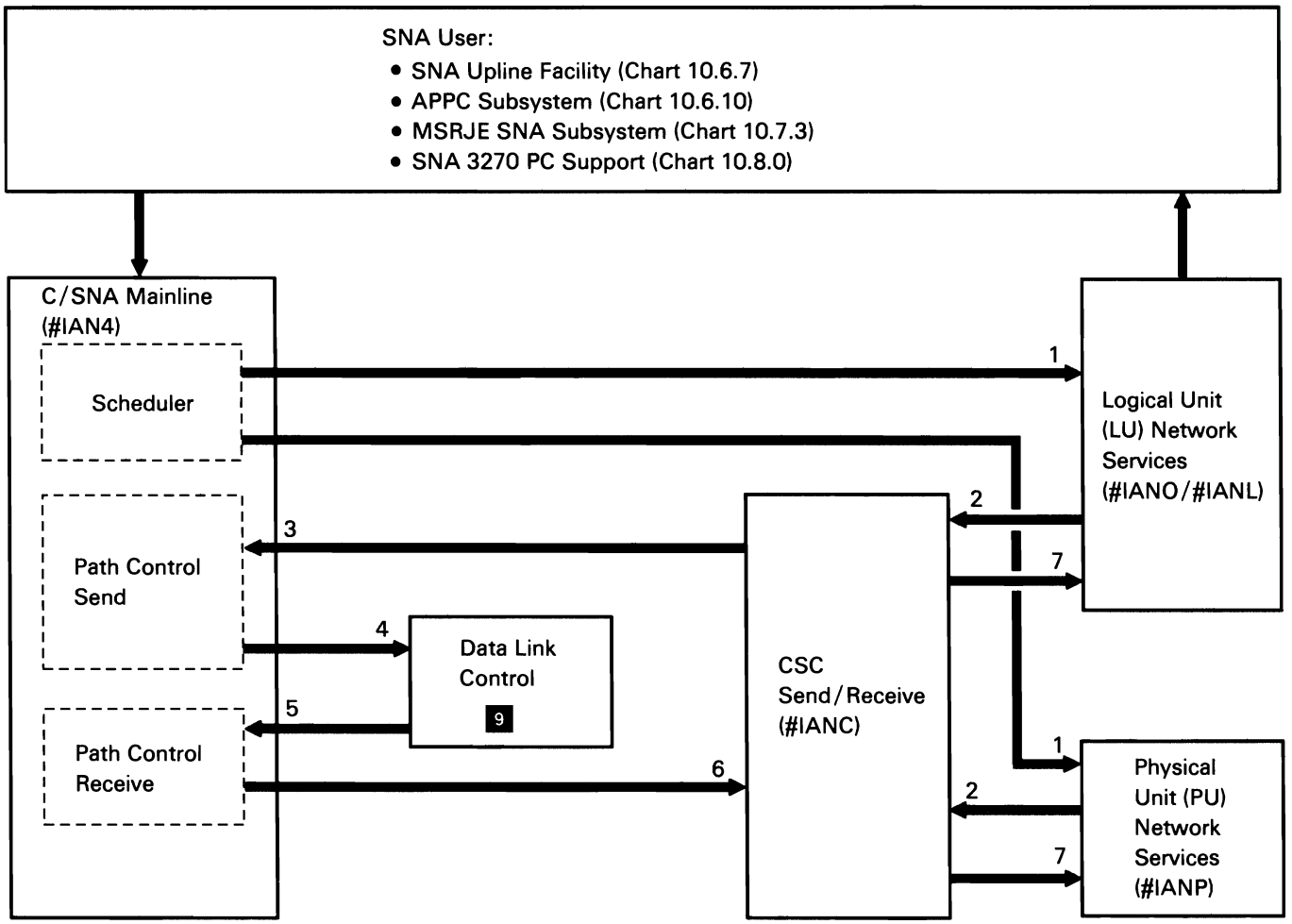
3 Convert BIU to path information unit (PIU) by adding transmission header (TH).

4 Convert PIU to basic link unit (BLU) by adding link header (LH) and link trailer (LT); send BLU.

5 Process transmission header (TH): find NAU half-session control block and, if found, pass to common session control.

6 Process request/response header (RH): validate command and route to LU services manager or PU services manager.

7 Interpret request/response unit (RU): if response required, build and issue; post subsystem, if applicable.



Note: Where two module names are given, the second-listed module performs all processing required for APPC subsystem (LU 6.2) support.

S0590343-2

Chart 10.4.2 SSP-ICF C/SNA Activate/Deactivate Control Flow

SSP-ICF USER AIDS SUBFUNCTION

SSP-ICF Debug

SSP-ICF debug records data in the task work area (TWA) for each user-specified SSP-ICF operation. The data can be displayed or printed by the user as an aid in program debugging. The debug routine, in conjunction with SSP-ICF data management trace, intercepts an SSP-ICF operation at the data management level and records (in the assigned TWA) a 64-byte debug trace entry, in addition to the SSP-ICF data management 64-byte trace entry. The debug trace routine and the SSP-ICF data management trace routine (if not already loaded) are loaded in system queue space (SQS) by SSP-ICF debug mainline.

The following debug processes are shown in Chart 10.5.1:

1 Activate debug trace according to parameters received with ICFDEBUG procedure call, print debug trace records, and deactivate debug trace recording, routing control as required.

2 If ICFDEBUG ON, allocate task work area and assign SQS space for load, and load debug trace and SSP-ICF data management trace.

3 If ICFDEBUG CRT, do the following:

- Allocate new task work area (TWA) for continuing trace.
- Get trace entries from old TWA.
- Build output buffer.
- Display trace entries to display station (CRT).
- Deallocate TWA.

4 If ICFDEBUG PRINT, do the following:

- Allocate new task work area (TWA) for continuing trace.
- Get trace entries from old TWA.
- Build output buffer.
- Allocate and open printer.
- Print trace entries.
- Deallocate TWA.

5 If ICFDEBUG OFF, deallocate TWA and free assigned SQS.

6 Terminate.

7 Create 64-byte debug trace entry and return.

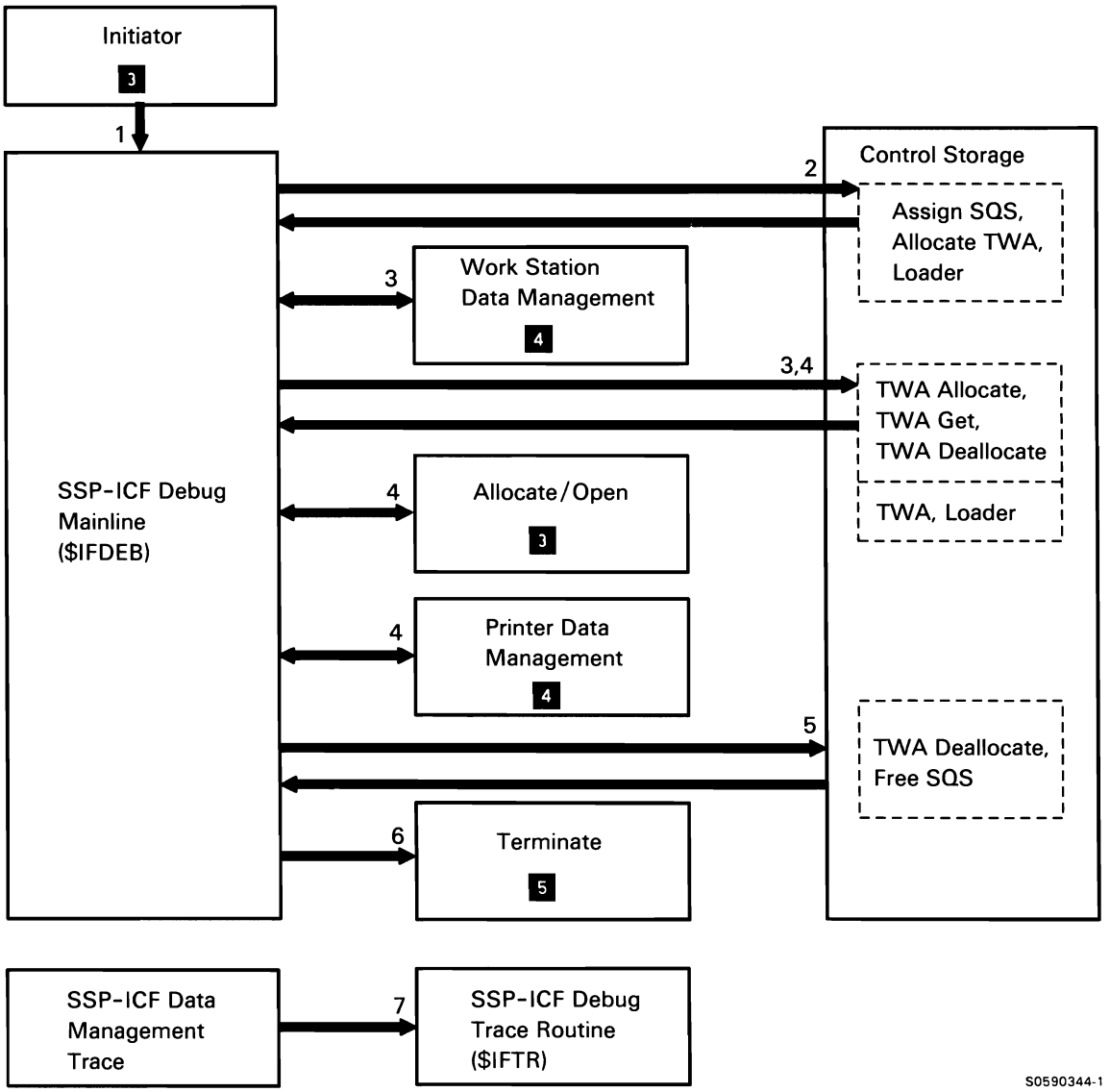


Chart 10.5.1 SSP-ICF Debug Control Flow

SSP-ICF Verify

SSP-ICF verify uses IBM-supplied application programs to verify the operation of communications links set up by the user for individual IBM data communications features. Verify is evoked with the ICVERIFY system procedure which, in addition to invoking the appropriate verification program, also sets UPSI switches 1 and 2 for the following:

- UPSI switch 1 on indicates BSC subsystem.
- UPSI switch 1 off indicates SNA subsystem.
- UPSI switch 2 on indicates the remote program is the evoker.

Chart 10.5.2 shows the control flow for verify:

1 Evoke ICBSCEL or ICPEER procedure on remote system.

2 Evoke CCPIVP task on the remote System/3.

3 Evoke the CSFE transaction on the remote system.

4 Issue IMS /TEST command to echo data.

5 Send a 3270 Clear key indication to the remote system and process response.

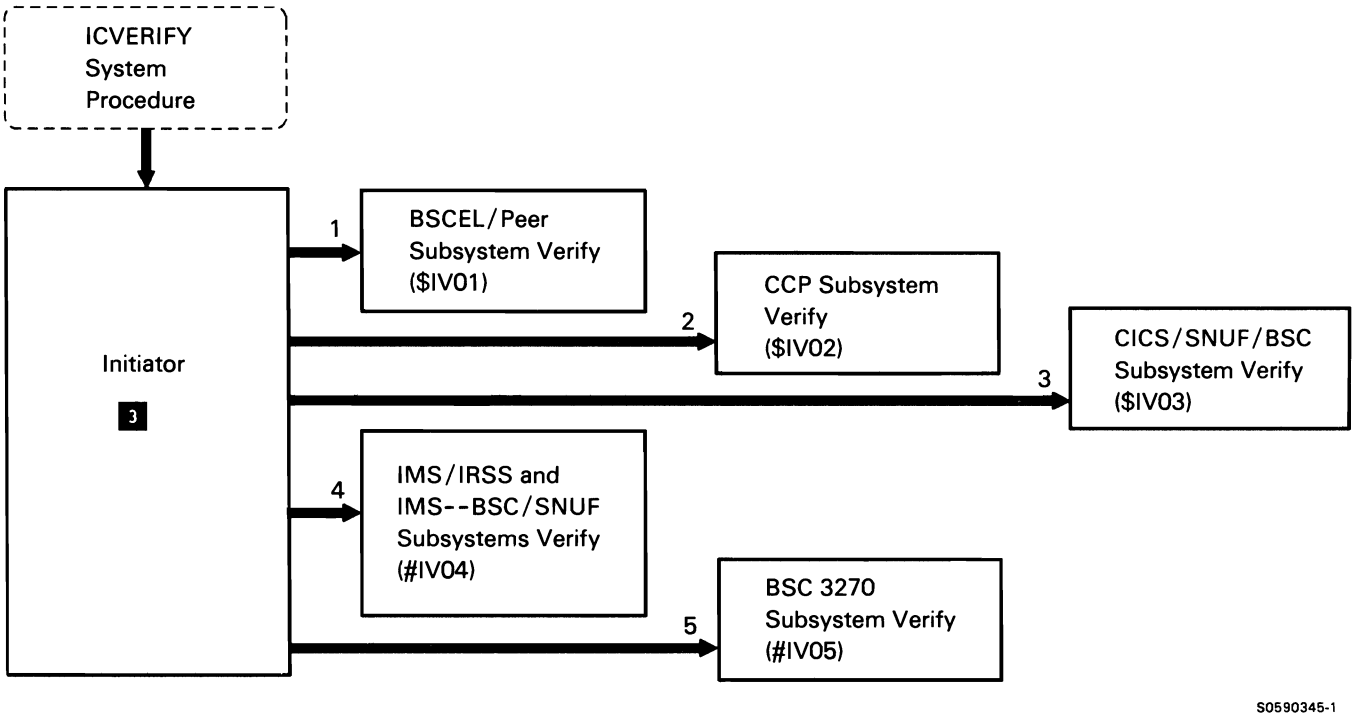


Chart 10.5.2 SSP-ICF Verify Control Flow

SSP-ICF PROGRAM-TO-PROGRAM SUBSYSTEMS

The SSP-ICF program-to-program subsystems allow System/36 programmers to write application programs that can communicate locally (Intra subsystem only) or remotely with other application programs or system programs without being concerned with the unique communications protocol of the remote system. They include:

- SSP-ICF Intra subsystem — Chart 10.6.2
- SSP-ICF IMS/IRSS — Chart 10.6.3
- SSP-ICF CICS subsystem — Chart 10.6.4
- SSP-ICF BSC CCP subsystem — Chart 10.6.5
- SSP-ICF BSCEL subsystem — Chart 10.6.6
- SSP-ICF SNA Upline Facility (SNUF) subsystem — Chart 10.6.7
- SSP-ICF Finance subsystem — Chart 10.6.8
- SSP-ICF Peer subsystem — Chart 10.6.9
- SSP-ICF APPC subsystem — Chart 10.6.10
- SSP-ICF APPN subsystem — Chart 10.6.11.n
- SSP-ICF Asynchronous subsystem — Chart 10.6.12

This topic also includes SSP-ICF BSC link control (Chart 10.6.1), which is used by all of the BSC program-to-program subsystems listed above.

The following map shows the storage layout of SSP-ICF program-to-program subsystem data areas. Logical address hex 0000 is the beginning of storage assigned to SSP-ICF:

Hex 0000	SSP-ICF 4K-byte Nucleus
Hex 1000	SSP-ICF Control (Chart 10.2)
Hex 2800 (see note)	SSP-ICF 2K-byte Transient Area (Optional)
Hex 3000	Applicable SSP-ICF Subsystem Programs
Variable	SSP-ICF Task Work Space and Map Area

Note: The SSP-ICF IMS/IRSS, Peer, SNUF, APPC, and APPN subsystems don't use the SSP-ICF transient area. Their subsystem programs start at logical hex address 2800.

Common Program-to-Program Subsystem Interfaces

Note: Although the Intra subsystem is an SSP-ICF program-to-program subsystem, much of the following description of general program-to-program communications does not apply to it.

The application programs written for use with SSP-ICF interface with SSP-ICF via SSP-ICF data management. Although SSP-ICF data management is described in this manual as a separate program module, it actually executes as a part of the application program task. It passes function requests and data from the application program to the applicable SSP-ICF subsystem and passes data and data-related information from the applicable SSP-ICF subsystem via posts to the application program.

The application tasks interface with the program-to-program subsystems via SSP-ICF control. Although SSP-ICF control is described in this manual as a separate program module, it actually executes as a part of the applicable program-to-program subsystem task. After executing a request by posting either the application program or the data link control task, the program-to-program subsystem goes to SSP-ICF control's wait routine to await the next request.

The SSP-ICF program-to-program subsystems generate the communications protocol needed to link the local application program (via the applicable SSP-ICF subsystem or equivalent program at the remote location) to the remote application program.

Program start requests for application programs written for use with SSP-ICF can come from the local system, via the initiator, or from the remote systems, via SSP-ICF procedure start requests.

Procedure Start Requests

All program-to-program subsystems are capable of initiating and executing procedure start requests. The following description assumes that the procedure start request is coming from a remote System/34 or a System/36. Requests from other system types would pass equivalent requests and control blocks to the local system:

- When issuing a procedure start request, the program at the remote system sets up an evoke operation code in the work station DTF and builds an evoke parameter list that indicates the user ID, the user's password, the system library containing the procedure, and the procedure name.
- SSP-ICF data management at the remote system puts pointers to the evoke list and the user's TWS buffer into the SUB, then posts SSP-ICF control out of its wait. SSP-ICF control branches to the new request entry point in the subsystem code. The remote subsystem sends the request to the local subsystem.
- The appropriate data link interrupt handler at the local System/36 posts SSP-ICF control (part of the applicable System/36 subsystem) out of its wait. The evoke operation at the remote system is now a procedure start request at the System/36.
- SSP-ICF control ensures that the subsystem is enabled, then converts the data in the evoke list to a procedure start parameter list, builds an SUB, posts the command processor to start the procedure, then goes to a wait.
- The command processor acknowledges the successful procedure and program start by posting the SUB, and the acknowledgement is routed back to the remote application program.

SSP-ICF BSC Link Control

Interactive BSC link control support binary synchronous communications between the BSC program-to-program subsystems and remote BSC systems. It is attached as a task by the open routine in #IBHP. As part of the task-attach process, the open routine also allocates the line, loads the proper microcode, assigns IOBs and work areas, and posts the interrupt handler task to enable the BSC adapter. After the initial setup, link control gets control when an IOB op-ends or when it receives a subsystem request post; in this capacity, it serves as a main storage interrupt handler. All requests to #IBHL come through an #IBHL wait routine.

Line protocols for SSP-ICF BSC are shown in *Appendix A: Data Communications Line Protocols.*

The following link control processes are shown in Chart 10.6.1:

1 Issue SVC instruction to process an IOB.

2 Perform IOB op-end processing and error checking

3 Perform SYSLOG message and response handling by posting the applicable subsystem with a multipurpose BUB.

4 Handle subsystem request:

- GET.
- PUT.
- Start monitor mode.
- Cancel (BUB).
- Clear.
- Select reject.
- Abort.
- Terminate (communications line).

5 Bring up data terminal ready and modem ready for enable.

6 Perform required abnormal termination processing based on contents of TEB (TEBWORK):

- Processor check caused by incorrect subsystem input.
- Line deallocation.
- Termination of subsystem.
- Detachment of link control task (#IBHL).

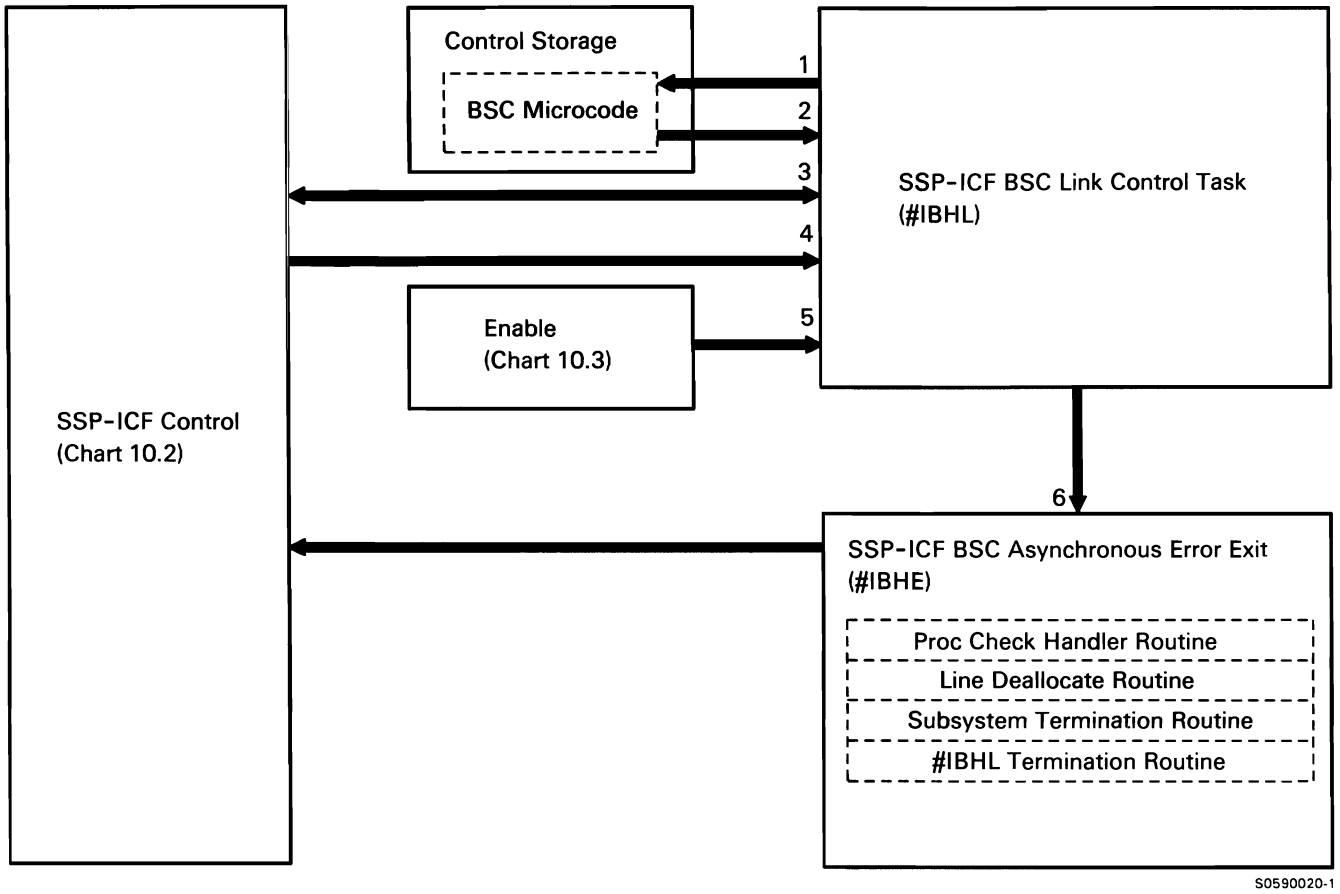


Chart 10.6.1 SSP-ICF BSC Link Control Flow

SSP-ICF Intra Subsystem

The Intra subsystem provides an interactive interface between user application programs on the same System/36. The Intra subsystem can support multiple application programs communicating concurrently. The Intra subsystem is generally used for the following purposes:

- To test new interactive communications applications without using a communications line (these programs may have to be slightly modified later).
- To allow for program-to-program communication within the local system.

The following Intra subsystem processes are shown in Chart 10.6.2:

1 Perform first enable of the subsystem.

2 Perform subsequent enables of the subsystem.

3 Route for new requests posted to the subsystem (individual operation code handler routines post user with RTC and data for input operations).

4 Process abnormal end of session.

5 Process successful program start post from the command processor or scheduler.

6 Process unsuccessful program start post from the command processor or scheduler.

7 Disable the Intra subsystem.

8 Process immediate subsystem disable.

9 Set subsystem processor check on subsystem error.

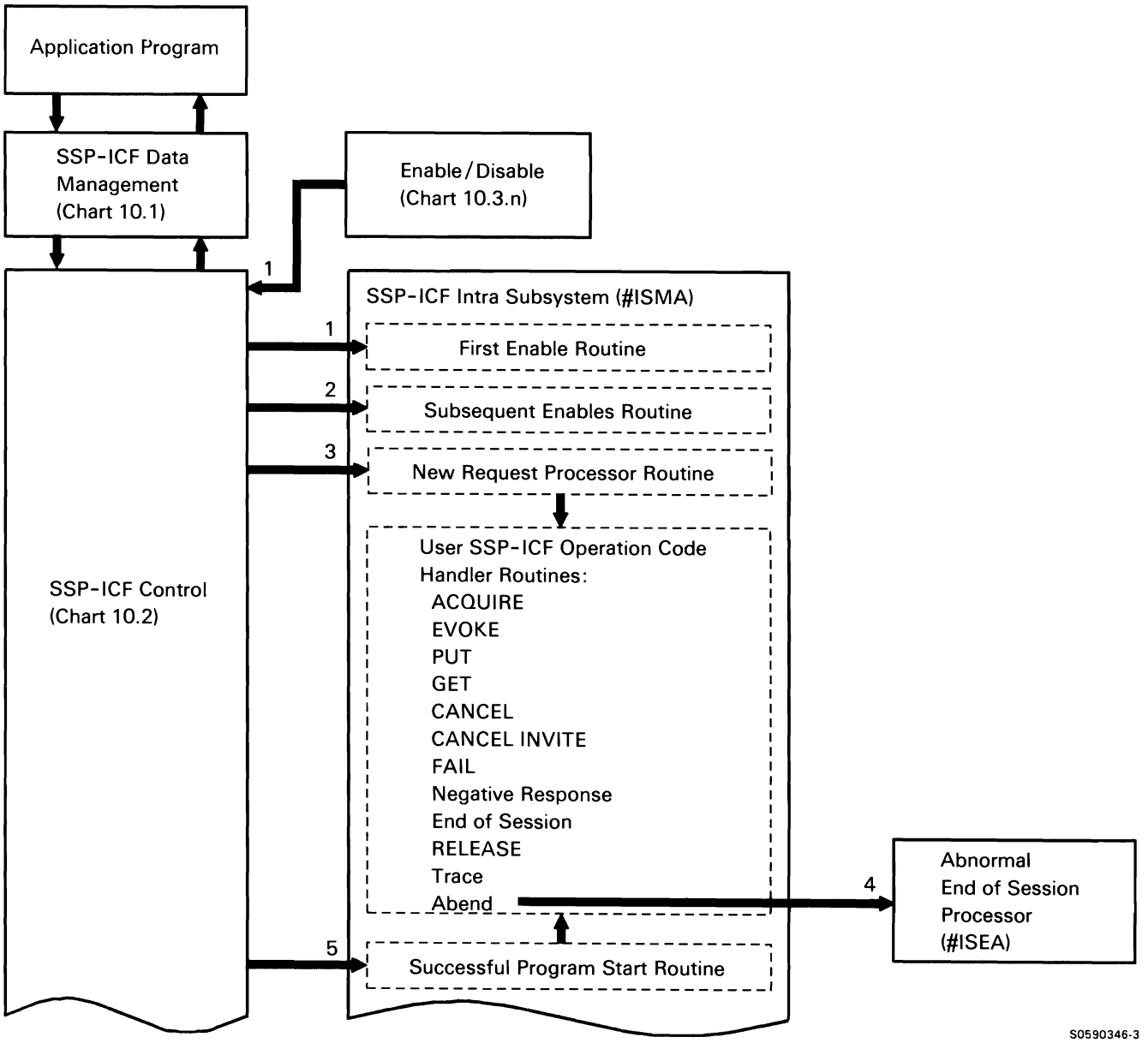


Chart 10.6.2 (Part 1 of 2) SSP-ICF Intra Subsystem Control Flow

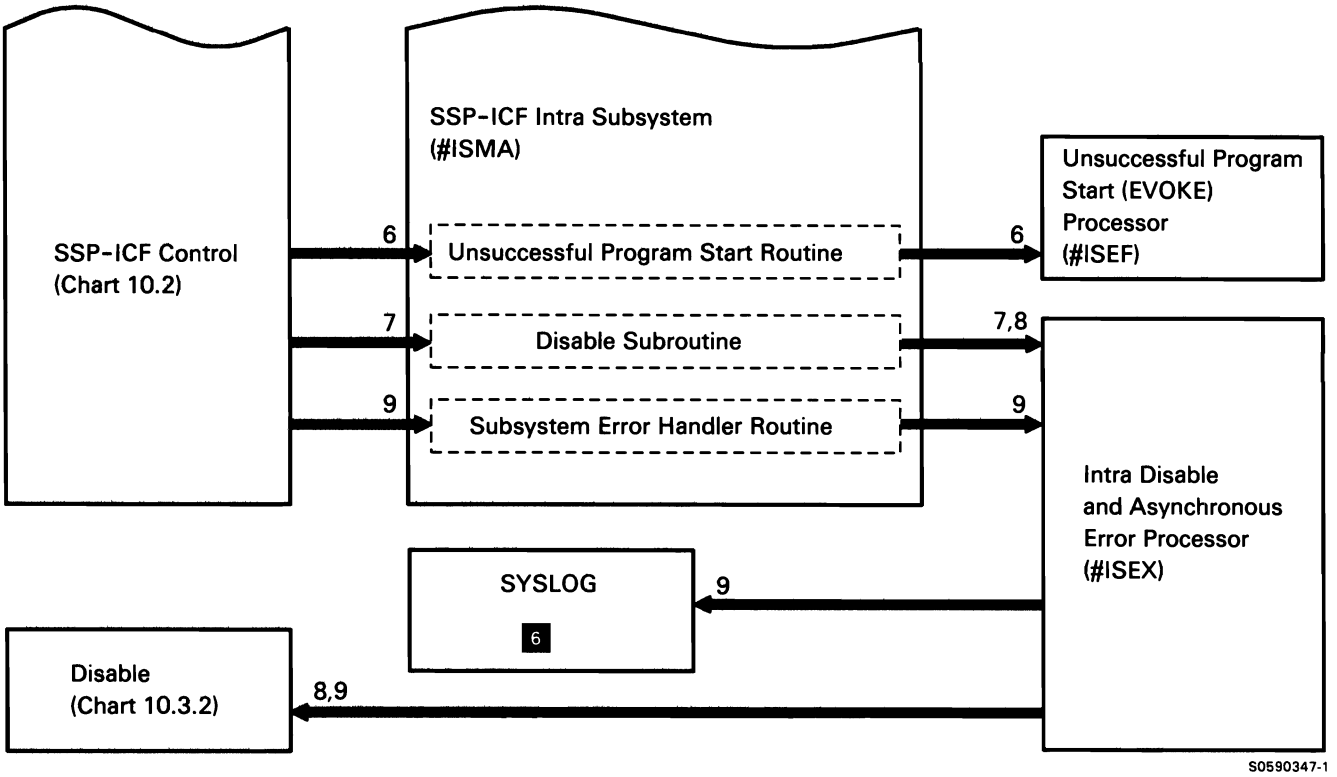


Chart 10.6.2 (Part 2 of 2) SSP-ICF Intra Subsystem Control Flow

SSP-ICF IMS/IRSS

System/36 IMS/IRSS (Information Management System/Intelligent Remote Station Support Subsystem) provides support that allows up to 16 concurrent sessions between the System/36 and IMS/VS (Information Management System/Virtual Storage) over a single BSC multipoint line.

IMS/VS views each session as a physical terminal and, therefore, must have an associated physical terminal address. The address must be defined at IMS/VS and at the System/36. The user program can request a specific physical terminal address on the session OCL statement or the subsystem can assign an address for the session.

The subsystem communicates with IMS/VS through the SSP-ICF BSC interrupt handler.

The following IMS/IRSS processes are shown in Chart 10.6.3:

1 Enable the subsystem:

- Allocate storage.
- Start receive operation.

2 Process user requests.

3 Process procedure start requests from host.

4 Process BSC interrupt handler operation end requests.

5 Display SYSLOG messages and process message responses.

6 Process BSC interrupt handler abnormal termination.

7 Perform subsystem disable processing.

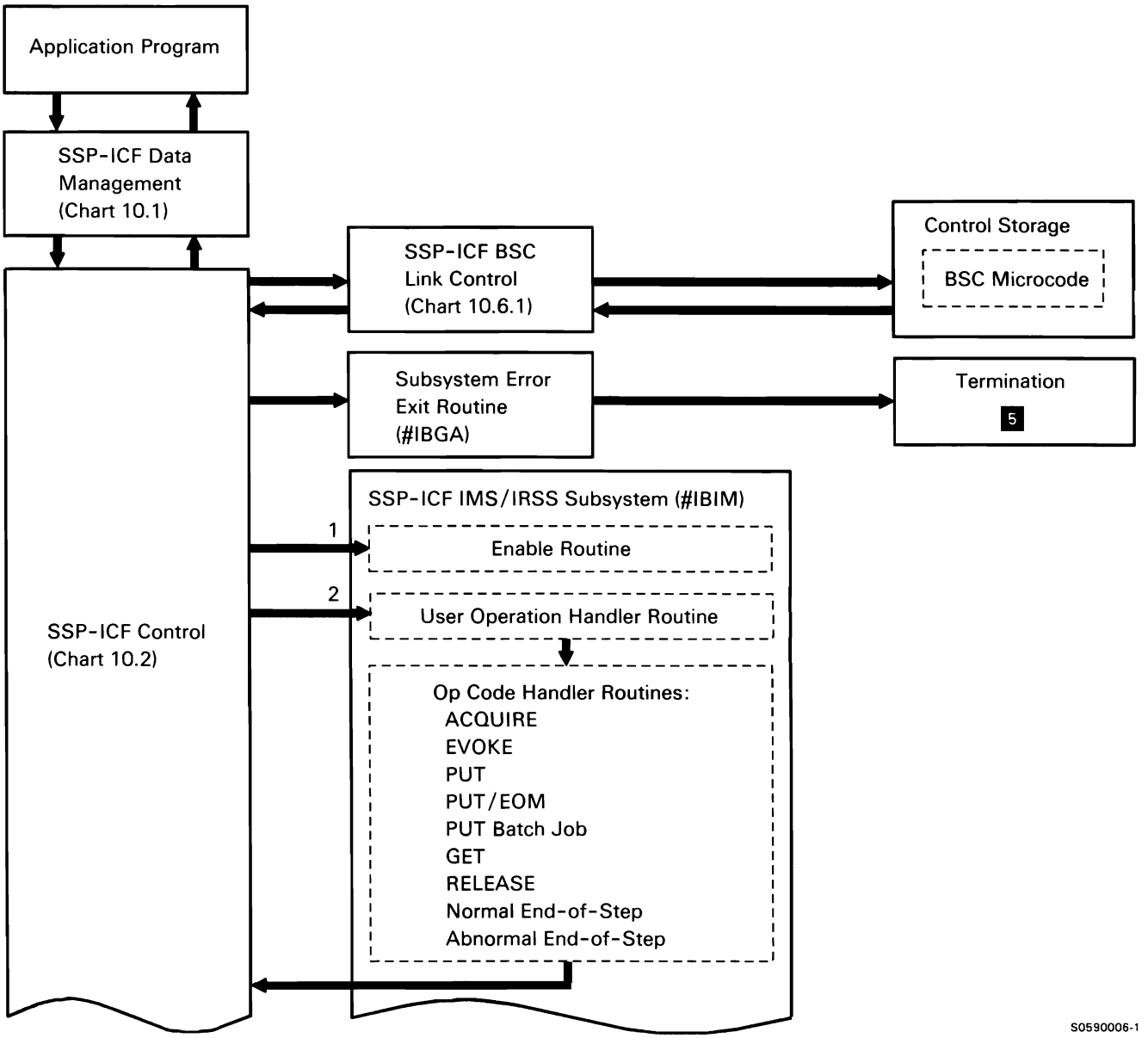


Chart 10.6.3 (Part 1 of 2) SSP-ICF IMS/IRSS Control Flow

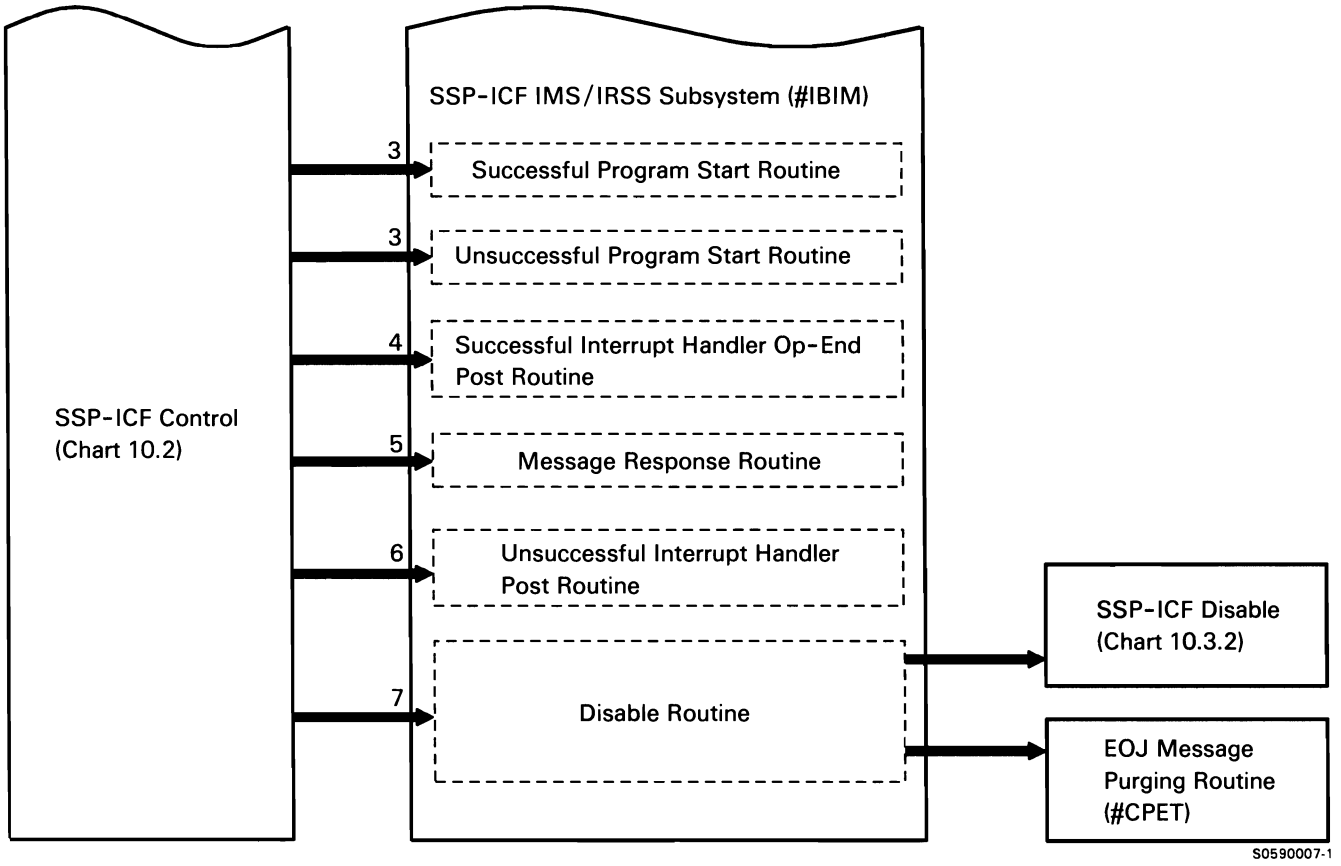


Chart 10.6.3 (Part 2 of 2) SSP-ICF IMS/IRSS Control Flow

This page is intentionally left blank.

SSP-ICF CICS Subsystem

The System/36 SSP-ICF binary synchronous communications (BSC) Customer Information Control System (CICS) Subsystem provides a session interface to IBM System/370 CICS/VS or equivalent system.

The CICS subsystem initializes the required storage areas, passes data between the application program and the BSC interrupt handler, initiates procedures based on data received from the remote system, and informs the operators of errors.

The following CICS subsystem functions are shown in Chart 10.6.4:

1 Initialize, and allocate storage and active session table (AST) on enable.

2 Deallocate storage and AST on disable.

3 Release session buffers and clean up after successful procedure start.

4 Release session buffers and clean up after unsuccessful procedure start.

5 Cleanup and disable subsystem after interrupt handler abnormal termination.

6 Process user responses to subsystem messages.

7 Process communications line operations (op ends via interrupt handler).

8 Process new user requests.

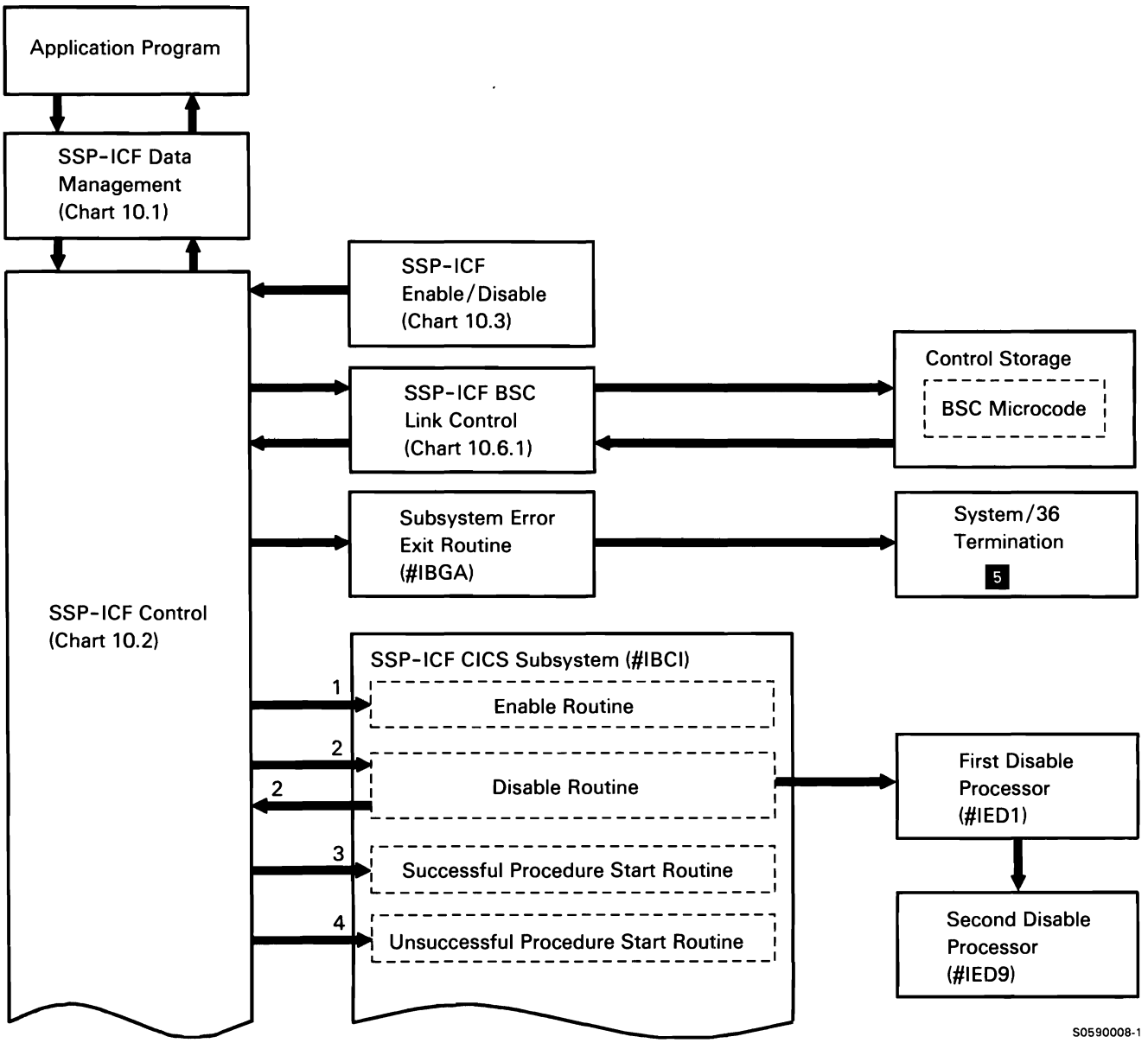


Chart 10.6.4 (Part 1 of 2) SSP-ICF CICS Subsystem Control Flow

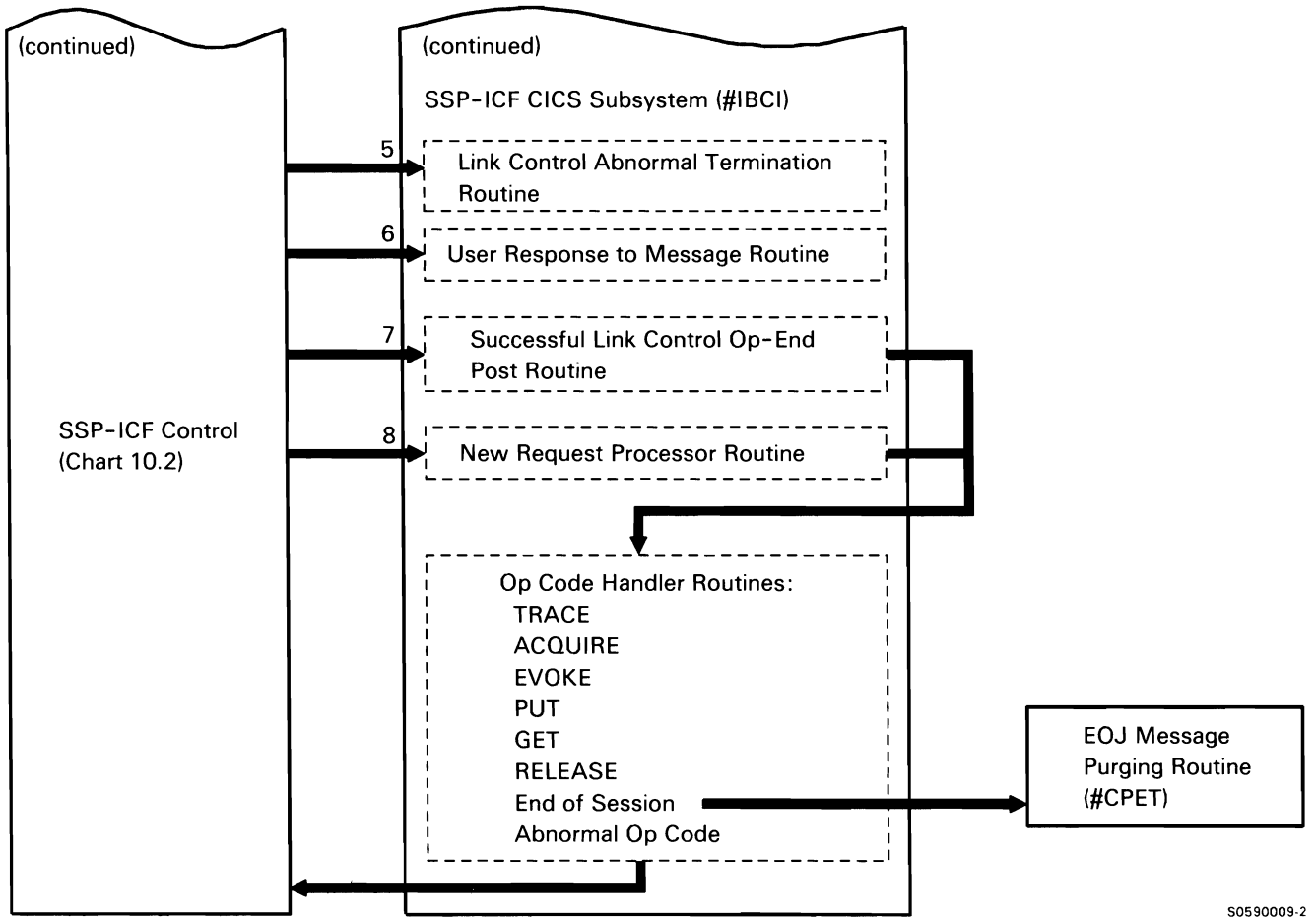


Chart 10.6.4 (Part 2 of 2) SSP-ICF CICS Subsystem Control Flow

This page is intentionally left blank.

SSP-ICF BSC CCP Subsystem

The System/36 SSP-ICF binary synchronous communications (BSC) Communications Control Program (CCP) subsystem provides System/36 SSP-ICF with a session interface to System/3 Model 15 CCP.

The CCP subsystem supports an EBCDIC/ASCII translation function that allows the user to put and get data in EBCDIC which is actually transmitted or received in ASCII.

The the following CCP subsystem processes are shown in Chart 10.6.5:

1 On serious errors, disable all System/36 communications lines associated with the CCP subsystem and shut down the CCP subsystem.

2 Enable SSP-ICF CCP by processing the ENABLE command from System/36 enable ($IENBL).

3 Disable SSP-ICF CCP by processing the DISABLE command from System/36 disable (#IEDS).

4 Initiate operations after successful procedure (program) start from the remote system.

5 Perform error recovery after an unsuccessful procedure (program) start attempt from the remote system.

6 Process after a BSC interrupt handler processor check.

7 Process after issuing a SYSLOG message.

8 Process after the BSC interrupt handler op ends an operation.

9 Process user-program operation codes.

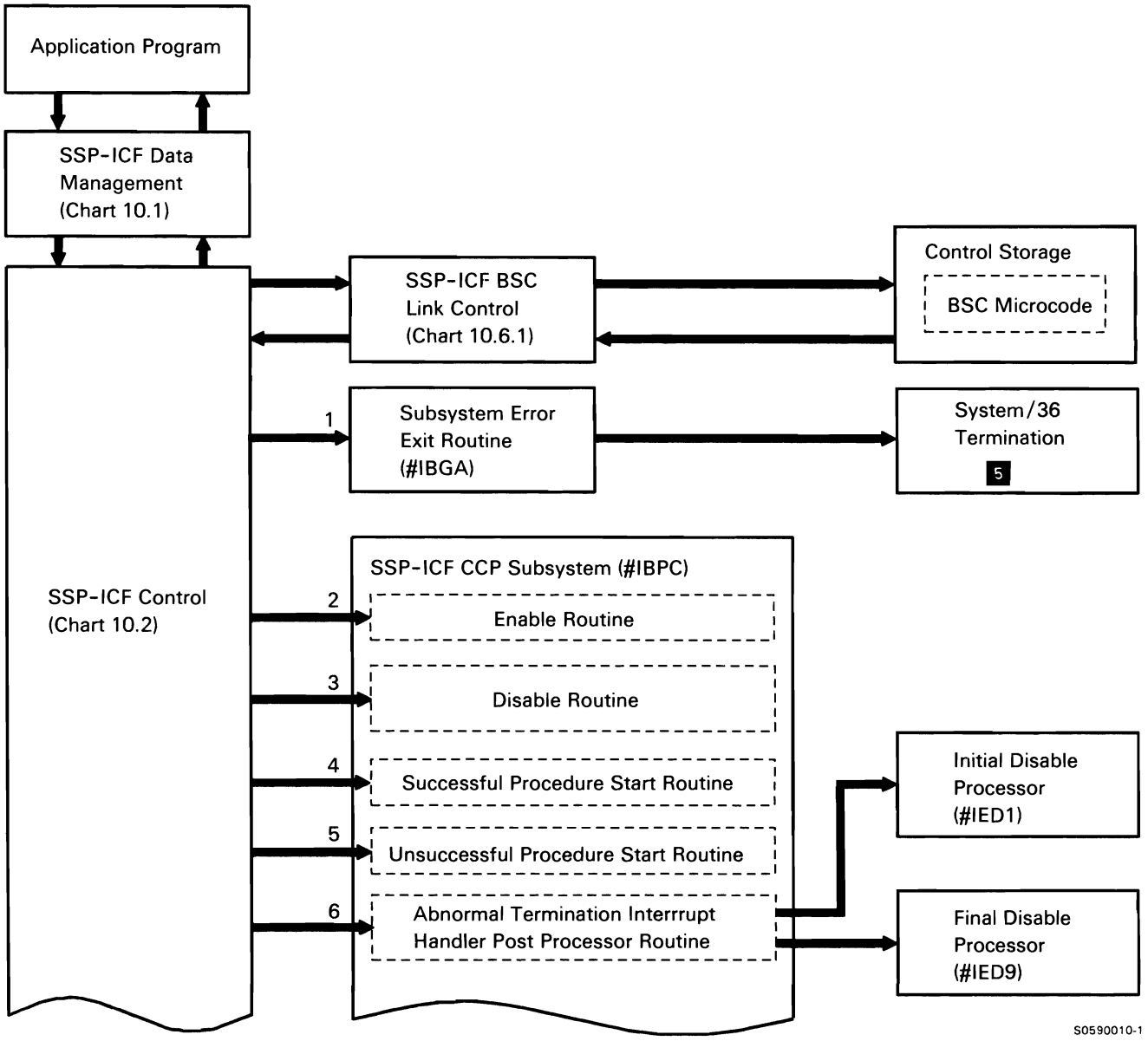


Chart 10.6.5 (Part 1 of 2) SSP-ICF CCP Subsystem Control Flow

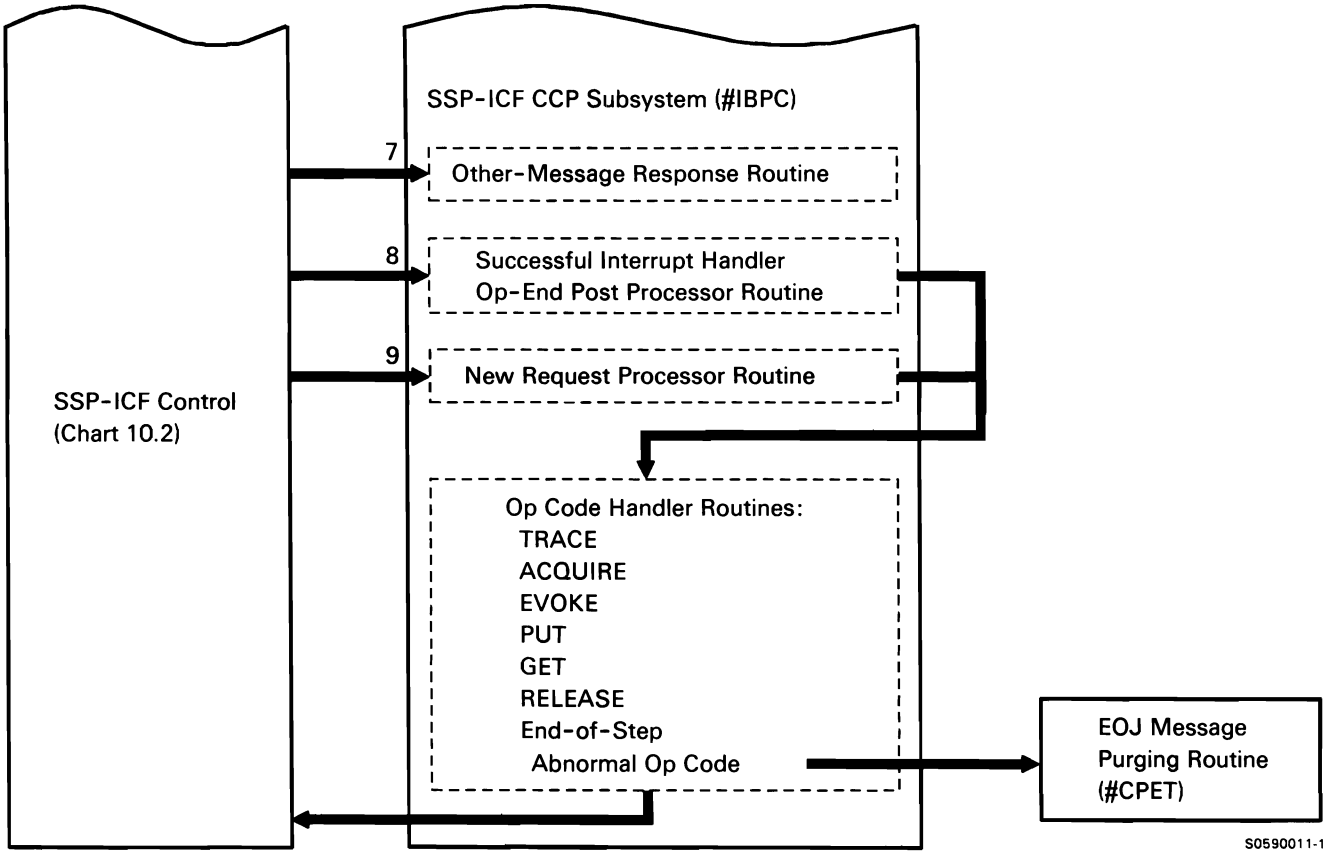


Chart 10.6.5 (Part 2 of 2) SSP-ICF CCP Subsystem Control Flow

SSP-ICF BSCEL Subsystem

The System/36 SSP-ICF Binary Synchronous Communications Equivalence Link (BSCEL) subsystem provides System/36 with an interface to another System/36, System/38, System/34, System/3, System/7, BTAM, TCAM, 3471, Series/1, or other system that presents a compatible interface. BSCEL supports EBCDIC or ASCII data and provides an automatic translation function that allows the application to put and get data in EBCDIC which is actually transmitted or received in ASCII. The BSCEL subsystem supports a variety of data formats including variable-length blocked records with a record separator or intermediate text block (ITB) separator, blank compression or truncation, 3740 multiple files and office systems support.

Line protocols used with SSP-ICF BSCEL are shown in *Appendix A: Data Communications Line Protocols*.

The following BSCEL subsystem processes are shown in Chart 10.6.6:

1 Process the ENABLE command (via the $IENBL utility) to enable the BSCEL subsystem on a specific line.

2 Process application program operation codes.

3 Process procedure start requests from the remote system.

4 Process requests and op ends from the BSC interrupt handler.

5 Process communications line failures.

6 Handle messages and responses related to the BSCEL task.

7 When trace is active, log trace entries for the BSCEL task.

8 Process the DISABLE command (via the $IEDS utility) to disable the BSCEL subsystem on a specific line.

9 Process BSCEL termination for BSC interrupt handler abnormal termination.

10 Process BSCEL subsystem abnormal termination.

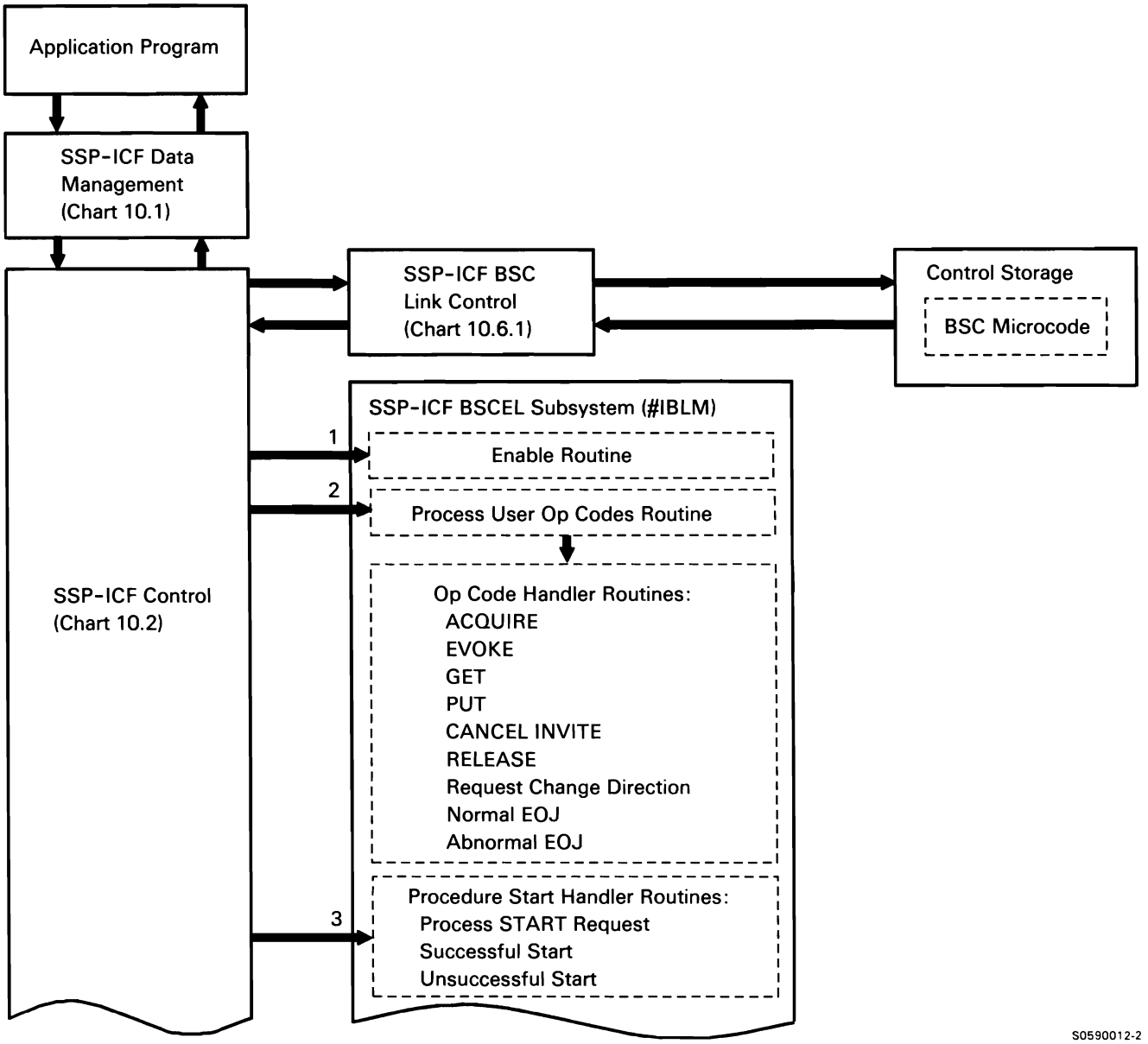


Chart 10.6.6 (Part 1 of 3) SSP-ICF BSCEL Subsystem Control Flow

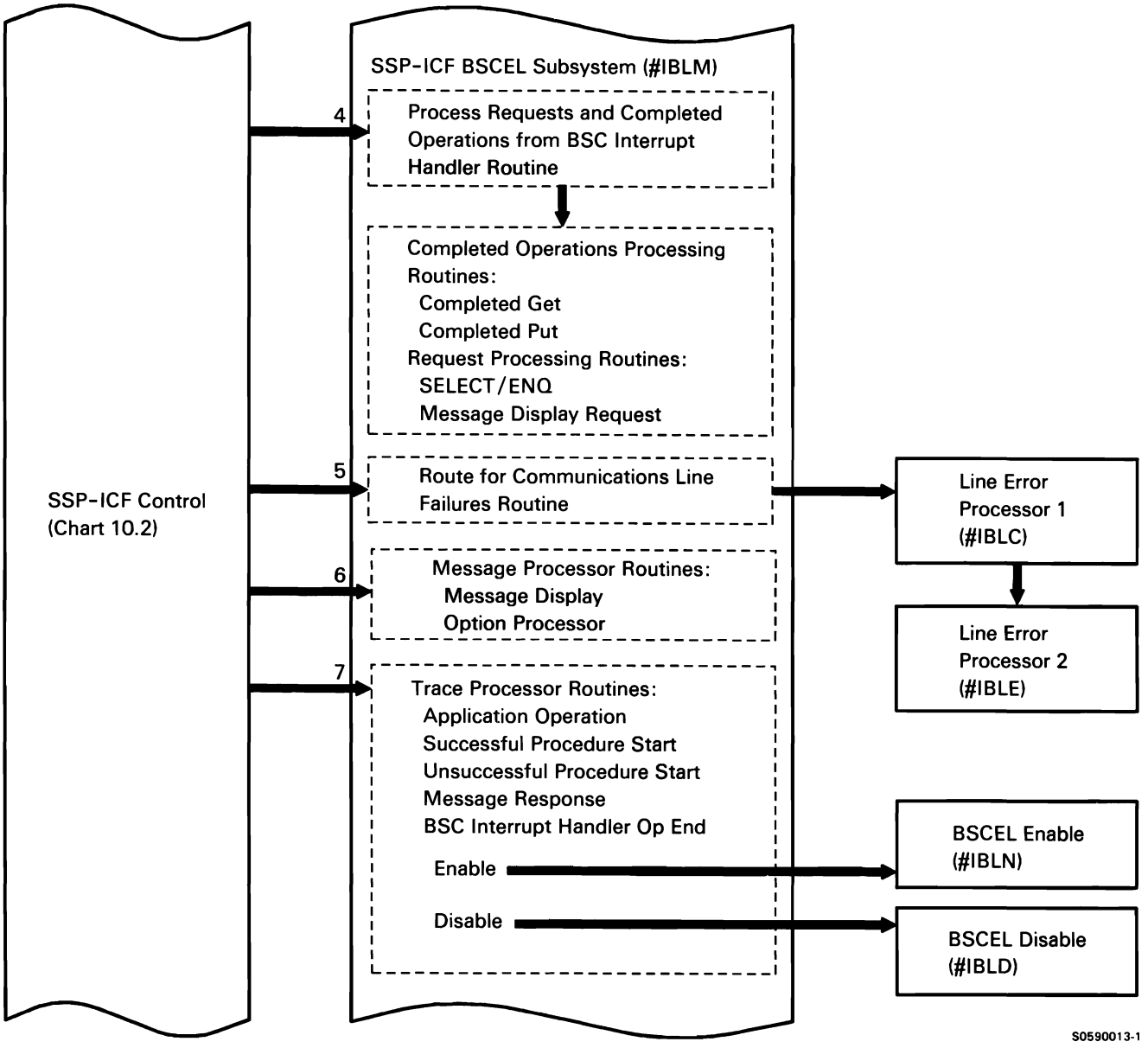


Chart 10.6.6 (Part 2 of 3) SSP-ICF BSCEL Subsystem Control Flow

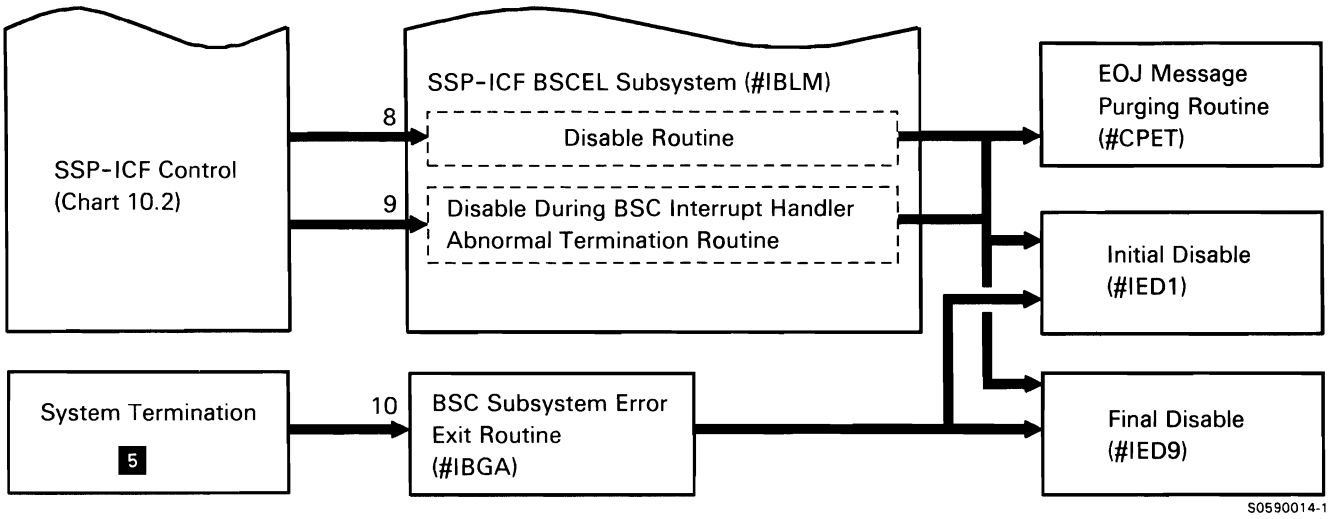


Chart 10.6.6 (Part 3 of 3) SSP-ICF BSCEL Subsystem Control Flow

SSP-ICF SNA Upline Facility (SNUF) Subsystem

The System/36 SSP-ICF SNA Upline Facility (SNUF) subsystem provides the SSP-ICF user with a session interface to the host subsystems, IMS/VS and CICS/VS, using SNA/SDLC protocols. The host subsystems run on System/370-type systems with Virtual Telecommunications Access Method (VTAM). The SNUF subsystem uses type-P protocols for sessions with IMS/VS and full function logical unit protocols with CICS/VS.

Line protocols with SSP-ICF SNUF are shown in *Appendix A: Data Communications Line Protocols.*

The following SNUF subsystem processes are shown in Chart 10.6.7:

1 Prepare for communications with a remote system.

2 Set up a session after successfully processing a program start request.

3 Perform error recovery after a program start request failure.

4 Process new requests from user programs.

5 Terminate communications with a remote host.

6 Process posts from C/SNA.

7 Handle program and enable errors.

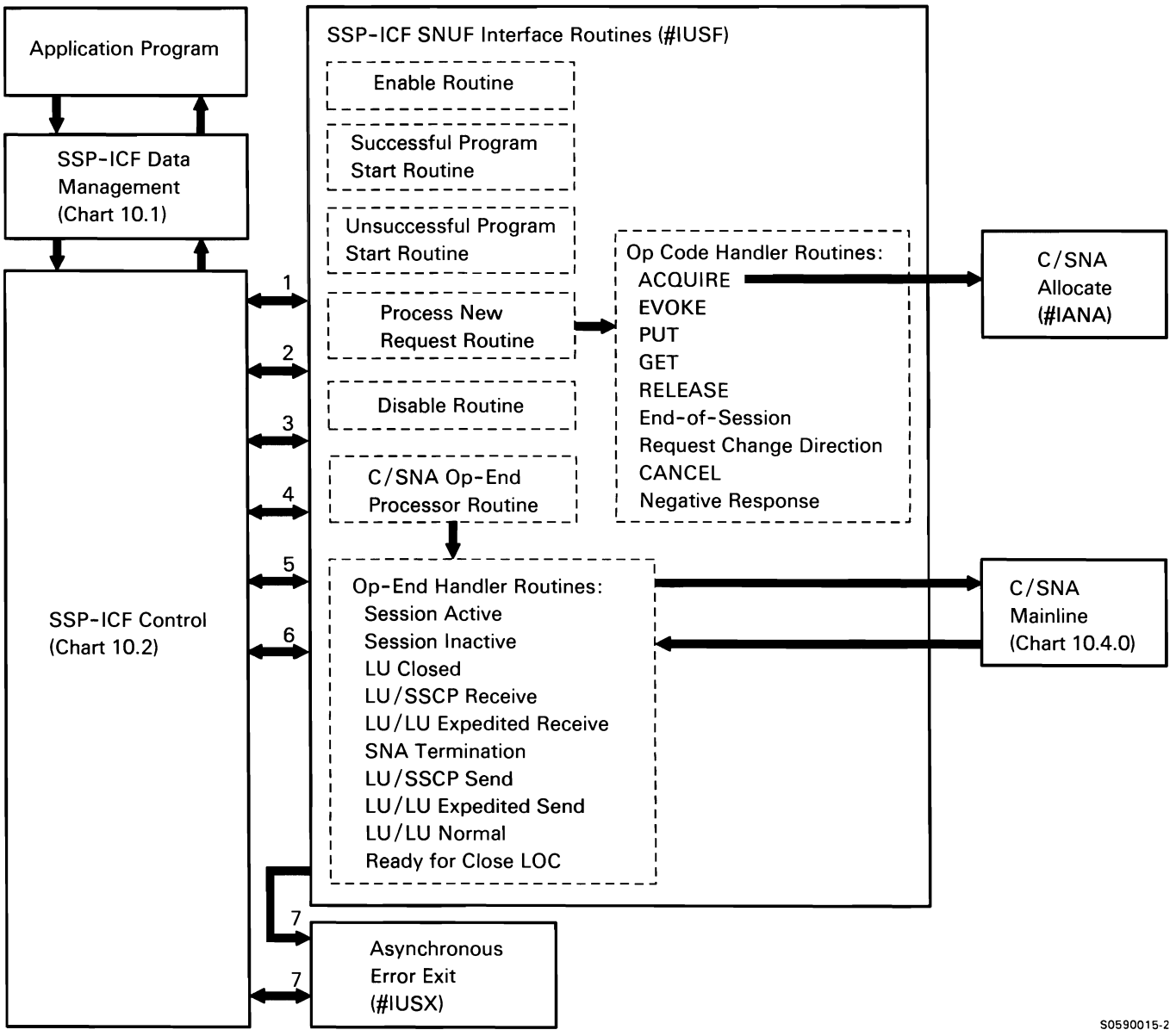


Chart 10.6.7 SSP-ICF SNUF Subsystem Control Flow

SSP-ICF Finance Subsystem

The System/36 SSP-ICF Finance subsystem provides the support that allows the System/36 user to write application programs to communicate with IBM 3601 or 4701 Finance Controllers and IBM 3694 Document Processors.

Line protocols used with the SSP-ICF Finance subsystem are shown in *Appendix A: Data Communications Line Protocols*.

The following Finance subsystem processes are shown in Chart 10.6.8:

1 Enable SSP-ICF Finance subsystem when control is received from System/36 enable ($IENBL).

2 Disable SSP-ICF Finance subsystem when control is received from System/36 disable (#IEDS).

3 Initiate operations after successful procedure start from the remote system.

4 Perform error recovery after an unsuccessful procedure start attempt from the remote system.

5 Process after issuing a SYSLOG message.

6 Process user program operation codes.

7 Process after SDLC op ends an operation.

8 On serious errors, disable the subsystem.

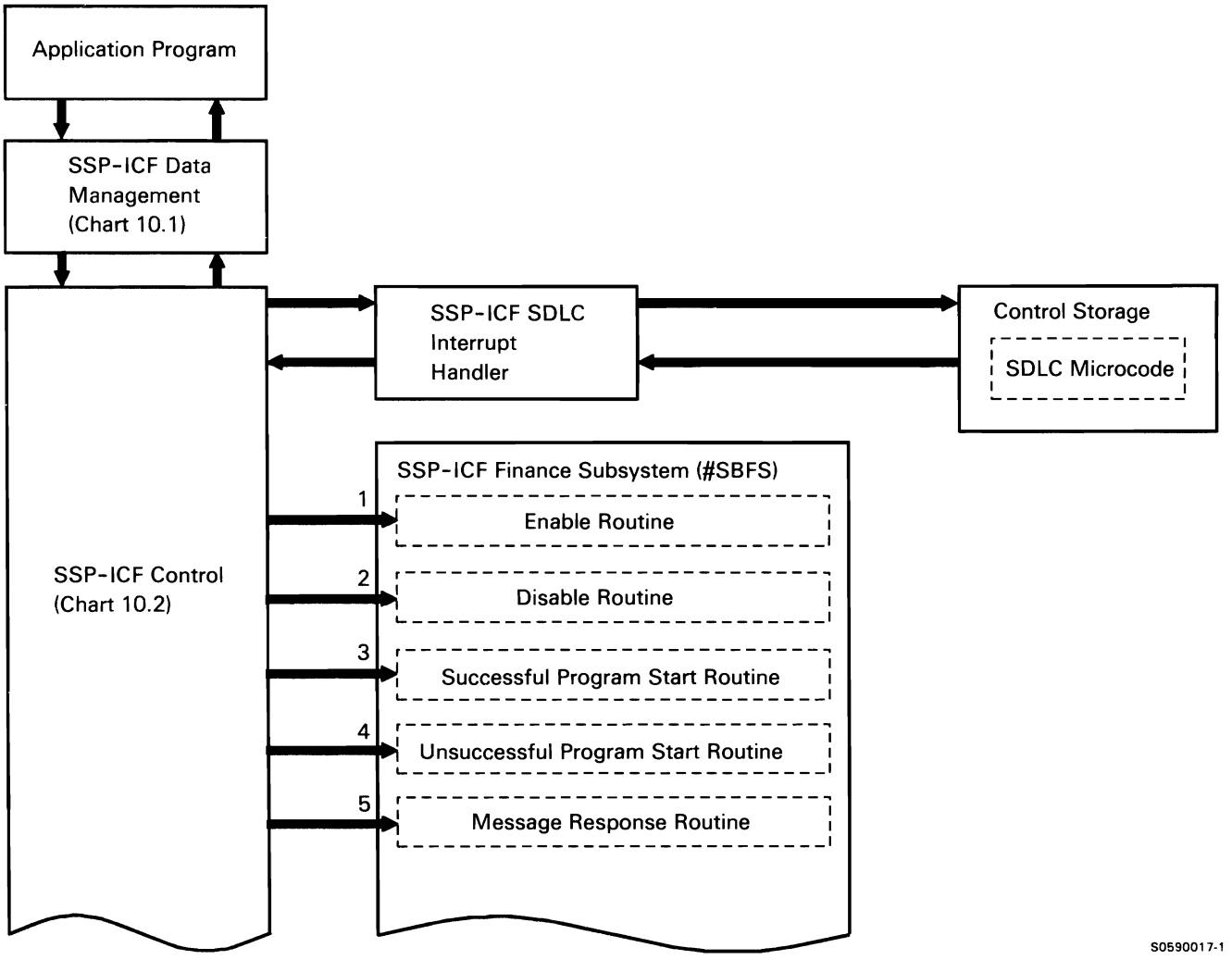


Chart 10.6.8 (Part 1 of 2) SSP-ICF Finance Subsystem Control Flow

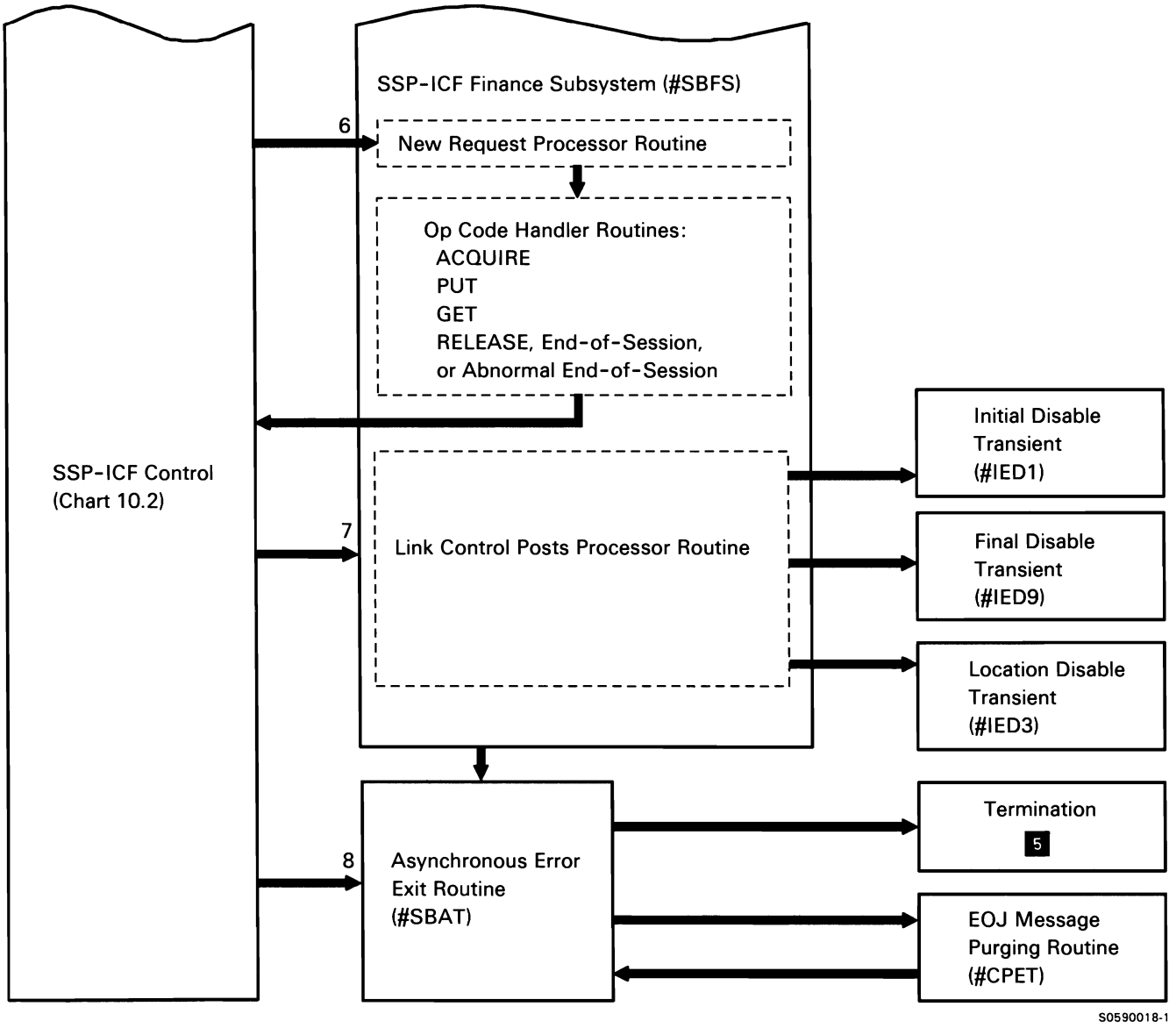


Chart 10.6.8 (Part 2 of 2) SSP-ICF Finance Subsystem Control Flow

SSP-ICF Peer Subsystem

The System/36 SSP-ICF Peer subsystem is SNA support that allows the System/36 user to write application programs to communicate with other programs on a remote System/34 or System/36. The Peer subsystem is prerequisite support for the Distributed Disk File Facility.

Line protocols used with the SSP-ICF Peer subsystem are shown in *Appendix A: Data Communications Line Protocols*.

The following Peer subsystem processes are shown in Chart 10.6.9:

1 Prepare for communications with a remote location.

2 Set up a session after successfully processing a program start request.

3 Perform error recovery after a program start request failure.

4 Process posts of IOBs from SDLC or process console queue elements (CQEs) for SYSLOG messages.

5 Process new requests from user programs.

6 Terminate communications with a remote location.

7 Process subsystem or enable asynchronous errors.

8 Post SDLC with appropriate data-to-line, and return to calling routine.

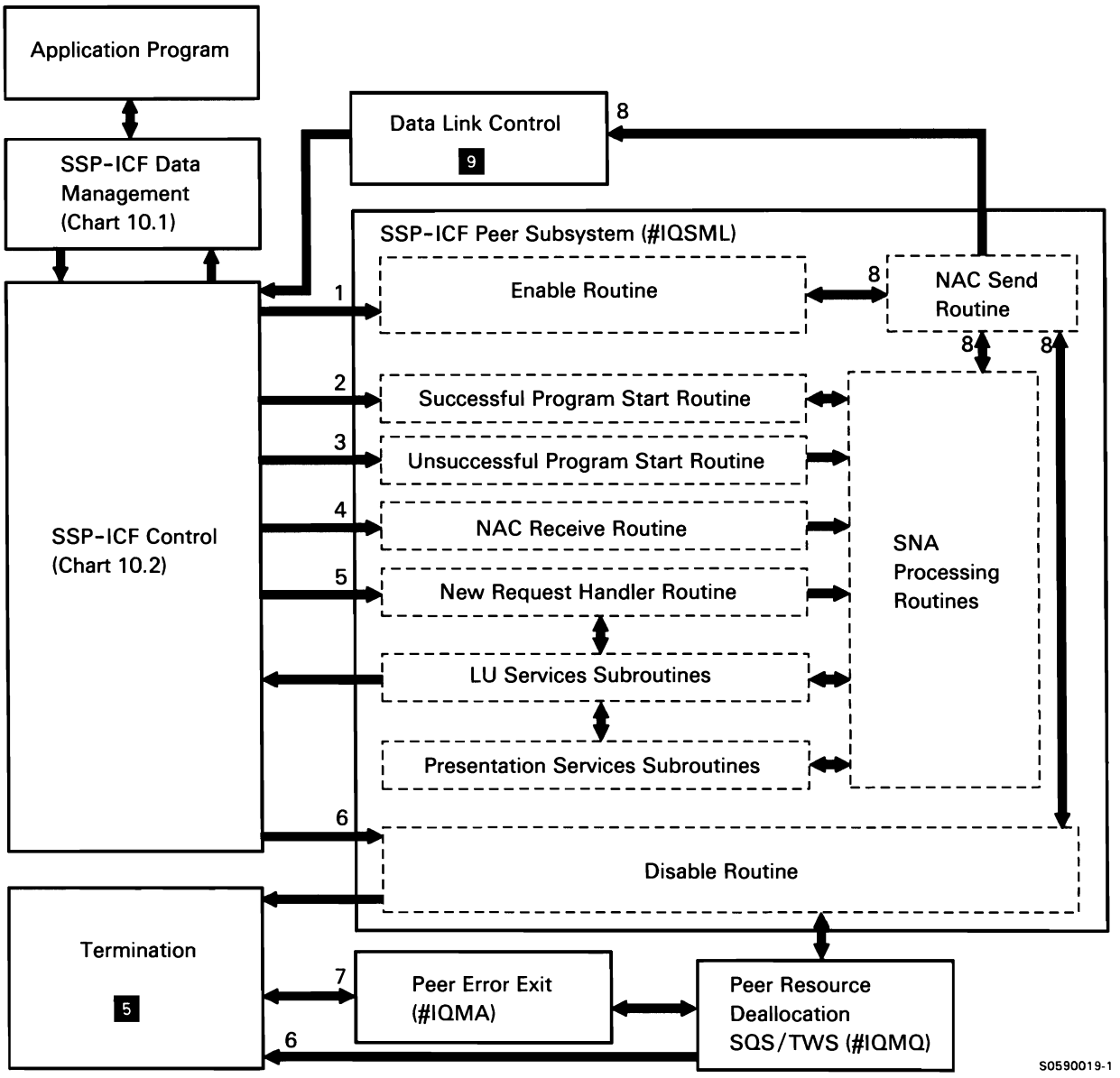


Chart 10.6.9 SSP-ICF Peer Subsystem Control Flow

SSP-ICF Advanced Program-to-Program Communications (APPC) Subsystem

The System/36 advanced program-to-program communications (APPC) subsystem provides for System/36 connectivity to other systems that support SNA LU type 6.2 protocols.

As described in this topic, the APPC subsystem supports application program to application program communication and includes support for the System/36 distributed data management (IDDM), distributed office system support, and Display Station Pass-Through features. C & SM problem management (alert support) also uses the APPC subsystem via a system (SSP-ICF control) interface.

In addition to the DTF interface, application programs use the following language-level interfaces to the APPC subsystem:

- $$FORMAT contains screen formats used for I/O performed during an SSP-ICF session.
- Externally-described data formats are built by IDDU.

Line protocols used with APPC are shown in *Appendix A: Data Communications Line Protocols.*

The following APPC subsystem processes are shown in Chart 10.6.10:

1 Process STRTGRP and STOPGRP commands by beginning/ending appropriate sessions and transactions according to command parameters:
   - Allocate and open work station DTF.
   - Process command parameters.
   - Exchange CNOS commands with remote system (parallel-session locations only).
   - Post APPC subsystem.
   - Release session acquired to execute CNOS, and terminate.

2 Process remote CNOS command by beginning/ending appropriate sessions and transactions according to command parameters:
   - Allocate and open work station DTF.
   - Issue an accept operation.
   - Initialize session group block (SGB) and post it to the APPC subsystem.
   - Release session acquired to execute CNOS, and terminate.

3 Route for change-number-of-sessions exchange with remote location.

4 Process the enable command:
   - If entire APPC subsystem is being activated, update status of each remote location in the specified configuration.
   - If a remote location is being activated, update status of that location in the specified configuration.

5 Set up a session after successfully processing a program start request.

6 Perform error recovery after a program start request failure.

7 Process posts from LU service models.

8 Process new requests (op-codes) from user programs:
   - Verify requested operation is valid.
   - Route for appropriate subsystem processing.
   - Post application program with operation complete return code.

9 Build/convert function management header 5 (FMH5):

- If request from local application program, build FMH5 to start a process at the remote APPC system.
- If request from remote application program, convert FMH5 to start a procedure at the local APPC subsystem.

10 Process posts from C/SNA.

11 Send problem management alert data to host on PU-to-SSCP flow.

12 Process the disable command:

- If entire APPC subsystem is being deactivated, perform deactivation processing for each remote location in the specified configuration.
- If a remote location is being deactivated, issue PREPARE TO CLOSELOC and CLOSELOC operations to C/SNA.
- If the remote location being deactivated is sending alerts, inform the alert task.

13 Free all SGBs associated with the LOC pointed to by XR2 and/or dequeue LOC pointed to by XR2.

14 Free and dequeue SCT pointed to by XR2. If there are no active subsystem configurations, go to end of job.

15 Handle program and enable errors.

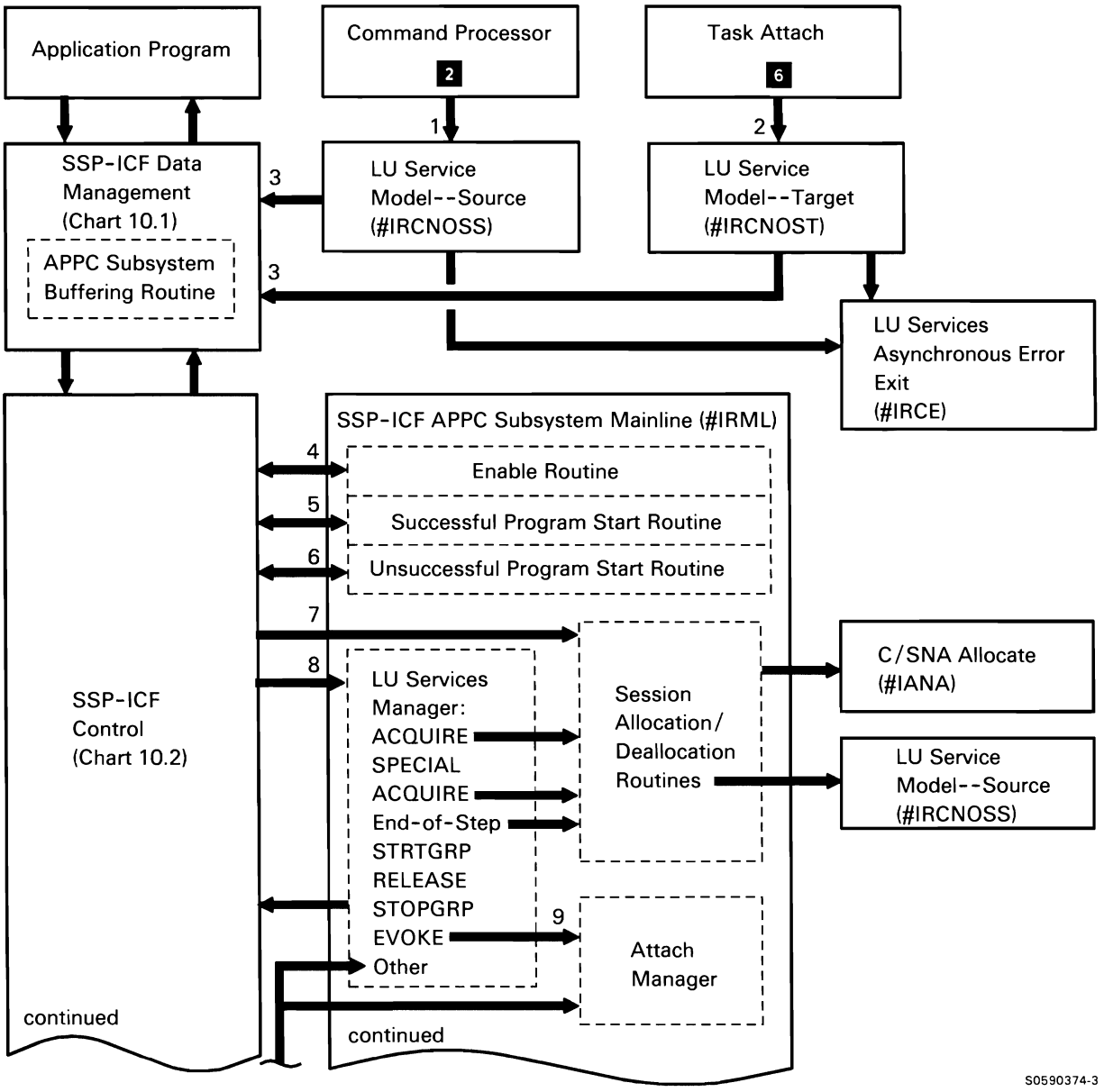


Chart 10.6.10 (Part 1 of 2) SSP-ICF APPC Subsystem Control Flow

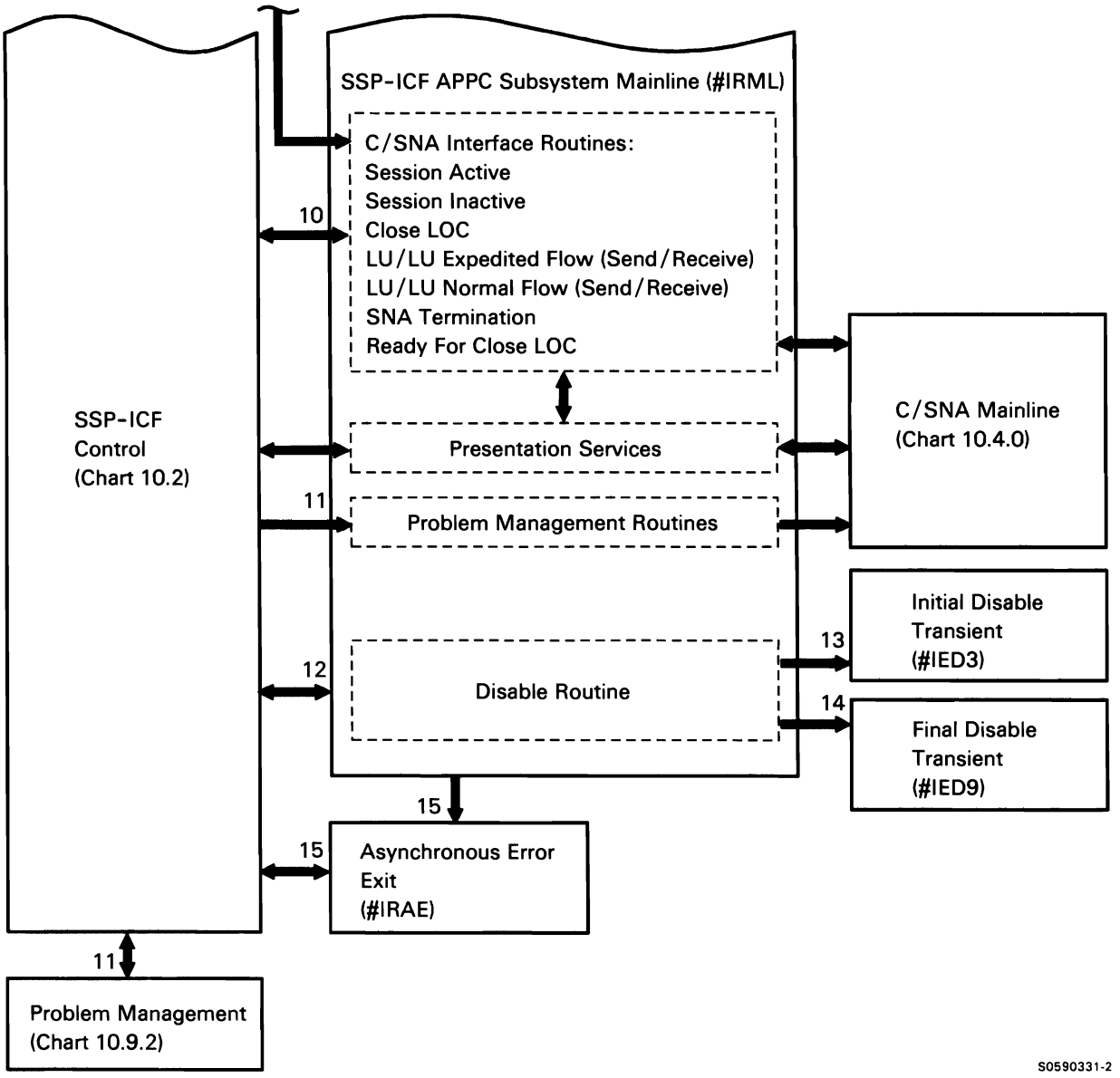


Chart 10.6.10 (Part 2 of 2) SSP-ICF APPC Subsystem Control Flow

SSP-ICF Advanced Peer-to-Peer Networking (APPN) Subsystem

The System/36 advanced peer-to-peer networking (APPN) subsystem provides for System/36 networking to other systems that support the APPN protocols. In addition, other systems that support SNA LU type 6.2 protocols can participate as end nodes (not including CICS/VS). See the *Advanced Peer-to-Peer Networking Guide*, SC21-9471-0, for more information.

As described in this topic, the APPN subsystem supports application program to application program networking and includes support for the System/36 distributed data management (DDM), distributed office system support, and Display Station Pass-Through features.

In addition to the DTF interface, application programs use the following language-level interfaces, which are the same as the APPC subsystem:

- $$FORMAT contains screen formats used for I/O performed during an SSP-ICF session.
- Externally described data formats are built by IDDU.

Line protocols used with APPN are shown in *Appendix A: Data Communications Line Protocols*.

The following APPN subsystem processes for a session request are shown in Chart 10.6.11.1.

Note: The numbers on Charts 10.6.11.1 and 10.6.11.2 indicate steps in each process.

1 Issue acquire for session not pre-established.

2 Pass session request.

3 Post for dynamic route.

4 Post with request.

5 Post to check if route is available before searching network.

6 Post with response. If positive, step 13 is executed.

7 If negative, response returns. Post to search for resource.

8 Pass special acquire through subsystem.

9 Pass responses from remote locations through subsystem.

10 Post with responses from remote locations.

11 Post with request for route.

12 Post with response.

13 Post with positive or negative reply.

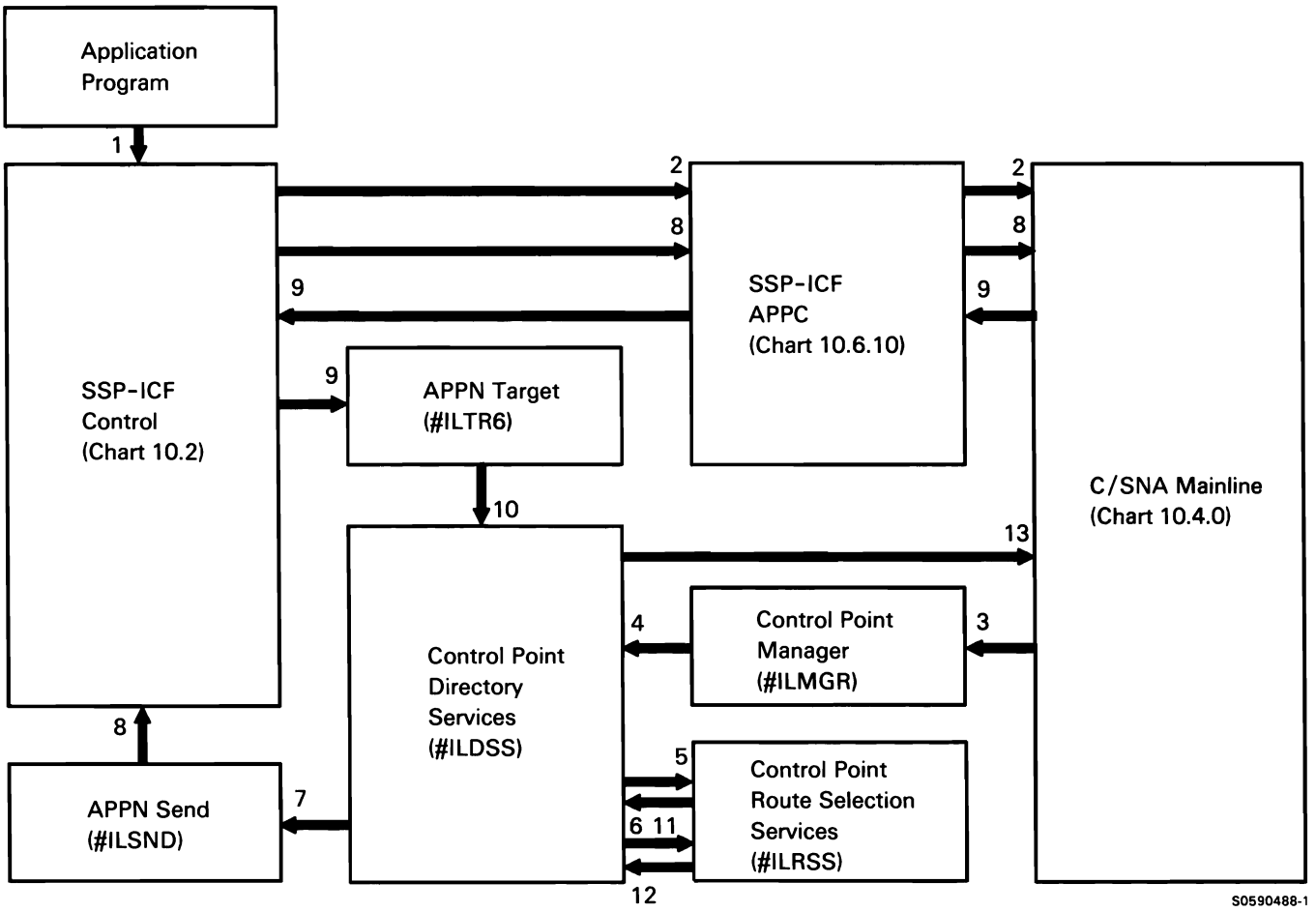


Chart 10.6.11.1 SSP-ICF APPN Subsystem Control Flow for Session Request

The APPNINFO procedure provides the user with access to information needed for problem determination in an APPN network.

The types of information made available are:

- Session trace
- Directory of locations
- Network configuration

The following APPN service functions are shown in Chart 10.6.11.2:

1 Control routed based on the first parameter entered on the procedure:
   - NETWORK starts network utility
   - LOCATIONS starts location utility
   - SESSION starts session utility

2 Request made to dump network information to disk.

3 Request made to dump location information to disk.

4 Read session data that was logged by C/SNA.

5 Retrieve necessary message text.

6 Process prompts and responses as required.

7 Read/write information from/to disk.

8 Issue any required error messages.

9 Perform any printing associated with user requests.

10 Terminate this job step.

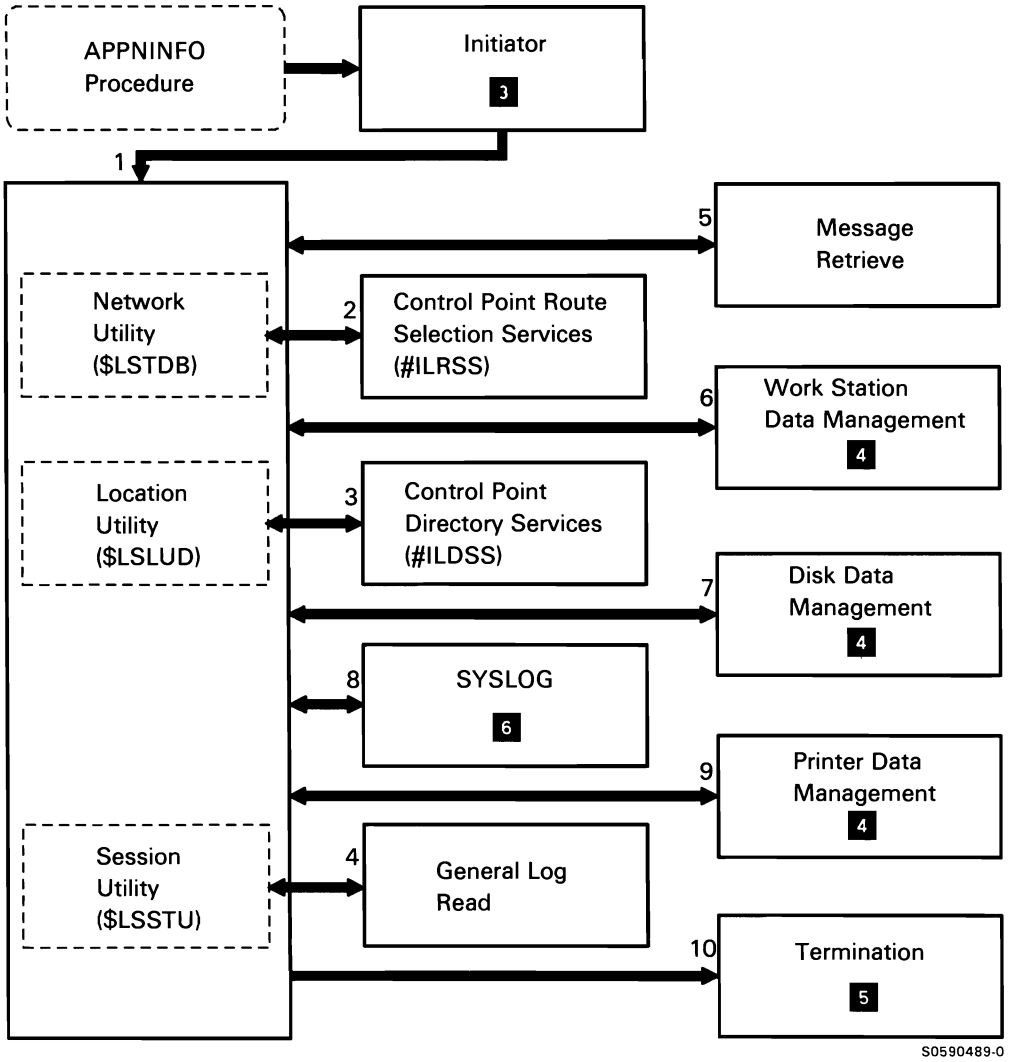


Chart 10.6.11.2 APPN Services Control Flow

SSP-ICF Asynchronous Subsystem

The System/36 SSP-ICF asynchronous subsystem provides the support that allows the System/36 user to write application programs to communicate with asynchronous devices.

Protocols used with the SSP-ICF asynchronous subsystem are defined within the user application program.

The following asynchronous subsystem processes are shown in Chart 10.6.12:

1 Enable SSP-ICF asynchronous subsystem when control is received from System/36 enable ($IENBL).

2 Initiate operations after a successful procedure start from the remote system.

3 Perform error recovery after an unsuccessful procedure start attempt from the remote system.

4 Process user program operation codes.

5 Disable SSP-ICF asynchronous subsystem when control is received from System/36 disable (#IEDS).

6 On serious errors, disable the subsystem.

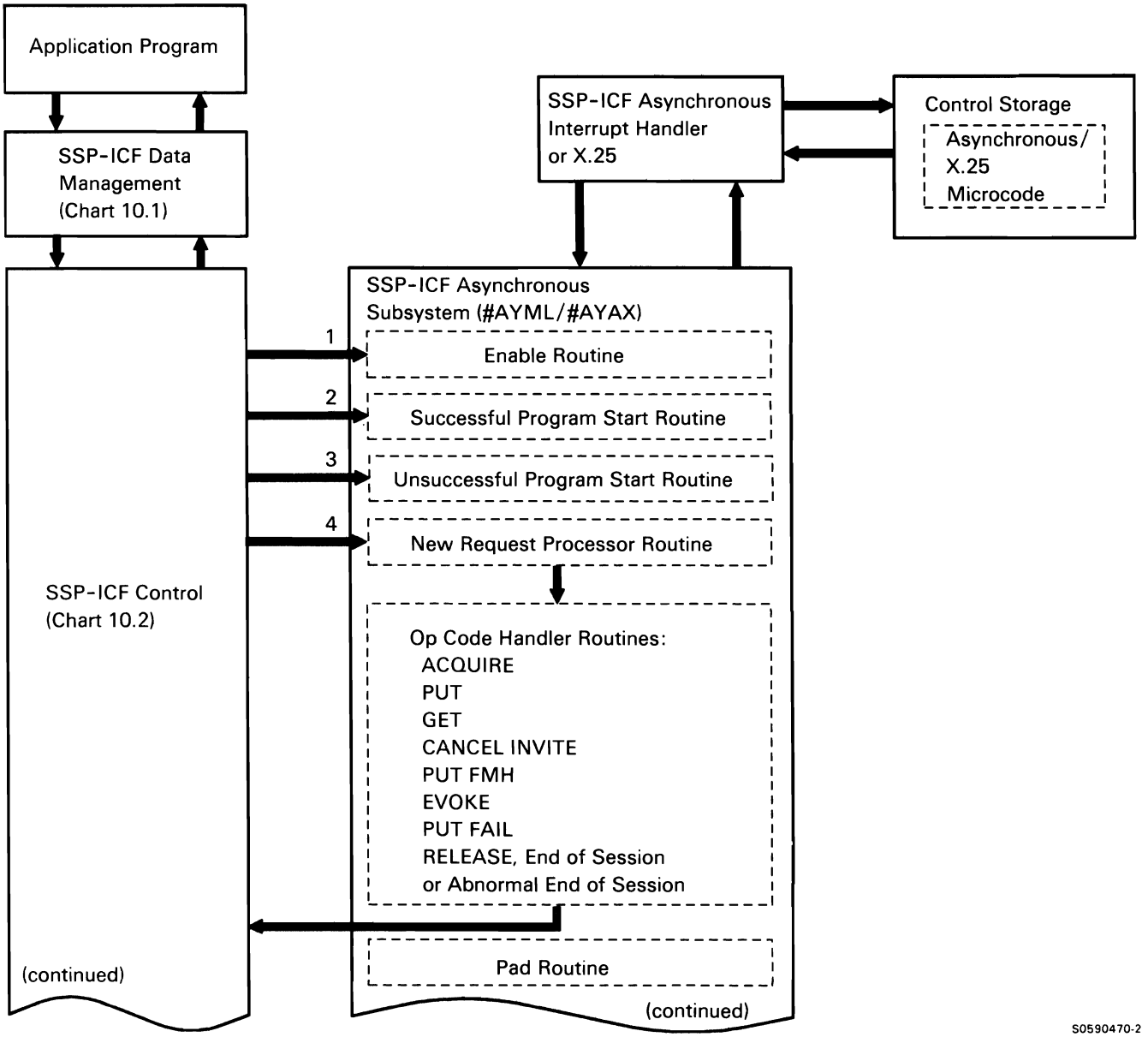


Chart 10.6.12 (Part 1 of 2) SSP-ICF Asynchronous Subsystem Control Flow

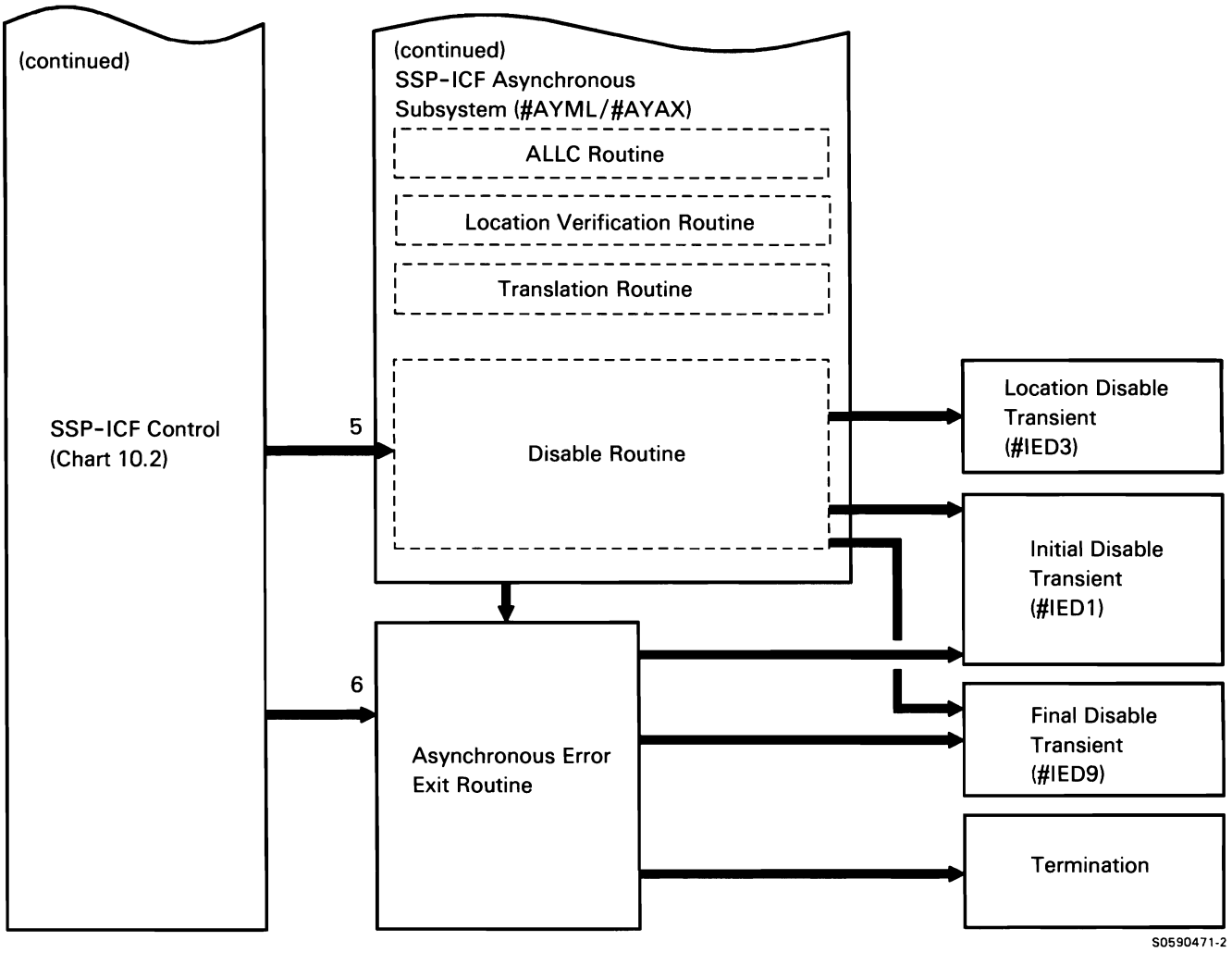


Chart 10.6.12 (Part 2 of 2) SSP-ICF Asynchronous Subsystem Control Flow

This page is intentionally left blank.

MULTIPLE SESSION REMOTE JOB ENTRY (MSRJE) SUPPORT SUBFUNCTION

The MSRJE support subfunction provides the user with comprehensive remote job entry capabilities to a System/370-type host system. It includes a command interface for reader/console input, and printer/punch support that interfaces directly with SSP printer data management and SSP disk data management. It also contains the support needed to interface with the System/36 BSC control storage code and C/SNA, and two data-handling utilities. The MSRJE support subfunction consists of the following components:

- MSRJE BSC interrupt handler — Chart 10.7.1
- MSRJE BSC subsystem — Chart 10.7.2
- MSRJE SNA subsystem — Chart 10.7.3
- MSRJE reader/console support — Chart 10.7.4
- MSRJE printer/punch support — Chart 10.7.5
- MSRJE forms control table utility — Chart 10.7.6
- MSRJE disk file utility — Chart 10.7.7

Line protocols used with MSRJE SNA and BSC support are shown in *Appendix A: Data Communications Line Protocols*.

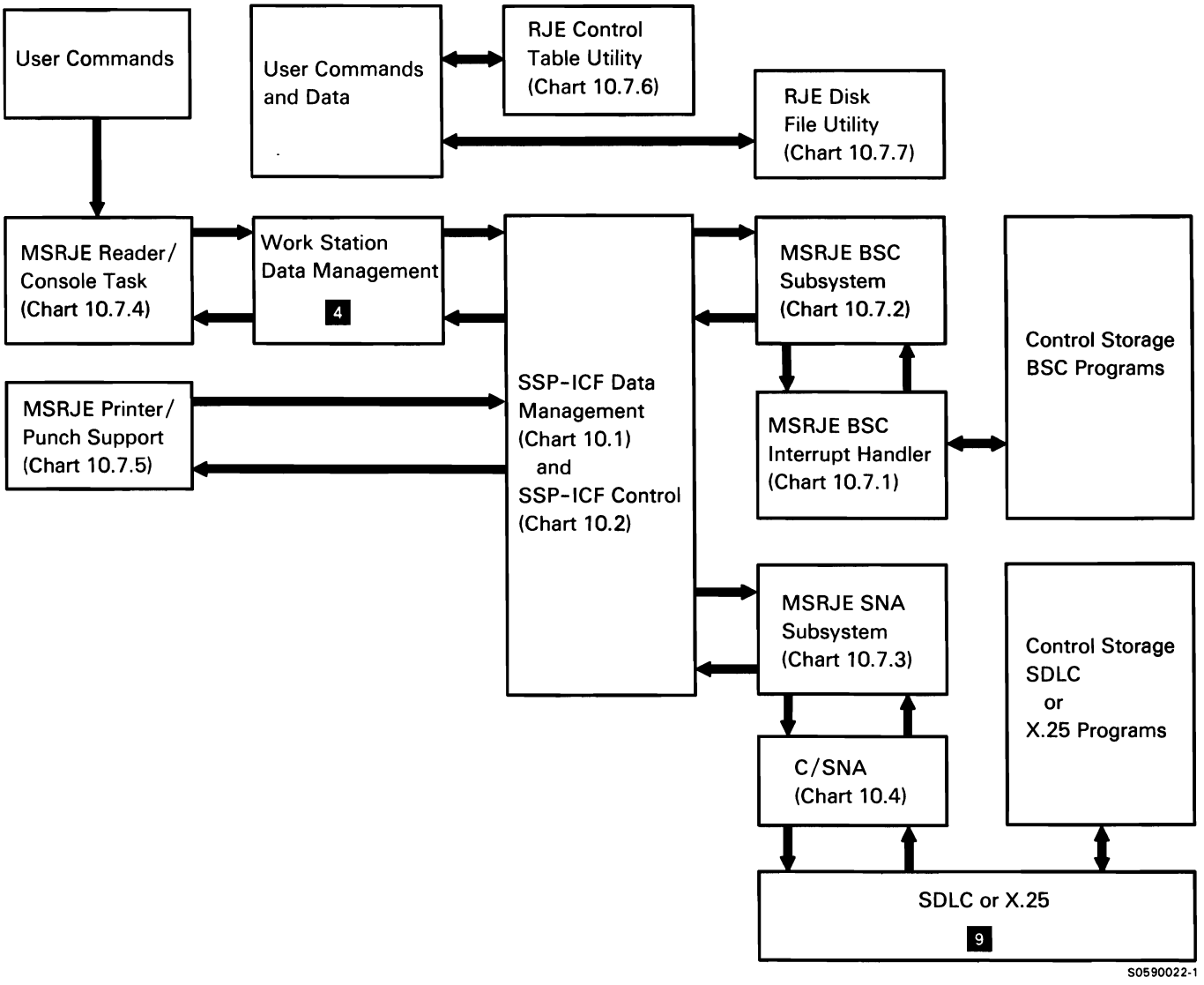


Chart 10.7.0 MSRJE Support Subfunction Overview Control Flow

MSRJE BSC Interrupt Handler

The MSRJE BSC interrupt handler (main storage) supports binary synchronous communications between the MSRJE subsystem and IBM System/370 or equivalent systems. It is transferred to as a task when the first line is allocated in #RJBP, the BSC subsystem protocol module. The protocol module also assigns IOBs and the MSRJE BSC block (MBB), and directs the line allocate subfunction to load the proper communications microcode. The protocol module then posts the interrupt handler for initialization. After the initial setup, the interrupt handler gets control when an IOB op ends, when the subsystem posts the interrupt handler, when a SYSLOG message that was issued by the interrupt handler is answered, and when one of its subtasks goes to end of job.

The following MSRJE BSC interrupt handler processes are shown in Chart 10.7.1:

1 At IOB op-end, check IOB return code and data stream for errors and issue the next communications operation.

2 Handle the following subsystem posts:

- Post to initialize MBB, and timer, and line IOBs for a communications line.
- Post to start the initiation subtask (#RJII) to enable the BSC adapter and send logon/signon.
- Post to post IOB containing transmit/receive overlay to BSC control storage code to transmit data.
- Post notifying interrupt handler of an available reserve buffer. (Reserve buffers are used by the interrrupt handler to maintain line activity by sending ACK0s.
- Post to start termination subtask (#RJIT) to disable adapter.
- Post to disable processing and, if last line, go to end of job.

3 Process SYSLOG message according to response.

4 Perform initialization subtask termination:

- Post subsystem informing it that line initialization was successful or unsuccessful.
- If required, finish disable.
- If required, perform abnormal termination of subtask.

5 Perform termination subtask termination:

- Post subsystem informing it that line is terminated.
- If required, finish disable.
- If required, perform abnormal termination of subtask.

6 If required, perform abnormal termination of MSRJE BSC interrupt handler.

The MSRJE BSC interrupt handler subfunction includes an error transient that handles its abnormal termination. If the interrupt handler or any of its subtasks terminate abnormally, all MSRJE BSC sessions are disabled.

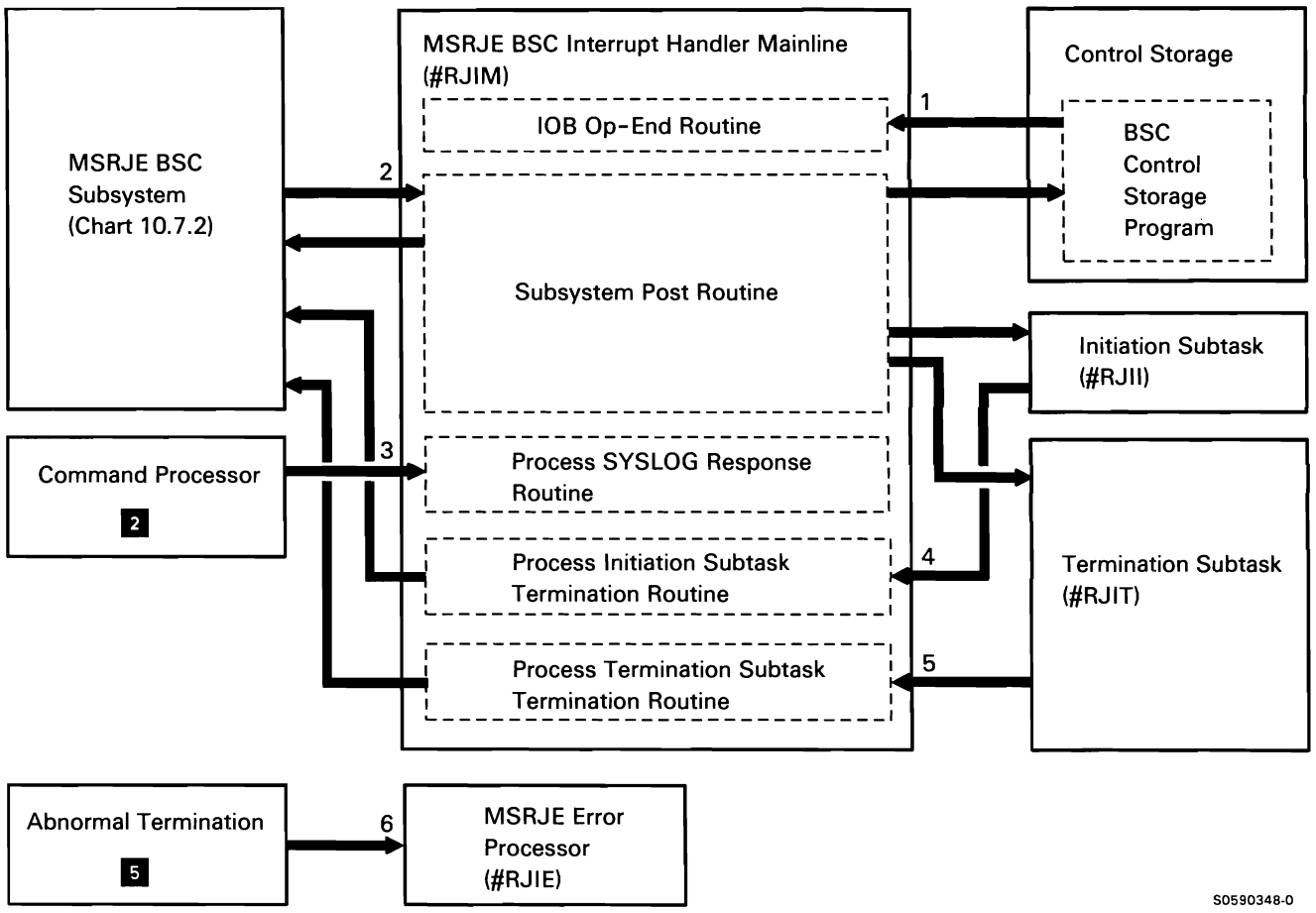


Chart 10.7.1 MSRJE BSC Interrupt Handler Control Flow

Multiple Session Remote Job Entry (MSRJE) BSC Subsystem

The MSRJE BSC subsystem is the interface between the device tasks and the MSRJE BSC interrupt handler. When an input device fills a line buffer, it posts the subsystem. The input device is given an empty line buffer and the interrupt handler is posted to send the full buffer to the host. When a text buffer is received from the host, the interrupt handler posts the subsystem. The subsystem checks the buffer; if it is for the printer/punch task, the subsystem queues the buffer to the task for processing, and starts the task if it is not active. If the buffer contains a console message, the subsystem decompresses the message and sends it to the reader/console task (if it is active) and to the history file. The subsystem also responds to forms mount messages and control sequences received.

The subsystem also handles initialization and termination. The initial ENABLE command causes the subsystem to be loaded. The subsystem protocol module allocates the line, loads the MSRJE BSC interrupt handler (if it is not already loaded), and allocates all control blocks and buffers for the subsystem. When the first MSRJE procedure for a line is entered, the subsystem establishes the connection to the host via the MSRJE BSC interrupt handler, then the interrupt handler sends the LOGON/SIGNON command to the host system. When the delay from the END command or the configured delay time has expired, the subsystem directs the interrupt handler to send LOGOFF/SIGNOFF to the host, thus ending the connection. The DISABLE command for the last line causes the subsystem and interrupt handler to go to end of job.

The following MSRJE BSC subsystem functions are shown in Chart 10.7.2:

1 Process ENABLE command for subsequent enables.

2 Process the DISABLE command from user by posting all active device tasks to terminate and posting the interrupt handler to disable the line.

3 Process post of successful program start.

4 Process post of unsuccessful program start by reissuing program start request (for first failure) or by issuing a message to the user (for subsequent failures).

5 Process SSP-ICF operation request (in SUB) from device task.

6 Process BSC MSRJE interrupt handler normal op-end (data has been received).

7 Process BSC MSRJE interrupt handler error op-end by posting all active tasks to terminate.

8 Process the following special requests:

- Timer op end.
- Command requests.
- Command responses.
- Inactive post from last device task on LOC.
- Interrupt handler; buffer free.
- Interrupt handler; communications established.
- Interrupt handler; answer disable.
- Interrupt handler; answer disable kill.
- Interrupt handler; disconnect received.

Note: In addition to the control flow shown, the reader/console and printer/punch tasks can also post SSP-ICF control directly.

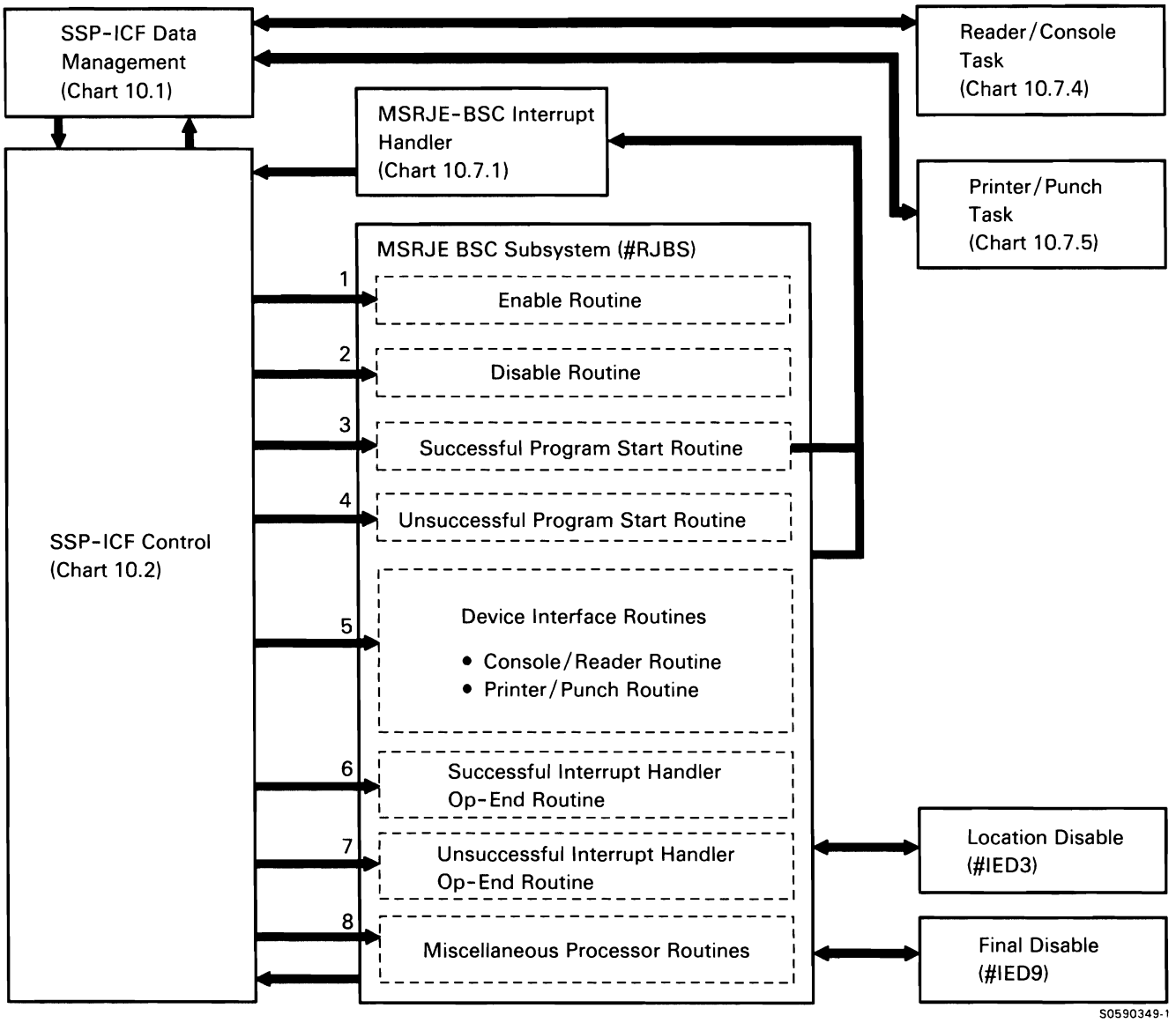


Chart 10.7.2 MSRJE BSC Subsystem Control Flow

Multiple Session Remote Job Entry (MSRJE) SNA Subsystem

The System/36 MSRJE SNA subsystem provides the interface for MSRJE device tasks to host applications RES, JES2, JES3, and DOS. The subsystem processes enable/disable and activation/deactivation requests, and manages all LU-to-LU data in conjunction with C/SNA.

The following MSRJE SNA subsystem functions are shown in Chart 10.7.3:

1 Process enable command for subsequent enables.

2 Process the disable command from the user and post C/SNA to disable the line.

3 Process post of successful procedure start.

4 Process post of unsuccessful procedure start by reissuing program start request (for first failure) or by issuing a message to the user (for subsequent failure).

5 Process SSP-ICF operation request from device task.

6 Process for the following:

- C/SNA post to subsystem.
- Timer op end.
- REQUEST or RESPONSE post.
- INACTIVE post from last device task on LOC.

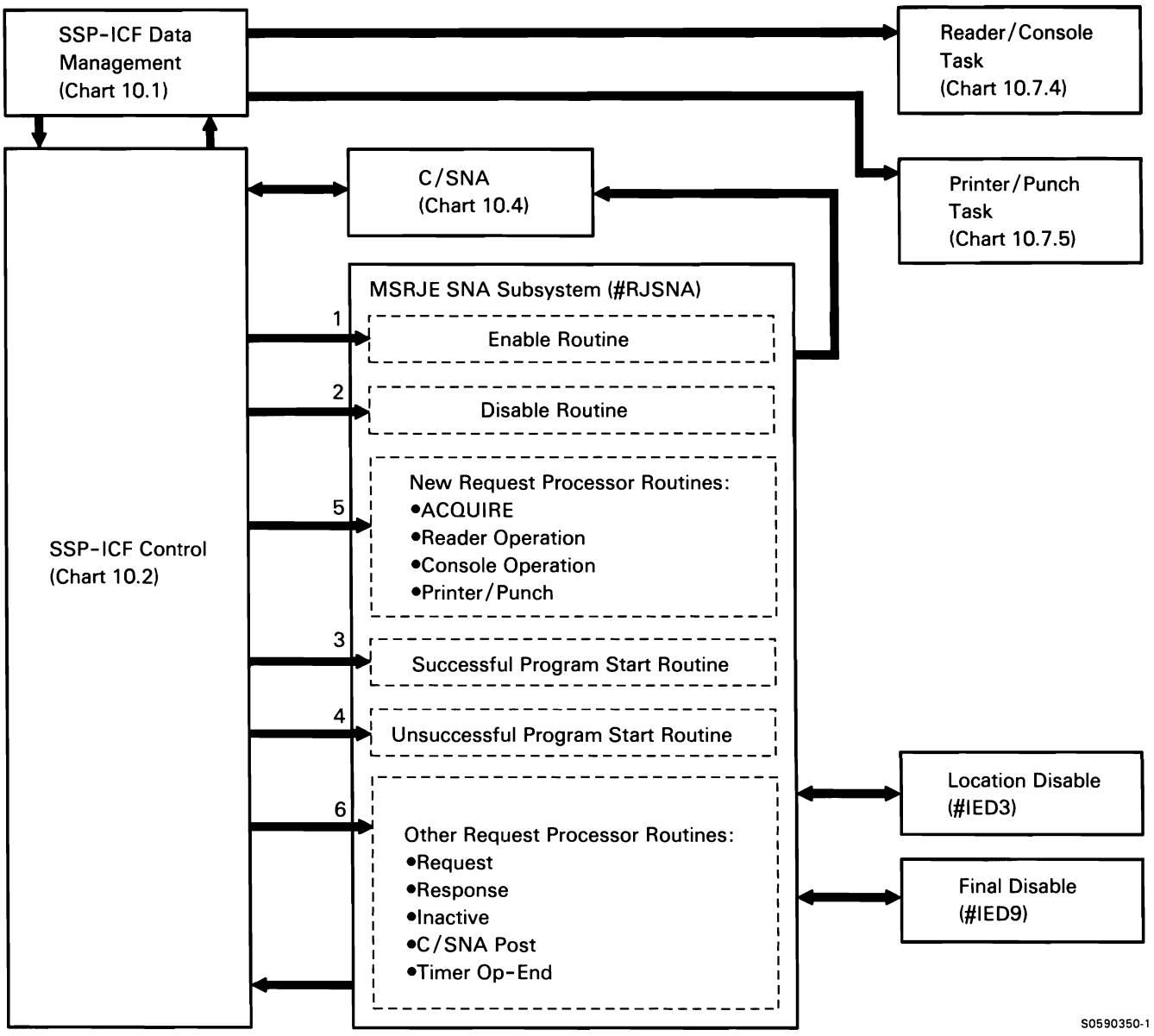


Chart 10.7.3 MSRJE SNA Subsystem Control Flow

MSRJE Reader/Console Task

The MSRJE reader/console (R/C) task is the user interface to the MSRJE function. It can be evoked by the // EVOKE OCL statement, run from the JOBQ, started by an ICF program, or activated by entering the MSRJE procedure statement from a display station. When the MSRJE reader/console task is activated by the MSRJE procedure statement, it starts the initiation phase with the host system. After the initiation phase is completed, it processes the following:

- Command keys, commands, and data from a work station keyboard
- Data from disk to be placed in buffers (supplied by the subsystem) and sent to the host system
- Return codes
- Timer time-outs
- Requests and responses from the MSRJE subsystem and SSP-ICF

The following MSRJE reader/console processes are shown in Chart 10.7.4:

1 At session operation-end, process session request or response, or save buffer address.

2 Initialize library parameters.

3 Check readfile parameters on MSRJE procedure for errors.

4 Syntax check MSRJE commands entered from keyboard or from command-capable disk file.

5 Process parameters for a LIBRARY command.

6 Process a READFILE command.

7 Display error messages and message requests received from the subsystem or the reader/punch task.

8 Process all commands other than LIBRARY and READFILE.

9 Close and deallocate files.

10 Syntax check readfile parameters on MSRJE procedure.

11 Place data into TWS line buffer supplied by subsystem.

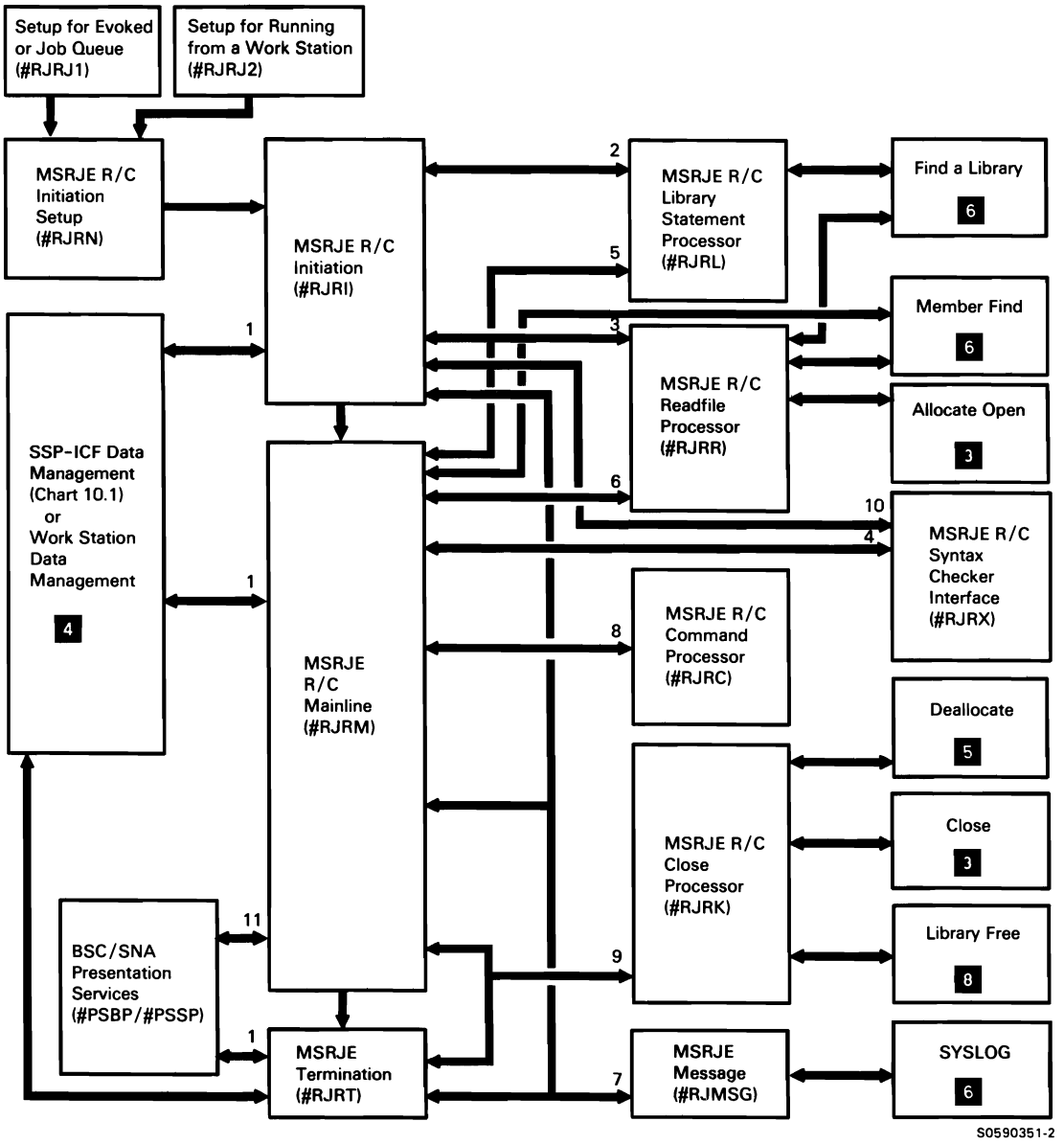


Chart 10.7.4 MSRJE Reader/Console Control Flow

MSRJE Printer/Punch Task

The MSRJE printer/punch task creates compressed disk (CDISK) files, punch files, and print data. The task is program-started by the MSRJE BSC or SNA subsystem. The data areas are initialized in #RJOI and control is transferred to the output task mainline routine.

The mainline routine examines the MSRJE function control block and forms control table entries, then produces the requested output.

The following MSRJE printer/punch processes are shown in Chart 10.7.5:

1 Process program start request from subsystem.

2 Allocate and open the work station DTF.

3 For work station operations, do the following:

- Get program load data.
- Set timer.

4 Get forms control table entry.

5 Allocate and open printer/disk file.

6 For work station operations, do the following:

- Get data buffer.
- Put negative response.
- Get message reply.
- Set timer.

7 Fill output buffer from line buffer received from subsystem.

8 Write to printer.

9 Write to disk file.

10 Close and deallocate printer/disk file.

11 Route messages to MSRJE console.

Note: If an output data set cannot complete normally because the device was canceled, option 2 was selected, or a permanent device error occurred, the termination transient receives control and the printer/punch task is terminated.

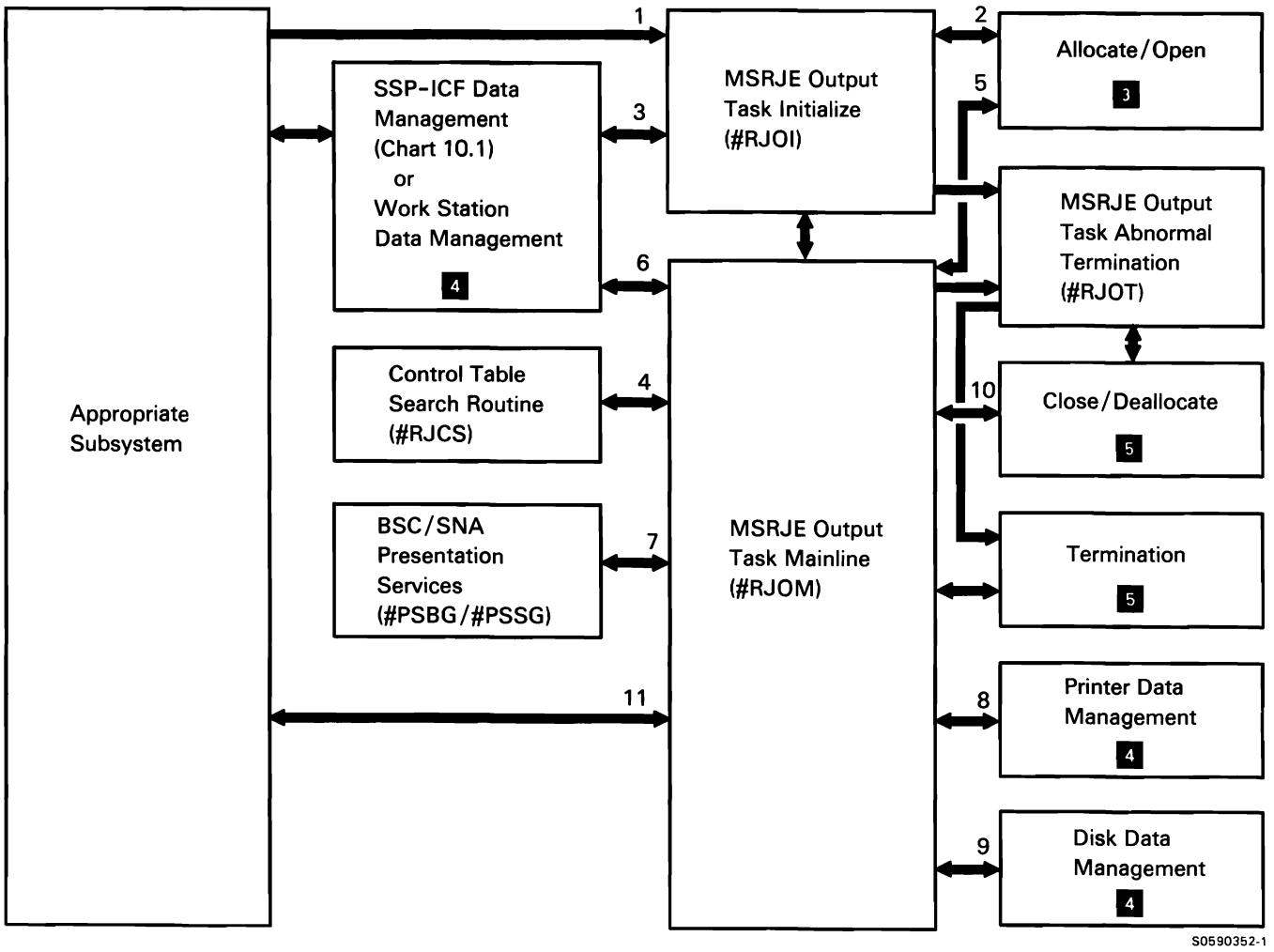


Chart 10.7.5 MSRJE Printer/Punch Control Flow

MSRJE Forms Control Table Utility

The MSRJE forms control table utility interactively builds, updates, removes, prints, or reviews an indexed file containing forms control information and disk file creation information. The user evokes the MSRJE forms control table utility with the RJTABLE procedure command.

The MSRJE control table utility processes are shown in Chart 10.7.6:

- Check and process the table menu option selected, the table name specified, and the table location.
- Check the return code from special allocate.
- Set up the opened file for entries.
- Process the entry type and output selected, and the entry name specified.
- Check and process all prompt displays.
- Convert character numbers to their binary equivalents.
- Convert binary numbers to their character equivalents.
- Syntax check the table name, file label, procedure name, library name, printer ID, and user ID.
- Process terminating error by dumping and exiting.
- Place print and punch default entries into the newly created table.
- Print entries from an existing control table file.
- Print the control table file.
- Review entries in a control table file.

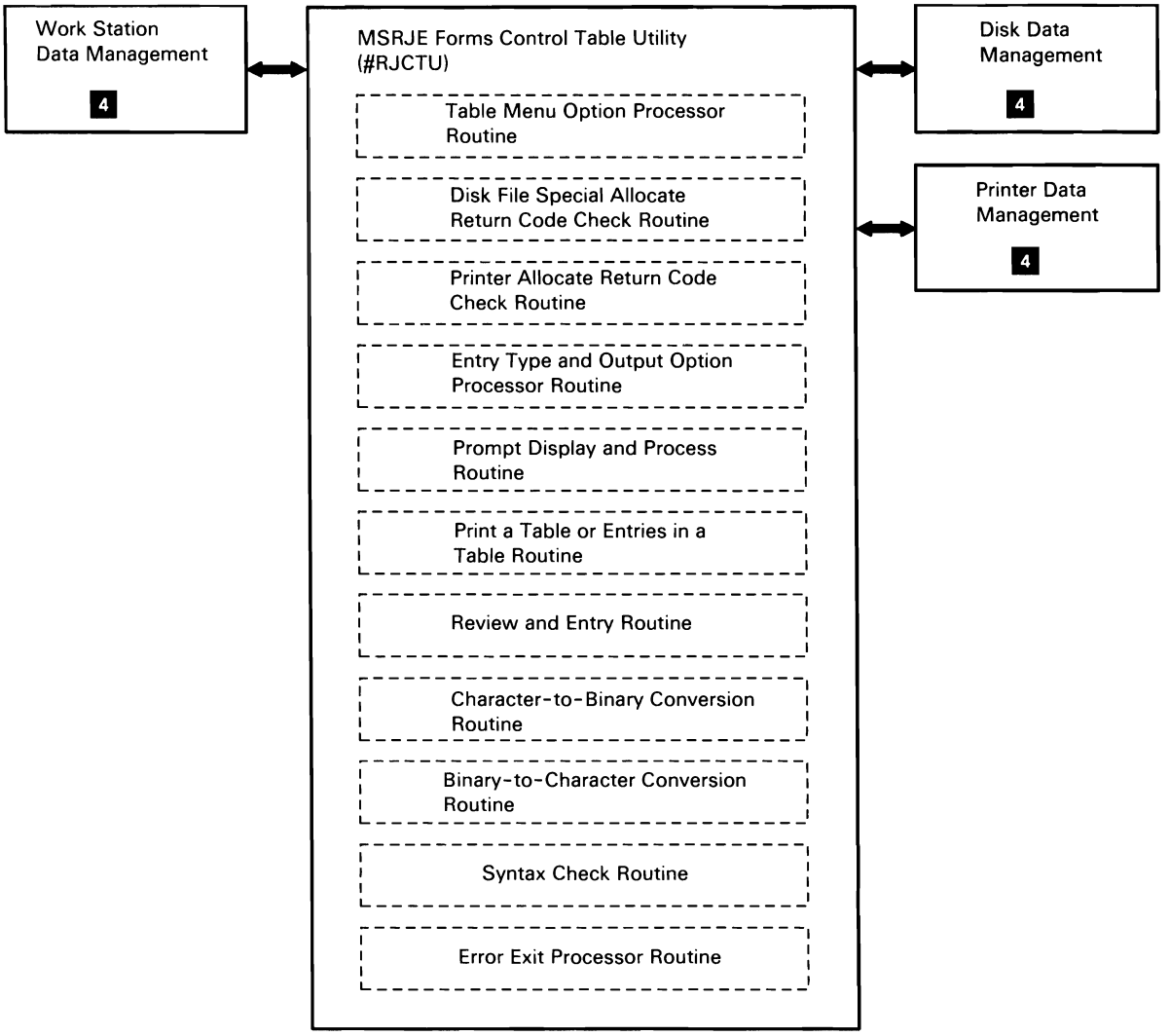


Chart 10.7.6 MSRJE Forms Control Table Utility Control Flow

MSRJE Disk File Utility

The disk file utility creates punch files and printer data from compressed disk (CDISK) files created during MSRJE sessions. It can be loaded from a work station, a procedure, or the job queue. The user specifies the CDISK file name on a COPYFILE utility control statement (or procedure parameters with the RJFILE procedure) along with an optional forms control table. The CDISK file can contain multiple datasets of SNA and BSC printer and punch data. The disk file utility creates a file for each dataset in the CDISK file. The user can provide as many COPYFILE statements per job step as needed. The END utility control statement terminates the utility.

The following MSRJE disk file utility processes are shown in Chart 10.7.7:

1 Syntax check utility control statements.

2 Retrieve forms control table entry.

3 Get SNA data record.

4 Get BSC data record.

5 Write punch record to data file or print record to data file.

6 Write print record to printer.

7 Terminate utility.

8 Process printer errors.

9 Process disk errors.

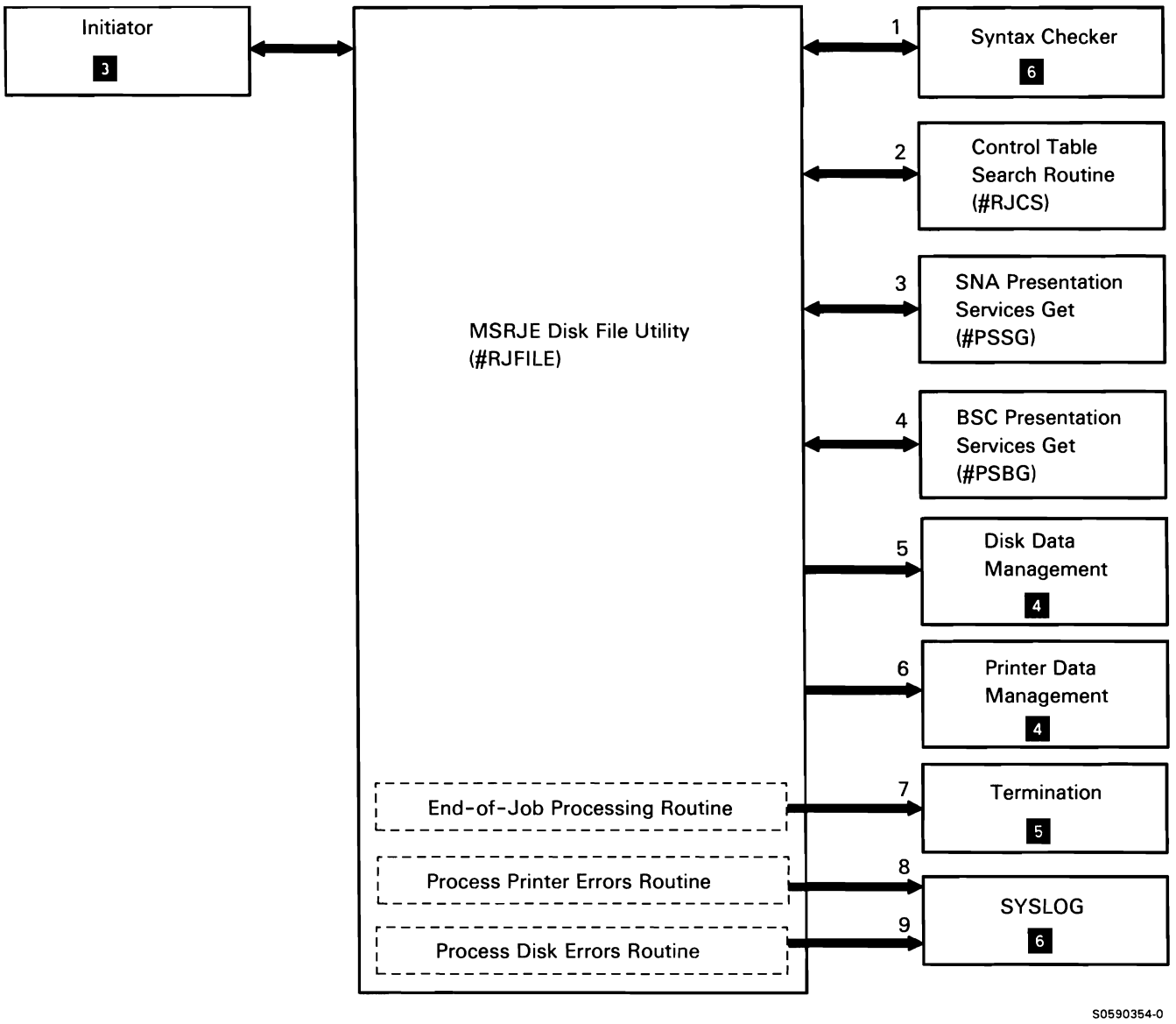


Chart 10.7.7 MSRJE Disk File Utility Control Flow

3270 SUPPORT SUBFUNCTION

The 3270 support subfunction allows System/36 and its display stations and printers to appear as a 3270 control unit with attached devices. (The 3270 device emulation translates the 3270 data stream from and to the 5250 data stream.) Support is provided for attachment to BSC, SNA/SDLC, and X.25 networks. When attached to a BSC 3270 network, the System/36 appears as a 3271 Model 2 cluster control unit with attached devices. When attached to an SNA/SDLC network, System/36 appears as a 3274 Model 1C control unit with attached devices. Host facilities for this emulation include System/370 VM/370 (BSC only), IMS/VS, CICS/VS, and TSO, and System/3 Model 15 CCP (BSC only).

Line protocols used with SNA 3270 (SDLC and X.25) and BSC support are shown in *Appendix A: Data Communications Line Protocols*.

Chart 10.8.0 shows the overview control flow for System/36 3270 device emulation. The display and printer emulation programs provide BSC and SNA terminal support that totally masks the 5250 device and causes the terminal to appear to the host system to be a 3270 device. Each System/36 device operating in emulation has its own copy of the applicable emulation program. With the exception of enable and disable processing, the BSC version of these programs interfaces directly with the BSC 3270 interrupt handler. User application programs can interface with BSC 3270 support via the BSC 3270 subsystem.

The SNA 3270 personal computer program provides the interface between the System/36 SNA 3270 subsystem and the EP3278 feature on the personal computer.

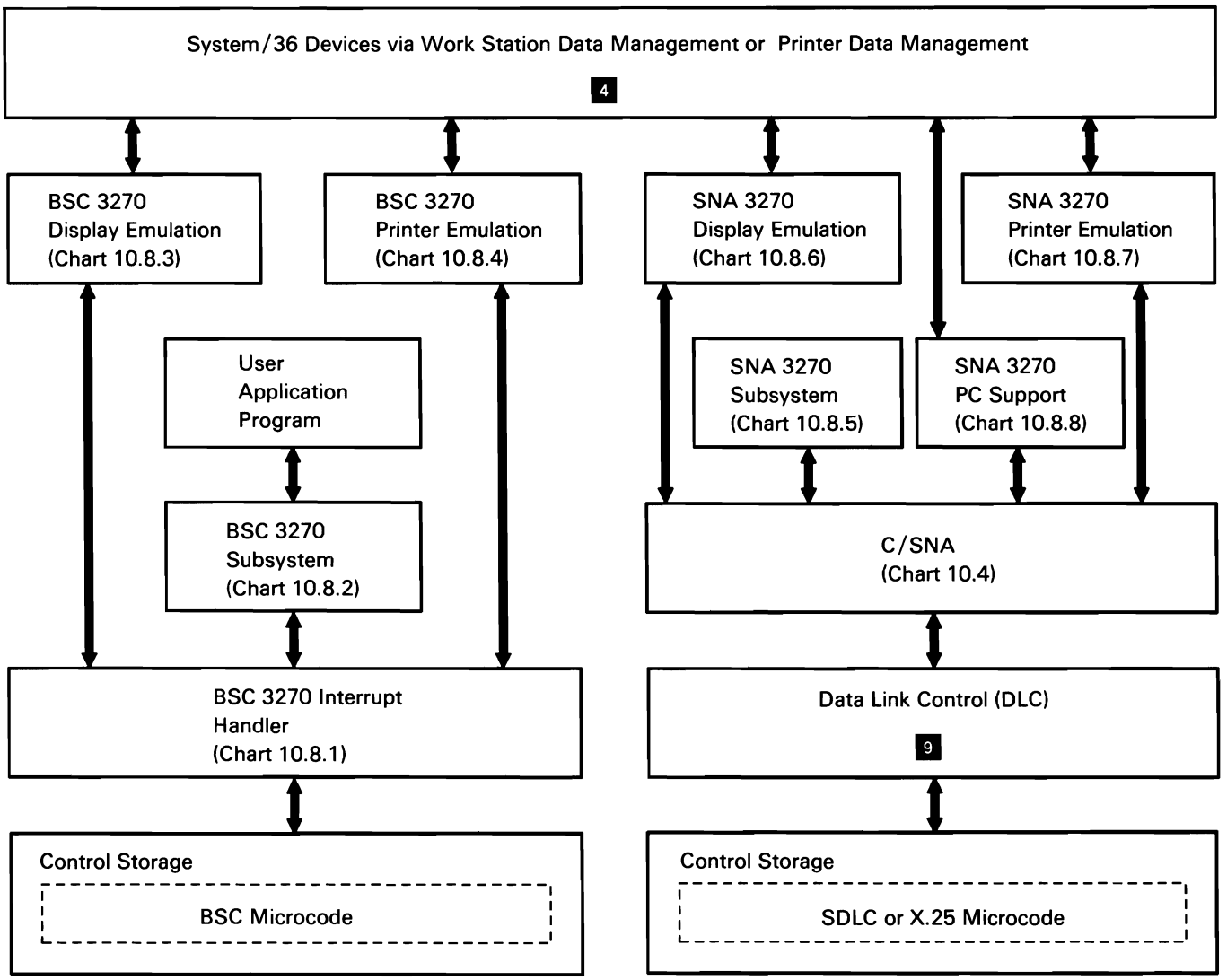


Chart 10.8.0 3270 Device Emulation Overview Control Flow

BSC 3270 Interrupt Handler

The BSC 3270 interrupt handler (in main storage) supports binary synchronous communications (BSC) between the BSC 3270 subsystem and a remote host system (for user program interface), or between a BSC 3270 device emulation task and a remote host system. It is attached by the BSC 3270 subsystem protocol module (#ITSCP) during enable as a nonswappable task. #ITSCP also allocates the line and assigns line buffers (in the TWS). The BSC 3270 subsystem posts the interrupt handler task to complete the enable.

The BSC 3270 interrupt handler communicates with the BSC 3270 subsystem and the BSC 3270 device emulation tasks via the realtime interface table (RIT). For more information on the RIT, refer to the *Data Areas* manual.

BSC 3270 Interrupt Handler Region Map

The following table shows the location of key data areas involved in the BSC 3270 interrupt handler:

Dump Address (in Hex)	Contents
0800	Nucleus
1000	BSC 3270 interrupt handler (#ITHML)
3000	BSC 3270 line buffers (2)
5000	Unused space

The following BSC 3270 interrupt handler processes are shown in Chart 10.8.1:

1 Process a disable post from the subsystem.

2 On disable, free IOBs.

3 Disable the adapter.

4 Complete enable processing (started by #ITSCP) by initializing and assigning IOBs and a work area, enabling the line, and posting the subsystem to acknowledge the enable request.

5 Make data terminal and modem ready.

6 Handle IOB op ends by verifying successful completion and routing to the post-task routine.

7 Update the RIT entry based on the IOB and post the task.

8 Handle task posts by setting up IOB for execution by control storage BSC microcode and posting it to the send-response routine.

9 Send a response to the host system by issuing an SVC instruction to control storage and inserting control characters in the data.

10 Handle time-outs from line or task (posted in IOB by control storage).

11 Perform error recovery by performing retry to limit set by configuration, then posting the subsystem to issue an error recovery message.

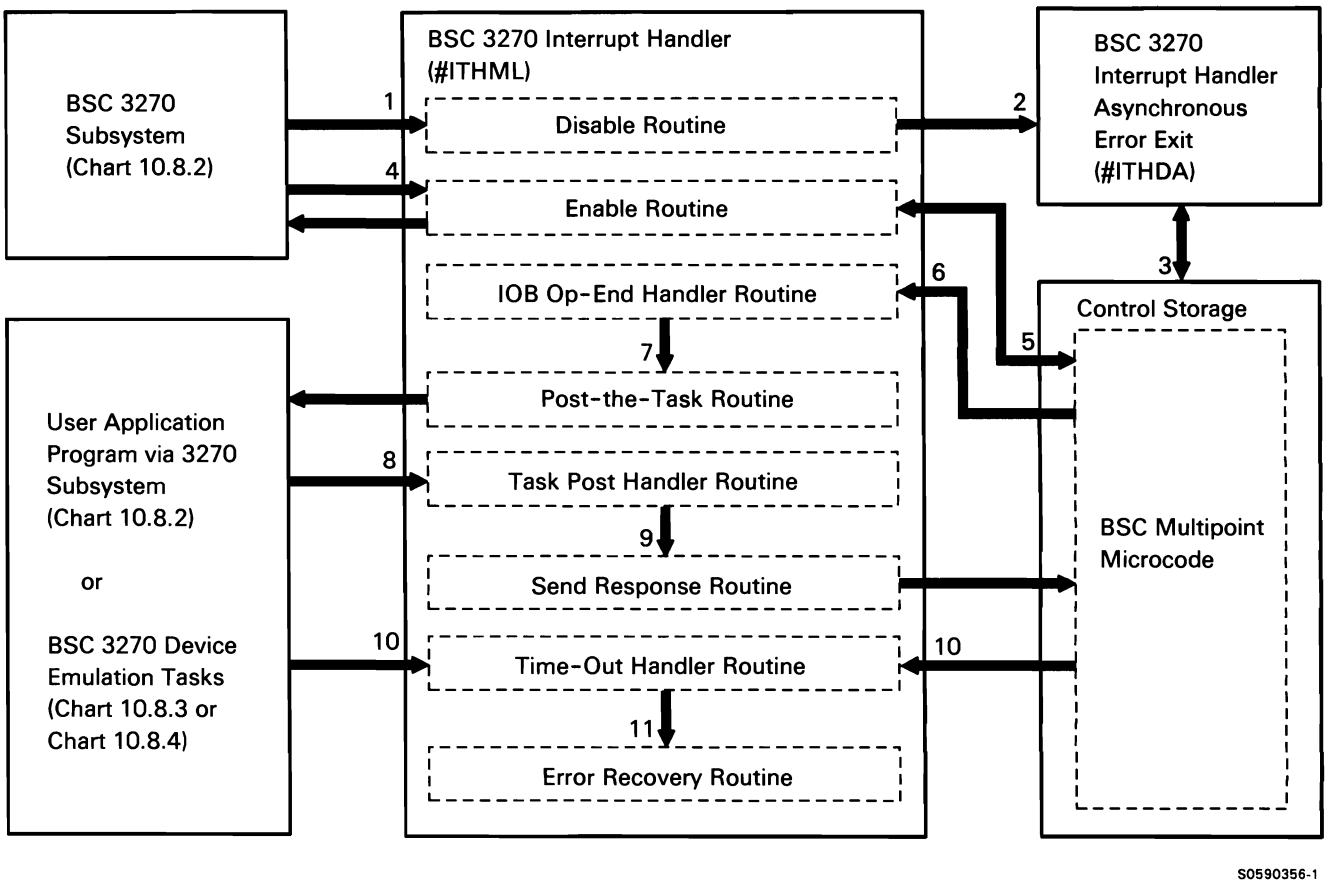


Chart 10.8.1 BSC 3270 Interrupt Handler Control Flow

BSC 3270 (Program Interface) Subsystem

The System/36 BSC 3270 subsystem performs enable and disable for all BSC 3270 support. It also provides System/36 user application programs with a BSC session interface to a host system using 3270 protocols. The BSC 3270 subsystem isolates the System/36 application program from 3270 device control information dependencies. If desired, the application programmer can still code with 3270 device control information.

All mainline BSC 3270 subsystem functions (operation codes) are processed by the BSC 3270 interrupt handler. Requests for these functions, and all status and control information relative to the requests, are contained in the realtime interface table (RIT) control block in SQS. The RIT is the control block interface between the BSC 3270 subsystem and the BSC 3270 interrupt handler. For more information on the RIT, refer to the *Data Areas* manual.

BSC 3270 Subsystem Region Map

The following table shows the locations of key data areas involved in the BSC 3270 subsystem:

Dump Address (in Hex)	Contents
0800	Nucleus
1000	SSP-ICF control
2800	BSC 3270 subsystem (#ITSPI)
4800	Task work space for line buffer
5800	Task work space for program interface put or get buffer

The following BSC 3270 subsystem processes are shown in Chart 10.8.2:

1 Enable subsystem by routing to BSC interrupt handler to complete enable.

2 Handle user operation requests passed in the SUB by setting up an RIT accordingly.

3 Process interrupt handler operation-ends (op ends).

4 Disable subsystem.

5 Handle line errors and interrupt handler abnormal termination.

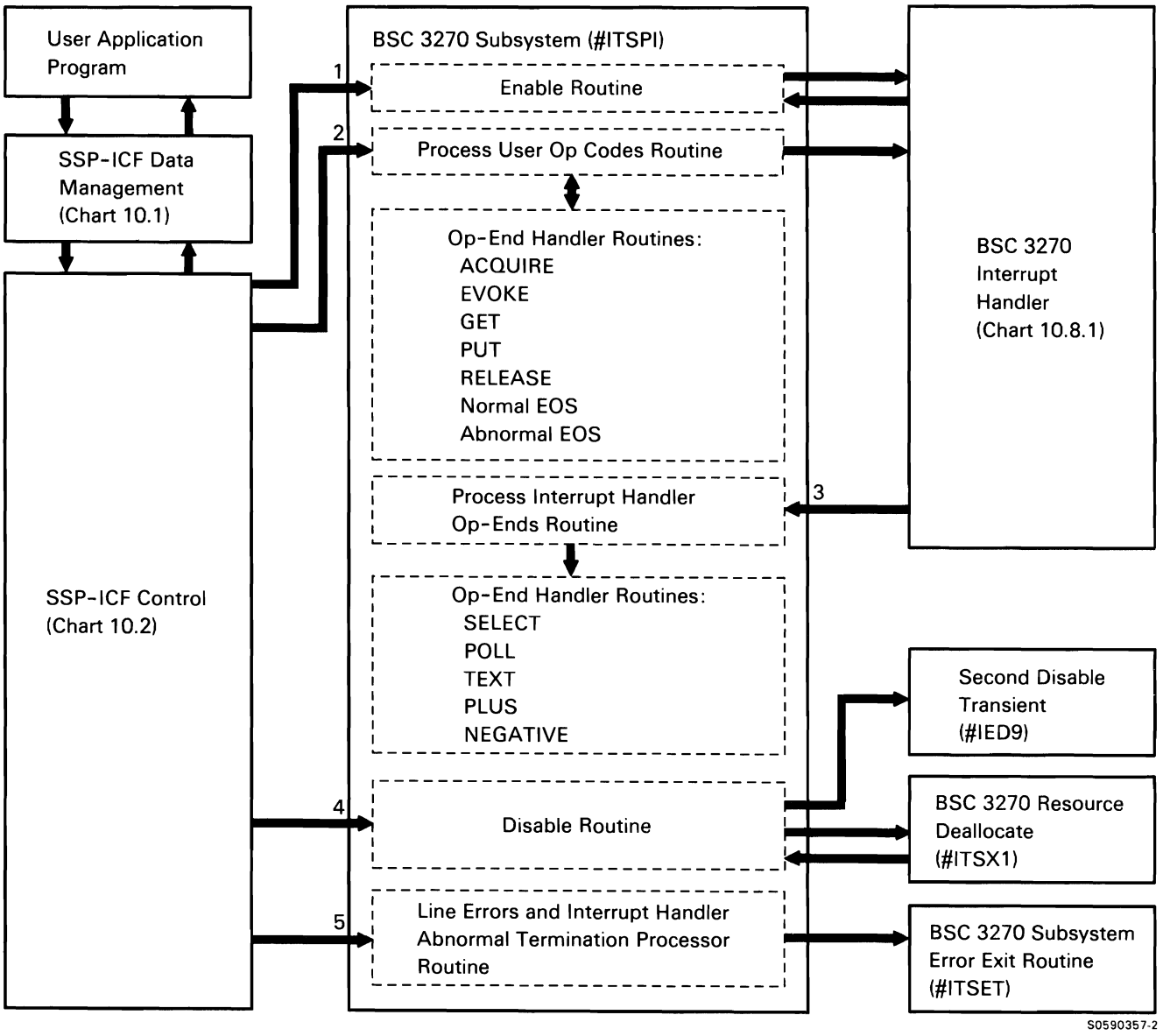


Chart 10.8.2 BSC 3270 Subsystem Control Flow

BSC 3270 Display Emulation

BSC 3270 display emulation provides the System/36 user with terminal support for a BSC 3270 network.

The following BSC 3270 display emulation processes are shown in Chart 10.8.3:

1 Process the EM3270 procedure command.

2 Recover when error detected by the procedure processor.

3 Initialize a session between a System/36 display and the host system by initializing the RIT, verifying that the line is enabled, and posting the interrupt handler with device ready.

4 Begin active session mainline processing after initialization.

5 During active session, handle command keys, commands, and data from a work station keyboard by converting data in #EMAD's 5250 buffer, placing the 3270-equivalent data in line buffer, and posting the interrupt handler.

6 During active session, handle commands and data from the host system by converting 3270 data in the line buffer, placing the 5250-equivalent data in #EMAD's 5250 buffer, and posting work station data management.

7 Perform normal termination of a session.

8 Perform abnormal termination of a session.

Note: Because #EMAX is reentrant, there will be only one copy of it for all active sessions.

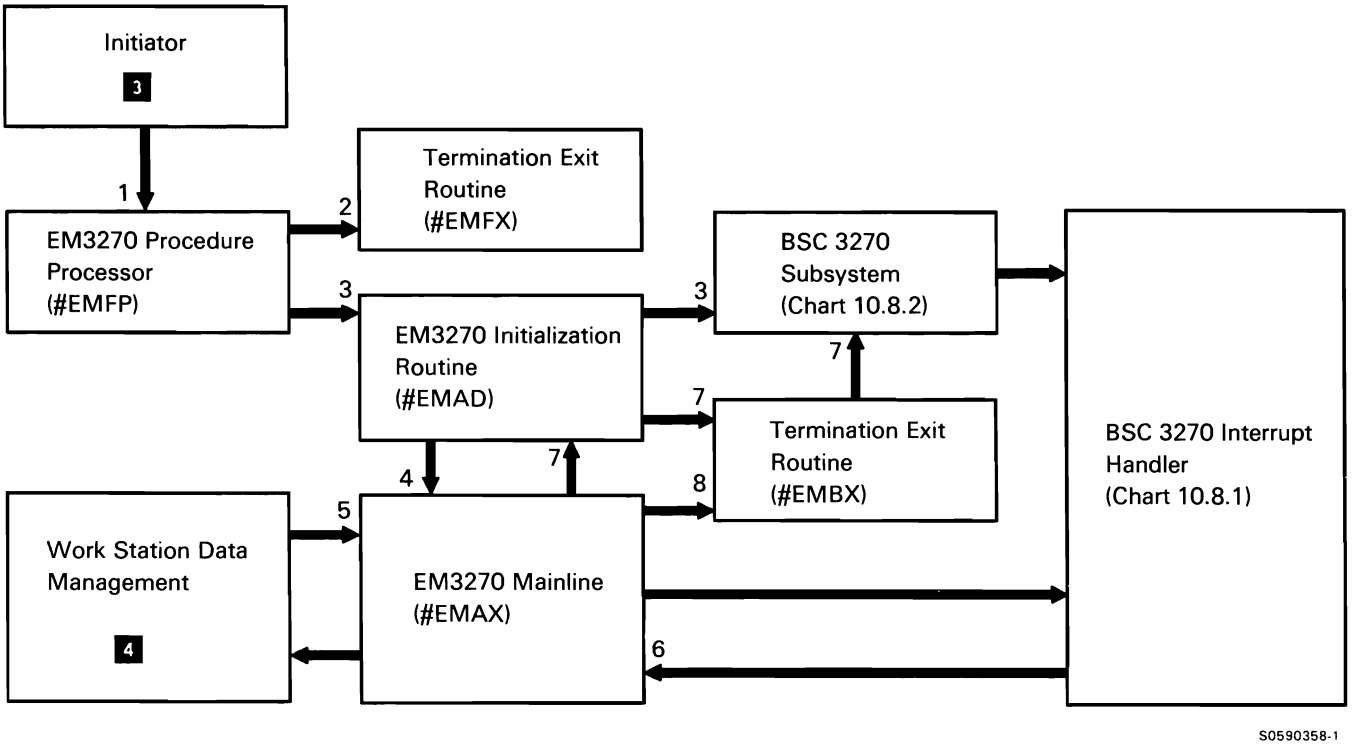


Chart 10.8.3 BSC 3270 Display Emulation Control Flow

BSC 3270 Printer Emulation

BSC 3270 printer emulation provides the System/36 user with terminal support for a BSC 3270 network.

The following BSC 3270 printer emulation processes are shown in Chart 10.8.4:

1 Process the EM3270 procedure command.

2 Recover when error detected by the procedure processor.

3 Initialize a session between a System/36 printer and the host system by initializing the RIT, verifying that the line is enabled, and posting the interrupt handler with device ready.

4 During active session, handle commands and data from the host system by converting 3270 data in the line buffer, placing the 5250-equivalent data in #EM9D's 5250 buffer, and posting printer data management.

5 Perform normal and abnormal termination of a 3270 device emulation session.

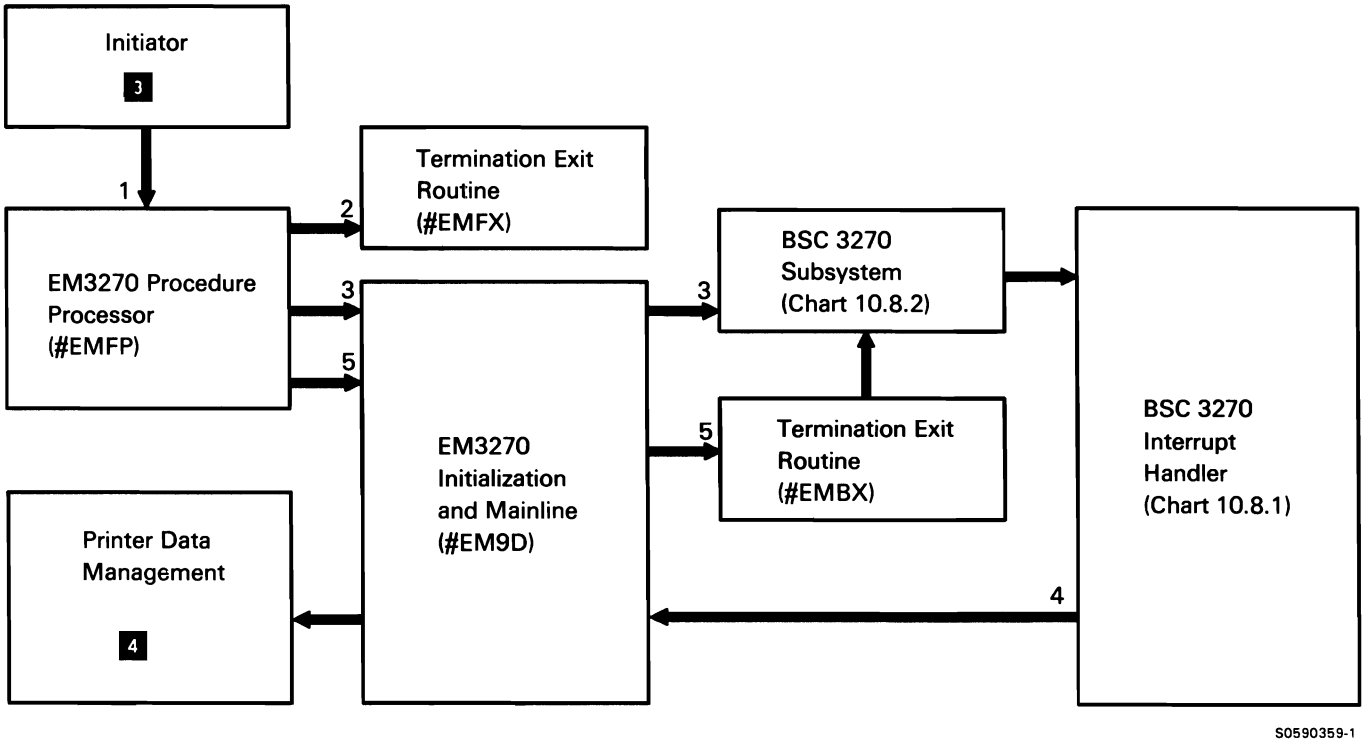


Chart 10.8.4 BSC 3270 Printer Emulation Control Flow

SNA 3270 Subsystem

The System/36 SNA 3270 subsystem performs enable, disable, initiate, and terminate processing for the SNA 3270 support.

SNA Subsystem Region Map

The following table shows the locations of key data areas involved in the SNA 3270 subsystem:

Dump Address (in Hex)	Contents
0000	Nucleus
1000	When active, termination exit routine (#IZAET); otherwise, SSP-ICF control
2800	SSP-ICF transient area and disable routine (#IZDBL)
3000	SNA 3270 subsystem (#IZMNL)

The following SNA 3270 subsystem functions are shown in Chart 10.8.5:

1 Process the ENABLE command from SSP-ICF enable ($IENBL) by allocating an RIT and 4K-byte buffer for each SNA 3270 device configured during the CNFIGICF procedure execution.

2 Set enable in the LOC and the SCT (subsystem control table).

3 At logon, verify that the bind sent from the host system is valid by checking for 3270 parameters in the bind request.

4 During active session, provide interface to the host system when data is transmitted or received.

5 Process the ES3270 procedure by doing the following:

- Set up the 3270 display emulation task (#ESAX).
- Build an SNUB, LUSCB, and RIT and assign a 4K-byte buffer for the session.
- Post initiation complete to the host system via C/SNA and post initiation complete to the emulation task.
- Return to SSP-ICF control.

6 Process the EP3270 procedure by doing the following:

- Set up the personal computer support task.
- Build an SNUB, LUSCB, and RIT, and assign a 4K-byte buffer for the session.
- Post initiation complete to the host system via C/SNA, and post initiation complete to the personal computer support task.

7 When device is terminated, process data received from the host system by sending negative responses via sense codes in the SNUB and LUSCB.

8 Route for DISABLE command from SSP-ICF disable ($IEDS).

9 Process DISABLE command by freeing the SCT and LOC and their associated SNUBs and buffers.

10 Process the ES3270 procedure termination post by doing the following:

- Terminate the 3270 display emulation task (#ESAX).
- Free session SNUB, LUSCB, and buffer.
- Post termination complete to the host system via C/SNA and post termination complete to the emulation task.
- Return to SSP-ICF control.

11 Handle all abnormal SNA posts or completions; call for dump if necessary. On serious errors, terminate the SNA 3270 session and disable the SNA 3270 subsystem.

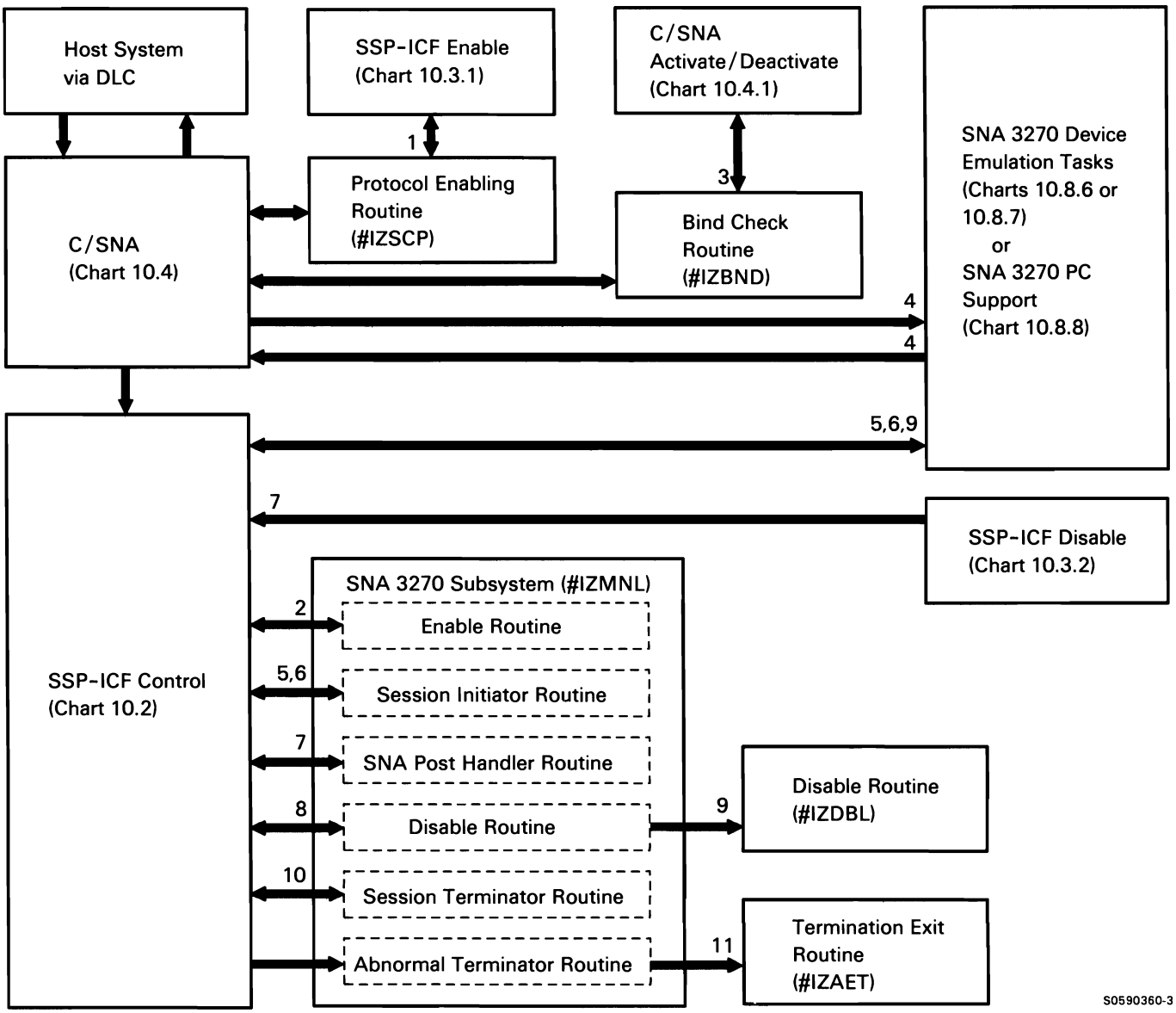


Chart 10.8.5 SNA 3270 Subsystem Control Flow

SNA 3270 Display Emulation

SNA 3270 display emulation provides the System/36 user with display terminal support in a SNA 3270 network.

The following SNA 3270 display emulation processes are shown in Chart 10.8.6:

1 Process the ES3270 procedure command.

2 Recover when error detected by the procedure processor.

3 Initialize a session between a System/36 display and the host system by initializing the RIT, verifying that the line is enabled, and posting C/SNA with initiation complete.

4 Begin active session mainline processing after initialization and receipt of the bind request from the host system.

5 During active session, handle command keys, commands, and data from a work station keyboard by converting data in #ESAX's 5250 buffer, placing the 3270-equivalent data in the display buffer, and posting work station data management.

6 During active session, handle commands and data from the host system by converting 3270 data in the session buffer, placing the 5250-equivalent data in the 5250 buffer, and posting work station data management.

7 Process SNA protocol.

8 Perform normal termination of a session.

9 Perform abnormal termination of a session.

Note: Because #ESAX is reentrant, there will be only one copy of it for all active sessions.

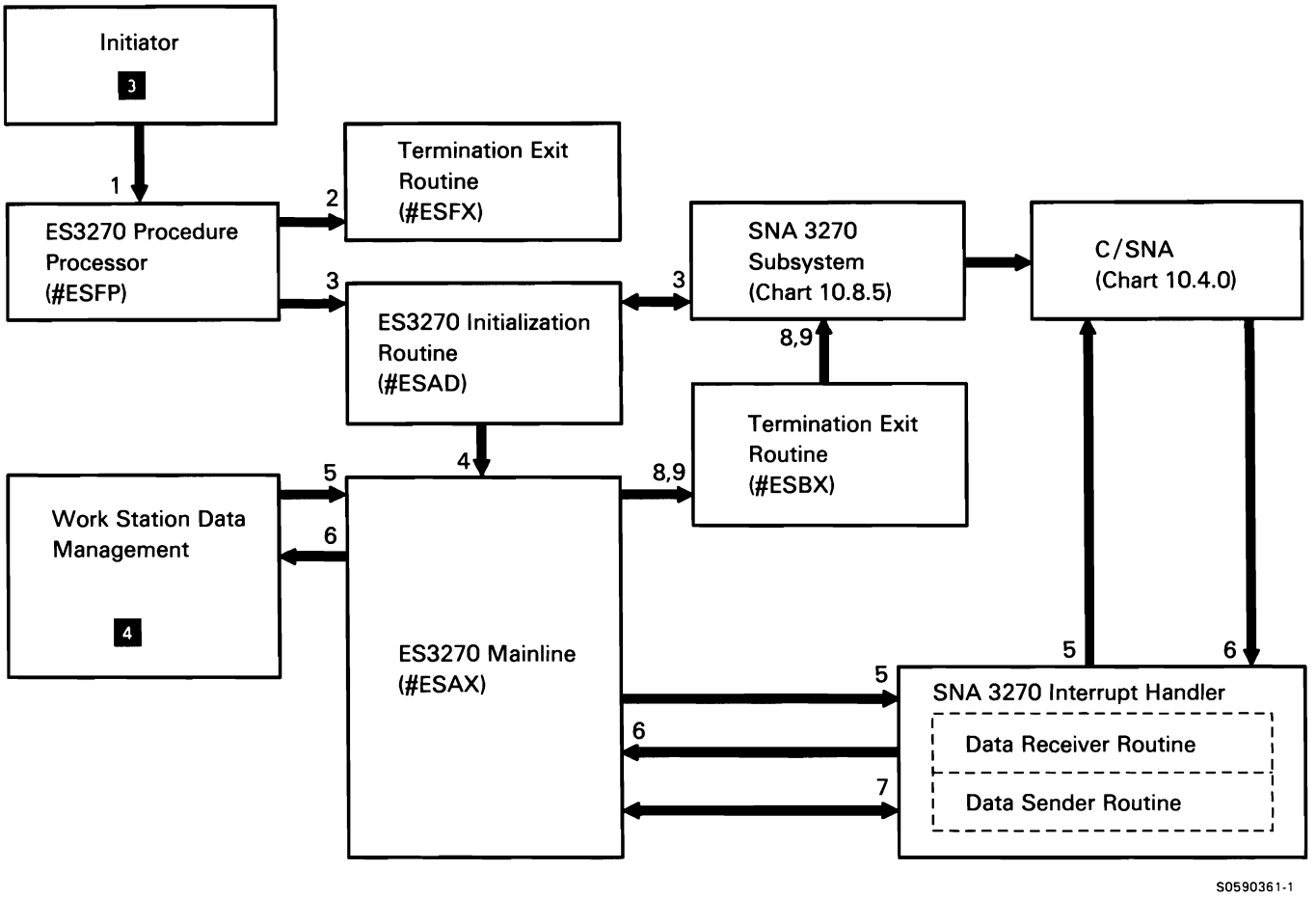


Chart 10.8.6 SNA 3270 Display Emulation Control Flow

SNA 3270 Printer Emulation

SNA 3270 printer emulation provides the System/36 user with printer support in a SNA 3270 network.

The following SNA 3270 printer emulation processes are shown in Chart 10.8.7:

1 Process the ES3270 procedure command.

2 Recover when error detected by the procedure processor.

3 Initialize a session between a System/36 printer and the host system by initializing the RIT, verifying that the line is enabled, and posting C/SNA with initiation complete.

4 During active session, handle commands and data from the host system by converting 3270 data in the line buffer, placing the 5250-equivalent data in the 5250 buffer, and posting printer data management.

5 Perform SNA 3270 protocol processing for SNA character string (SCS) or data stream compatibility (DSC) data streams.

6 Control spooling via continue option.

7 Perform normal and abnormal termination of a session.

8 Form logical printer records from the SCS data stream.

Note: Based on the BIND command received, either #ES9D or #ESSD will be loaded after logon.

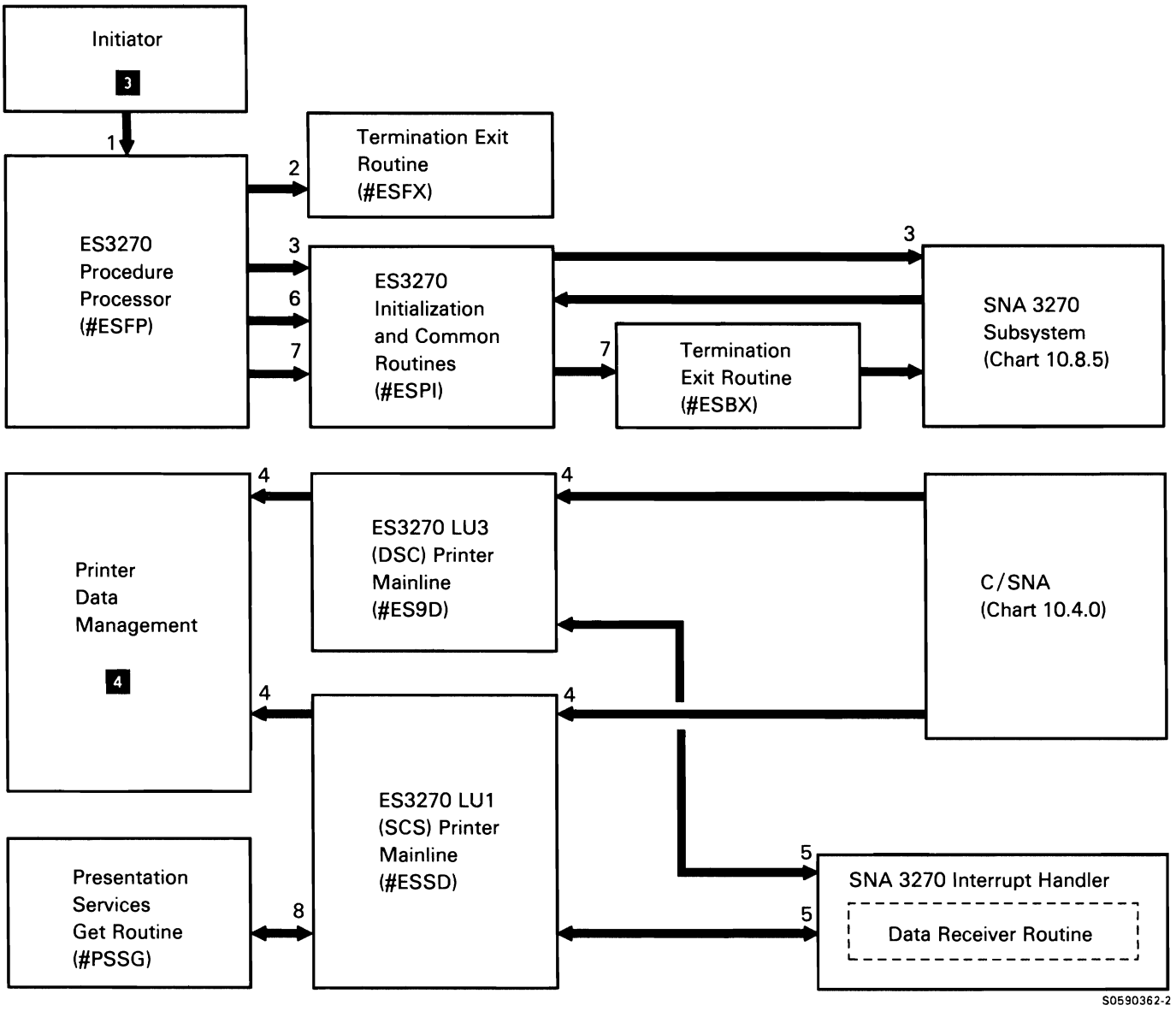


Chart 10.8.7 SNA 3270 Printer Emulation Control Flow

SNA 3270 Personal Computer Support

SNA 3270 personal computer support provides the interface between the System/36 SNA 3270 subsystem and the EP3278 feature on the personal computer.

The following SNA 3270 personal computer support processes are shown in Chart 10.8.8:

1 Process the EP3270 procedure command.

2 Recover when error detected by the procedure processor.

3 Initialize a session between a System/36 display and the host system by initializing the RIT, verifying that the line is enabled, verifying that EP3278 is active on the personal computer, and posting C/SNA when the initiation is complete.

4 Begin active session mainline processing after initialization and receipt of the bind request from the host system.

5 During active sessions, pass all personal computer operations and 3270 data to the SNA 3270 interrupt handler.

6 During active sessions, pass all SNA 3270 interrupt handler operations and 3270 data to the personal computer.

7 Process SNA protocol.

8 Perform normal termination of a session.

9 Perform abnormal termination of a session.

Note: Because #ESPC is reentrant, there will be only one copy of it for all active sessions.

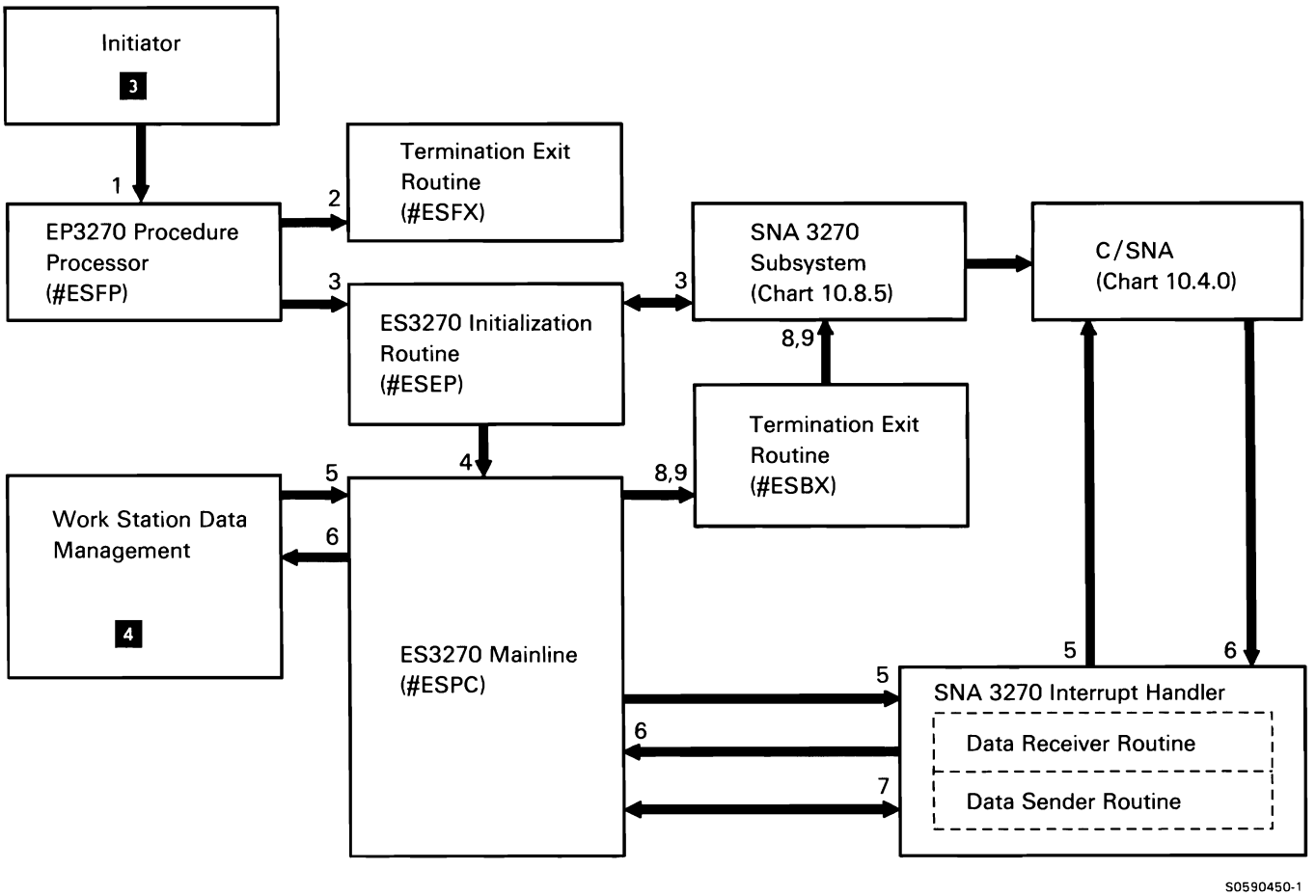


Chart 10.8.8 SNA 3270 Personal Computer Support Control Flow

This page is intentionally left blank.

COMMUNICATIONS AND SYSTEMS MANAGEMENT (C & SM)

System/36 communications and systems management (C & SM) support consists of the following:

- Change management
- Problem management
- Remote management

Note: C & SM change management may be referred to elsewhere as distributed systems node executive (DSNX); similarly, C & SM problem management may be referred to elsewhere as alert support, and C & SM remote management may be referred to as distributed host command facility (DHCF).

Change Management

The communications and systems management (C & SM) change management feature allows users to distribute and manage files, libraries, and members in a network of System/36s attached to a System/370-type system.

C & SM change management communicates with the host system via the SSP-ICF SNUF subsystem, C/SNA, and SDLC. All files passed from the host system are in the form of data sets. The data set name determines data set type (file, library, load member, source member, procedure member, or subroutine member) that System/36 is to process.

System/36 change management executes the following requests from the host system:

- Add data set
- Replace data set
- Send data set
- Delete data set
- Initiate received procedure
- Inform operator

The following C & SM change management functions are shown in Chart 10.9.1:

1 Process SNUF subsystem procedure start request.

2 Process initial accept-input operation.

3 Perform mainline routing for change management.

4 Verify user ID and password.

5 Process received data.

6 Transmit requested data.

7 For operations involving files, allocate, open, close, and deallocate files as required.

8 For operations involving library members, find library, find member, protect member, delete member, and perform puts and gets as required.

9 Read or write files as required.

10 Delete files as required.

11 Issue messages received as a result of an inform-operator request from the host system and put System/36 change management messages to the history file.

12 Evoke procedures sent by change management with an initiate-function request.

13 Notify change management of completion of a procedure started by an initiate-function request.

14 Perform change management processing required when change management is canceled or if unrecoverable error occurs.

15 Terminate this task.

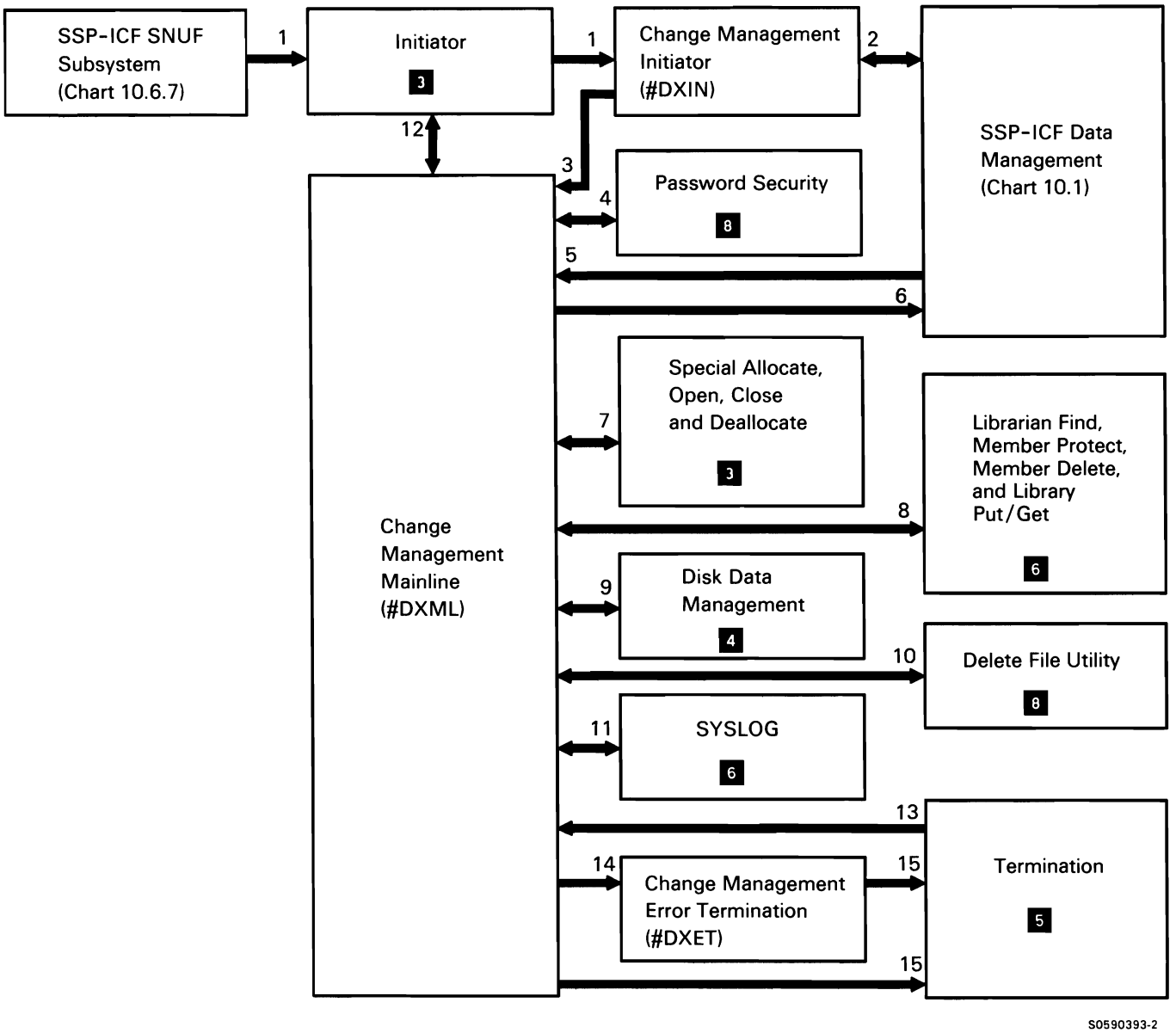


Chart 10.9.1 C & SM Change Management Control Flow

Problem Management

The communications and systems management (C & SM) problem management feature allows users to set up the System/36 to automatically alert the host system about the unavailable status of system hardware. It also provides a mechanism for signaling that a loss of availability is impending via notification alerts. (The host system is a System/370-type system running the Network Communications Control Facility (NCCF) and the Network Problem Determination Application (NPDA) programs. C & SM problem management consists of two alert task modules, #CZAT and #CZAE, and an alert utility module, $CZUT.

##MSG1 contains approximately 200 hardware error, intervention required, thermal check, and unavailable messages that are shipped from IBM with their alert generation indicators on. When the alert task is active and SYSLOG or the console SYSLOG message builder transient (#SVWER) detect that the alert generation indicator is on in a message they are issuing, the alert task is posted to send an alert to the host in the form of a network management vector transport (NMVT RU). The NMVT RU is the architected request/response unit that carries alert information to the host.

The alert utility, described later in this topic, allows the user to customize the System/36 portion of the network's problem management system by modifying the alert generation indicator of any alert-capable message contained in ##MSG1 and provides the interface for sending a notification alert.

The C & SM problem management alert task is attached by the APPC subsystem protocol module when an alert location is enabled. The alert task communicates with the host system via its special interface (bypassing SSP-ICF data management) to the APPC subsystem.

The following C & SM problem management functions are shown in Chart 10.9.2.1:

1 Process ZO or NO parameter list containing the error message that defines alert.

2 Wait for posts from subsystem or disable request; route (based on XR1 value) for processing by appropriate routine.

3 Build ZA parameter list (containing the alert address) and post it to APPC for transmission on the PU-SSCP session. Return to mainline routine.

4 Check return code in ZA parameter list on post back (alert sent) from APPC subsystem. If line is inactive, requeue alert ZO or NO parameter list, then return to mainline routine.

5 Process remote node active (RNA) post from APPC subsystem, then return to mainline routine.

6 Process for termination. If no other LOCs are still enabled, terminate; otherwise, return to mainline routine.

7 Perform normal termination or route for alert task abnormal termination.

8 Perform alert task abnormal termination:

- Wait for completion of pending requests.
- Post APPC to disable all enabled C & SM LOCs.
- Issue message to local system console via SYSLOG.
- Quiesce all pending events and free SQS.
- Return to system termination.

9 Process LOC disable posts from abnormal termination.

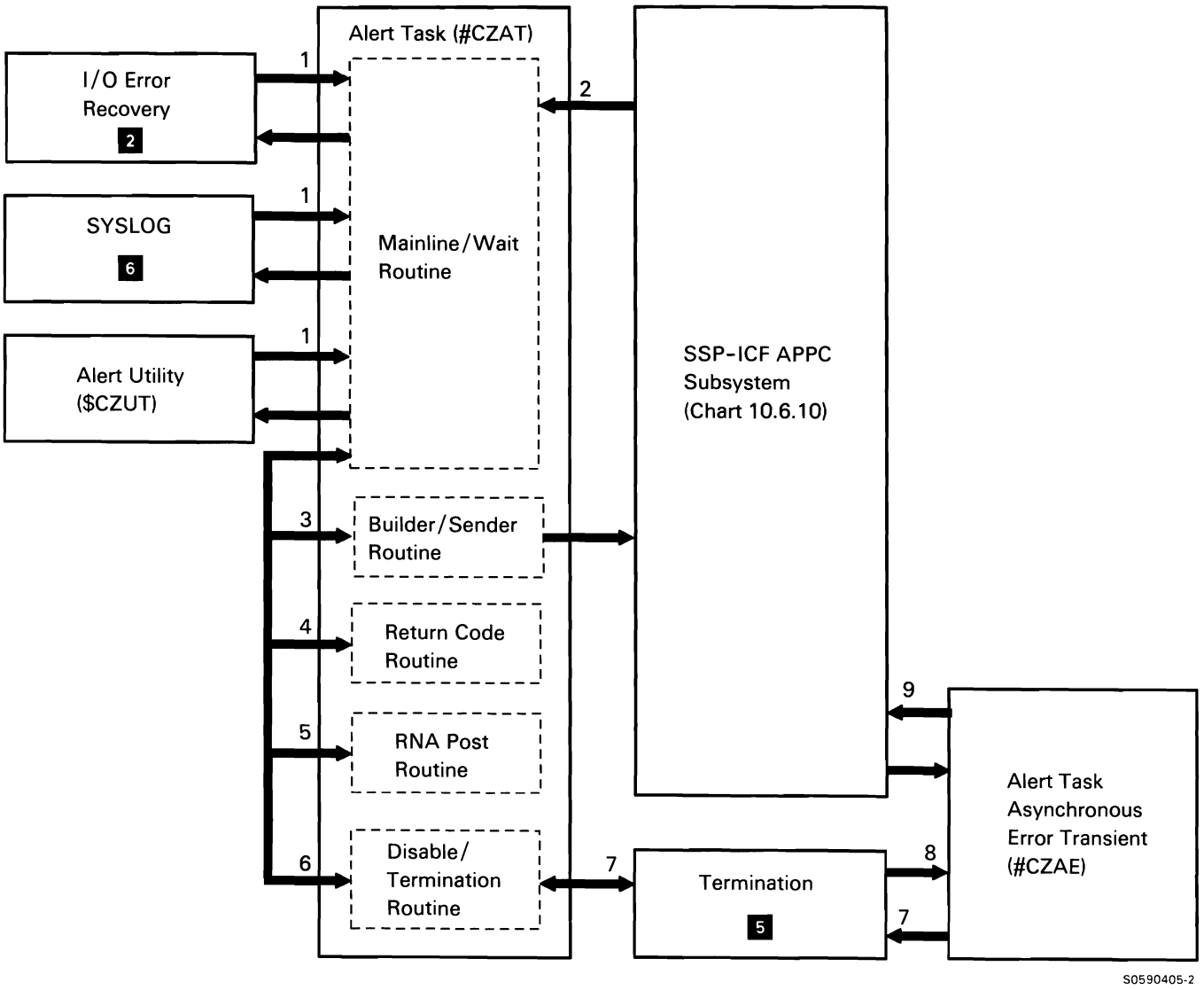


Chart 10.9.2.1 C & SM Problem Management Control Flow

The alert utility allows a user with system operator or higher security classification to access the alert generation indicators in the ##MSG1 system message member or in an alert source member for any of the following purposes:

- View/print alert generation indicator settings in ##MSG1 message member.
- View/print alert generation indicator settings in alert source member.
- Reset alert generation indicator settings in ##MSG1 to those originally set in alert MIC table (AMT) by IBM.
- Apply an alert source member to ##MSG1 to modify the alert generation indicator settings.
- Create, update, copy, remove, or rename an alert source member.

The alert utility also provides the interface for sending a notification alert which allows you to generate your own message and send it to the host system that is receiving alerts.

The following C & SM alert utility functions are shown in Chart 10.9.2.2:

1 Do the following, routing control where necessary:
   - Initialize common area.
   - If security is active, check for proper security classification (system operator or above).
   - Allocate and open work station DTF.
   - Prompt user for function to be performed.
   - Perform requested function.

2 Get message text and alert generation indicator.

3 Perform any printing associated with user requests.

4 Process operator prompts and responses, as required.

5 Reset/modify the alert generation indicator for specified messages in system message member.

6 Process alert source member:
   - Verify existence of specified library and member.
   - Lock member against simultaneous use.
   - Read/write member as required.

7 Issue any required error messages.

8 Send notification alert.

9 Terminate this job step.

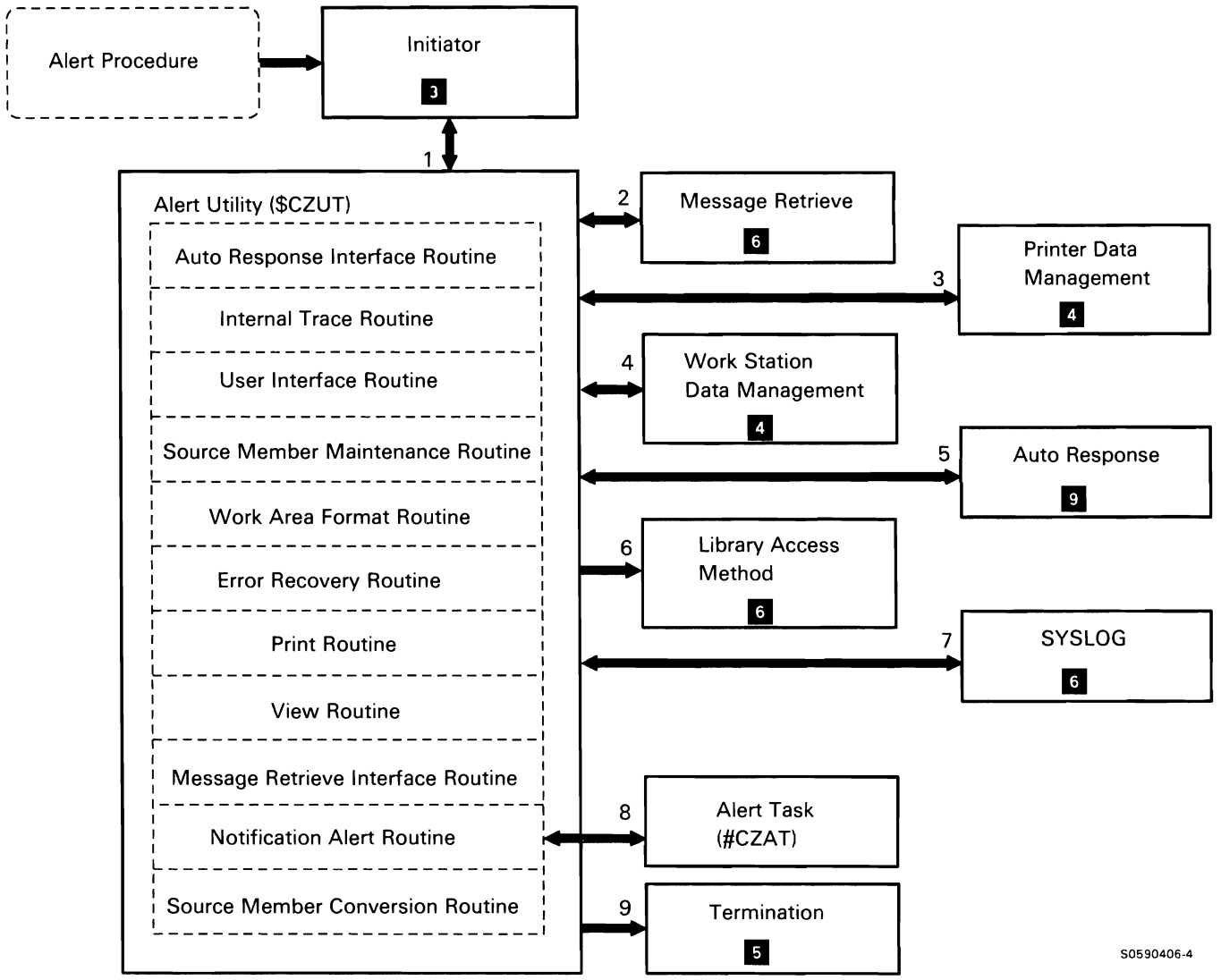


Chart 10.9.2.2 Alert Utility Control Flow

Remote Management

The C & SM remote management (DHCF) feature allows 3270 users on a host system (through the HCF application) to access the System/36 as a remote work station.

C & SM remote management communicates with the host system via the SSP-ICF SNUF subsystem, C/SNA, and SDLC. Data passed between System/36 and the host system is enveloped within the frame. The 7-byte envelope describes the type of operation for HCF and DHCF, and carries the device RH, some relevant TH bits, and the sequence numbers.

While the 3270 is attached to a DHCF session, all normal commands, procedures, and applications can be accessed on the System/36 as if the 3270 device were a 5250 remote device.

The following C & SM remote management functions are shown in Chart 10.9.3:

1 Process SNUF subsystem procedure start request.

2 Process initial accept input function.

3 Perform session bring-up and device bring-up.

4 Process received data. Data stream translation 3270-5250.

5 Process data to be transmitted. Data stream translation 5250-3270.

6 Interface to work station data management for outbound data.

7 Interface to work station data management for inbound data.

8 Save and restore screen data to/from disk.

9 Interface to $NUPD for error handling.

10 Issue messages.

11 Perform processing required when session is canceled by either System/36 or host.

12 Perform termination when the last session is canceled.

13 Perform keyboard map functions (display, change) initiated from keys procedure.

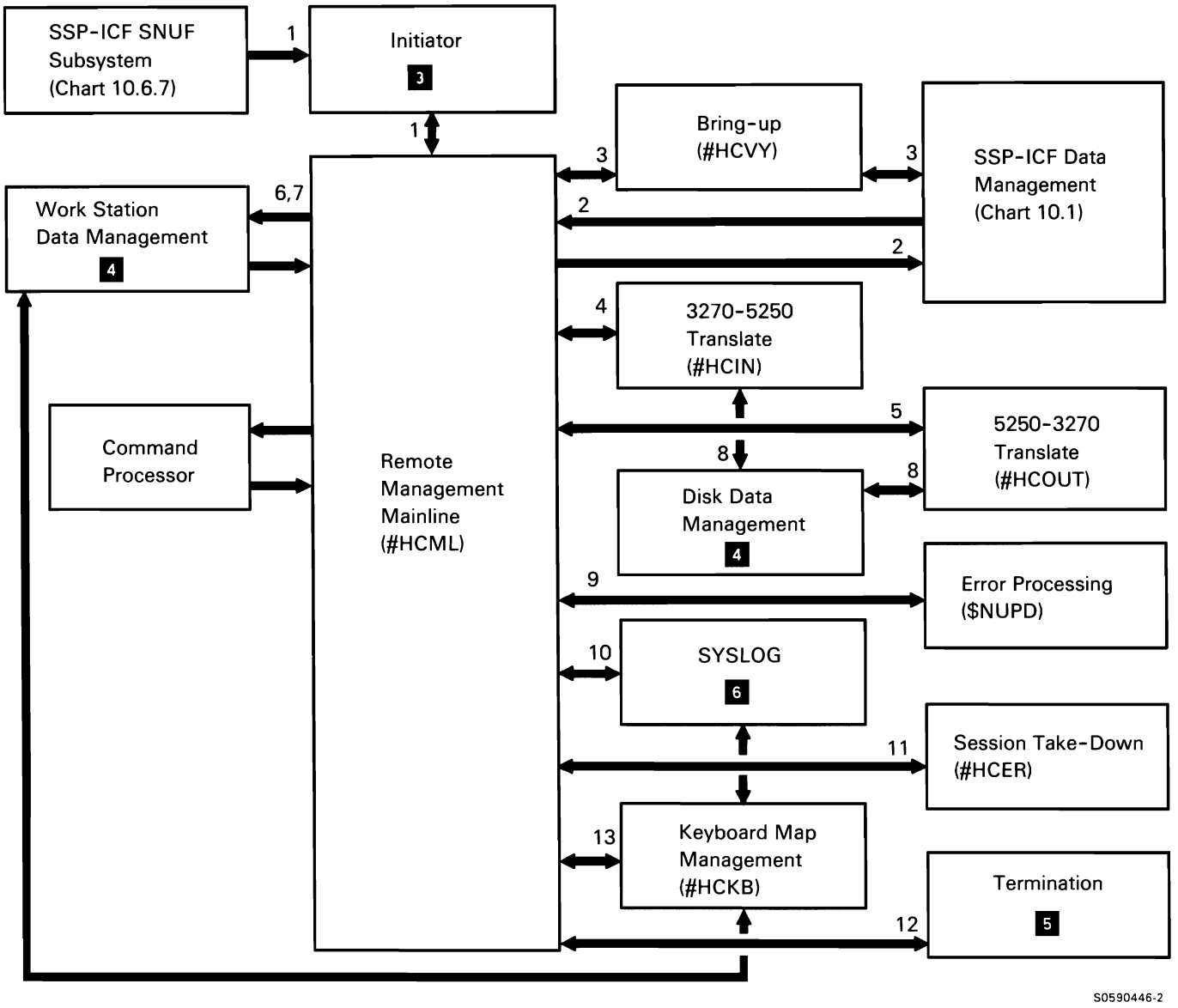


Chart 10.9.3 Remote Management Control Flow

PERSONAL COMPUTER SUPPORT SUBSYSTEM

The personal computer support subsystem provides server functions to personal computers that are attached to a System/36 via a 5250 emulation program or via the IBM Token-Ring Network.

The following personal computer support subsystem processes for a 5250 environment are shown in Chart 10.10.1:

1 Perform the SSP-ICF control program start or post user task subroutine.

2 Perform the SSP-ICF control wait subroutine. SSP-ICF control branches to the defined entry points in the router to fall out of the wait.

3 Post SSP-ICF control with data received.

4 Transfer termination to the router termination entry point.

5 Perform one of the following router subroutines:

- Put-display-buffer
- Get-display-buffer
- Wait
- Terminate-conversation

6 Perform one of the following operations:

- Display-buffer-sent
- Display-buffer-received
- Display-buffer-sent-received
- Display-buffer-received-not-sent
- SSP-ICF-post-received

7 Issue operations.

8 Read or write a virtual disk.

9 Issue printer operations.

10 Issue query operations.

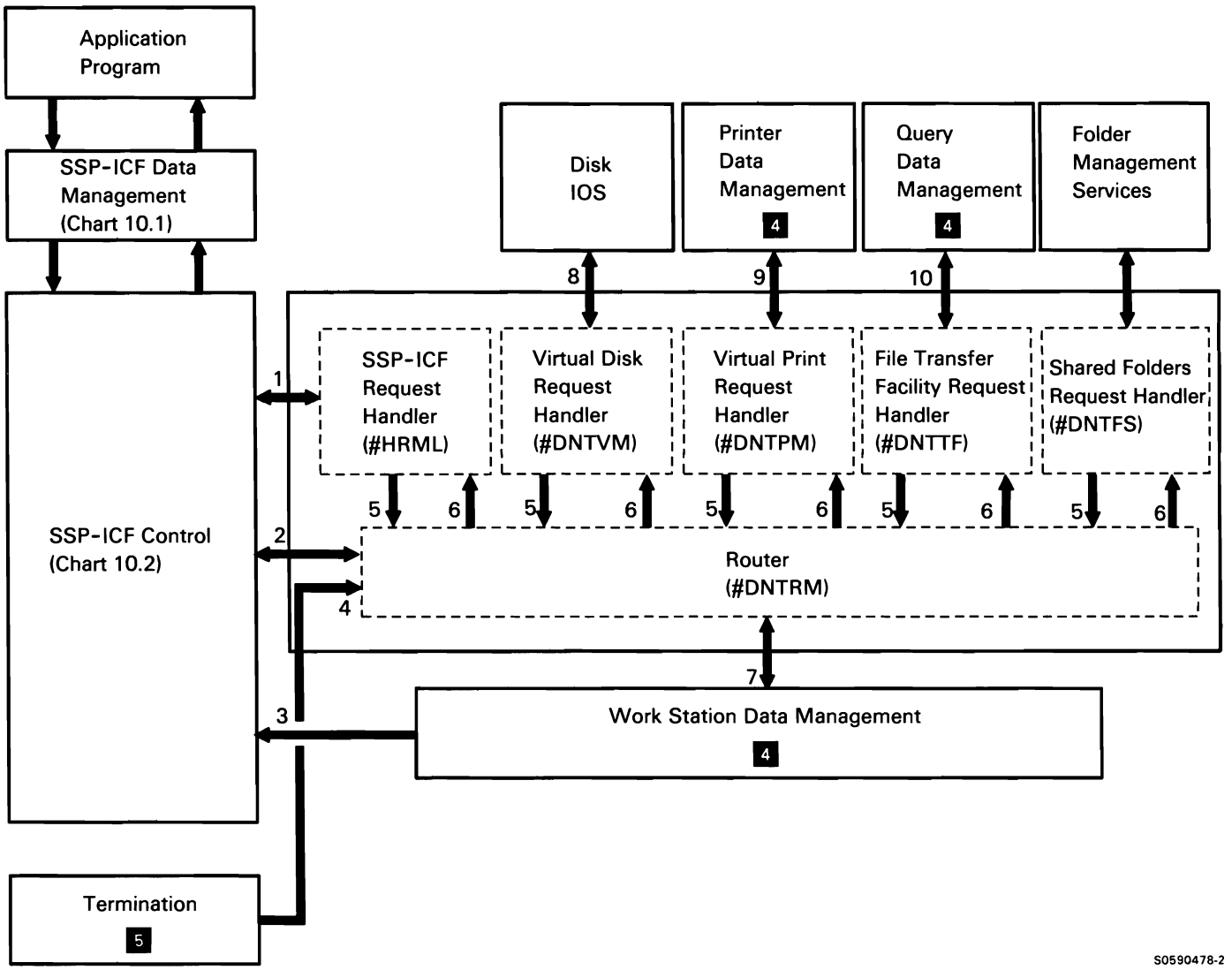


Chart 10.10.1 Personal Computer Support Subsystem Control Flow for a 5250 Environment

The following personal computer support subsystem processes for an environment of the LAN are shown in Chart 10.10.2:

1 Perform the SSP-ICF control program or post user task subroutine.

2 Perform the SSP-ICF control wait subroutine. SSP-ICF control branches to the defined entry points in the router to fall out of the wait.

3 Perform one of the following router subroutines:

- Put-display-buffer
- Get-display-buffer
- Wait
- Terminate-conversation

4 Perform one of the following operations:

- Display-buffer-sent
- Display-buffer-received
- Display-buffer-sent-received
- Display-buffer-received-not-sent
- SSP-ICF-post-received

5 Issue operations.

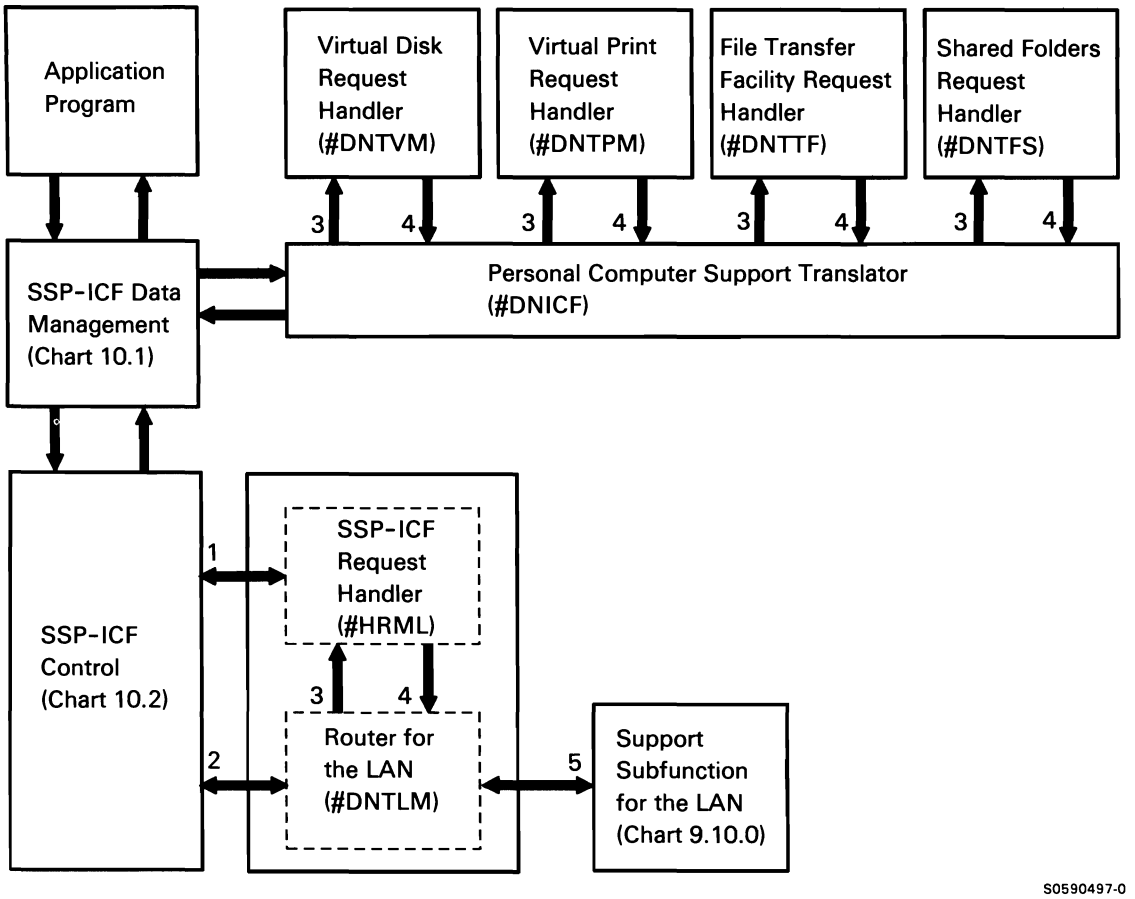


Chart 10.10.2 Personal Computer Support Subsystem Control Flow for an Environment of the LAN

Section 3: SSP Programming Interfaces

This section describes interfaces between the SSP and other IBM-supplied programming. The interface descriptions are intended for use in isolating a suspected control storage, programming language, or Utilities Program Product problem from an SSP problem, for the purposes of writing an accurate APAR.

For information on the control blocks and data areas used in this section, refer to the *System Data Areas* manual.

The following interfaces are described:

- SSP to control storage
- SSP intermodule
- SSP to overlay linkage editor
- SSP to sort utility
- SSP to Ideographic Generator/Sort Program Product
- SSP to language program products
- SSP to Utilities Program Product
- SSP to OFFICE/36
- SSP to Development Support Utility

Reverse numbers in the text in this section (3, for example) refer to major function descriptions in *Section 2: System Function/Control Flow*.

SSP to Control Storage Interfaces

The following SSP to control storage main interfaces are described:

- Control storage executable code
  - SVC instruction
  - XFER instruction

- Control storage data areas (located in main storage)
  - System queue header area
  - System communications area

SSP TO CONTROL STORAGE EXECUTABLE CODE

SVC (Supervisor Call) Instructions

The SVC instruction (op code hex F4) is the basic SSP interface to control storage and is used by all executable code in main storage. Main storage programs, either at the machine language or assembler level, use the SVC instruction to request control storage functions. When an SVC instruction is encountered in the instruction stream, main storage processing is halted, main storage registers are saved by hardware, and the control storage entry routine (operating at a dedicated interrupt level) is started and determines what operation has been requested. The operation is determined by the Q-byte and R-byte of the SVC instruction and parameters that are passed with the instruction. Based on the SVC request operation, the entry routine determines if the operation is valid and if it is to be executed immediately or processed in overlapped or delayed mode.

If the SVC instruction is to be executed immediately, the main storage processor remains stopped, and the entry routine gives control to the immediate SVC routine to perform the requested function. After the function has been performed, the immediate SVC routine returns control to the entry routine which restarts the main storage processor. The main storage processor resumes processing at the next instruction either of the current task or, if a task switch has occurred, of a different task.

If the SVC instruction is to be processed in overlapped or delayed mode, the entry routine determines if another task can be run; if another task is ready, the entry routine requests the SVC processor, then starts the main storage processor which causes the MSP hardware to load the registers of the main storage processor for a different task, if possible, and exits the entry routine. Overlapped or delayed functions receive control on the main program level and are used either to control data flow between main or control storage and the I/O devices or to maintain control information.

For more information on this interface, refer to the *Control Storage Service Information* manual.

XFER (Transfer) Instruction

The XFER instruction (op code hex F5) is a request for specialized control storage functions. When the main storage processor attempts to execute the transfer instruction, it is stopped; it is not started again for the current task until the specialized control storage processing is completed (it can be started for the execution of a different task). The control storage entry routine requests the extended control storage (ECS) processor to execute the transfer instruction.

The extended control storage (ECS) processor, operating on the control storage processor main interrupt level, receives control and executes the instruction on the main program level.

After determining that the Q-byte of the transfer instruction is valid, the ECS processor uses the Q-byte to determine whether the instruction is from a BASIC or a FORTRAN program and passes control to either the scientific instruction set interpreter or the BASIC instruction set emulator.

For more information on this interface, refer to the *Control Storage Service Information* manual.

SSP TO CONTROL STORAGE DATA AREAS

System Queue Header Area

The system queue header area is located at hex 0B00 of the main storage fixed nucleus and contains 1024 bytes. Each queue header contains a 3-byte address of the first associated element of a chain. Each chain represents work to be done by various system components or resources used by the components.

The system queue header area serves as the main work-unit interface between the main storage and control storage processors. Work unit elements (control blocks) required for main storage task execution are chained to the various queues by main and control storage. When control storage executes a work unit requested in a queue element, it removes that element from the chain.

For more information on this interface, refer to *Queue Header Area* in the *System Data Areas* manual.

System Communications Area

The system communications area is a 768-byte common area that is used by various components of the SSP to communicate with each other. It contains configuration data, addresses (disk and main storage) of system areas as well as other system information.

The system communications area is located in the fixed nucleus starting at hex 0800 of main storage.

For more information on this interface, refer to *System Communications Area* in the *System Data Areas* manual.

SSP Intermodule Interfaces

The following SSP intermodule interfaces are described in the following text:

- System interlocks
- Program types
- Program attributes
- Program control
- Subtasking

SYSTEM INTERLOCKS

The System/36 multitask environment requires system interlocks to prevent the simultaneous use of certain system resources by two or more tasks. The *initiator* 3 is the primary component ensuring that resources are properly interlocked. It obtains these resources by setting the interlocks in the task blocks or queue headers, usually via the resource enqueue and resource dequeue SVC instructions. For locations of the following interlocks, see *Task Block* and *Queue Header Area* in the *System Data Areas* manual:

- System trace
- Console SYSLOG
- Reject file
- Dedication
- Scheduler
- VTOC (includes main storage format 1's)
- Format 5
- Procedure name
- History file
- Multiple SNA line
- Spool file
- Messages file
- Security utility
- User ID file
- Resource utility
- Resource security file
- CONFIG record
- MRT termination
- File modify
- Address compare dump
- SCT queue
- MSRJE
- TUB chain interlock (in SCA)
- Procedure parameter save area
- Job control block
- ERP interlock
- Spellcheck

PROGRAM TYPES

Resident programs are loaded during initial program load and are present at all times in the fixed nucleus.

Transient programs are reusable (see *Program Attributes*) and run in the System/36 transient area (a 4K-byte area starting at address hex 001000).

Both resident and transient programs can be addressed using real instruction addressing.

Translated programs are those for which address translation is used for instruction addressing. These programs can be executed anywhere in main storage and do not require contiguous pages of storage.

PROGRAM ATTRIBUTES

The following attributes define bits in the IDMAC bytes of each SSP program:

Reentrant Program: Multiple tasks can execute concurrently within a reentrant program. With more than one task using the same code, it is essential that data relevant to specific tasks be stored somewhere other than in the reentrant program. Because the dispatcher can preempt a reentrant program at any soft stop point to allow another task to execute it or to take its storage (without writing it to the swap area), the program must be designed to be refreshable at all soft stop points (see *Soft Stop*). When the program needs storage again the reentrant program is read in from its library.

Reusable Program: Multiple tasks can execute a reusable program but, because the code can be altered during execution, it must be reinitialized with each use. The dispatcher assumes it can preempt a reusable program at any refreshable wait point by releasing its storage without writing it to the swap area. A refreshable wait point is a point in the reusable program at which no data dependencies exist; the supervisor can read in a new copy of the program from the appropriate library, with the ownership of the program possibly changing to a different task.

Single Copy: A program that cannot be shared by multiple tasks; each using task has its own copy of the program. When its storage is taken, the single copy program must be written to the swap area.

Soft Stop: A hardware function requested by control storage when a task switch is necessary. Soft stop can occur at logical breakpoints in a program such as true branch or jump backward conditions, SVC instructions, transfer instructions, or the loading of the IAR for any purposes. Because a task switch can occur at any soft stop point, the supervisor stops the main storage processor and may start up another task.

Swap Required: When taking storage from a swap-required program, the supervisor must write the current copy of the program to the swap area before releasing its storage.

Pinned Program: The supervisor cannot release the storage of a pinned program.

PROGRAM CONTROL

On System/36, programs receive control by either the load or the transfer supervisor call instruction. The programs evoked by LOAD-RUN OCL statements are started as tasks by #SVAT, the task-attach function. The attach function either loads or transfers to the program.

Programs evoked via transfer receive control either via transfer by ID or by disk sector address. On a transfer by ID, the program always receives control at its load address; other than at IPL time, resident programs are always called by ID. A nonresident program may be requested using an ID, if one is defined, or it may be called via sector address. If the transfer is by sector address, an entry point table may be used to specify a start control address other than the program load address. An entry point number other than 0 indicates that an entry point table is to be used. If an entry point table is used, the value of the ENTRY parameter, minus 1, times 2, is the displacement into the entry point table where the start control address is contained.

Resident programs and *transient programs* occupy real addressable storage locations. Except during IPL time, resident programs are transferred to by ID; transient programs receive control via transfer either by ID or by sector.

Nonresident programs are loaded when needed, and are present as long as the storage they occupy is not needed by a higher priority task. They can execute anywhere in main storage and do not require contiguous pages of main storage; these are translated programs. Control is passed to them by transfer either by ID or by sector address.

For more information on program control, refer to *Program Management* in the *Control Storage Service Information* manual.

SUBTASKING

Subtasking allows SSP modules to execute subroutine-equivalent tasks (subtasks) in parallel. An asynchronously executing subtask does not return control to the originating task in the normal subroutine manner, but it can post the originating task during execution; on subtask termination, control storage termination posts the parent task if the transfer request specified RETURN-YES.

For more information on subtasking, refer to *Program Management* in the *Control Storage Service Information* manual.

SSP to Overlay Linkage Editor Interfaces

The overlay linkage editor (OLE) can be entered two ways: compiler entry, directly from a language processor; or user entry, from the OCL load statement. The following routines are used by OLE for I/O interfaces:

- *Compiler access method (CAM)* macroinstruction, using *control storage disk IOS* to get or put sectors of $WORK and $SOURCE files.
- *Find library F1 address* 6 to find library sector address.
- *Librarian find routine* 6 to read subroutine and load members from a library.
- *Librarian sector get/put routine* 6 to write subroutine and load members into a library.
- *Printer data management* 4 to print messages, map storage, and cross reference print data.
- *Disk DTF data management* 4 to write messages to the diagnosed source file.

If OCL statements are used to evoke OLE, OLE calls the *syntax checker* 6 to check the syntax of the control statements. It calls *allocate* 3 or *special allocate* 3 to allocate $WORK and $SOURCE files.

SSP to Sort Utility

The sort utility is part of the base SSP. This utility may be evoked in the following ways:

- User-written OCL statements to load and run the #GSORT program.
- SORT procedure.
- Transfer supervisor call (SVC) instruction directly from a user-written COBOL or assembler program.

The sort utility supports the $SRT and the $SORT macroinstructions in the Assembler and Macro Processor Program Product. The $SRT macroinstruction generates a sort parameter list; the $SORT macroinstruction generates an SVC instruction to call the sort utility from an assembler program.

When a user program calls the sort utility directly by using a transfer SVC instruction, the #GSLS module runs in the system transient area. This module then transfers control to the first sort module (#GSORT). The #GSLS module varies the program size depending on available storage, the minimum sort design point (18K bytes), and the program size of the caller. This same sort module also sets itself up as a termination exit routine so that it receives control if sort terminates abnormally. Control can then be returned to the user with a success or failure indication.

Setup Phase

For the first step in sorting, the program reads the sort specifications, generates code for selecting the records to be sorted, and builds the appropriate work record. The sort utility uses the following system interfaces during the setup phase:

- If called from a user program, use control storage mapping to map the user parameter list to the sort program area.
- *Member find* 6 to find the sort message member (#GS#MM) and set the program level 1 message member pointer in the job control block to that member.
- *Disk VTOC read/write* 6 to read the format 1's for the input, work and output files. From this information, the sort utility can determine the number of input files, the number of records in the input file, the input file record length, whether or not a user WORK FILE statement was specified, and whether or not the output file overlaps an input file.
- *Allocate* 3 to allocate the sort work file if the user specifies a FILE statement with a NAME-WORK parameter. Otherwise, *special allocate* 3 to allocate a disk area large enough to process the input.
- *SYSIN* 6 to read sort specifications if they are inline in the procedure.
- *Source get* 6 if sort specifications are in a source member.
- *Message retrieve* 6 to retrieve sort error and informational messages from the sort message member.
- *SYSLIST* 6 to print specifications, status, and errors.

Execution Phases

These phases read the input file(s) and write work records into the work file. The work records are sorted and then written into the output file. The sort utility uses the following system interfaces during the execution phases:

- *Allocate* 6, *open* 6, and *close* 5 for all input and output files.
- *Deallocate* 3 for input files as they are read.
- *Consecutive get disk data management* 4 to read the input file(s) and *consecutive put disk data management* 4 to write the output file.
- *Control storage disk IOS* to read and write work records in the work file and to clear the work file area to zeros at the end of sort.
- *SYSLOG* 6 to write error or informational messages requiring a response.

SSP to Ideographic Utilities

CHARACTER GENERATOR UTILITY

The character generator utility allows the user to create and maintain a file of ideographic characters that are not contained in the IBM-supplied extended ideographic character set for System/36. The character generator is evoked with the CGU procedure.

The SSP initiator 3 loads #CGIN to initialize for character generation.

The utility uses SSP special allocate 3 to allocate the following files:

- #EXTN, #EXT1818, and #EXT2424 files contain matrix patterns for the extended ideographic character set, the character generator utility special characters, and the user-defined characters.
- #KAMAST file, contains master sort information for all IBM-supplied ideographic characters and user-defined characters.
- #KACTIVE file, contains active collating sort information for all ideographic characters.

SSP SYSLOG 6 receives control from any of the character generation utility modules to issue messages and prompt for operator options.

SSP termination 5 receives control from #CGML to terminate the character generator utility.

IDEOGRAPHIC SORT UTILITY

The ideographic sort utility can sort up to eight disk files by one of three methods: (1) addrout sort, (2) tagalong sort, or (3) summary tagalong sort. The ideographic sort is invoked by the SRTX procedure.

The SSP initiator 3 loads #CGIN to initialize for character generation.

The utility uses SSP special allocate 3 to allocate the following files:

- #KAMAST file, contains master sort information for all ideographic characters and user-defined characters.
- #KACTIVE file, contains active collating sort information for all ideographic characters.

SSP SYSLOG 6 receives control from any of the sort utility modules to issue messages and prompt for operator responses.

SSP termination 5 receives control from #KA4A or from #KATB to terminate the sort utility.

SSP to Language Program Product Interfaces

This section describes the language program product interfaces to the SSP. For information on the language program product execution-time and compile-time modules, refer to *Appendix B: Programming Language Directory*.

The following SSP to language program product interfaces are described:

- RPG
  - Compile-time interfaces
  - Execution-time interfaces
- BASIC
- COBOL
  - Compile-time interfaces
  - Execution-time interfaces
- FORTRAN
  - Compile-time interfaces
  - Execution-time interfaces
- Assembler Language and Macro Processor

SSP TO RPG INTERFACES

Compile-Time Interfaces

The RPG compiler uses the following interfaces to SSP and system functions:

- *Compiler information block (CIB):* The CIB is a system block created by the *initiator* 3 when a compilation is requested. The compiler extracts the modification reference number, date, and time from the CIB for the compiler prologue. The compiler also places information into the CIB (the current system date and time, the library submember type, the compiler work file extents, and the region size available) for later processing by the overlay linkage editor. The compiler reads/writes the CIB using *information retrieval* 6.
- *System local data area (LDA):* The system LDA is used to pass the parameters from the compiler procedures to the compiler. The compiler reads/writes this area using *information retrieval* 6.
- *Work file IOBs:* The auto report function of the RPG compiler uses the compiler access method (CAM) macroinstruction to access compiler work files. CAM uses the IOB as input to *control storage disk IOS*, which reads/writes the files. The rest of the compiler accesses the $SOURCE and the $WORK files without the use of CAM, by updating the IOB and calling disk IOS directly.
- *Define the file (DTF):* If a diagnosed source member was requested, the compiler uses *open* 3 and *allocate* 3 to prepare another work file ($WORK2) using the DTF control block. Input and output to the file are performed with the $GETD and $PUTD macroinstructions (*disk data management* 4).
- *$WORK file:* The compiler passes the generated object program to the overlay linkage editor in the $WORK file.
- *Where-to-go table:* Each of 13 RPG compiler modules contain a where-to-go table, which contains entries for each of the next several phases to be executed. The compiler routine that calls the next phase uses a where-to-go table entry as the loader parameter list. This table is updated automatically by the *cross-reference resolver* 6 whenever library maintenance takes place in the #RPGLIB library.

Execution-Time Interfaces

The following RPG II data areas can be accessed by SSP whenever an RPG II user program is executed:

- *Reserved object communications area (ROCA):* A reserved data area that is contained in the first 256 bytes of the root segment of every RPG II object program. SSP accesses the indicator table, which contains an entry for every possible RPG II indicator.

- *Define the file (DTF):* The DTF is the primary external interface to data management. The program contains a preopen DTF for each file specified by the program. When the program is executed, the *job start function* 3 modifies the preopen DTFs and returns them as post-opened DTFs.

  After external requests have been made (generally to the *data management function* 4, the *data communications function* 9, or the *SSP-ICF function* 10), the requested SSP function sets the postopen DTF completion code and returns the DTF.

  After a DTF is opened, the following system macroinstructions are used for I/O to the file:

  - $GETD and $PUTD–*Disk data management* 4
  - $PUTP–*Printer data management* 4
  - $GETB and $PUTB–*BSC data management* 9
  - $WSIO–*Work station data management* 4

- *Input and output buffers:* Every RPG program specifies an input buffer for each input file specified. The input buffer address is contained in the DTF at postopen time.

  An output buffer is assigned for each output device specified. If the output buffer length is less than 144 bytes, the buffer is located in the prime work area in ROCA. If the buffer length is greater than 144 bytes, a separate buffer area is allocated for the program. At postopen time, the address of the output buffer area is contained in the DTF.

- *WORKSTN DTF extension:* This area contains additional information, used by the RPG II program, about the work stations. The *initiator* 3 and *data management* 4 use this to get the address of the WORKSTN ID table.

  WORKSTN ID table: The WORKSTN ID table is an array of entries, describing the work stations associated with a program. An entry exists for each work station in a file. Within a given file, each work station entry is unique. The *initiator* 3 and *data management* 4 help maintain this table by setting the requester flag bit on for each entry, up to the MRTMAX value (number of display stations that can request a MRT program).

- *RPG II halt processor transient ($$SYRP):* This module is used when an error occurs during the execution of an RPG II user program. The module processes halts when requested by an RPG II compiler or RPG II user program. It sets up a parameter list for *SYSLOG* 6 and processes operator responses to the message. If a 0, 1, or 2 option is taken, $$SYRP returns to the RPG II program; if the operator cancels, $$SYRP passes control to the *termination function* 5.

  Option 0: Return to next instruction
  Option 1: Go to input mainline (unless otherwise stated)
  Option 2: Go to close mainline, close file, and go to end job

- *SSP-ICF WORKSTN file:* The RPG programmer uses a WORKSTN file as the interface to *SSP-ICF data management* 10. For all WORKSTN file operations, *work station data management* 4 checks the ID field in the DTF. If the first character of the ID is numeric, the operation is routed to *SSP-ICF data management* 10; otherwise, it is routed to *work station data management* 4.

SSP TO BASIC INTERFACES

BASIC Program Product Overview

The BASIC Program Product operates in either *edit* mode or *run* mode.

During edit mode, the BASIC program product takes user source statements from the display station or a user library source member and converts the statement operators and operands into intermediate text. Intermediate text is the form in which BASIC source statements are stored on disk or in the main storage work area; the intermediate text provides a standardized notation scheme that allows faster compiler execution. During run mode, the compiler converts intermediate text into BASIC instruction set instructions. The instructions are executed by the BASIC instruction set emulator in control storage. Because the BASIC Program Product is interactive, it has many interfaces to SSP functions.

Common BASIC Interfaces

The following common interfaces are used by most BASIC functions and components:

- Attn key processing by all BASIC modules accesses the TB and JCB.
- Device I/O operations use *open, allocate* and *deallocate* 3.
- File operations use *open, allocate,* and *deallocate* 3.
- Library operations use *open, allocate,* and *deallocate* 3, *librarian find* and *member find* 6, and *control storage disk IOS*.
- Message operations use *message retrieve* 6 and *SYSLOG* 6.
- Print operations use *message retrieve* 6 for headings, control storage time and date retrieve for time and date information, and *printer data management* 4 for printing.
- For local data area access, the local data area I/O handler uses *information retrieval* 6 to read, write, reread, or rewrite to the local data area.

Special BASIC Interfaces

The following BASIC components and functions use special interfaces to SSP:

User Sign-On Commands

At sign-on time, the SSP *initiator* 3 processes the BASIC sign-on procedure commands. The commands and the initialization modules they evoke via the *initiator* 3 are:

BASIC	Normal initialization (#BLINC)
BASIC MRT	MRT run mode initialization (#BLMIC)
BASICP	Procedure initialization (#BLPIC)
BASICR	Special run mode (#BLSIC)
BASICS	BASICS initialization (#BLSNC)

The initialization modules use the following interfaces:

- *Control storage disk IOS* to read and write from the work file
- The XFER SVC instruction to transfer between modules ($XFER macroinstruction)
- *Information retrieval* 6 to read the system local data area
- *Special allocate* 3 to allocate the work file
- *Work station data management* 4 to determine system attributes and to open the work station file
- *Folder management services folder open* 4 to open a data dictionary

Other BASIC Commands

Other (non-initialization) BASIC commands and the system functions they evoke are:

AUTO — The display station module (#BLDSE), which handles the AUTO command, uses *work station data management* 4 to display the line numbers on the screen

ALERT — The display station module (#BLDSE), which handles the ALERT command, uses *work station data management* 4 to display the message on the screen

COLSEQ — The COLSEQ processor (#BLCSE) uses:

- *Work station data management* 4 to display the sequence
- *Control storage time and date* to get the current time and date

FREE — The FREE module (#BLFRE) uses:

- *Library member delete* 6 to delete the member
- *Special allocate* and *special deallocate* 3 to deallocate the file

HELP — The HELP module (#BLHPE) uses:

- *Message retrieve* 6 to get screen output
- *Work station data management* 4 to process prompts

HISTORY — The command processor (#BLCPE), which handles the HISTORY command, uses *history file put* 8 to log the history file for the HISTORY command

LIBRARY — The LIBRARY module (#BLLDE) uses *librarian find* 6 to find the specified library

DATADICT — The command processor (#BLCPE) which handles the DATADICT command uses *FMS folder open* 4 to open a data dictionary

LIST — The LIST module (#BLLIE) uses *work station data management* 4 to list the work space

LISTP — The LISTP module (#BLLPE) and the XREF parameter processor (#BLXRE) use:

- *Special allocate* and *open* 3 to allocate and open the printer
- *Printer data management* 4 to print the lines
- *Close* 5 and *deallocate* 3 to close the printer

LOAD — The LOAD module (#BLLDE) uses:

- *Source get* 6 to load source or data
- *Control storage disk IOS* to load a SUBR member
- *Librarian find* and *library member find* 6 to find the library and member

MERGE — The MERGE processor (#BLMGE) uses:

- *Control storage disk IOS* to read and write the work file for programs
- *Source get* 6 to get the records from the data source member

PRINT — The formatted print processor (#BLPFR) and the unformatted print processor (#BLUPR) use *work station data management* 4 or *printer data management* 4 to process print command requests to the work station or printer, respectively

The library I/O processor (#BLDIR) uses control storage disk IOS to put the data in a work file

PROC — The load module (#BLLDE), which handles the PROC command, uses:

- *Librarian find* and *library member find* 6 to find the library and member, respectively
- *Source get* 6 to load source

RENUM — The RENUM processor (#BLRNE) uses *control storage disk IOS* to read and write the work file

SAVE or REPLACE — The SAVE module (#BLSVE) uses *librarian find* and *library member find* 6 to find the library and member, respectively

STATUS — The BASIC trace module (#BLRER) uses:

- *Message retrieve* 6 to get the headings
- *Control storage time and date* to get get heading data
- *Allocate* and *open* 3 to open the printer
- *Printer data management* 4 to print the information
- *Close* 5 and *deallocate* 3 to close the printer.

SUBPROC — The load module (#BLLDE), which handles the SUBROC command, uses:

- *Librarian find* and *library member find* 6 to find the library and member, respectively
- *Source get* 6 to load source

BASIC Statements

BASIC statements with external interfaces, and the modules they evoke are:

BREAK — The trace module (#BLRER), which handles the BREAK statement, uses *work station data management* 4 to display the character expression on the screen

CHAIN — The chain module (#BLCHR) uses:

- *Librarian find* and *library member find* 6 to find the library and member, respectively
- *Source get* 6 to load source
- Control storage disk IOS to load an SUBR member

OPTION COLLATE — The run time module (#BLRUE), which handles the OPTION COLLATE statement, uses:

- *Librarian find* and *library member find* 6 to find the library and member, respectively
- *Source get* 6 to load source

PAUSE — The trace module (#BLRER), which handles the PAUSE statement, uses *work station data management* 4 to display the character expression on the screen

RESTORE	For input disk files, the open module (#BLOPR) uses *disk data management* 4 For output disk files the open module (#BLOPR) uses *special close* and *special deallocate* 3, and *special allocate* and *special open* 3 to restore an output disk file. For library members the open module (#BLOPR) uses *source get* 6
TRACE	The trace module (#BLRER) uses *message retrieve* 6 to retrieve messages, *work station data management* 4 to put messages to the screen, *special allocate* and *open* 3 to allocate and open the printer, and *printer data management* 4 to print the trace output. The trace module also uses control storage time and date retrieval in performing the requested trace

Intrinsic Functions

Intrinsic functions and the interfaces they use are:

ATTRIBUTE$	*Work station data management* 4
CURCOL and CURROW	*Work station data management* 4
DATE and DATE$	*Information retrieval* 6
MSG$	*Librarian find, member find,* and *message retrieval* 6
TIME and TIME$	*Control storage time and date* functions
TIMER	*Work station data management* 4 or *SSP-ICF data management* 10
UPSI$	*Information Retrieval* 6
USERID$	JCB
RETCODE$	*Work station data management* 4 or *SSP-ICF data management* 10

Source Entry

BASIC source entry takes place between the time the BASIC or BASICP command was processed and the time the programmer enters the RUN command. The BASIC edit-time display station manager 1 (#BLDSE) interfaces between SSP *work station data management* 4 and the BASIC statement and command processors.

Special Run Mode

At the beginning of special run mode, the initialization module (#BLSIC) uses *librarian functions* 6 to load the SUBR member containing the BASIC program.

I/O Interfaces

- *PUT/GET I/O Handlers*: During the run-time execution of a BASIC program, the PUT I/O handler (#BLPSR) and the GET I/O handler (#BLGSR) use *Disk data management* 4 to write to or read from a stream file or unformatted disk file.

- During the run-time execution of a BASIC program, the BASIC I/O handler (#BLIOR) uses *disk data management* 4 to process disk read, write, delete, and restore requests.

- The BASIC SSP-ICF module (#BLCMR) uses *SSP-ICF data management* 10 to read from or write to a communications session.

- The BASIC WAITIO module (#BLWAR) uses *SSP-ICF data management* 10 and *work station data management* 4 to perform an accept operation.

- The SFGR work station I/O module (#BLIOR) uses *work station data management* 4 to perform read, write, and rewrite operations to the display station.

- *BASIC Print*: During the run-time execution of a BASIC program, the following print and input operations use SSP interfaces:

  Print to work station or printer: The formatted print processor (#BLPFR) and the unformatted print processor (#BLUPR) use *work station data management* 4 or *printer data management* 4 to process print requests to the work station or printer.

  Print to library members: The library member I/O processor (#BLDIR) uses *control storage disk IOS Full screen processor*: The full screen processor module (#BLFSR) uses *work station data management* 4 to process PRINT FIELDS and INPUT FIELDS statements.

  INPUT/LINPUT from work station: The INPUT/LINPUT I/O handler (#BLIPR) uses *work station data management* 4 to perform INPUT/PRINT operations.

  INPUT/LINPUT from library members: The library member I/O module (#BLDIR) uses *source library get* 6 to read from library members.

BASIC Procedures

In running BASIC procedures, the load source module (#BLLSE) uses *source get* 6 to get the next line from the applicable procedure.

BASIC Commands Processor

In processing BASIC commands, the BASIC command processor module (#BLCPE) uses *history file put* 8 to log to the history file, control storage disk IOS to read/write the work file and *termination* 5 to process the OFF command.

BASIC Compiler

The compiler (#BLICR) uses control storage disk IOS, *special deallocate* 3, and *work station data management* 4.

BASIC Run-Time Monitor

The run-time monitor (#BLRUE) uses *control storage disk IOS*, *work station data management* 4, and *librarian find* 6.

Open

The BASIC OPEN module (#BLOPR) uses the following SSP interfaces:

- For a disk file:
  - *Special allocate* and *special open* 3 to allocate and open the file
  - *Active format-1 area access* 6 to scan fields in the format 1

- For an input library member: *Library find, single name find,* and *source get* 6, to set up the source get parameter list

- For an output library member: *library find, single name find, member protect* 6 and *special disk allocate* 3 to allocate the scratch file

The BASIC OPEN module (#BLPOR) uses *special allocate* and *special open* 3 to allocate and open the printer file

The BASIC OPEN module (#BLWOR) uses the following SSP interfaces:

- For a work station file:
  - *Library find* 6 to find the SFGR library
  - *Single name find* 6 to find the SFGR load member
  - *Work station data management* 4 to open the work station file

- For an SSP-ICF file: *Special acquire* 10 to open the communications file

Close

The BASIC close module (#BLCLR) uses the following SSP interfaces:

- For a disk file:
  - *Special deallocate* and *special close* 5 to deallocate/close the file
  - If close is requested for a stream file, *disk data management* 4 to write the last record.

- For a printer file:
  - *Printer data management* 4 to print the last record
  - *Special deallocate* and *special close* 5 to close the printer file.

- For a work station file: *Work station data management* 4 to close the file

- For an SSP-ICF file: *SSP-ICF data management* 10 to end the SSP-ICF session

- For an input library member: If FREE is specified in the close request, *library member delete* 6

- For an output library member:
  - *Source get* to read records, if writing to the end of an old member
  - *Source put* to write records to the member
  - *Special deallocate* 3 to deallocate the scratch file
  - *Control storage time-of-day* to set the directory entry
  - If FREE is specified in the close request, *Library find* and *Library member delete*

- For output to a new member (or if BEGIN was specified and no records were written): *Library member protect* 6

Termination

The BASIC termination module (#BLXTR) and special run mode termination (#BLSXR), use *control storage disk IOS* to read/write the work file, and *deallocate* and *close* to close the trace print file. The special run mode termination module (#BLSXR) also uses *SYSLOG* 6 and *termination* 5 to perform error message handling and end of job for the BASIC program.

SSP TO COBOL INTERFACES

Compile-Time Interfaces

The COBOL compiler uses the following interfaces to SSP and system functions:

- *Compiler Information Block (CIB):* The CIB is a system block created by the *initiator* 3 when a compilation is requested. The compiler extracts the modification reference number, date, and time from the CIB for the compiler prologue. The compiler also places information into the CIB (the current system date and time, the library submember type, the compiler work file extents, and the region size available) for later processing by the overlay linkage editor.

- *Extendable compiler work files:* When the value that the user specified for allocating the compiler work files becomes too small, the compiler attempts to extend the work files automatically using the compiler access method (CAM) macroinstruction. If it extends the work files successfully, the new compiler work file extents are placed in the CIB.

- *Diagnosed source member:* When the user requests a diagnosed source member, another compiler work file is allocated to contain this member. At the end of the compilation, this work file is passed to SEU (for the COBOLONL procedure) or to the *library maintenance utility* 8 (for other compilation procedures) for replacement of the original source member.

- *Work file IOBs:* CAM uses the IOB as input to *control storage disk IOS*, which physically reads/writes the files. The compiler accesses the $SOURCE and the $WORK files via CAM, which updates the IOB and calls *control storage disk IOS*.

- *Where-to-go table:* One COBOL compiler module (#CB00) contains a where-to-go table, which contains an entry for each of the compiler phases to be executed. The compiler routine (in #CB00) that calls the next phase uses the where-to-go table entry as the loader parameter list. This table is updated automatically by the *cross-reference resolver* 6 whenever library maintenance takes place in #COBLIB.

- *System local data area (LDA):* The system LDA is used to pass the parameters from the compiler procedures to the compiler. The compiler reads/writes this area using *information retrieval* 6.

Execution-Time Interfaces

COBOL object programs use the following interfaces to SSP and system functions:

- *Define the file (DTF):* The DTF is the primary external interface to data management. Every user-defined file in a program creates an associated DTF. The COBOL compiler also creates a pre-DTF that is located in front of each DTF. The generated program maintains data in the pre-DTF, which contains information essential to the proper execution of the file. All file input/output operations in the COBOL-generated user program operate through standard interfaces with *disk data management* 4, *printer data management* 4, *work station data management* 4, and *SSP-ICF data management* 10.

- *SSP-ICF transaction files:* Multiple transaction files can be defined for a COBOL program. A single transaction file can be repeatedly opened and closed, but no more than one transaction file may be opened at a time.

  The COBOL program uses the transaction (work station) files as the interface to *SSP-ICF data management* 10. For all transaction file operations, *work station management* 4 checks the ID field in the DTF. If the first character of the ID is numeric, the operation is routed to *SSP-ICF data management* otherwise, it is routed to *work station data management*

- *Execution-time halts:* For input/output operations to user-defined files, the user must provide error handlers for exceptional conditions. If no appropriate error handler is specified for a given file or operation and an exception occurs, the COBOL-generated program issues a halt. The COBOL display error message routine sets up a parameter list for *SYSLOG* 6 and processes operator responses to the error message. The operator can end the program or return to the program at the next logical statement. If the operator ends the program, the routine passes control to the *termination function* 5.

SSP TO FORTRAN INTERFACES

The FORTRAN Program Product consists of three parts: a compiler, an execution-time library, and the scientific interpreter. The compiler creates an object program which the overlay linkage editor (OLE) combines with the library routines to create an executable load module. The scientific interpreter is a control storage module that emulates the scientific instruction set.

Compile-Time Interfaces

The FORTRAN compiler uses the following interfaces to SSP and system functions:

- *$SOURCE file:* The FORTRAN compiler uses the compiler access method (CAM) macroinstruction to read the source program from the $SOURCE file. The *initiator* 3 copies the source member or the *SYSIN* 6 data stream to this file.

- *Compiler information block (CIB):* The CIB is a system block created by the *initiator* 3 when a compilation is requested. The compiler gets the source reference number, modification time, and date from the CIB and prints them, using the $DTFP and $PUTP macroinstructions to interface with *printer data management* 4. It saves the compile time and date in the CIB for use by OLE.

- *System local data area (LDA):* The compiler also uses the system LDA for passing parameters that are not passed by the CIB from the procedure to the compiler. The $INFO macroinstruction is used.

- *Cross-reference resolver (OXRF):* The compiler uses the *cross-reference resolver* 6 to locate the next compiler phase. The useful information from the OXRF entries is moved into the non-overlayable portion of the root module. The next compiler phase is then loaded by the $LOAD macroinstruction.

- *$WORK file:* After the FORTRAN compiler has generated scientific instructions for the FORTRAN source statements, it uses *disk data management* 4 to write the object code to the $WORK file. This file is used by the overlay linkage editor to create the load and/or subroutine modules.

- *Define the file (DTF):* The DTF is the primary external interface to data management. The compiler generates a DTF for each file and/or device used in the FORTRAN program. The $DTFP, $DTFW, and $DTFD macroinstructions are used to build skeleton DTFs (for *printer, work station, and disk data managements* 4, respectively). The DTFs are written to the $WORK file.

Execution-Time Interfaces

When a // LOAD OCL statement is used on a FORTRAN-generated load module, the *allocate function* 3 loads the FORTRAN microcode into control storage.

The FORTRAN program communicates with the scientific interpreter through a communications area within the program. This area, called FINCOM, is defined by the DFIN macroinstruction. Normally, index register 1 points to this area during program execution. The FORTRAN program transfers control to the interpreter by executing an XFER instruction. (For more information on this instruction, refer to *XFER Instruction* in this section.)

The library routines provide access to the following system functions:

- *Printer data management* 4
- *Disk data management* 4
- *Work station data management* 4
- *SYSIN* 6
- *SYSLIST* 6
- *Message retrieve* 6
- *Information retrieval* 6
- *Control storage time of day*

SSP TO THE ASSEMBLER LANGUAGE AND THE MACRO PROCESSOR

The macro processor reads the $SOURCE file and writes the source program into the $ASMINPT file, expanding all macroinstruction call statements. The assembler uses $ASMINPT as the source file ($SOURCE if the macro processor was not invoked), $WORK2 as a work file, and writes the object records to $WORK. All files are extendable.

SSP to Assembler

The assembler uses:

- *Disk data management* 4 to read and write files
- *Disk data management* 4 (via the CAM macroinstruction) to read and write the $WORK2 file when creating the cross-reference listing
- *System find* 6 to retrieve the assembler phases
- *Control storage loader* to run the phases
- *Printer data management* 4 to create the program listing

Parameters are passed from the ASM procedure to the assembler via the system local data area (LDA). The assembler reads the LDA using *information retrieval* 6.

The assembler gets the modification reference number, date, and time from the compiler information block (CIB), and it puts the date and time of assembly, the library member subtype, the work file extents, and the region size into the CIB using *information retrieval* 6.

The assembler calls *SYSLOG* 6 to issue any displayed messages.

The compiler passes the generated object program to the overlay linkage editor in the $WORK file.

For information on the the execution-time interface between SSP and a user-written assembler program, see the *Programming With Assembler* manual and the *Interactive Communications Feature: Reference* manual.

SSP to Macro Processor

The macro processor uses *disk data management* 4 to read and write files.

The macro-processor phases are retrieved and run using $FIND and $LOAD macroinstructions. A system find and load is performed to make *source get* 6 resident. Source get retrieves the macroinstruction definitions that are to be expanded into source.

The macros can be stored in any library. This is referred to as the macro library. The macro processor locates the macro library via *active format-1 area access* 6.

SSP to the Utilities Program Products

The following SSP to the Utilities Program Product interfaces are described:

- DFU
- SEU
- WSU
- SDA

SSP TO DFU

DFU has two major phases: setup phase and execution phase. The user signs on DFU by entering the DFU command. If the DFU program does not exist, DFU goes through setup to allow the user to define his DFU program.

Setup Phase

During setup, DFU maintains a common area pointed to by index register 1. This area contains constants, keyword translations, work file pointers, display station DTFs, and several subroutines used by all setup modules.

DFU retrieves the modification reference number from the library control block and increments it if the user requests the DFU specification source to be saved.

DFU also invokes the *screen format generator* 8 to convert the DFU execution format source to load member form. The load member has the same name as the DFU program.

If a severe error is encountered or if the user wishes to cancel, DFU invokes *SYSLOG* 6 to issue an error message.

Execution Phase

After the subroutine and load members are created, DFU begins the execution phase. DFU maintains an execution common area. This area contains constants, keyword translations, control block pointers, and subroutines used by all execution-time modules.

DFU uses *disk data management* 4 for record insertion, retrieval, and update. DFU uses the $DTFW and $WSIO macroinstructions to interface with *work station data management* 4.

If a severe error is encountered or if the user cancels, DFU invokes *SYSLOG* 6 to issue an error message.

SSP TO SEU

Most of the SSP interfaces with SEU occur during initialization and at end of job. SEU maintains a common area, which contains the work station DTF, where-to-go tables (maintained by OXREF), an IOB, SEU's data management routines (which use disk IOS), and various other subroutines and data areas.

Initialization

The user enters the SEU command to sign on to SEU. *SYSIN* 6 is used to retrieve the control statement. SEU uses *SYSLOG* 6 to issue messages. Keywords are retrieved from SEU's message member by *message retrieve* 6.

SEU maps to the format index area and builds a table of user format names and lengths for future use.

The *library maintenance utility* 8 is used for finding libraries and library members and for protecting the member being edited.

If a diagnosed source member is specified in the control statement, the diagnosed source file is allocated as a consecutive output file. *Disk data management* 4 is used to read the records from the file.

Special allocate 3 is used to build the work file used by SEU. *Source get* 6 retrieves the statements from the member, and disk IOS (control storage) is used to put the statement into the work file.

End of Job

SEU reads the library control sector to get information in the library. The library member subtype and reference number are updated in the library directory using the *library maintenance utility* 8.

Special allocate 3 is used to allocate a work library, which temporarily contains the new or updated member.

Disk IOS (control storage) is used to get the statements out of the work library and *library sector get/put* 6 is used to put the statements into the work library. *Library sector get/put* 6 is then used to update the user's library from the work library.

SSP TO WSU

WSU has two major phases: the generation phase and the execution phase. During the generation phase, the user's WSU source program is read and three library members can be produced:

- An executable R-module
- A user procedure to execute the R-module
- Any necessary screen format members

During the execution phase, the R-module is executed and any of the 24 WSU execution phases are linked to it as necessary.

Generation Phases

During WSU program generation, WSU uses the first 80 bytes and the last 18 bytes of the system local data area for various internal considerations. The reference number, reference date, and reference time are placed in the library directory entries of the generated libary members.

The system macroinstructions used, and the system functions evoked, during the generation phase are:

$ALOC	*Allocate* 3
$DTFD	*Disk data management* 4
$DTFO/$OPEN	*Open* 3
$PUTP	*Printer data management* 4
$DTFP	*Printer open* 3
$IOBD	*Control storage disk IOS*
$FIND/$FNDP	*System find* 6
$LOAD	*Control storage loader*
$TOD, $TRB	*Control storage timer*
$XFER	*Control storage transfer*

Execution Phases

All execution initialization functions are performed by code that is executed once and then overlaid. Except for a small number of services provided by resident WSU routines, post-initialization functions are performed by WSU transient routines. These routines are packaged in load modules (#WSX00-#WSX20) that are accessed by call/exit linkage subroutines, which handle module loading.

Because WSU transients are not loaded at predetermined locations, entry point addresses must be resolved dynamically. Address contents for individual routines are in a table in front of the module to aid resolutions.

The DTF (define the file) is the primary external interface to data management. One DTF is assigned for each logical disk file used in the WSU program. However, here is only one work station DTF and screen data buffer assigned for the entire WSU program, regardless of the number of user formats defined or the number of work stations supported.

Nonresident data other than data file records and transients are kept in a work file. At initialization time, data blocks (messages and calculation specifications) are transferred from the object program to the work file and then retrieved as needed.

The system macroinstructions used, and the system functions evoked, during the execution phase are:

$ALOC	*Allocate* 3
$CLOS	*Close* 5
$DTFO/$OPEN	*Open* 3
$DTFW/$WSEQ	*Work station data management* 4
$FIND/$FNDP	*System find* 6
$INFO	*Information retrieval* 6
$IOBD	*Control storage disk IOS*
$LOAD	*Control storage loader*
$LMSG/$LOG/ $LOGD	*SYSLOG* 6
$TRB	*Control storage timer*
$XFER	*Control storage transfer*

SSP TO SDA

SDA consists of four major phases: initialization, selection, definition, and termination.

Initialization

The initialization phase uses the following SSP interfaces:

- The system local data area (LDA), which is initialized to establish the user's SDA profile.
- Existence checking (*special allocate function* 3) on SDA work files to determine recovery potential.

Selection

The selection phase uses the following SSP interfaces:

- *Work station data management* 4 and the *message retrieve* 6 to support display screen presentation during selection.
- The system LDA for retrieving and maintaining SDA profile information.
- Source member retrieval (*record mode source get* 6).
- Work file allocation/initialization of primary file.
- Load and execution of definition and termination phases.

Definition

The definition phase uses the following SSP interfaces:

- *Work station data management* 4 and *message retrieval* 6 to support display presentations during the definition phase.
- Primary work file I/O to save definition work.
- Work file allocation/initialization/deletion of the secondary file.

Termination

The termination phase uses the following SSP interfaces:

- *Work station data management* 4 and *message retrieval* 6 to support display presentation for end-of-function option display screens.
- Primary work file I/O and deletion.
- *Control storage time and date* for directory and printer heading.
- Source member output (*record mode source put* 6).
- *Printer data management* 4.
- System LDA access for linkages to compilers (invokes the *screen format generator* 8, *build menu* 6, and WSU).

SSP to OFFICE/36

OFFICE/36 includes Query/36, Personal Services/36 (PS/36), DisplayWrite/36 (DW/36), and PC Support/36. Because of obvious PS/36 and DW/36 interdependencies, the interface descriptions for PS/36 and DW/36 are combined. As a convenience to the reader, the screen names associated with these PS/36-DW/36 user functions are listed under the names of the functions they invoke. Query/36 interfaces are described separately.

SSP TO OFFICE/36 EXECUTABLE CODE

SSP to Query/36 Interfaces

Query/36 uses *special allocate* 3 to allocate output disk files, overflow disk files, printers, and all disk files being queried.

For all operations to display stations, Query/36 calls *work station data management* 4 via a transfer SVC instruction with a work station parameter list (WSPL).

Query/36 uses *query data management* 4 to retrieve records from disk files being queried.

Query/36 calls *control storage disk IOS* to do the following:

- Read a library directory to build a list of queries in a library.
- Read Query/36 members from a library.
- Read the VTOC to build a list of libraries or files.
- Read/write the overflow disk file, if necessary.

Query/36 uses *folder management services (FMS)* 4 for the following purposes:

- To read file definition information from dictionaries including formats contained in file, format record ID codes, fields in formats, and field locations, types, and sizes.
- To interface to the DW/36 editor for the purposes of putting output data, or the data field instructions that define the output fields, into a DW/36-created work document.
- To interface to DW/36 print to put Query/36 output data into a special page in the DW/36 document.

Query/36 uses *VTOC access* 6 to determine what, if any, file definition and dictionary a file's format 1 is linked to.

Query/36 uses *librarian functions* 6 for the following purposes:

- To find a library and a Query/36 member (SUBR) within the library, to make the library the new working library, and to free the old library.
- To protect a Query/36 member.
- To write a new Query/36 definition to a library, or update an existing Query/36 member in a library.

Query/36 uses *system evoke* 2 to build the default Query/36 library #QUERY.

Query/36 uses *query data management* 4 to retrieve records from the input file specified for a report.

Query/36 interfaces to *security support user profile services* 8 to save information about a specific user ID.

SSP to PS/36 and DW/36

As a convenience to the reader, the display names associated with the following functions are listed under the names of the functions they invoke.

Maintain Mail Folder and Mail Control Flow

The maintain mail folder function is invoked by the following displays:

- OFCMNT
- MAINTAIN MAIL FOLDER
- MOVE DOCUMENT TO PRIVATE FOLDER
- DELETE DOCUMENTS FROM MAIL FOLDER
- CHANGE DOCUMENT COUNTERS

The maintain mail folder function uses *librarian functions* 6 to find the DW/36 library and load member and to find the resource security module.

The maintain mail folder function uses *folder management services* 4 to open, close, copy, delete, update, search, and list the mail folder; and to open, close, and update the users' folders. Also, the mail folder uses *Resource Security* to secure documents in the users' text folder.

The mail control flow function uses the following interfaces:

- *Special allocate* 3 to allocate disk and printer files.
- *Work station data management* 4 to read from/write to display stations.
- *Folder management services* 4 to access mail logs, access mail folders, and to access the DW/36 print function temporary folder.

The maintain mail folder and mail control flow functions uses *profiles transforms* 4 to convert documents from L2 or L3 to DW/36 format.

Document Library Services

The document library services function is invoked by the OFCDLS display.

The document library services function uses the following interfaces:

- *Librarian function* to find the DW/36 library and load members and to find the PC Support/36 library and load members.
- *Special allocate* to allocate disk and printer files.
- *Disk data management* to read from/write to disk files.
- *Work station data management* to read from/write to display stations.
- *Folder management services* to access (open, close, copy, delete, update, and search) the library services folder and to access (open, close, update, and search) users' folders.
- *VTOC access* (read) to determine file type.

Maintain Communications Routes and Queue Definitions

The maintain communications routes and queue definitions functions are invoked by the following displays:

- OFCMNT
- MAINTAIN COMMUNICATIONS ROUTES
- ADD COMMUNICATIONS ROUTES
- CHANGE COMMUNICATIONS ROUTES
- DELETE COMMUNICATIONS ROUTES
- MAINTAIN COMMUNICATIONS QUEUE DEFINITIONS
- ADD COMMUNICATIONS QUEUE DEFINITIONS
- CHANGE COMMUNICATIONS QUEUE DEFINITIONS
- DELETE COMMUNICATIONS QUEUE DEFINITIONS

The maintain communications routes and queue definitions functions use *folder management services* 4 to open and close the mail folder.

Administration Installation

The administration installation function is invoked by the following displays:

- SET UP OR MAINTAIN PS/36 FUNCTIONS
- CREATE DIRECTORY FILES
- CREATE GROUP FILE
- CREATE CALENDAR INDEX FILE
- CREATE ACTIVITY QUEUE FILE
- CREATE MAIL FOLDER
- CREATE MAIL INFORMATION FILE
- CREATE ACKNOWLEDGMENT FILE
- CREATE MAIL QUEUE FILE
- CREATE PROFILE FILE
- CREATE STORAGE FOLDER
- MAINTAIN PS/36 DEFAULTS

The administration installation function uses the following interfaces:

- *Special allocate* 3 to allocate all disk files it creates.
- DDM to read and write to disk files.
- VTOC access (read/write) to determine if a file's format is on the system.
- *Work station data management* 4 to read from/write to display stations.
- *Folder management services* 4 to create or perform existence checks on the mail folder and the safe folder.
- *Resource security file and edit* 8 to secure PS/36 files and folders.

Enrollment

The enrollment function is invoked by the following displays:

- SELF ENROLLMENT
- ENROLL NEW PS/36 USER
- USER DEFAULTS
- ENROLL NEW INDEPENDENT WORK STATION USERS
- ENROLLMENT INFORMATION
- CHANGE ENROLLMENT FOR PS/36 USERS
- CHANGE ENROLLMENT FOR IWS USERS
- DELETE FROM PS/36
- DELETE ENROLLMENT

The enrollment function uses the following interfaces:

- *Special allocate* 3 to allocate all disk files it creates or uses.
- *Disk data management* to read from/write to disk 4 files.
- *Work station data management* 4 to read from/write to display stations.
- *Folder management services* 4 to open, close, or delete mail folders, and to create, open, delete, read, or update mail logs.
- *Resource security file and edit* 8 to secure mail logs and check authorization for access to files and the mail log.

Group Control

The group control function is invoked by the following displays:

- USE PERSONAL SERVICES/36
- WORK WITH GROUPS

The group control function uses the following interfaces:

- *Special allocate* 3 to allocate all disk files it creates or uses.
- *Disk data management* to read from/write to disk files.
- *Work station data management* 4 to read from/write to display stations.
- *Resource security* 8 to check authorization for access to files.

SSP TO OFFICE/36 DATA AREAS

Query/36 Data Areas

During its report function initialization, Query/36 retrieves QRYRUN procedure parameters from the procedure parameter save area (PPSA). Information from the PPSA is used to build the Query/36 parameter list, which tells query data management 4 what record selection tests to perform, what fields to select for Query/36 output, and what calculations and/or sorting to perform.

All Query/36 work areas are allocated in query TWS (task work space). When a module needs to access a data area contained in TWS, the module maps to the needed information. Initially, Query/36 is allocated 180K bytes of TWS; the following table shows the general allocation of the query TWS:

Address	Contents
0000	Query definition prompting work area
E800	Query subroutine member build area
18000	Query runtime area

Query/36 receives up to 222K of TWS for a field table. Only the amount needed is obtained.

The Query/36 data management work area is also on TWS. Query/36 receives 60K of TWS for each file, up to a maximum of 300K.

SSP TO DEVELOPMENT SUPPORT UTILITY

The Development Support Utility (DSU) is a program product used to create, edit, remove, view, or print source members and procedure members. DSU has a full screen editor that allows editing for an entire screen of data at a time. The product also provides a wide range of line commands, RPG syntax checking, prompts, and format lines to help enter and change language and utility statements.

DSU uses Session Init to:

- Perform SYSIN.
- Perform SYSLOG.
- Open a session.

DSU uses the List functions to:

- Obtain a list of members.
- Delete a member.
- Read member records for the view or print option.

DSU uses the Edit Init Library Access Method to:

- Protect the edit member.
- Load the member into the file.
- Read the user profile.

DSU uses the Edit function to:

- Handle disk data management.

DSU uses the Exit processing Library Access Method to:

- Protect and find a new member name if specified at end of job.
- Save a member.
- Protect the edit member or new member specified at end of job.
- Delete the workfile using special allocate.

Mail Log

A mail log is a folder management services folder that is created with only the data descriptor area (DDA) of the folder; the mail log entries reside in the DDA.

Mail Folder

The mail folder is a normal folder management services folder that contains a special DDA record for each document. This record is a count of the number of mail log entries and DIA queue entries referring to the document and is called the associate record.

PS/36 and DW/36 Storage

OFFICE/36 uses a flexible storage management scheme that allows it to partition its 48K-byte region into separate areas for program code and data. The following storage map shows the general layout of PS/36 and DW/36 storage:

Address	Contents
0000	DW/36 load module area
0800	System nucleus
1000	PS/36 or DW/36 load module area
Variable	Dynamic storage stack
Variable	Caller parameter control block (optional)

Storage in the dynamic storage stack is allocated in 8-byte multiples. Storage is allocated and freed as a LIFO stack. Executing programs are responsible for sequencing their data area storage request in order by degree of permanence.

Section 4: Directory

This directory serves as a reference to the control flow charts in this manual and to the microfiche listings. The entries under *Chart Number* directly refer to control flow charts in Section 2, and the labels under *Entry Point* (listed for modules with multiple-entry points) can be used in the cross-reference portion of the microfiche listings. Modules are listed in alphabetic order by module name.

Each directory entry contains the module name, control flow chart references (if any), the module entry point(s), the descriptive name, and a brief description of the module's function.

ADDITIONAL MODULE INFORMATION

For additional information on module detail attributes, sizes, and link addresses, enter the following OCL statements:

```
// LOAD $MAINT
// RUN
// COPY FROM-#LIBRARY,NAME-DIR,LIBRARY-O,
// TO-PRINT,LIST-DETAIL,PAGE-NO
// END
```

To decode the attribute bytes given in the $MAINT listing, refer to *Library Directory, Macro DIREQ* in the *System Data Areas* manual.

MODULE STORAGE LOCATION

For the purposes of locating modules in storage, SSP modules can be classified according to three types:

- *Real transients* must execute in the system transient area. The word *transient* is included in the descriptive names of all real transients.

  Note: SSP-ICF subsystems use a transient area defined by SSP-ICF control that is in no way associated with the system transient area. In text, this transient area is always referred to as the SSP-ICF transient area.

- *Resident modules* are loaded according to configuration options specified, or otherwise in effect, at IPL time. The word *resident* is included in the descriptive names of all resident modules.

- *Logical transients* (also called translated transients) execute as transients, but the supervisor can load them in any available storage, real or translated.

DIRECTORY CHART REFERENCES

Many module entries have more than one chart reference. The chart references for each module are listed according to the relative amount of information they contain. Therefore, the first chart reference should provide you with the most information on the subject module's control flow.

Note: Overview charts (those that have chart numbers ending with .0) are not referenced, except when the overview chart is the only reference for a module. If you need more control flow information than you find in the referenced chart(s), you should determine whether Section 2 contains an overview chart for the module reference. (For example, if the first chart referenced is Chart 9.2.3, you should look for a chart numbered 9.2.0 or 9.0.) For more information on the organization of control flow information in this manual, refer to *Chart Numbering* in Section 2.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$ARSP	8.1	$ARSP	Autoresponse utility	Place autoresponse severity values and alert indicators into a message load member.
$BICDI	8.2.1	$BICDI	Basic exchange file display	Display a basic exchange diskette file on the SYSLIST device.
$BICET	8.2.2	$BICET	I-exchange diskette-to-disk	Copy (or add) records from an I-exchange diskette to a disk file.
$BICFI	8.2.3	$BICFI	Basic exchange disk to diskette	Copy (or add) records from a disk file to a basic exchange diskette file.
$BICIF	8.2.2	$BICIF	Basic exchange diskette to disk	Copy (or add) records from a basic exchange diskette file to a disk file.
$BICR	8.2.0	$BICR	Basic or I-exchange mainline	Control the copying or displaying of records from a diskette basic or I-exchange file.
$BICST	8.2.1	$BICST	I-exchange file display	Display an I-exchange diskette file on the SYSLIST device.
$BICTE	8.2.3	$BICTE	I-exchange disk to diskette	Copy (or add) records from a disk file to an I-exchange diskette file.
$BMENU	8.3	$BMENU	Build Menu utility	Build SFGR load module from a source menu.
$BUILD	8.4	$BUILD	Sector rebuild utility	Prompt for and correct bad disk sectors.
$CNAD	—	$CNAD1	Configuration utility: add support	Set configuration record update flags based on values set in system local data area.
$CNAD1	1.2.2	$CNAD1	Configuration utility: install program product support	Set current, add, drop, and release update flags in configuration record.
$CNAD2	1.2.2	$CNAD2	Configuration utility: install SSP features and optional SSP support	Prompt for and set flags in configuration member for changes to SSP features and optional SSP.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$CNAID	1.2.1 1.2.2	AIDBEG	System configuration support aids	Provide the following capabilities: • Condense or extend a specified library • Get the size of user available area and #LIBRARY. • Calculate number of diskettes required to back up specified library. • Calculate number of diskettes needed to save a library.
$CNAM0	1.2.4.1	CNAMBEG	Configuration for display station on remote controller	Provide for updating of device configuration for 5294 Remote Control Unit.
$CNDED	—	$CNDED	System configuration dedicated interface	Establish dedicated interface for system configuration.
$CNDP	1.2.4	DBEG	System configuration default printer assign	Prompt for and set default printers for each display station on the system.
$CNDRP	1.2.5	$CNDRP1	Configuration utility: drop support	Set configuration record update flags based on values set in the system local data area.
$CNDWS	1.2.4.1	$CNDWSO	System configuration utility: work station default assignment	Set default logical IDs, controller IDs and addresses, and unit addresses for local and remote work stations.
$CNFIG	1.2.1 1.2.5		System configuration utility: root phase	Set IPL defaults.
$CNF1	1.2.1	$CNF1	System configuration utility: main menu	Prompt for and set IPL defaults.
$CNHLP	1.2.1		System configuration help utility	Set up help text menu.
$CNMAT	1.2.4	$CNMAT	Configuration utility: modify work station attributes	Prompt for and set attributes assigned to a work station.
$CNMID	1.2.4	$CNMID	System configuration utility: modify work station logical IDs.	Perform for and set work station logical IDs, subconsole assignment, and default printer assignment.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$CNMSR	1.2.2 1.2.1	$CNMSR	System configuration utility: create, edit, or delete utility	Prompt for the creating, editing, or deleting of a configuration member.
	1.2.1	$CNUPT	System configuration update rebuild	Alter master configuration member.
$CNMUP	1.2.1	CNMUPBEG	Update master configuration record	Prompt for and make changes to master configuration record.
$CNNAP	1.2.4.1	NAPBEG	System configuration of 3262 Printer	Prompt for 3262 Printer configuration.
$CNPLN	1.2.2	$CNPLN1	System configuration: planning information	Present planning information display.
$CNPRT	1.2.3	$CNPRT	CNFIGSSP print function	Display configuration print menu and route for options selected.
$CNREM	1.2.4.1	CNRMBEG	Configuration for display station on remote controller	Provide for updating of device configuration for 3274 Remote Control Unit.
$CNRLD	1.2.2	$CNRLD	System configuration utility: modify work station attributes	Prompt for and set work station attributes.
$CNRWS	1.2.4.1	RWSBEG	Set remote line attributes	Set the following attributes: • Switch type. • Auto-reconnect. • Modem speed. • DDS adapter attached. • EIA interface.
$CNSA	1.2.4	SABEG	System configuration subconsole assign	Prompt for and set assignment of a subconsole for each printer on the system.
$CNSET	1.2.1	SETENTRY	Configuration: update LDA	Set flags in LDA based on changes to be made in configuration record.
$CNSKP	1.2.1	SKPENTRY	Configuration: update LDA	Set flags in LDA based on changes to be made in configuration record. Used for Kanji systems.
$CNSTK	1.2.1	STKENTRY	Configuration: update LDA	Set flags in LDA based on changes to be made in configuration record. Used for Kanji-preferred systems.
$CNSYP	1.2.4	SYSBEG	System configuration system printer assign	Prompt for and set assignment of system printer.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$CNSYS	1.2.2	$CNSYS	Configuration utility: system parameters	Prompt for and set system, print, and spooling features.
$CNTRT	1.2.2	CNTRT	Configuration: select 3270 device emulation translate tables	Select user-specified translate tables to be used by BSC or SNA 3270, 3278, and 3279 device emulation.
$CNWD	1.2.4.1	CNWDBEG	System configuration work station add, delete, or modify remote controllers	Define controllers and specify number of devices on each controller port.
$CNWD2	1.2.4.1	CNWD2BEG	Error processing for $CNWD	Check for controller errors. Update error flags.
$CNWD3	1.2.4.1	CNWD3BEG	System configuration work station add, delete, or modify remote controllers	Handle the OFF DROP prompt for $CNWD.
$CNWN	1.2.4.1	CNWNBEG	System configuration work station add, delete, or modify local or remote work stations and printers	Prompt for and set additions, deletions, and modifications to configuration data for local and remote devices.
$CNWO	1.2.4 1.2.4.1	$CNW01	System configuration utility: work station menu	Display option menu for editing work station portion of configuration member.
$CNWP	1.2.4	CNWP00	System configuration utility: generalized work station select routine	Provide the interface to the displayed screen for $CNSA.
$COADD	8.5.2	$COADD	High-speed add of disk file to diskette file	Add a disk file to an existing diskette file.
$COCRT	8.2.1 8.5.1 8.31.1	$COCRT	Display to the CRT	Interface between $COGET, $CORMT, or $TCTSL and work station data management.
$COGET	8.5.1.3	$COGET	Get/put interface with record-mode data management	Interface for record mode data management and/or $COCRT and $COPRT.
$COINT	8.5.1.1 8.5.1.2	$COINT	Record mode/remote file copy input initialization	Initialize files to be copied in record mode. Disk file may be local or remote.
$COOUT	8.5.1.2 8.5.1.3 8.5.1.4	$COOUT	Record mode/remote file copy output initialization	Prepare for output to disk, diskette, or display. Disk file may be local or remote.
$COPRT	8.2.1 8.5.1 8.31.1	$COPRT	Display to the printer	Interface between $COGET, $CORMT, or $TCTSL and printer data management.
$COPY	8.5.0	$COPY	Copy user file(s) utility	Mainline module of the $COPY utility. Invoke the syntax checker and route control to $COINT, $COADD, $COZFF, $COZRS, or $COZSV according to the function specified.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$CORMT	8.5.1.4	$CORMT	Remote file copy through direct interface with distributed data management	Get/put interface with both record mode data management and distributed data management.
$COZFF	8.5.3	$COZFF	Interface with sector mode data management for disk-to-disk copy	Control for disk-to-disk copy.
$COZRS	8.5.4	$COZRS	Interface with sector mode data management for diskette-to-disk RESTORE operation	Control for diskette-to-disk RESTORE operation.
$COZSV	8.5.5	$COZSV	Interface with sector mode data management for disk-to-diskette SAVE operation	Control for disk-to-diskette SAVE operation.
$CPMG	—	$CPMG	MSGFILE procedure	Display/update status of system message file.
$CPPE	8.6	$CPPE	Procedure interface to SYSLOG	Issue error message.
$CZUT	10.9.2.2	$CZUT	Alert utility	Change alert generation indicators in specified alertable messages.
$DACHK	7.1.8	DACHK	Diskette drive check	Perform track 0 read, worn head test, and alignment test.
$DALOG	7.1.8	$DALOG	Display diskette log	Display diskette log and allow operator update.
$DAMP	7.1.8	DAMP	Diskette utilities supervisor	Allocate the diskette drive, prompt for option selection, load and call program for option selected, and clean up after specified utility terminates.
$DARCV	7.1.8	$DARCV	Diskette sector recovery	Perform multiple reads on specified sector, saving any successful read data and informing user of error read. Multiple write back, with verification.
$DASCN	7.1.8	DASCN	Scan diskette for errors	Read verify all sectors on the diskette and print a summary report.
$DCOPY	8.37	DCOPY	Diagnostic diskette copy utility	Copy an entire diskette, regardless of content; sectors 3, 5, and 6 and cylinder 0 head 0 are not copied.
$DDST	8.7	$DDST	Key sort utility	Sort the keys of a specified file.
$DELET	8.8	$DELET	File/Library delete control module	Read input parameters, determine which function is to be performed, and load the proper module.
$DELF1	8.8	$DELF1	Delete files from fixed disk	Delete files from fixed disk.
$DELI1	8.8	$DELI1	Delete files from diskette	Delete files from diskette.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$DSBO	8.36.7	$DSBO	IDDU common open	Opens data dictionaries and file definitions within them.
$DUPFX	8.9	$DUPFX	Diskette copy utility sector fix transient	Process diskette read errors detected by diskette copy utility modules.
$DUPRD	8.9	$DUPRD	Diskette copy utility mainline	Perform mainline processing for diskette single-file or all-file copying.
$DUPIA	8.9	$DUPIA	Diskette all file input copy	Copy all files from diskette to scratch disk files.
$DUPOA	8.9	$DUPOA	Diskette all file output copy	Copy files from scratch disk files to diskette.
$DUP1F	8.9	$DUP1F	Diskette single file input copy	Copy specified file from diskette to common area.
$DUP2F	8.9	$DUP2F	Diskette single file output copy	Copy specified file from common area to diskette.
$ERAP	7.1.1	$ERAP	Error recording analysis procedure mainline	Format and display or print error history information for all system devices.
$ERA2	7.1.1	DEVFMT	ERAP 21ED disk drive error formatter	Put formatted 21ED disk drive SIO counts, error counts, and error history table information to requester display station or requested printer.
$ERA3	7.1.1	DEVFMT	ERAP 10SR disk drive error formatter	Put formatted 10SR disk drive SIO counts, error counts, and error history table information to requester display station or requested printer. If requested, reset SIO and error counts.
$ERCE	7.1.1	DEVFMT	ERAP local display station and printer error formatter	Put formatted local display station and printer SIO counts, error counts, and error history table information to requester display station or requested printer. If requested, reset SIO and error counts.
$ERD2	7.1.1	DEVFMT	ERAP diskette drive error formatter	Put formatted diskette drive SIO counts, error counts, and error history table information to requester display station or requested printer. If requested, reset SIO and error counts.
$ERD3	7.1.1	DEVFMT	ERAP diskette magazine drive error formatter	Put formatted diskette magazine drive SIO counts, error counts, and error history table information to requester display station or requested printer. If requested, reset SIO and error counts.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$ERE2	7.1.1	DEVFMT	ERAP line printer error formatter	Put formatted line printer SIO counts, error counts, and error history table information to requester display station or requested printer. If requested, reset SIO and error counts.
$ERSM	7.1.1	DEVFMT	ERAP summary formatter	Put formatted ERAP summary data to requester display station or requested printer.
$ER02	7.1.1	DEVFMT	ERAP control processor error formatter	Put formatted control storage processor error history table information to requester display station or requested printer.
$ER03	7.1.1	DEVFMT	ERAP main storage processor error formatter	Put formatted main storage processor error history table information requester display station or requested printer.
$ER6C	7.1.1	DEVFMT	ERAP local attachment (controller) formatter	Put formatted attachment SIO counts, error counts, and error history table information to requester display station or requested printer.
$ER68	7.1.1	DEVFMT	ERAP MLCA/ELCA error formatter	Put formatted MLCA/ELCA error history table information to requester display station or requested printer.
$ER8A	7.1.1	DEVFMT	ERAP remote work stations controllers error formatter	Put formatted remote work station controllers error history table information to requester display station or requested printer.
$ER8E	7.1.1	DEVFMT	ERAP remote display stations and printers error formatter	Put formatted remote display station and printer error history table information to requester display station or requested printer.
$ER80	7.1.1	DEVFMT	ERAP BSC formatter	Put formatted BSC SIO counts, error counts, and error history table information to requester display station or requested printer.
$ER81	7.1.1	DEVFMT	ERAP SDLC formatter	Put formatted SDLC SIO counts, error counts, and error history table information to requester display station or requested printer.
$ER82	7.1.1	DEVFMT	ERAP autocall formatter	Put formatted autocall SIO counts, error counts, and error history table information to requester display station or requested printer.
$ER83	7.1.1	DEVFMT	ERAP X.21 formatter	Put formatted X.21 SIO counts, error counts, and error history table information to requester display station or requested printer.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$FBAIB	4.11.10 4.11.11 8.5.4 8.10	$FBAIB	Alternative index build	Build an alternative index file from an existing physical file.
$FBLD	8.10	$FBLD	File build utility	Create a null disk or control the building of an alternative index file.
$FEACD	7.1.6	FEACD	Diagnostic aids setdump or address compare dump	Provide for user determined limits for debug-type dumps.
$FEAIDS	7.1.0	FESTARTZ	Diagnostic aids mainline	Initialize for diagnostic aids utilities and provide common subroutines:
		ABORT		Abnormally terminates a utility.
		BINTODEC		Converts binary to zone decimal.
		BINTOHEX		Converts binary to printable hexadecimal.
		DECTOBIN		Converts zone decimal to binary.
		HEXTOBIN		Converts printable hexadecimal to binary.
		DISKIO		Processes disk I/O requests.
		ENDIT		Terminates a utility.
		ERROR		Processes halts or SYSLOG calls.
		FETCH		Loads and passes control to a diagnostic aids utility.
		FORMAT		Formatting message/data merge function.
		FINDMIC		Retrieves a message.
		GENIO		Interfaces between utility and printer or work station.
		MOVEIT		Moves blocks of data.
		OVERLAY		Loads and calls overlay modules.
		PRINT		Processes print requests.
		RETURN		Returns to caller in push stack.
		SAVREGS		Saves caller's registers in push stack.
		WSIO		Processes work station I/O requests.
$FEALK	7.1.5	$FEALK	Diagnostic aids APAR utility 2	Copy data needed to diagnose a software problem from disk to diskette.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$FEAPR	7.1.5	$FEAPR	Diagnostic aids APAR utility 1	Create the following APAR files on diskette by copying data from disk: • APARFILE (task or stand-alone dump). • CONTFILE (control storage). • IOCFILE (I/O controller). • TRCEFILE (trace header file). • [name]FILE (trace file with user-specified name). • PTFFILE (#PTFLOGS for program products). • SYSFILE (configuration record, VTOC, trace work area and diskette work area). • MCDEFILE (microcode tables). • HISTFILE (history file). • SERVFILE (service log). • SPOLFILE (spool file). • JOBQFILE (input job queue file). • ERAPFILE (error logging tables). • CKPTFILE (checkpoint restart file).
$FEBTR	7.1.4	$FEBTR	Diagnostic aids batch trace utility	Create or modify batch environment trace tables.
$FECHK	7.2	$FECHK	PTF prerequisite checking	Process a PTF file to determine if all prerequisites are present.
$FECNT	7.1.0	FECNT	Diagnostic aids router	Read utility control statements, set up any required diskette processing, and load and pass control to requested function.
$FECNV	7.1.1	FECNV	Diagnostic aids conversion routines	Provide conversion routines for use by diagnostic aids programs:
		TUTOD		Convert timer units to time of day.
		TODTU		Convert time of day to timer units.
		MULTIPLY		Multiply two numbers.
		DIVIDE		Divide two numbers.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$FECON	7.1.2.8	FECON	Diagnostic aids control storage dump utility	Dump control storage from either the control storage library dump area or from a disk or diskette dump file.
$FECRE	—		Dump file analysis utility 1	Build a DFA file.
$FEDFA	7.1.7	$FEDFA	Diagnostic aids dump file analysis	Format dump selected areas of main, control, and translated storage control blocks from either a task or a stand-alone dump file.
$FEDSK	7.1.2.2 7.1.3	FEDSK	Dump disk/diskette and patch	Dump disk or diskette storage and process user modification of the data.
$FEDSP	—	FEDSP	Dump file analysis utility 2	Display the DFA file.
$FEDTR	7.1.2.1 7.1.2.10	$FEDTR	Dump trace file utility	Print or display a system trace file's content in a formatted dump.
$FEFIX	7.3	FEFIXMNL	PTF utility	Apply program temporary fixes to system programs.
$FEIOC	7.1.2.5	$FEIOC	Diagnostic aids I/O controller storage dump utility	Dump I/O control storage either from control storage dump area or from a stand-alone disk or diskette dump file.
$FEKEY	—	FEKEYMNL	F.E. CRT window display	Display on the CRT screen an 80-byte window of a block of data passed to it.
$FEMCD	7.1.2.7	$FEMCD	Diagnostic aids microcode tables dump utility	Dump the following from either the control storage library or from an APAR diskette: • Prerequisite list. • Microcode level table. • Controlled patch table. • Free lance patch table.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$FEMNT	7.1.2.1 7.1.7 7.1.5 7.1.0	$FEMNT	Dump file maintenance	Perform the following multiple-dump-file-maintenance functions: • Allocate the existing dump file specified by the caller. • Find the last-created task dump file. • Display status of all existing dump files. • Display status of all existing dump files and allow the user to select of one of them.
$FEPDD	7.1.2.4	$FEPDD	Dump product level data	Print or display a product level file from a disk or from an APARFILE on disk.
$FEPRM	—	$FEPRM	PTF remove utility	Provide the capability to remove PTFs.
$FEPRO	7.1.4	FESTART	Standard profile retrieve routine (RAS overlay 1)	Retrieve trace profiles and copy them to a $FEPRO parameter list.
$FEPR1	7.1.4	FESTART	Standard profile retrieve routine (RAS overlay 2)	Retrieve trace profiles and copy them to a $FEPRO parameter list.
$FEPTF	7.1.2.9	FEPTF	Diagnostic aids PTF log dump	Print or display a PTF log from disk or from a diskette APAR file.
$FESBT	—	FESBT	Decoding tables for SB type field	Return the 5-character label for the SB type whose address is in POINT@1.
$FESI1	7.1.0 7.1.2.1 7.1.2.3 7.1.2.4 7.1.2.5 7.1.2.6 7.1.2.7 7.1.2.8 7.1.2.9 7.1.2.10 7.1.2.11 7.1.7	FESI1	Diagnostic aids copy diskette APAR file to disk	Copy an APAR diskette file to a resident or temporary disk file.
$FESRV	7.1.2.6	FESRV	Diagnostic aids dump service log utility	Print or display system service log file either from current log or from an APAR diskette file.
$FESTR	7.1.2.1	$FESTR	Main storage dump utility	Dump main storage, virtual storage, the control storage tables and trace tables/files from a task dump file or a stand-alone dump file on disk to a printer or display station.
$FESVC	—	$FESVC	Trace event descriptors (RAS overlay)	Provide a table for decoding SVCs, component IDs, and control storage trace events.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$FESYS	7.1.2.3	$FESYS	System area dump utility	Format dump the following system areas either from storage or from an APAR diskette: • Disk VTOC. • Task work area (TWA). • Input job queue. • Spool files.
$FETAP	7.1.2.12	$FETAP	Diagnostic aids dump tape utility	Dump contents of a tape volume to the specified printer.
$FETI1	7.1.2.1 7.1.2.3 7.1.2.4 7.1.2.5 7.1.2.6 7.1.2.7 7.1.2.8 7.1.2.9 7.1.2.10 7.1.2.11 7.1.7	$FETI1	Diagnostic aids copy tape APAR file to disk	Copy a tape APAR file to a resident or temporary disk file.
$FETRC	7.1.4	$FETRC	Diagnostic aids trace utility	Display events currently being traced and prompt for any changes. Pass new events to control storage.
$FREX	8.20	$FREX	Free disk space: cleanup	Unsuspend command processor, update security and spool file, and cross-reference library.
$FREE	8.20	$FREE	Free disk space: phase 1	Interface with syntax checker to read utility control statements.
$FRE0	8.20	$FRE0__00	Free disk space: initialization	Initialize work space and map the used and free disk blocks into main storage as a bit mask.
$FREQ	8.20	$FREQ__00	Table sort routine	Sorts the table entries based on start of allocated space.
$FRE1	8.20	$FRE1__00	Free disk space: high	Accumulate free space at high addresses of spindle by moving files to low addresses.
$FRE2	8.20	$FRE2__00	Free disk space: low	Accumulate free space at low addresses of spindle by moving files to high addresses.
$FRE3	8.20	$FRE3__00	Free disk space: cleanup	Reset format-5 entries to indicate free space.
$HELP	8.11	$HELP	System/36 help utility	Prompt the user for the parameters on an SSP procedure.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$HETO	8.11	$HETO	System/36 OCL help table	Create tables for $HELP.
$HETP	8.11	$HETP	System/36 PCE help table	Create tables for $HELP.
$HET1	8.11	$HET1	System/36 help table 1 for procedures A-C	Create tables for $HELP.
$HET2	8.11	$HET2	System/36 help table 2 for procedures D-F	Create tables for $HELP.
$HET3	8.11	$HET3	System/36 help table 3 for procedures G-O	Create tables for $HELP.
$HET4	8.11	$HET4	System/36 help table 4 for procedures P-R	Create tables for $HELP.
$HET5	8.11	$HET5	System/36 help table 5 for procedures S	Create tables for $HELP.
$HET6	8.11	$HET6	System/36 help table 6 for procedures T-Z	Create tables for $HELP.
$HIST	8.12	$HIST	History file display utility	Display contents of history file to the SYSLIST device.
$HSSCRL	8.12	$HSSCRL	History file scroll	Provide forward and backward scrolling capabilities in user copy or system history file.
$ICFA	1.3.0 1.3.2 1.3.3	$ICFB	SSP-ICF subsystem parameter utility	Set parameters supplied by operator responses in the configuration member.
$ICFB	1.3.1	ENTRY	SSP-ICF line parameter configuration	Set up line information for the selected subsystem in the SSP-ICF member data set.
$ICFC	1.3.2	$ICFC	SSP-ICF BSC general configuration parameters	Display and edit BSC subsystem parameters.
$ICFF	1.3.3	$ICFF	SNUF subsystem location parameters	Display SNUF location screen information.
$ICFH	1.3.2	$ICFH	SSP-ICF BSC multipoint session address configuration	Display and edit parameters applicable to multiple session BSC subsystems.
$ICFIG	1.3.0	LDPOINT	SSP-ICF subsystem configuration utility	Set up SSP-ICF subsystem configuration data sets in the user-specified library.
$ICFJ	1.3.2	$ICFJ	SSP-ICF CCP general configuration parameters	Display and edit parameters applicable to the SSP-ICF CCP subsystem.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$ICFK	1.3.2	$ICFK	IMS subsystem PTERM (physical terminal) configuration	Set up PTERMs (physical terminals) for the IMS/IRSS subsystem.
$ICFLU	1.3.1	$ICFLU	SSP-ICF logical unit table	Create, edit, or review line logical unit table.
$ICFMS	1.3.0	$CNFIG	SSP-ICF system configuration utility	Set up for configuration requested, write config member to disk, route for further CNFIGICF processing.
$ICFN	1.3.3	$ICFN	Display Peer subsystem location configuration parameters	Display Peer subsystem location parameters.
$ICFO	1.3.2	$ICFO	SSP-ICF 3270 general configuration parameters	Display and edit parameters applicable to BSC 3270 emulation subsystem.
$ICFP	1.3.2	$ICFP	SSP-ICF 3270 device configuration parameters	Display and edit parameters applicable to BSC 3270 device emulation.
$ICFQ	1.3.3	$ICFQ	SSP-ICF SNA 3270 device parameters	Display 3270 device parameters.
$ICFR	1.3.1	ENTRY	SSP-ICF line attributes for remote system configuration	Define the remote system in the line configuration member.
$ICFS	1.3.2 1.3.3	$ICFS	SSP-ICF remote system and location parameter configuration	Get remote system and location parameters for BSC MSRJE and SNA subsystems.
$ICFT	1.3.3	$ICFT	SSP-ICF Finance subsystem location parameters	Display Finance subsystem location parameters.
$ICFW	1.3.0	$ICFW	Write configuration member	Write configuration member for current library.
$IC34	1.3.2 1.3.3	$IC34	Display MSRJE location parameters	Display screen 34.0 (MSRJE location parameters).
$IC35	1.3.2 1.3.3	$IC35	Display MSRJE logon parameters	Display screen 34.0 (MSRJE logon parameters).
$IC36	1.3.2 1.3.3	$IC36	Display printer/punch parameters	Display screen 36.0 (MSRJE printer/punch parameters).
$IC37	1.3.2 1.3.3	$IC37	Display MSRJE reader parameters	Display screen 37.0 (MSRJE reader parameters).
$IC41	1.3.3	ENTRY	SSP-ICF APPC location definition	Prompt user to define APPC location.
$IC43	1.3.3	$IC43	SSP-ICF APPC location definition	Prompt user to define APPC location.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$IDSET	8.13	$IDSET	SSP-ICF DEFINEID utility	Route for DEFINEID syntax checking system file maintenance.
$IDSM	8.13	$IDSM	SSP-ICF DEFINEID module	Maintain remote ID list on a disk file for SSP-ICF system.
$IEDS	10.3.2	$IEDS	Disable an SSP-ICF subsystem	Terminate an SSP-ICF subsystem task.
$IENBL	10.3.1	$IENBL	Initialize SSP-ICF subsystem	Build required control blocks and give specified SSP-ICF subsystem control.
$IFDEB	10.5.1	$IFDEB	SSP-ICF debug mainline	Execute as a work station SRT to activate, deactivate, or display debug trace file.
$IFTR	10.5.1	$IFTR	SSP-ICF debug trace routine	Log SSP-ICF trace entries to disk.
$INIT	8.14	$INIT	Diskette initialization routine	Format, rename, or delete files from a diskette.
$IRERD	—	$IRERD	SSP-ICF APPC error data handler	Process data errors detected by APPC subsystem.
$IRMAP	—	$IRMAP	SSP-ICF APPC mapped conversion support	Process APPC subsystem get/put mapped data.
$IV01	10.5.2	$IV01	SSP-ICF BSCEL installation verification	Verify installation of SSP-ICF BSCEL subsystem by sending/receiving a record.
$IV02	10.5.2	IVGO	SSP-ICF CCP installation verification	Verify installation of SSP-ICF CCP subsystem by testing for working link.
$IV03	10.5.2	$IV03	SSP-ICF CICS/VS installation verification	Verify installation of System/36 SSP-ICF subsystem and remote host CICS/VS system by testing for working link.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$IV04	10.5.2	$IV04	SSP-ICF IMS/VS installation verification	Verify installation of System/34 subsystem and remote IMS/VS host by testing for working link.
$IV05	10.5.2	$IV05	SSP-ICF installation verification for BSC 3270 subsystem	Verify installation of SSP-ICF BSC 3270 subsystem.
$LABEL	8.15 8.34.4	LABSTART	Display VTOC initialization routine	Initialize and route for diskette or disk VTOC display.
$LABF1	8.15 8.34.4	LABF1	Display disk VTOC routine	Display on the SYSLIST device the disk VTOC entries.
$LABI1	8.15	$LABI1	Display diskette VTOC routine	Display on the SYSLIST device the diskette VTOC entries.
$LABST	8.15	$LABST	Disk VTOC sort routine	Sort the disk VTOC entries by location or by name.
$LATLC	8.15	LABSTART	Tape label display routine	Display tape header label information on SYSLIST device.
$LATNM	8.15	LABSTART	Tape label sort routine	Retrieve and sort active tape entries.
$LSLUD	10.6.11.2	$LSLUD	APPN location utility	Dump location information to disk.
$LSSTU	10.6.11.2	$LSSTU	APPN session utility	Read session data that is logged by C/SNA.
$LSTDB	10.6.11.2	$LSTDB	APPN network utility	Dump network information to disk.
$MACHG	8.16.14	$MACHG	Library directory entry change routine	Change library member name, subtype, and/or reference number in the member's directory entry.
$MACK	8.16.1	$MACK	Save library	Save a library on a diskette.
$MACMP	6.1.7 6.1.9 8.16.1 8.16.3 8.16.4 8.16.14	$MACMP	COMPACTOR	Compact active directory entries in ascending order within library directory.
$MACOM	6.1.5 6.1.6	$MACOM	Open/close routine	Check open/close request.
$MADLT	8.16.0	$MADLT	Delete routine	Delete entries unless ALL is specified.
$MADSP	8.16.4	$MADSP	Library print routine	Print library directory entries, library members, and system information.
$MAFIR	8.16.7 8.16.11	$MAFIR	Library directory fast insert routine	Place stack of directory entries into their respective alphabetic locations.
$MAFND	6.1.3	$MAFND	Librarian find routine	Locate library directory entries by type(s) and full or partial name.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$MAILD	6.1.9 8.16.14	$MAILD	Library directory insert routine	Place directory entry into alphabetic location within library directory.
$MAINT	8.16.0	$MAINT	Library maintenance mainline routine	Use syntax checker routine, #USYX, to process control statement from SYSIN.
$MALAM	6.1.1.0	$MALAM	Library access method routine	Perform mainline function for library access method.
$MALCO	6.1.5 6.1.6	$MALCO	Open/close module	Check open/close requests.
$MALFN	6.1.3 8.16.4	$MALFN	Find routine	Locate directory entries by type(s) and full or partial name.
$MALIL	—	$MALIL	Library directory insert	Place directory entry into alphabetic location within library directory.
$MALOC	8.16.0	$MALOC	Library allocate module	Allocate user libraries, and change directory or member size of a library.
$MALTL	8.16.7	$MALTL	Library-to-library copy	Copy library member(s) from library to library.
$MAPGS	6.1.6	$MAPGS	Library sector get/put routine	Place modules in sector mode from library and pass them.
$MAPTF	6.1.6	$FETRC	PTF log handler	Get, put, update, and delete entries in the PTF log.
$MAPUR	6.1.5	$MAPUR	Library record put routine	Place source or procedure records into the library.
$MARAL	8.16.13	$MARAL	Library reallocate module	Reallocate libraries with extents, change library, size, or change directory size.
$MARCK	8.16.3 8.16.13	$MARCK	Library compress driver	Remove gaps in the member space of the requested library.
$MARDR	8.16.12	$MARDR	Librarian from reader driver routine	Put records from SYSIN device in requested library.
$MARES	8.16.2.2	MARESMNL	Main storage nucleus initialization program	Initialize main storage nucleus.
$MARFF	8.16.9	$MARFF	Copy from file to library in record mode routine	Copy source and procedure members.
$MARPF	8.16.8	$MARPF	Library copy from file to print in record mode	List member type and member names in record mode file.
$MARS	8.16.2.1	$MARS	Restore libraries routine	Restore user library or #LIBRARY if it has been saved.
$MARTF	8.16.5	$MARTF	Copy from library to file in record mode routine	Copy members.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$MAR2K	8.16.3	$MAR2K	Update member pointers after library compress	Perform necessary updates after a library compress.
$MAR3K	8.16.2.2 8.16.3	$MAR3K	Suspend command processor	Issue suspend for command processor TCB.
$MASDF	8.16.10	$MASDF	Print sector mode librarian file	List directory information for members in System/32 or System/34 sector mode files.
$MASPC	—	$MASPC	Specification module for library utilities	A table of specification values used by syntax checker routine (#USYX).
$MATDS	8.16.10	$MATDS	Copy from file to printer in sector mode routine (tape)	List directory information for library members in sector mode tape files.
$MATFS	8.16.6	$MATFS	Copy from library to file in sector mode routine	Copy members.
$MATLS	8.16.11	$MATLS	Copy from file to library in sector mode routine	Copy members.
$MAXNT	—	$MAXNT	Update transfer table	Update system transient transfer table
$MGBLD	8.17	@MASYG	Message build utility	Create a message load member in a disk library.
$MMSP	8.18	$MMSP	Stop automonitor	Terminate the automonitor function.
$MMST	8.19	$MMST	Start monitor	Initiate automonitor function.
$MNDF	—	$MNDF	Multinational data file conversion	Convert national character set data to multinational character set or vice versa.
$MNML	—	$MNML	Multinational mainline	Convert source or procedure member data to multinational character set data or vice versa.
$PACK	8.20	$PACK__00	Free disk space: compatibility phase	Provide compatibility with System/34 free disk space; performs basic $FREE function.
$PDST	9.7	$PDST	Mainline link station test	Use the SDLC test command to verify that secondary SDLC stations are operational.
$PNLM	8.21	$PNLM	DEFINEPN phone list utility	Build or modify a load member of phone numbers for use by the autocall task.
$POSGI	8.22	$POSGI	Special E-format diskette to disk	Copy a special E-format diskette file to disk.
$POST	8.22	$POST	5260 diskette copy: mainline	Copy diskette files processed by the 5260 point-of-sales terminal to disk files.
$PRCED	8.23.1.12	$PRCED	Security location profile edit utility	Update, add, delete, and/or locate location profiles in the user ID file.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$PRCVT	8.23.1.16	$PRCVT	System/34 to System/36 conversion utility	Convert System/34 password and/or security file to System/36 user ID file and/or security file.
$PRLST	8.23.1.5	$PRLST	User profile list utility	List user ID file to the SYSLIST device.
$PRUED	8.23.1.2	START	User profile edit utility	Edit user profiles.
$PRUID	8.23.1.1	$PRUID	Security user ID utility	Control user ID file, password security, and badge security.
$PRURS	8.23.1.4	$PRURS	User ID file restore utility.	Restore user ID file on disk.
$PRUSV	8.23.1.3	$PRUSV	User ID file save utility	Save user ID file on disk.
$RENAM	8.24	$RENAM	Rename utility	Rename files/libraries on disk.
$RREDT	8.23.1.8	START	Resource security file edit utility	Edit file or user records.
$RRESC	8.23.1.7	$RRESC	Resource security definition utility	Control resource security.
$RRLST	8.23.1.11	$RRLST	Resource file list utility	List resource security file to the SYSLIST device.
$RRSAV	8.23.1.9	$RRSAV	Resource file save utility	Save resource security file on disk.
$RRSTR	8.23.1.10	$RRSTR	Resource file restore utility	Restore as the current resource security file a saved copy on disk.
$RRTED	8.23.1.14	$RRTED	Resource file edit utility	Edit file for folder and folder members.
$RRTLT	8.23.1.15	$RRTLT	Resource security file list utility for folders	List information about folders from the resource security file.
$SETCF	8.25	SET00010	Work station configuration utility	Update the work station communications configuration records.
$SETCP	8.26	SET00000	Communications configuration utility	Update the system communications configuration record.
$SFDEP	8.27	$SFDEP	Screen format diagnose and print routine	Diagnose and print source input specifications.
$SFDSP	—	$SFDSP	Screen format diagnose S entry and print routine	Diagnose and print S record errors.
$SFFDB	8.27	$SFFDB	Screen format FDT and data stream build routine	Build the data stream and FDT entries (if required) for each format.
$SFGR	8.27	$SFGR	Screen format generator mainline	Perform mainline processing for screen format generation.
$SFLMM	8.27	$SFLMM	Screen format load member maintenance	Add, replace, or delete screen formats in a load member.
$SFRSQ	8.27	$SFRSQ	Screen format resequence routine	Sort the data stream input field into assending sequence.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$SFSPC	—	$SFSPC	START	A table of specification values used by the syntax checker routine for parsing control statements.
$SINA	8.34	$SINA	NRD create	Allocate a NRD file as a new file.
$SINE	8.34	$SINE	NRD edit	Prompt for and make requested changes to an NRD file.
$SINCT	8.34.2	$SINCT	NRD conversion	Convert System/34 NRD file to System/36 NRD file or restore a System/36 user NRD file to the System/36 system NRD file.
$SINDL	8.34.3	$SINDL	NRD delete	Delete an NRD file from disk.
$SINL	8.34.4	$SINL	NRD list	List entries in the NRD file.
$SINR	8.34	$SINR	NRD edit mainline	Initialize and route for NRD file maintenance.
$SLFL	—	$SLFL	Setfile utility	Provide procedure compatibility upward from System/34.
$SMFLG	7.3.2	$SMFLG	SMF utility report writer	Print an SMF report.
$SMFMC	—	$SMCSTART	System measurement utility for MLCA/ELCA	Collect communications line data at specified time intervals for MLCA/ELCA communications.
$SMFML	7.3.1	SS00010	SMF utility	Initialize for data collection phase.
		SM00010		Collect SMF data at specified time intervals.
		SC00010		Collect communications line data at specified time intervals for non-MCLA/ELCA communications.
$SMFTT	7.3.1	$SMFTT	SMF task data cleanup routine	Cleanup for SMF termination.
$SMFXT	7.3.1	$SMFXT	SMF EOJ routine	Perform end-of-job processing for SMF mainline routine.
$TCFDT	8.31.2	$TCFDT	Disk-to-tape copy module	Copy disk file to a tape file, or add disk file to an existing tape file.
$TCOPY	8.31.0	$TCOPY	Tape copy utility	Copy files between tape and disk or SYSLIST device.
$TCTFD	8.31.3	$TCTFD	Tape-to-disk copy module	Copy tape file to a disk file or add tape file to an existing disk file.
$TCTSL	8.31.1	$TCTSL	Tape display module	Display the contents of a tape file on the SYSLIST device.
$TINIT	8.30	$TINIT	Tape initialization utility	Initialize tape.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$TMSERV	8.35	$TMSERV	FMS procedure interface routine	Read and process FMS utility control statements.
$UASC	8.28	$UASC	Display spool file entries utility	Display spool file entries placed in a user file by $UASF.
$UASF	8.29	$UASF	User access to spool file utility	Copy compressed spool file to a user file in expanded format.
$XNLM	9.4.3	$XNLM	X.21 phone list load-module builder	Build or change an X.21 telephone list load member.
$XNSH	9.4.4	$XNSH	X.21 SHM line configuration load module builder	Create, update, print, or remove an X.21 SHM line configuration.
$XREST	8.32	$XREST	Extended character file restore utility	Restore the extended character file from a diskette file.
$XSAVE	8.33	$XSAVE	Extended character file save utility	Save the extended character file to a diskette file.
$XTAL	9.8.4.3	$XTAL	X.25 phone list utility	Build or modify a list of phone numbers for use by an X.25 task.
$XTCD	9.8.4.1	$XTCD	Delete member	Delete member during virtual circuit configuration.
$XTCE	9.8.4.1	$XTCE	Edit/update configuration	Edit or update during virtual circuit configuration.
$XTCF	9.8.4.1	$XTCF	X.25 configuration mainline controller	Route for X.25 configuration main menu input.
$XTCM	9.8.4.1	$XTCM	Main menu screen processor	Edit the main menu input.
$XTCN	9.8.4.1	$XTCN	Configuration network and logical channels	Edit network and logical channel data.
$XTCP	9.8.4.1	$XTCP	Print network and logical channels	Print the contents of the configuration record under format.
$XTCR	9.8.4.1	$XTCR	Format and print virtual circuit configuration	Format and print configuration during virtual circuit configuration.
$XTCS	9.8.4.1	$XTCS	Library member select	Select library member for virtual circuit configuration.
$XTCV	9.8.4.1	$XTCV	Virtual circuit controller	Route for virtural circuit menu input.
$XTDS	9.8.4.2	$XTDS	Members-with-tariff totals selection list	Control the storage and selection of a virtual circuit member name for members with a tariff file.
$XTLC	9.8.4.2	$XTLC	Logical channel status	Control the storage and display of the logical channel status for all subscribed channels.
$XTMS	9.8.4.2	$XTMS	Active-member selection list	Control the selection and display of a virtual circuit's member name from active member in main storage.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
$XTMT	9.8.4.2	$XTMT	X.25 maintenance utility mainline	Display and control maintenance option menu. Allocate, open, and close file and device DTFs.
$XTWQ	9.8.4.2	$XTWQ	Call wait queue status	Control the storage and display of the call wait queue status for all active virtual circuit members.
#AYAX	10.6.11	#AYAX	Asynchronous subsystem	Acts as an interface between all System/36 functions and the asynchronous subsystem.
#AYML	10.6.11	#AYML	Asynchronous subsystem (X.25)	Acts as an interface between all System/36 functions and the asynchronous subsystem.
#AYPM	—	#AYPM	Line and task open routine	Perform enable-time processing or open functions for the subsystem. Allocate and initialize resources.
#BSCL	9.1.4		BSC close	Close BSC files. If called by termination at disable time, free BSC I/O area, and detach BSC task if there are no other users.
#BSCM	9.1.3 9.1.1 9.1.2.1 9.1.4	—	BSC mainline	Process all BSC data management requests and perform line initialization.
	—	#BSAUTO	Receive-initial initiator	Set up IOB and issue SVC for receive-initial command. Start automatic response.
	9.1.1	#BSCKRI	Check receive-initial command	Check for pool or address response with NAK, EOT ACKO; start 2-second time-out. Start another receive-initial command after NAK or EOT.
	—	#BSHB	Operation and processor	Analyze operation end interrupts and (1) give control to error recovery routines, or (2) set good completion code in IOB.
		#BSIO	BSC IOS subroutine	
	—	#BSIW	BSC wait	Wait for I/O request to complete.
	9.1.1	#BSLI	Line initialization routine	
	9.1.1	#BSMD	BSC message handler	Issue messages via SSP SYSLOG for batch BSC.
		#BSMPEOT	Send EOT subroutine	
	9.1.1	#BSPOST	BSC post	Post data management after user request is processed.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#BSDB	9.1.1 9.1.2.1 9.1.2.2 9.1.4	—	BSC data management subroutines	Contain BSC resident data management subroutines required during the execution of a BSC task.
		#BSCP	Blank compression subroutine	
		#BSDM	BSC data management mainline	Perform router function for #BSDB subroutines.
		#BSEXIT	Exit-to-user NSI subroutine	
		#BSIW	BSC wait	Wait for I/O request to complete.
		#BSMG	GET deblocking subroutine	
		#BSMO	Data management move subroutine	
		#BSMP	PUT blocking subroutine	
		#BSPOST	BSC post	Post BSC (main storage) interrupt handler to perform I/O operations.
		#BSXP	Blank expansion subroutine	
#BSER	9.1.4	BSER	BSC abnormal termination	Handle BSC task processor checks.
#BSL0	9.1.1	#BSLO	Line initialization	Determine line type; if multipoint, return to #BSCM; if other, call appropriate transient.
#BSL2	9.1.1	#BSL2	Switched line calling initialization	Initialize manual or autocall line.
#BSL3	9.1.1	#BSL3	Switched line answer line initialization	Initialize manual or autoanswer BSC line.
#BSL4	9.1.1	#BSL4	Point-to-point nonswitched line initialization	Initialize for ENQ/ACK0 exchange.
#BSMD	9.1.1 9.1.3 9.1.4	#BSMD	BSC message handler	Issue BSC messages via SYSLOG.
#BSOB	9.1.1	#BSOB	BSC open	Open BSC DTFs on the preopened DTF chain.
#CABC	3.2.4.1 3.2.4.2 3.2.1	#CACM	BSC data communications allocate	Allocate communications DTFs.
		#CAIC	Communications task allocate/deallocate	Attach communications data management task, allocate desired line, and set up CSB and ACE for each DTF.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CADV	—	N/A	Device allocate table resident	Provide IPL with the device allocate table framework.
#CAD1	3.2.3	#CAD1	Deallocate mainline	Delete/deallocate disk files.
#CAD2	3.2.3	#CAD2	Deallocate device processor	Deallocate the diskette drive and printers.
#CAD3	3.2.3	#CAD3	Deallocate file processor	Perform required processing for deallocating a file: free space, keysort, and clean up reserve area.
#CAF1	3.2.1 3.2.2	#CAF1	Disk file allocate	Set up the amount of space to get in the format-5 area for new files. Get required reserve area space.
#CAF3	3.2.1 3.2.2	#CAF3	Disk file allocate	Calculate disk file capacities and format the disk space.
#CAMG	3.2.1 3.2.2 3.2.3 3.2.4.1 3.2.4.2	#CAMG	Allocate message processor	Issue messages and halts for allocate.
#CAMI	3.2.2 3.2.3	#CAMI	Allocate multiple indexed processing	Perform the building and removing of multiple index family chains.
#CAML	3.2.1 3.2.4.1	#CAML	Allocate mainline	Allocate devices and perform router processing for allocate modules.
#CAM2	3.2.1 3.2.2	#CAM2	Disk file allocate	Complete allocate for non-output files; initialize control blocks for new disk files.
#CAPT	3.2.1 3.2.2	#CAPT	Printer allocate	Allocate spooled and nonspooled printer files.
#CARW	3.2.4.2	#CARW	SDLC line allocate/ deallocate	Allocate/deallocate a line for an SDLC task.
#CAR1	3.2.1	#CAR1	Diskette file allocate	Allocate diskette files.
#CAS1	3.2.2	#CAS1	Special allocate mainline	Allocate disk files and perform router processing for allocate modules.
#CATL	3.2.1	#CATL	Tape allocate	Allocate for standard label (SL) tape file processing.
#CATM	3.2.1	#CATM	Tape allocate	Allocate for nonstandard label (NS), nonlabeled (NL), and bypass-label tape file processing.
#CATP	3.2.1	#CATP	Tape allocate	Allocate for standard label (SL) tape file put processing.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CATS	3.2.1	#CATS	Tape allocate	Allocate for nonstandard label (NS) tape file processing.
#CCAS	2.3.1	#CCAS	ASSIGN command	Process ASSIGN command.
#CCAT	—	#CCAT	ASSIGN command	Process ASSIGN command.
#CCCM	2.3.2 2.7	#CCCM	CANCEL command	Process CANCEL command and inquiry cancel.
#CCCO	2.3.2 2.12	#CCCO	CONSOLE command phase 1	Reassign system console when an I/O error occurs at the system console.
#CCCP	2.3.2	#CCCP	Spool CANCEL PRT command	Remove spool entries from the spool file.
#CCGP	2.3.2	#CCGP	Spool CHANGE command	Change spool file entries and spool writer attributes.
#CCHM	2.9	#CCHM	Help text for user menus	Display user-specified help text or default general help text.
#CCID	2.3.1	#CCID	INFOMSG command	Process INFOMSG command.
#CCJC	2.3.2	#CCJC	Job queue management	Perform CANCEL JOBQ, CHANGE JOBQ, and CHANGE JOBS commands.
#CCJH	2.3.2	#CCJH	Job queue management	Change the status of a job, based on the HOLD or RELEASE JOBQ commands
#CCJQ	2.3.1 2.16	#CCJQ	JOBQ command (input)	Process JOBQ command.
#CCJS	2.3.2	#CCJS	START JOBQ command (input)	Process START JOBQ and STOP JOBQ commands.
#CCMG	2.3.1	#CCMG	MSG command phase 1	Process MSG command diagnostics and queuing.
#CCMQ	2.11	#CCMQ	Message file manager	Process message to/from the system message file.
#CCMU	2.2 2.3.1 2.9 2.16	#CCMU	MENU command processor	Process MENU command.
#CCMX	2.7	#CCMX	Message display	Display messages at work station.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CCOF	2.3.1 2.6	#CCOF	SIGN-OFF/MODE command	Process OFF and MODE command.
#CCPP	2.3.2	#CCPP	Spool STOP PRT command	Stop the spool writers.
#CCPS	2.3.2	#CCPS	Spool START PRT command	Start the spool writers.
#CCPT	2.3.1	#CCPT	Spool RESTART PRT command	Restart the spool writers.
#CCPW	2.3.1	#CCPW	Power OFF command	Power off the system after clearing all active jobs.
#CCPY	2.3.1	#CCPY	PRTY command	Change or set a job's priority.
#CCRE	2.3.1	#CCRE	REPLY command	Process REPLY command to system messages.
#CCRG	—	#CCRG	Reply command for STATUS MESSAGE	Process REPLY command.
#CCRJ	2.3.3	#CCRJ	Command processor MSRJE status routine	Display active MSRJE locations.
#CCRS	—	CRS00000 CRS00100	Display second-level message text	Retrieve second-level text.
#CCRT	2.3.2 2.5	#CCRT	STOP/START command	Process START and STOP commands.
#CCSC	—	#CCSC	Command processor communications status routine	Display the status of the communications configuration record.
#CCSG	2.3.3	#CCSG	STATUS MESSAGE command handler	Allow system operator to view subconsole messages.
#CCSJ	2.3.3	#CCSJ	STATUS JOBQ command	Display JOBQ status.
#CCSM	2.3.3 2.7 2.9 2.13	#CCSM	STATUS mainline	Route control to proper status module.
#CCSP	2.3.3	#CCSP	Spool STATUS PRT command	Display spool file status.
#CCSQ	2.3.3	#CCSQ	Spool STATUS WRT command	Display spool writer status.
#CCSS	2.3.3	#CCSS	STATUS SESSION command	Display session information (pages 1 and 2).
#CCST	2.3.3	#CCST	STATUS SYSTASK routine	Display active tasks.
#CCSU	2.3.3	#CCSU	STATUS USERS command	Display active user task information.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CCSW	2.3.3	#CCSW	STATUS WORKSTN command	Display device status.
#CCSX	2.3.3	#CCSX	STATUS WORKSTN command segment 2	Build work station information for segment 2 of status work station.
#CCS2	2.3.3	#CCS2	STATUS SESSION command #2	Display session information to the requesting display station (pages 3, 4, and 5).
#CCTD	2.3.1	#CCTD	TIME command	Return time of day and system date.
#CCTU	—	#CCTU	Subconsole TUB builder	Build TUBs for subconsoles.
#CCVA	2.3.1	#CCVA	VARY command	Vary operator-specified devices on or off.
#CDCP	—	#CDCP	Loader for concurrent DCP main routine	Load and initialize concurrent DCP.
#CDCV	—	#CDCV	Concurrent DCP conversions	Perform conversions.
#CDKT	—	#CDKT	Concurrent DCP diskette	Handle diskette operations.
#CDMN	—	#CDMN	Concurrent DCP: mainline	Provide the dialog through which the CE/CSR may invoke and control various main storage diagnostic programs.
#CDPM	—	#CDPM	Concurrent DCP primary menu	Show the concurrent DCP primary menu.
#CDSK	—	#CDSK	Concurrent DCP disk	Locate a module in the disk C/S library or load a program from disk into M/S.
#CDWS	—	#CDWS	Concurrent DCP work station	Process display requests.
#CHHT	2.9	#CHHT	Help text	Initially process Help key requests.
#CHID	—	#CHID	Help text lookup	Retrieve help text format.
#CHIN	2.8 2.9	#CHIN	Help input processor	Process help menu item selected.
#CHKY	—	#CHKY	Help keyword processor	Retrieve the FIB for a command or keyword passed from the caller and pass the caller the 8-character keyword or command name which corresponds to the format ID specified in the parameter list.
#CHLP	2.3.1 2.8 2.11	#CHLP	HELP control command processor	Select and display the proper help menu.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CHMD	2.9	#CHMD	Help command keys processor	Process command keys when help support is active.
#CHOH	—	#CHOH	#CH@12: #CH@22 format index routine	Find the format index for #CH@12: #CH@22.
#CHOI	—	#CHOI	#CH@23: #CH@NN format index routine	Find the format index for #CH@23: #CH@NN.
#CHOM	—	#CHOM	Home key processor	Process Home key, based on current display.
#CHOP	2.9 2.10	#CHOP	Help output processor	Display help menus and prompts, help text for help menus and prompts, and command processor screens.
#CHPB	2.8 2.9	#CHPB	Help page-back processor	Display previous help support page.
#CHVE	—	#CHVE	CAPS verify routine	Verify MICS for CAPS design.
#CIAD	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CIAL	// ALLOCATE processor	
		#CIDL	// DEALLOC processor	
#CIAM	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CIAT	// ATTR processor	
		#CIMT	// INCLUDE (MRT) processor	
#CIEP	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CIEV	// EVOKE processor	
		#CIPP	// PROMPT processor	
#CIER	3.1	IER00100	Initiator error routine	Process all initiator-detected errors by setting up error MICs and processing operator responses.
#CIFL	3.1	#CIFL	// FILE OCL statement processor	Process disk FILE statements.
#CILD	3.1	#CIIC	// INCLUDE and // LOAD OCL statement processor	Process OCL verbs from initiator.
#CILP	8.16.3 8.20 8.11	#CILF #CIPA	Library free and procedure parameter access routines	Free libraries not being used and provide access to procedure parameter save area.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CIML	3.1	IML00010	Initiator mainline	Read OCL statements and pass control to proper OCL verb processor.
#CIM0	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CILB	// LIBRARY processor	
		#CILC	// LOCAL processor	
		#CIMM	// MEMBER processor	
		#CISL	// SYSLIST processor	
		#CISW	// SWITCH processor	
#CIM1	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CIIN	// INFOMSG processor	
		#CIJM	// JOBQ, MENU, MSG, POWER, and VARY processor	
		#CILG	// LOG processor	
		#CINH	// NOHALT processor	
		#CIOF	// OFF processor	
#CIM2	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CIFM	// FORMS processor	
		#CIIM	// IMAGE processor	
		#CIRG	// REGION processor	
		#CIRS	// RESERVE processor	
#CIM3	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CICM	// COMPILE processor	
		#CICO	// COMM processor	
		#CIDT	// DATE processor	
		#CISN	// SESSION processor	
#CIM4	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CIAB	// ABEND processor	
		#CIDB	// DEBUG processor	
		#CIWT	// WAIT processor	

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CIPW	3.1		OCL verb processing	Process OCL verbs from initiator.
		#CIPR	// PRINTER statement processor	
		#CIWK	// WORKSTN statement processor	
#CIRN	3.1	IRN00100	// RUN OCL statement processor	Process RUN statements.
#CISU	—	ISU00010	Initiator procedure start setup	Perform procedure startup.
#CITD	3.1	#CIMN	// FILE OCL statement processor	Process diskette FILE statements.
#CLAE	6.2.7	—	IF ACTIVE and // EVALUATE processing routines	Process active procedure test and evaluate statements.
		CLAC	Active procedure existence tester	
		CLEV	EVALUATE statement routine	
#CLBEF	6.2.7	—	SYSIN IF, existence, and and file size test routines	
		CLBL0000	IF block existence test	
		CLEX0000	IF file, member switch existence test	
		CLFS0000	File size substitution	
#CLEKM	6.2.7	—	SYSIN keyboard, error, and message routines	Process keyboard input, errors detected by SYSIN, and messages issued by SYSIN.
		CLER		Issue messages for SYSIN.
		CLKB		Process keyboard input for SYSIN.
		CLMP		Log *, **, and pause messages for SYSIN.
#CLIS	6.2.7	—	SYSIN IF statement and substitution processing routines	
		CLIF		Process procedure IF statements.
		CLSB		Perform procedure substitution processing.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CLMSW	6.2.9 2.14 2.15	CLMG0000 CLSR0000 CLWO0000	Interface to CLMG, CLSR, CLWO	Set up messages interface.
#CLSG	6.2.9	#CLSG	SYSLOG	Log messages to a display station and/or the history file, and provide for responses to messages.
#CLSR	2.14 2.15	#CLSR	Dual message routing	Route a copy of messages that require a response and that were sent from the console back to any work stations attached to that task.
#CLSS	6.2.7	#CLSS #MASYL	SYSIN mainline Source get	Get and process procedure statements from a library member or the keyboard.
#CLST	6.2.8	#CLST	SYSLIST	Perform system list function.
#CMCD	2.8 2.11 2.12	#CMCD	Console display	Display messages at system console or subconsole.
#CMCI	2.11 2.5	#CMCI	Console light manager	Manage the message waiting light for display stations.
#CMCR	—	#CMCR	Subconsole reassign	Reassign subconsole messages to the system console.
#CMCU	2.4 2.11 2.13	#CMCU	Command processor cleanup routine	Perform log printer, history file put, message retrieve, and message display for command processor.
#CMEJ	2.11	#CMEJ	Console EOJ	Mark all messages responded to by EOJ with '**' on console.
#CMSM	—	#CMSM	Console management special message	Handle unsolicited message requests from ICF.
#CMWO	9.1.3 9.1.4		System logging utility	Perform data management function between SYSLOG and a work station.
#CNMID	1.2.4	$CNMID00	System configuration utility: modify work station	Prompt for and set work station logical IDs, subconsole assignment, and default printer assignment.
#CPDK	2.8	#CPDK	Dup key processor	Put saved data in input field when Dup key is pressed.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CPET	4.5.3 10.6.3 10.6.4 10.6.5 10.6.6 10.6.8	#CPET	EOJ message purging routine	Search console and subconsole queues for outstanding task messages.
#CPHK	—	#CPHK	Command processor Help key router	Route the Help key to the appropriate module for processing.
#CPIN	2.16	#CPIN	OCL interface to command processor, running under user TCB	Provide an interface to allow commands to be issued by an SVC or OCL.
#CPIO	2.10	#CPIO	Command processor input/output processor	Create WSDM parameter list and call WSDM to handle save/restore requests, display command processor screens, display user menus, and display user menu help text.
#CPIP	2.9	#CPIP	Sign-on procedure attach	Attach the sign-on procedure.
#CPIQ	2.7 2.9	#CPIQ	Command processor input/output inquiry menu processor	Process inquiry options, rename, and other exception conditions.
#CPKS	—	#CPKS	Keysort processor	Perform keysort on files during STOP system.
#CPMF	2.9	#CPMF	System error display	Display error messages on system error display.
#CPML	2.1 2.5 2.14 2.15 2.16	#CPML	Command processor mainline task and wait resident	Route control to appropriate command processor routine and wait for action.
#CPOC	2.16	#CPOC	Initiator-command processor for OCL commands running under the command processor TCB	Provide an interface to allow commands to be issued by an SVC or OCL.
#CPON	2.2 2.9	#CPON	Sign-on processor	Process sign-on.
#CPPS	2.1	#CPPS	Command processor procedure start processor	Process procedure start requests from programs.
#CPRL	2.6	#CPRL	Work station release processor	Release a work station from a job.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CPRS	2.3.2 2.3.1	#CPRS	First operand router processor	Route for command processing based on the first operand of the Cancel, Hold, Release, Start, or Stop commands.
#CPRT	2.2 2.3.1 2.4 2.7 2.8	#CPRT	Command processor input router 1	Initially process input routine commands to appropriate module.
#CPSC	2.1 2.5 2.16	#CPSC	Command processor scheduler	Schedule command processor events.
#CPSI	2.1 2.2 2.3.1 2.4 2.5 2.6 2.7 2.8 2.12 2.15 2.16 2.17	#CPSI	Command processor subtask initialization	Initialization/termination processes for command processor subtasks.
#CPSP	2.9	#CPSP	Command processor special key processor	Initiate processing required by special function keys on keyboard.
#CPTC	2.1 2.5 2.13	#CPTC	Command processor task-to-task communications router	Process system request inquiry, and release functions.
#CPTS	2.11	#CPTS	Command processor sign-on displayer	Display sign-on screen.
#CPT2	2.5	#CPT2	Attention key processor	Process Attn key.
#CPT3	2.12	#CPT3	System Request and Power-On key processor	Process Sys Req and Power On keys.
#CSAF	6.2.1	#CSAF	Active format-1 access transient	Read or write a format 1 from or to the active format area (AFA).
#CSBT	6.2.3 8.5.1 8.16.4	#CSBT	Build membership table module	Build printable character table for command processor and $MAINT and $COPY utilities.
		#CSBM	Build membership table from source member	

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CSMS	6.2.18	#COMS	Diskette magazine drive search	Search for specified file or diskette.
#CSPL	—	#CSPL	Phone list retrieval module	Search AFA or disk for specified phone list.
#CSRD	6.2.16	#CSRD	Disk-to-diskette save/restore	Transfer data from disk to diskette for system save/restore functions performed through the data storage controller.
#CSSM	3.2.1 3.2.2	#CSSM	Printer interface routine (image verify)	Send the proper image and translation table to either a 5211 or 3262 Printer.
#CSSR	6.2.16	#CSSR	Diskette data save/restore	Perform SAVE and RESTORE data transfer.
#CSTR	6.2.17	#CSTR	Disk-to-tape save/restore 2	Transfer data from disk to tape for system save/restore functions performed through the data storage controller.
#CSTS	6.2.17	$#CSTS	Disk-to-tape save/restore 1	Transfer data from disk to tape for system save/restore functions performed through the data storage controller.
#CSVF	6.2.4	#CSVF	Disk VTOC access transient	Read or write format 1's in the disk VTOC on disk.
#CSVI	6.2.5	#CSVI	Diskette VTOC read/write	Read/write diskette VTOC and prepare.
#CSVJ	6.2.5	#CSVJ	Diskette VTOC read/write	Read/write diskette VTOC and prepare.
#CTCP	5.2.3	#CTCP	Reserve file compress	Accumulate free space within the bounds of the reserve file space for the task.
#CTECM	5.2.1		Termination communications interface transient	Terminate data communications task.
#CTEDM	4.11.9 5.2.2	#CTEDM	Remote file termination processing	Terminate the processing of a remote file.
#CTEEX	—	#CTEEX	Exit interface for termination	Build or reset a termination exit block (TEB).
#CTEFP	5.2.2	#CTEFP	Termination file processing routine	Update format 1's, format 5s, and FSBs for all files used.
#CTEIF	5.2.1	#CTEIF	Termination interface transient	Perform initial termination of step and job. Interface to the main terminator.
#CTEJB	5.2.3	#CTEJB	Termination mainline JCB processing	Update JCB and prepare for next step.
#CTEKS	5.2.2	#CTEKS	Termination keysort	Determine which files, if any, require keysort.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#CTEML	5.2.2	#CTEML	Termination mainline	Determine which termination routines are required and transfer to required modules.
#CTES	8.20	#CTES	Update spool file, JOBQ, message file, and security file SCA addresses	Update the spool file master index and SCA file addresses.
#CTEVI	5.2.2	#CTEVI	Process alternative indexes	Update format 1's, format 5s, and FSBs for alternative index files used.
#CUTST	7.7	#CUTST	Communications test supervisor	Display screen message support.
#CZAE	10.9.2.1	#CZAE	Alert task asynchronous error transient	Perform alert task abnormal termination.
#CZAT	10.9.2.1	#CZAT	Alert task	Issue alert to host system when posted by SYSLOG or I/O error recovery.
#DDCL	5.1.1	#DDCL	Disk close	Close disk DTFs.
#DDDM	4.1.1 4.1.2 4.1.3 4.1.4 4.1.5	—	Disk data management	Provide data management router and disk data management access method drivers, base functions and subroutines.
	4.1.5	DDAD	Add	
	4.1.3	DDADMOV	Move/write record	
	4.1.1	DDBEOF	Set cursor to BOF/EOF state	
	4.1.1	DDBLOCK	Disk data management request processor	
	4.1.4	DDDL	Delete record	
	4.1.1	DDERROR	Abnormal terminate	
	4.1.3 4.1.4	DDESTBFU	Establish buffer addressability	
	4.1.1	DDESTIMP	Implicit update/delete	
	4.1.1	DDEXTINT	File extend initialization	
	4.1.2	DDGT	Get record, keyed get, unkeyed get	
	4.1.2	DDGTR	Get record	
	4.1.5	DDIDXDP	Check for invalid duplicate key	

Module Name	Chart Number	Entry Point	Descriptive Name	Function
	4.1.1 4.1.2 4.1.3 4.1.4 4.1.5	DDMAIN	Disk data management detail processing	
	4.1.3	DDUP	Update record	
	4.1.1	DDXKSORT	Keysort	
#DDD2	4.1.1 4.1.5	#DDDM2	Disk data management index insert routine	Insert an index entry with the specified key/RRN in each index in the specified key list.
#DDD3	4.1.1	#DDD3	Disk data management selection expression evaluation routine	Select records meeting the conditions specified in the selection expression string.
#DDD4	4.1.1	DD4MAIN	Disk data management exception processing transient	Processes the following exception conditions: • Set cursor to BOF/EOF state. • Abnormal termination. • Implicit update/delete read. • File extend initialization. • Fix end-of-data request. • Set lower limit. • Keysort required. • Selection expression initialization.
#DDD5	3.3.1	#DDD5	Quicksort	Sort index overflow area for keyed files.
#DDKAA	4.7	AA0000	Keysort control	Perform keysort mainline processing.
#DDKBB	4.7	BA0100	Keysort preprocessor	Initialize and check for sort level required.
#DDKLL	4.7	LL0000	Keysort 3-phase sort control	Coordinate three phase sort in work area.
#DDKMM	4.7	MM0000	Keysort merge function	Perform keysort merge.
#DDKSS	4.7	SS0000	Keysort sort-in-place	Perform keysort without work area.
#DDKZZ	4.7	ZZ0100	Keysort postprocessing	Perform keysort cleanup.
#DDK1A	4.7	#1A000	Keysort phase-1 control	Perform phase 1 of 3-phase keysort.
#DDK2A	4.7	#2A000	Keysort phase-2 control	Perform phase 2 of 3-phase keysort.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#DDK3A	4.7	#3A000	Keysort phase-3 control	Perform phase 3 of 3-phase keysort.
#DDSM	4.6	#DDSM	Sector data management to disk	Read or write multiple sectors of data to or from disk files.
#DDXL	8.23.1.8 8.23.1.12 8.23.1.14	#DDXL	Disk file extend	Extend the space of disk data files and update their active and VTOC format 1's.
#DD1OP	3.3.1	#DD1OP	Disk open Phase 1 transient	Perform the following open functions: • Check all DTFs on chain for valid devices specified. • And that all DTFs to be opened have been allocated. • If no disk DTFs on chain, call other device opens. • Perform initial diagnostic checking on all disk DTFs. • Complete processing for any ZPAM DTFs and route control to #DD2OP for other disk DTF processing.
#DD2OP	3.3.1	#DD2OP	Disk open Phase 2	Perform the following open functions for (non-ZPAM) DTFs: • Set up required file space and SQS data areas. • Perform disk pseudo open if requested. • Call keysort if required. • If indexed file or multiple index file, call #DD3OP.
#DD3OP	3.3.1	#DD3OP	Disk open Phase 3	Perform the following disk DTF open functions: • Set up high-key data areas required for data files. • Set up a storage index for primary portion of the file index.
#DEBUG	—	#DEBUG	Dump program	Generate dumps.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#DNAE	—	#DNAE	PC Support/36 abnormal error transient of the LAN	PC Support/36 abnormal error error of the LAN.
#DNICF	10.10.2	#DNICF	PC Support/36 translator	Provide SSP-ICF interface for PC Support/36 request handlers.
#DNPM	—	#DNPM	PC Support/36 protocol module	Line open function on the LAN and PC Support/36.
#DNTFS	10.10.1 10.10.2	#DNTFS	Shared folder request handler	Provide server function for PC Support/36 shared folders.
#DNTLM	10.10.2	#DNTLM	PC Support/36 router for the LAN	Establish and handle communications to a personal computer for PC Support/36 via a connection of the LAN.
#DNTPM	10.10.1 10.10.2	#DNTPM	Virtual print request handler	Provide server function for PC Support/36 virtual printer.
#DNTRM	10.10.1	#DNTRM	PC Support/36 router	Establish and handle communications to a personal computer for PC Support/36.
#DNTTF	10.10.1 10.10.2	#DNTTF	File transfer facility request handler	Provide server function for PC Support/36 transfer facility.
#DNTVM	10.10.1 10.10.2	#DNTVM	Virtual disk request handler	Provide server function for PC Support/36 virtual printer.
#DPAL	—	#DPAL	Printer alignment	Give user capability to align printer forms.
#DPCL	5.1.1	#DMCL	Common close	Close printer and work station DTFs and route for closing of other device DTFs.
#DPDM	4.3.1	#DPDM	Printer data management	Issue IOBs to printer IOS (main storage) to perform requested printer I/O operations.
#DPOP	3.3.1 9.1.1	#DPOP	Printer data management open	Process printer DTFs from common open.
#DQEPG	4.12.1	#DQEPG	Query data management (QDM) expression generator	Process source expressions during QDM compile.
#DQM1	4.12.1	#DQM1	Query data management (QDM) merge phase 1	Merge records from five or more sort buffers.
#DQM2	4.12.1	#DQM2	Query data management (QDM) merge phase 2	Merge records from four or less sort buffers.
#DQOPG	4.12.1	#DQOPG	Query data management (QDM) sort generator	Perform required sort code generation during QDM compile.
#DQQOP	4.12.1	#DQQOP	Query data management (QDM) open	Perform initialize processing during query compile and route control for execution processing.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#DQQS	4.12.1	#DQQS	Query data management (QDM) quicksort	Sort records in output buffer(s).
#DQSPG	4.12.1	#DQSPG	Query data management (QDM) select generator	Perform required select code generation during QDM compile.
#DQWPG	4.12.1	#DQWPG	Query data management (QDM) WHERE generator	Process WHERE statements during QDM compile.
#DQ1DU	4.12.1	#DQ1DU	Query data management (QDM) data scan to user driver	Select user-specified records and put them in user buffer.
#DQ2DB	4.12.1	#DQ2DB	Query data management (QDM) data scan build list driver	Set up for QDM select and sort processing (for records to be returned to user).
#DQ3DD	4.12.1	#DQ3DD	Query data management (QDM) data scan to disk driver	Set up for QDM select and sort processing (for records to be written to disk).
#DQ4XU	4.12.1	#DQ4XU	Query data management (QDM) index scan to user driver	Select records using RRN buffer in query common.
#DQ5XB	4.12.1	#DQ5XB	Query data management (QDM) index scan build list driver	Select and sort records using RRN buffer in query common.
#DRCL	5.1.1	#DRCL	Diskette close	Close diskette DTFs.
#DRDI	4.2.1	#DRDI	Diskette data management for unspanned I-files	Transfer I-format files to or from a diskette in record mode.
#DRDM	4.2.1	#DRDM	Diskette data management	Process puts and gets in record mode to diskette.
#DROP	3.3.1	#DROP	Diskette file open transient	Open diskette files.
#DRSI	4.2.1	#DRSI	Diskette data management for unspanned I-files in sector mode	Transfer I-format files to or from a diskette in sector mode.
#DRSM	4.2.1	#DRSM	Diskette sector data management	Handle gets and puts in sector mode to diskette.
#DSAF	8.36.3	#DSAF	IDDU link definition to label	Prompt for data that links a definition to a file label.
#DSA2	8.36.1.2	#DSA2	IDDU field selection and ordering	Prompt to verify that WHAT-USED table entries for fields used by a format contain current information.
#DSA3	8.36.1.3	#DSA3	IDDU format selection and ordering	Prompt to verify that WHAT-USED table entries for fields used by a format contain current information.
#DSBP	8.36.4	#DSBP	IDDU batch print routine	Print DDA information for the definition name requested.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#DSCA	8.36.1.2	#DSCA	IDDU communications attributes	Prompt for and validity check communcations attributes.
#DSCD	8.36.2	#DSCD	IDDU dictionary create	Create data dictionary.
#DSCE	8.36.1.2	#DSCE	IDDU evoke process	Prompt for evoke build.
#DSCF	8.36.3	#DSCF	IDDU disk file create	Create a disk file.
#DSCP	8.36.1.2	#DSCP	IDDU evoke process	Prompt for evoke parameters selection and ordering.
#DSC1	8.36.1.1	#DSC1	IDDU field attributes	Prompt for field attributes.
#DSDT	8.36.1	#DSDT	IDDU definitions options	Prompt for changes to definitions and route for definition type requested.
#DSD1	8.36.1.1	#DSD1	IDDU field definition	Prompt for changes to field definitions.
#DSD2	8.36.1.2	#DSD2	IDDU format display	Prompt for changes to format definitions.
#DSD3	8.36.1.3	#DSD3	IDDU file definition	Prompt for changes to file definitions.
#DSE1	8.36.1.1	#DSE1	IDDU field edit	Prompt for editing of numeric field definitions.
#DSFI	8.36.5	#DSFI	IDDU F and I specifications to IDDU conversion	Convert from F and I specifications format to IDDU format.
#DSF1	8.36.1.1	#DSF1	IDDU multiple field create display	Prompt for creating multiple field definitions.
#DSG2	8.36.6	#DSG2	IDDU format data stream generator	Generate record-ID code and/or commmunications format data streams.
#DSIN	8.36.0	#DSIN	IDDU initialization	Initialize and route for IDDU processing.
#DSLL	8.36.3 8.36.9	#DSLL	IDDU label list	Read VTOC and list specified file labels.
#DSLS	8.36.1 8.36.3 8.36.8.1 8.36.8.2 8.36.8.3	#DSLS	IDDU dictionary list	List data dictionary names for selection by user.
#DSL1	8.36.8.1	#DSL1	IDDU member list 1	List field definitions.
#DSL2	8.36.8.2	#DSL2	IDDU member list 2	List format definitions.
#DSL3	8.36.3 8.36.8.3	#DSL3	IDDU member list 3	List file definitions.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#DSM2	8.36.9	#DSM2	IDDU format mark	Use affected format list to update applicable DDA fields.
#DSM3	8.36.9	#DSM3	IDDU file mark	Use affected files list to update applicable DDA fields.
#DSOD	8.36.2	#DSOD	IDDU data dictionary options display	Prompt for type of dictionary option and route for requested option.
#DSOF	8.36.3	#DSOF	IDDU disk file options display	Prompt for disk file option to be performed against specified disk label.
#DSOP	8.36.4	#DSOP	IDDU print options display	Prompt for print option to be performed against specified definition.
#DSOR	8.36.1.1 8.36.1.2 8.36.1.3 8.36.2	#DSOR	IDDU rename options display	Prompt for and rename a file, format, field, or dictionary.
#DSO1	8.36.1.1	#DSO1	IDDU field options	Prompt for and process field options for definition.
#DSO2	8.36.1.2	#DSO2	IDDU format options	Prompt for and process format options for definition.
#DSO3	8.36.1.3	#DSO3	IDDU file options	Prompt for and process file options for definition.
#DSP1	8.36.4	#DSP1	IDDU field print routine	Print requested field definitions.
#DSP2	8.36.4	#DSP2	IDDU format print routine	Print requested format definitions.
#DSP3	8.36.4	#DSP3	IDDU format print routine	Print requested file definitions.
#DSRB	8.36.6	#DSRB	IDDU dictionary rebuild	Rebuild requested dictionary.
#DSRD	8.36.2	#DSRD	IDDU dictionary revise	Change dictionary description and/or dictionary security.
#DSRI	8.36.1.2	#DSRI	IDDU record-ID codes	Display requested record-ID codes.
#DSS1	8.36.5	#DSS1	IDDU field exit	Update data dictionary with field definition changes.
#DSS2	8.36.5	#DSS2	IDDU format exit	Update data dictionary with format definition changes.
#DSS3	8.36.5	#DSS3	IDDU file exit	Update data dictionary with file definition changes.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#DSVD	8.36.1.1 8.36.1.2 8.36.1.3 8.36.2	#DSVD	IDDU confirm delete	Confirm a delete request from any options display.
#DSVW	8.36.1.1 8.36.1.2 8.36.1.3	#DSVW	IDDU confirm WHERE-USED	Display WHERE-USED information for a definition.
#DSV2	8.36.9	#DSV2	IDDU format verification	Confirm that changes made to a low-level definition will not result in errors in a higher-level definition.
#DSV3	8.36.9	#DSV3	IDDU file verification	Confirm that changes made to a low-level definition will not result in errors in a higher-level definition.
#DSWC	8.36.8.2 8.36.8.3	#DSWC	IDDU copy WHAT-USED definitions	Copy a list of items used by a file or format definition.
#DSWU	8.36.4 8.36.9	#DSWU	IDDU WHERE-USED tracking and control	Build a list of affected items used by a particular field, format, or file definition and place it in the dictionary control member's DDA.
#DSXD	8.36.1.1 8.36.1.2 8.36.1.3 8.36.2	#DSXD	IDDU long comment display	Prompt for long comment.
#DSX1	8.36.1.1	#DSX1	IDDU field exit	Update data dictionary with field definition changes.
#DSX2	8.36.1.2	#DSX2	IDDU format exit	Update data dictionary with format definition changes.
#DSX3	8.36.1.3	#DSX3	IDDU file exit	Update data dictionary with file definition changes.
#DSY1	8.36.8.1	#DSY1	IDDU copy field selection	Prompt for and copy requested field definitions.
#DSY2	8.36.8.2	#DSY2	IDDU copy format selection	Prompt for and copy requested format definitions.
#DSY3	8.36.8.3	#DSY3	IDDU copy file selection	Prompt for and copy requested file definitions.
#DSZDD	8.36.5	#DSZDD	IDDU data dictionary processing	Create data dictionary definitions from data in work file.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#DSZIN	8.36.5	#DSZIN	IDDU F and I specifications conversion initialization	Initialization for data dictionary processing.
#DSZSR	8.36.5	#DSZSR	IDDU F and I specifications source processing	Process all source members specified in the IDDUXLAT procedure parameters.
#DSZ1	8.36.8.1	#DSZ1	IDDU copy field	Prompt for and copy specified field definitions.
#DSZ2	8.36.8.2	#DSZ2	IDDU copy format	Prompt for and copy specified format definitions.
#DSZ3	8.36.8.3	#DSZ3	IDDU copy file	Prompt for and copy specified file definitions.
#DTCC	2.3.3	#DTCC	Display status of system communications configuration	Display communications lines status information.
#DTDC	2.3.3	#DTDC	Command processor status command–communications	Display communications lines status information.
#DTDL	2.3.3	#DTDL	Display communications line activity	Display communications line dynamic activity.
#DWDM	4.4.0	#DWDM	Work station data management router resident	Call work station data management routines.
#DXET	10.9.1	#DXET	C & SM change management termination	Terminate change management.
#DXIN	10.9.1	#DXIN	C & SM change management initiator	Initialize for change management.
#DXML	10.9.1	#DXML	C & SM mainline	Perform mainline processing for change management.
#EMAD	10.8.3	EMGOA	3270 BSC device emulation initialization	Initialize display for emulation.
#EMAX	10.8.3	EMEWA	3270 BSC device emulation driver	Emulate 3277 display.
#EMBX	10.8.3 10.8.4	#EMBX	3270 BSC emulation termination exit routine	Detach device from 3270 emulation.
#EMFI	—	EMFGO	3270 BSC emulation installation routine	Determine the name of the translation tables for the 3270 I/O interface code and 5250 character set.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#EMFP	10.8.3 10.8.4	EMFGO	3270 BSC emulation EM3270 procedure processor	Verify the procedure parameters.
#EMFX	10.8.3 10.8.4	#EMFX	3270 BSC emulation abnormal exit routine for procedure processor	Perform abnormal termination for #EMFP.
#EM0101 through #EM1516	—	—	3270 BSC emulation translate tables	Non-executable load modules that provide foreign language translate tables for device drivers.
#EM9D	10.8.4	EMGO9	3270 BSC device emulation printer driver	Emulate 3288 Printer.
#ESAD	10.8.6	ESG02	3270 SNA device emulation intialization	Initialize display for emulation.
#ESAX	10.8.6		3270 SNA device emulation mainline	Emulate 3277 display.
#ESBX	10.8.6 10.8.7	#ESBX	3270 device emulation termination exit routine	Detach device from 3270 emulation.
#ESFI	—	ESFGO	3270 device emulation, installation routine	Determine the name of the translation tables for the 3270 I/O interface code and 5250 character set.
#ESFP	10.8.6 10.8.7	ESFGO	3270 SNA emulation ES3270 procedure processor	Verify the procedure parameters.
#ESFX	10.8.6 10.8.7	#ESFX	3270 SNA emulation termination exit routine procedure processor	Perform abnormal termination for module #ESFP.
#ESPI	10.8.7	ESGO5	3270 device emulation, SNA printer initialization	Initialize printer for emulation of a 328x Printer.
#ESSD	10.8.7	ESEW5	3270 SNA SCS LU1 printer mainline	Emulate 3287 Printer in SCS format.
#ES0116 through #ES1916	—	—	3270 SNA emulation translate tables	Non-executable load modules that provide foreign language translate tables for device drivers.
#ES9D	10.8.7	DSCPNT	3270 SNA DSC LU3 printer mainline	Emulate 328x Printer in DSC format.
#FEIN	7.4.2	#FEIN	Service log allocate/terminate transient	Allocate the service log file for a service logging session or terminate the service logging session.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#FERX	7.4.1	#FERX	Service log recording transient	Log data to the service log file.
#FESLG	7.1.2.10 7.1.5	#FESLG	Start/stop disk trace logging	Start or stop logging of trace tables to disk for a specific trace table.
#FETDP	—	#FETDP	Main storage task dump program (router)	Determine the output device for a task dump, and load in the appropriate device routine.
#FETFD	—	#FETFD	Main storage task dump program (disk routine)	Output a task dump to a disk dump file.
#FETLK	7.1.5	#FETLK	Diagnostic aids APAR utility 4	Perform APAR termination processing.
#FETPR	—	#FETPR	Main storage task dump program (printer routine)	Format and print a task dump.
#FETRD	—	#FETRD	Main storage task dump program (diskette routine)	Output a task dump onto an APAR diskette.
#GAER	—	#GAER	EXTN transient for RCL recovery	Determine which RCL entries are necessary when the RCL is full.
#GAIC	—	#GAIC	EXTN transient for input and CGU	Perform EXTN processing for input and CGU operations.
#GAML	2.5 4.4.1 4.4.2 9.6.4	GAMINIT	EXTN task mainline	Perform mainline processing for ideographic character devices on the system.
#GAPO	—	#GAPO	EXTN printer transient	Perform printer EXTN processing.
#GAPR	4.3.1 4.5.3	#GAPR	EXTN character print scan	Search printer data stream for EXTN character codes requiring translation to RAM address.
#GASR	4.4.1 4.4.2	#GASR	EXTN task save/restore RCL	Save or restore RAM contents list (RCL) for a work station.
#GAWT	—	#GAWT	EXTN where-to-go tables	Contains where-to-go tables in case OXRF has run.
#GCAC	9.5	#GCAC	MLCA/ELCA autocall dial module	Process switched-line call requests from protocol link control.
#GCAT	9.4.0 9.4.2	#GCAT	#GCFR (REQUESTX) abnormal termination	Perform cleanup for REQUESTX abnormal termination by deallocating line, freeing data areas, and posting the X.21 MLCA/ELCA dial task.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#GCFR	9.4.2	#GCFR	X.21 dynamic registration/ cancellation task	Process operator requests for registration for, or cancellation of, Public Data Network services.
#GC21	9.4.1 9.1.1	#GC21	X.21 MLCA/ELCA autocall task	Load X.21 connection task and process autocall requests from protocol link control.
#HCER	10.9.3	#HCER	Session take-down	Perform processing required when session is canceled by either System/36 or host.
#HCIN	10.9.3	#HCIN	3270-5250 translate	Process received data. Data stream translation 3270-5250.
#HCKB	10.9.3	#HCKB	Keyboard map management	Perform keyboard map functions (display, change) initiated from keys procedure.
#HCML	10.9.3	#HCML	Remote management mainline	Acts as an interface to work station data management for inbound and outbound data.
#HCOUT	10.9.3	#HCOUT	5250-3270 translate	Process data to be transmitted. Data stream translation 5250-3270.
#HCVY	10.9.3	#HCVY	Session and device bring-up	Perform session bring-up and device bring-up.
#HFCOPY	6.2.10	#HFCOPY	High-speed history file copy	Copies the history file to a user file when history file becomes full.
#HFPU	6.2.10	#HFPU	History file put	Put OCL, error messages, and operator responses in the history file.
#HPDA	4.4.2	#HPDA	Abnormal termination	Perform cleanup for help task abnormal termination.
#HPDO	4.4.2	#HPDO	Application help document processor	Browse through an online document.
#HPKP	4.4.2	#HPKP WAIT	Help key processor task	Display help formats.
#HPLM	4.4.2	HPDA HPCLR HPDG	Help support list maintenance routine	Documentation is provided at each module entry point.
#HPMN	—	#HPMN	Application help menu read task	Read through an online document (evoked by READINFO procedure).
#HRML	10.10.1 10.10.2	#HRML	SSP-ICF request handler	Provide SSP-ICF server program interface.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#IANA	10.6.7 10.6.10	—	C/SNA half-session allocate nonresident subroutine	Allocate or deallocate C/SNA half-session control blocks (SNUB, LUSCB, and PHUB).
#IANB	—	#IANB	C/SNA session reset	Reset C/SNA session.
#IANC	10.4.2	#IANC	Common session control	Process requests and responses for session activation and deactivation.
#IANE	10.3.1	#IANE	SSP-ICF C/SNA protocol processor	Initialize for C/SNA processing.
#IANJ	10.4.0	#IANJ	C/SNA link manager	Handle exception conditions for C/SNA.
#IANL	10.4.1 10.4.2	#IANL	LU network services (for LU6.2)	Perform FMD requests, notify command, and term-self command network services functions.
#IANO	10.4.1 10.4.2	#IANO	LU network services	Perform FMD requests, notify command, and term-self command network services functions.
#IANP	10.4.2	#IANP	PU network services	Perform configuration and maintenance services.
#IANS	10.4.0	#IANS	Single-node control point (SNCP) configuration services	Process C/SNA-detected errors.
#IANT	10.4.1	#IANT	Transmission control-session control	Perform session service and activation/deactivation functions.
#IANX	—	#IANX	C/SNA error exit program	Receive control when the C/SNA task causes a program check.
#IANY	10.4.1	#IANY	C/SNA data flow control (DFC) send/receive (for FMP-4)	Perform all required processing for sending and receiving DFC commands/responses and subsystem requested data messages/responses.
#IANZ	10.4.1	#IANZ	C/SNA data flow control (DFC) send/receive (for FMP-19)	Perform all required processing for sending and receiving DFC commands/responses and subsystem requested data messages/responses.
#IAN1	—	#IAN1	C/SNA send message	Send a message unit (MU) or a SIG to a SNA procedure.
#IAN4	10.4.0 10.4.1 10.4.2	#IAN4	C/SNA mainline	Perform mainline and routing functions for C/SNA.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#IBCI	10.6.4	#IBCI	SSP-ICF CICS/VS subsystem	Establish and maintain one or more logical communications sessions between the System/36 and CICS/VS via BSC.
#IBGA	10.6.3 10.6.4 10.6.5 10.6.6	#IBGA	SSP-ICF BSC subsystems error exit routine	Process abnormal termination for all SSP-ICF BSC subsystems.
#IBGT	—	#IBGT	SSP-ICF EBCDIC/ASCII translation	Translate EBCDIC to ASCII or ASCII to EBCDIC for SSP-ICF functions.
#IBHE	10.6.1	IBHE	SSP-ICF BSC asynchronous error exit	Perform abnormal termination processing for the BSC mainline task.
#IBHL	10.6.1	—	SSP-ICF BSC link control task	Provide link control to support System/36 BSC users.
#IBHP	10.3.1	IBHP	SSP-ICF BSC protocol module	Perform SSP-ICF BSC open functions at enable.
#IBIDM	—	—	IMS/IRSS data management	(Link-edited in #IBIM.)
#IBILM	—	—	IMS/IRSS subsystem line manager	(Link-edited in #IBIM.)
#IBIM	10.6.3	#IBIxxx	SSP-ICF IMS/IRSS subsystem	IMS/IRSS link-edited object module. Includes the following source modules (listed under their respective directory entries): #IBIDM, #IBILM, #IBIQM, #IBISG, #IBISP, and #IBIWM.
#IBIQM	—	—	IMS/IRSS queue management subroutines	(Link-edited in #IBIM.)
#IBISG	—	—	IMS/IRSS SY block generator	(Link-edited in #IBIM.)
#IBISP	—	—	IMS/IRSS SY block processor	(Link-edited in #IBIM.)
#IBIWM	—	—	IMS/IRSS work manager	(Link-edited in #IBIM.)
#IBLA	10.4.5	IBLXACQ	SSP-ICF BSCEL acquire operation code processor	Process acquire operation for BSCEL subsystem.
#IBLC	10.6.6	IBLCERR1	SSP-ICF BSCEL line error processor 1	Process line error messages for BSCEL subsystem.
#IBLD	10.6.6	IBLDDABL	SSP-ICF BSCEL disable processor	Disable BSCEL subsystem.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#IBLE	10.6.6	IBLEERR2	SSP-ICF BSCEL line error processor 2	Complete processing of line errors for BSCEL subsystem.
#IBLF	—	multiple	SSP-ICF BSCEL subsystem F SSP-ICF transient	Process operation failures for the SSP-ICF BSCEL subsystem.
#IBLM	10.6.6	—	SSP-ICF BSCEL subsystem mainline	Interface between all System/36 functions and the BSCEL subsystem.
#IBLN	10.6.6	IBLXNABL	SSP-ICF BSCEL subsystem enable	Enable the BSCEL subsystem.
#IBLR	—	IBLREOSA	SSP-ICF BSCEL abnormal termination	Perform abnormal termination for the BSCEL subsystem.
#IBPC	10.6.5	—	SSP-ICF CCP subsystem mainline	Interface between all System/36 functions and the CCP subsystem.
#ICCNT	10.2	ICSPGMS	SSP-ICF control resident subroutines	Provide an interface from SSP-ICF subsystem tasks to the command processor procedure start SVC interface.
#ICD2	2.3.3	#ICD2	Command processor status SSP-ICF sessions command	Display status of SSP-ICF sessions.
#ICD2	—	#ICD2	Subsystem session status command processor	Process requests for subsystem session status.
#ICDB	10.1	#ICDB	SSP-ICF data management end session processor	#ICDB is called by #ICDM to perform subroutine execution of subsystem instructions.
#ICDC	10.1	#ICDC	SSP-ICF data management message processor	Issue all SSP-ICF data management error messages, and error messages for every hex 80 or greater return code in a SUB.
#ICDI	2.3.3	#ICDI	Subsystem status command processor	Process requests for subsystem status.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#ICDM	10.1	#ICDM	SSP-ICF data management mainline	Called whenever a user issues an SVC to data management with a device code for an SSP-ICF session and the Interactive Communications Feature has been configured on the system. If the operation is for a work station, transfer control to work station data management to handle the operation. If the operation is for an Interactive Communications Feature session, validate the operation and post the appropriate subsystem to handle the operation. Within the user's DTF is a work station parameter list (WSPL). Within the WSPL is a 2-byte op-code modifier and op-code that instruct #ICDM the required operation(s) to perform.
#ICDN	10.1	#ICDN	SSP-ICF data management mainline	Called to continue the processing of an SSP-ICF user request. #ICDN validates the operation and posts the appropriate subsystem to handle the operation.
#ICDU	10.1	#ICDU	Interface between IDDU and #ICDM	Called when user request is an IDDU request for communications. Validates the operation and converts IDDU communications format to valid WSPL operation code. Calls #ICDM to continue processing.
#IED1	10.3.2 10.6.4 10.6.5 10.6.6 10.6.8	DS100000	Initial disable	Perform initial disable processing.
#IED3	10.3.2 10.6.8	DS300000	Free LOC disable	Dequeue and free a LOC element and free the associated phone list.
#IED9	10.3.2 10.6.4 10.6.5 10.6.6 10.6.8 10.8.2		Final disable	Perform final disable processing.
#IELOC	—	#IELOC	Enable implicit APPN location	Build an APPN LOC.
#IEN3	10.3.1	N3000000	Enable multiple locations	Build LOCs for locations being enabled.
#IESGB	—	#IESBG	Build/free SGB	Build and free SGBs for APPN.
#ILDSS	10.6.11.1 10.6.11.2	#ILDSS	APPN directory services	Conduct and reply to APPN directory searches.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#ILMGR	10.6.11.1	#ILMGR	APPN control point manager	Control CPSVCMG session.
#ILRSS	10.6.11.1 10.6.11.2	#ILRSS	APPN route selection services	Maintain network topology and calculate search and session routes.
#ILSND	10.6.11.1	#ILSND	APPN send	Transmit data to adjacent nodes for local control point.
#ILTR6	10.6.11.1	#ILTR6	APPN target	Receive data from adjacent control points.
#IPLPROC	1.1.5	N/A	IPL sign-on procedure	Invoke IPL overrides according to user responses to prompts. **Note**: This is a system procedure, not an SSP load member.
#IQMA	10.6.9	#IQMA	Peer asynchronous error exit	Terminate task and clean up control blocks after abnormal termination, processor check, or 2K-page failure in Peer task.
#IQMB	—	#IQMB	Build SPUB Peer SSP-ICF transient	Build SPUB for SSP-ICF Peer subsystem task, abnormal termination, processor check, or 2K-page failure in Peer task.
#IQMD	—	#IQAQR	Attach manager request decode Peer SSP-ICF transient	Process received attach FMH5 by setting up for program start.
#IQMQ	10.6.9	#IQMQ	SSP-ICF Peer free SSQS/CQS	Free unneeded SQS/TWS and dequeue messages.
#IQMS	10.3.1	IQMS	SSP-ICF Peer protocol processor	Activate the SDLC task and allocate TWS.
#IQSML	10.6.9	IQSxxx	SSP-ICF Peer subsystem mainline	Perform all SSP-ICF data management operations requested of the Peer subsystem by the end user.
#IRAE	10.6.10	#IRAE	APPC subsystem asynchronous error exit	Handle program and enable errors in the APPC subsystem.
#IRATR	10.1	#IRATR	APPC subsystem get-attributes routine	Retrieve information for get-attributes and get-status operations.
#IRCE	10.6.10	#IRCE	LU services model asynchronous error exit	Handle program and other errors detected by the APPC subsystem source and target LU service models.
#IRCNOSS	10.6.10	#IRCNOSS	APPC LU service model–source	Process change number of sessions requests from local application programs.
#IRCNOST	10.6.10	#IRCNOST	APPC LU service model–target	Process change number of sessions requests from remote systems.
#IRDA	2.3.3	#IRDA	APPC status	Display status for SSP-ICF APPC subsystem.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#IRML	10.6.10	#IRML	APPC subsystem mainline	Process all requests issued to APPC subsystem.
#IRPM	10.3.1	#IRPM	APPC subsystem protocol processor	Update LOC and SCT during enable.
#ISEA	10.6.2	#ISEA	SSP-ICF Intra end-of-step abnormal processor	Provide end-of-step abnormal processing for the SSP-ICF Intra subsystem.
#ISEF	10.6.2	#ISEF	SSP-ICF Intra unsuccessful program start (evoke) processor	Provide evoke failure processing for the SSP-ICF Intra subsystem.
#ISEX	10.6.2	#ISEX	SSP-ICF Intra disable and asynchronous error processor	Get control from SSP-ICF disable, or a soft processor check in the Intra task.
#ISMA	10.6.2	ISMxxxx	SSP-ICF Intra subsystem mainline	Provide user programs the ability to use SSP-ICF to start and communicate with other programs on the same system.
#ITHDA	10.8.1		3270 BSC interrupt handler asynchronous error exit	Disable 3270 emulation after a disable command or an interrupt handler processor check.
#ITHML	10.8.1	—	3270 BSC device emulation interrupt handler	Emulate 3271 control unit.
#ITSCP	10.3.1	#ITSCP	3270 emulation: communications protocol module	Assign system queue space, attach interrupt handler, and allocate line and load microcode.
#ITSET	10.8.2		3270 BSC subsystem error exit routine	Process abnormal termination on subsystem op-end or interrupt handler ABEND.
#ITSPI	10.8.2	—	3270 BSC subsystem	Interface between all System/36 functions and the 3270 BSC subsystem.
#ITSX1	10.8.2		3270 subsystem resource deallocate processor	Release all buffers and control blocks owned by subsystem.
#IUSBC	—	#IUSBC	SNUF BIND check processor	Check BIND images received from C/SNA.
#IUSD	—	#IUSD	Evoke data formatting routine	Format the user's input on an evoke operation to the proper format for transmitting to the host.
#IUSF	10.6.7	multiple	SSP-ICF SNA Upline Facility (SNUF) mainline	Provide an interface between SSP and SSP-ICF at SNUF task's initial startup. Process event completions.
#IUSN	10.3.1	#IUSN	SNUF enable protocol processor	Perform initial processing required before mainline is given control.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#IUSS	—	#IUSS	Procedure start processor	Process procedure start requests from host.
#IUSX	10.6.7	#IUSX	SSP-ICF SNUF error processor	Perform abnormal termination processing for the SNUF subsystem.
#IUS4	—	#IUS4	Process STSN command	When a set and test sequence numbers command is received, this routine is called.
$IUS5	—	#IUS5	Exception handling routine for DFC commands received	Receive control when a data flow control command is received and is not expected.
#IZAET	10.8.5		SNA 3270 termination exit routine	Handle abnormal termination of SNA 3270 subsystem.
#IZBND	10.8.5		SNA 3270 bind check routine	Ensure bind parameter are acceptable.
#IZDBL	10.8.5		SNA 3270 disable routine	Handle normal disable or termination of an SCT or LOC.
#IZMNL	10.8.5	#IZMNL	3270 SNA mainline	Interface between all System/36 functions and the 3270 SNA subsystem.
#IZSCP	10.3.1 10.8.5	#IZSCP	SNA 3270 protocol processor	Initialize to allow enabling of subsystem.
#IZSRT	10.8.5	IZSNx	SNA 3270 interface subroutines	Process SNA 3270 event completions.
HRSTAT	10.1	HRSTAT	PC Support/36 get-attributes routine	Retrieve information for get-attributes and get-status operations.
#LNAT	9.10.3	#LNAT	Initialization/termination abnormal termination of the LAN	Perform initial abnormal termination for errors detected by initialization or termination of the LAN, and perform abnormal termination for all processes of the LAN.
#LNIT	9.10.1.1 9.10.1.2 9.10.2	#LNIT	Initialization/termination of the LAN	Process line open requests from SNA, handle open station and enable IOBs from the SNA, process all messages for the LAN, and complete processing for terminate/detach from the LAN.
#MACG	6.1.9	#MACG	Library member change routine	Change member name, subtype, and/or reference number.
#MADT	6.1.7	#MADT	Library directory entry delete routine	Delete specified non-IBM-supplied members from a library.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#MAFLB	6.1.1	#MAFLB	Find a library routine	Find a library by 8-byte name, and set up for its use.
	6.1.1	@CSVF	Disk VTOC read/write	
#MAMPM	6.1.8	#MAMPM	Library member protection	Interlock library member.
#MANOP	6.2.2	#MANOP	No-op routine for cross-reference resolver	Handles errors when SSP module in a WTG table is not found in the system library
#MAPR	6.1.5	#MAPUR	Library record put routine	Place source or procedure records into the library.
#MASEC	6.1.6	#MASEC	Library sector get/put	Get modules in sector mode from the library and pass them to a calling routine.
#MASFN	6.1.2	#MAFLB	Single name find routine	Find a library member in either the system or user library.
#MASGT	6.1.4	#MASGT	Source library find/get routine	Find either source or procedure members when requested.
#MASYL	6.1.4 8.16.4	@MASYG	Source library get routine include and load version	Get records from source or procedure members in the requested library.
#MAXRF	6.2.2	#MAXRF	Cross-reference resolver resident	Resolve absolute disk addresses of routines in where-to-go (WTG) tables so those routines can be loaded without the use of system find. Also, resolve format member index tables in modules.
#MAXT	—	#MAXT	Librarian extent	Create a library extent when a full library exists on a $MAINT to library sector mode copy.
#MERP	4.9	#MERP	MCR data management error recovery	Perform error recovery for the MCR data management task.
#MLCK	9.3.1	#MLCK	MLCA/ELCA controller check handler	Initiate and control MLCA/ELCA error recovery procedures.
#MLCT	2.5 9.3.1	#MLCT	MLCA/ELCA controller recovery	Provide interface between the error processing transient (#SVERP) and the MLCA/ELCA error recovery task (#GCCE) by attaching #GCCE.
#MGRET	6.2.6 6.2.8	MGR00000	Message retrieve routine	Retrieve a message from the specified message member.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#MIML	4.9	#MIML	MCR data management mainline	Provide data management mainline routing for the MCR device.
#MION	4.9	#MION	MCR data management mainline	Provide data management functions for the MCR device.
#MNER	—	#MNER	Multinational conversion error utility	Issue errors for the #MNER procedure.
#MRER	4.9	#MRER	MCR data management error recovery	Perform error recovery for the SUBR25 subroutine.
#MRON	4.9	#MRON	MCR data management open	Provide data management open routine for the SUBR25 subroutine.
#MNER	—	#MNER	Multinational conversion error utility	Issue error messages for the #MNER procedure.
#MSBEL	1.1.6 8.20	#MSBEL__0	IPL file rebuild: initialization and non-indexed file checking module	Initialize common area, and verify and correct VTOC format 1's for non-indexed files.
#MSBFL	1.1.6	#MSBFL__0	IPL file rebuild: indexed file check module	Verify and correct VTOC format 1's for indexed files.
#MSBGL	1.1.6	#MSBGL__0	IPL file rebuild: date chain module	Ensure all format 1's for files with the same label and different dates are properly marked.
#MSBHL	1.1.6	#MSBHL__0	IPL file rebuild: VTOC compactor	Place all format 1's in the minimum number of sectors possible.
#MSBIX	8.5.3 8.5.4 1.1.6	MSIX0000	Index extraction	Get keys from indexed file data and build index.
#MSBLD	1.1.6	#MSBLD__0	IPL file rebuild: initial module	Initialize common area for IPL file rebuild.
#MSCFG	1.1.5	MSIPLMNT	Main storage IPL overrides	Prompt for and process override requests from the system operator.
#MSCPR	1.1.4	MSCPRMNT	Command processor function	Initiate sign-on sequence and perform sign-on initialization for configuration records, subconsole, and other miscellaneous functions.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#MSIPL	1.1.3	MSIPLMNT	Main storage IPL phase 3	Initiate sign-on sequence and perform sign-on initialization for configuration records, subconsole, and other miscellaneous functions.
#MSJQ	4.5.1	#MSJQ	Job queue IPL setup	Initialize job queue file during IPL.
#MSNIP	1.1.1	MSNIPMNT	Main storage IPL phase 1	Start main storage IPL by setting up initialization data areas.
		#MSREL	Main storage IPL from diskette	Transfer system library from diskette to disk. **Notes:** 1. #MSNIP begins execution in the system transient area but does a translated transfer to itself in order to run translated. 2. #MSREL is link-edited to #MSNIP as a routine.
#MSPR	8.23.2.1	#MSPR	Security IPL initialization processor	Control security.
#MSRR	8.23.2.1	#MSRR	Resource security IPL initialization	Do the following at IPL: • Perform resource security checking. • Attempt to correct any resource security status errors. • Sets up resource record in SCA.
#MSSC	2.1	#MSSC	IPL command processor mainline	Route control to appropriate command processor routine and wait for more work to do during IPL.
#MSSP	4.5.1	#MSSP	Spool IPL initialization	Allocate, deallocate, reformat or audit spool file. Start spool writers for autowriter.
#MSTRC	1.1.3	#MSTRC	IPL trace reset procedure	Initialize the microcode trace function.
#MSTWA	1.1.2	MSTWAMNT	Main storage IPL phase 2 resident	Initialization configuration record for communications and local and remote terminals.
#OLAF	—	AFA000	Build auto-link segment list	Read $WORK and rewrite to it any R-dots found there.
#OLAH	—	OLAH00	Build cross-reference segment list and set dependency bits	Build the cross-reference segment list.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#OLAJ	—	AJA000	Sort segment list into full overlay candidate structure	Sort segment list into sublists.
#OLAP	—	OLAP00	Overlay design I	Use the segment list as generated by previous phases in calculating the core requirements of a given object program.
#OLAR	—	OLAR00	Overlay design II	Build the overlay segment list.
#OLAT	—	AT000	Overlay linkage editor core MAP phase	Print core usage MAP and error messages on the printer.
#OLBE	—	START	Relocate, resolve externs and build load module	Read the segment list, and build an ESL table, and read the R-modules for each overlay, relocate them, and resolve externs.
#OLBO	—	OLBO00	Overlay linkage editor library control phase	Determine type of output requested, and call the proper phase.
#OLDSF	—	DSF000	Overlay linkage editor diagnosed source file	Add OLE sysprint messages on the end of the diagnosed source file (DSF).
#OLER	—	ER000	Overlay linkage editor error routine	Call SYSLOG to issue a message.
#OLINK	—	INK000	Overlay linkage editor user interface phase	Perform initial processing of user requests for the overlay linkage editor.
#OLI1	—	INK105	Overlay linkage editor user entry phase 2	Accept // PHASE and // OPTIONS control statements, syntax check the statements, and save the information in LOMMON.
#OLI2	—	INK105	Overlay linkage editor user entry phase 3	Accept // MODULE statements from SYSIN, find the module(s), and copy them to $WORK.
OLI3	—	INK000	Overlay linkage editor user entry phase 4	Accept // EQUATE, // CATEGORY, and // GROUP statements from SYSIN and build segment entries in the $SOURCE buffer.
#OLMSG	—	OLMSG0	Error message print phase	Issue error messages for errors detected by other phases of the linkage editor.
#OLYNX	—	YNX000	Overlay linkage editor compiler entry phase	Initialize for overlay linkage editor.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#OPCPD	7.8	START	OLPD communications test	Set up for control storage communications tests and route control.
OPETY	7.8	START	OLPD remote work station entry	Get line number for logical work station ID passed.
#OPFCH	7.8	START	OLPD configuration fetch	Using communications line data configuration data in LDA, build index into 'next' field of control record.
#OPFWA	7.8	START	OLPD WSWA configuration find	Get work station communications configuration data from WSWA and place the data in the LDA.
#OPMOS	7.8	START	OLPD MLCA/SLCA determination	Determine communications adapter type installed.
#OPSP	7.8	START	OLPD driver	Perform mainline routing for OLPD data retrieval modules.
#OPVOF	7.8	START	OLPD vary off successful check	Check to determine if previously issued Vary-Off command was successful.
#OPVON	7.8	START	OLPD vary on successful check	Check to determine if previously issued Vary-On command was successful.
#OPWIF	7.8	START	OLPD work station information	Get and display work station configuration data.
#PDAB	9.7	#PDAB	SDLC station test abnormal termination processor	Terminate SDLC station list.
#PRBND	8.23.2.8	#PRBND	Security bind processor	Check that remote location name passed by APPC subsystem is authorized for System/36 access.
#PRCG	8.23.2.2	X'1000'	Change user profile	Change any or all default user menu, default user library, and default help menu fields in a user profile based on the parameter list supplied.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#PRERR	8.23.1.2 8.23.1.5 8.23.1.6 8.23.1.8 8.23.1.10 8.23.1.11 8.23.1.12 8.23.1.13 8.23.1.14 8.23.1.15	#PRERR	Security error processor	Called by various security utilities to return to mainline code on abnormal terminations.
#PRSD	8.23.2.3	#PRSD	Password security processor	Retrieve user's security file and make sign-on security check.
#PSBG	10.7.5 10.7.7	#PSBG	BSC presentation services get	Decompress a buffer received from the host into a print, punch, or console output buffer, if requested.
#PSBP	10.7.4	#PSBP	BSC presentation of services put	Compress an input line of data, either console or reader, put it in a BSCA buffer for transmission.
#PSSG	10.7.5 10.7.7 10.8.7	#PSSG	Presentation service get module	Decompress input records with support SCS characters being processed.
#PSSP	10.7.4	#PSSP	Presentation services put routine	Block input records into output records.
#RDDFF	—	#RDDFF	Distributed disk file facility	Perform System/36 disk data management function requested by a System/34 source DDFF. (This module is evoked through the System/36 SSP-ICF Peer subsystem and includes no System/34 user interface.)
#RDSA	4.11.4	#RDSA	Source DDM allocate	Allocate a remote file.
#RDSC	4.11.4	#RDSC	Source DDM close	Close a remote file.
#RDSD	4.11.7	#RDSD	Source DDM delete	Delete a remote file.
#RDSE	4.11.5	#RDSE	Source DDM error handler	Process for errors detected in source DDM modules; also process abnormal SSP-ICF return codes for the RDICFDM procedure used by SDDM components.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#RDSG	4.11.4	#RDSG	Source DDM get request processor	Map disk data management get DTF into DDM request and send to remote target system. Process response from target system and post results to requester.
#RDSO	4.11.4	#RDSO	Source DDM open processor	Process disk data management open request for a remote data file.
#RDSP	4.11.4	#RDSP	Source DDM put request processor	Map disk data management put DTF into DDM request and send to remote target system. Process response from target system and post results to requester.
#RDSQ	4.11.2	#RDSQ	DDM source RENQ/RDEQ	Enqueue or dequeue the remote file locally, and transmit equivalent request to remote system's target DDM.
#RDSU	—	#RDSU	DDM suspend	Perform initiator processing for remote files owned by a JCB that is being suspended by the initiator to give control to a NRT or MRT program's JCB.
#RDSX	4.11.1	#RDSX	DDM extract	Search NRD for file label specified by VTOC access. Process file format 1 as specified by caller of VTOC access.
#RDSZ	4.11.5	#RDSZ	DDM release	Perform termination processing for remote files by releasing unneeded ICF sessions and dequeueing, freeing, and unlocking nonshared remote files.
#RDTA	4.11.10	#RDTA	Target DDM allocate	Process an allocate request from a remote system's SDDM.
#RDTC	4.11.10	#RDTC	Target DDM close	Process a close request from a remote system's SDDM.
#RDTD	4.11.6	#RDTD	Target DDM delete	Process a delete request from a remote system's SDDM.
#RDTM	4.11.10	#RDTM	Target DDM mainline	Perform mainline processing for requests received from the remote system's SDDM.
#RDTO	4.11.10	#RDTO	Target DDM open processor	Process disk data management open request for a local file received from a remote system.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#RDTQ	4.11.10	#RDTQ	DDM target RENQ/RDEQ	Process enqueue or dequeue request received from remote system.
#RDTT	4.11.10	#RDTT	DDM TDDM relay	Relay file request to next remote system if file is not at this target location. If next target system finds file, relay request response back to original requester.
#RDTX	4.11.10	#RDTX	DDM TDDM extract	Process file requests from remote system's source DDM extract.
#RJBP	10.3.1	#RJBP	MSRJE BSC protocol module	Assign space for an MBB and two IOBs, and allocate the communications line if there is not another BSC MSRJE LOC enabled.
#RJBS	10.7.2		MSRJE BSC subsystem	Perform BSC protocol subsystem functions.
#RJBT	—	#RJBT	BSC error termination exit processor	Process BSC subsystem abnormal terminations.
#RJCS	10.7.5 10.7.7	#RJCS	Forms control table search routine	Search the control table for a carriage or file build entry whose forms name matches the host forms name given in the search parameter list.
#RJCTU	10.7.6	RJTABLE	Forms control table utility	Build an indexed file containing forms control information and disk file creation information.
#RJFILE	10.7.7	#RJFILE	Remote job disk file utility	Perform MSRJE disk file processing.
#RJIE	10.7.1	#RJIE	MSRJE BSC interrupt handler error processor	Process every MBB on the MBB queue.
#RJII	10.7.1	#RJII	MSRJE BSC interrupt handler link initiate	Enable BSC adapter.
#RJIM	10.7.1	#RJIM	MSRJE BSC interrupt handler mainline	Perform input for and handle output from the BSC adapter.
#RJIT	10.7.1	#RJIT	MSRJE BSC interrupt handler link termination	Disable BSC adapter.
#RJMSG	10.7.4	#RJMSG	MSRJE message processor	Display messages for MSRJE utility or subsystems and return operator reply when requested.
#RJOI	10.7.5	#RJOI	MSRJE output task initialize	Perform initial setup for MSRJE output task mainline.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#RJOM	10.7.5	#RJOM	MSRJE output task mainline	Perform MSRJE printer/punch functions.
#RJOT	10.7.5	#RJOT	MSRJE output task termination processor	Perform abnormal termination for MSRJE printer/punch.
#RJRC	10.7.4	#RJRCSTRT	MSRJE reader/console command processor	Interpret MSRJE utility control commands.
#RJRI	10.7.4	#RJRINIT	MSRJE reader/console initiation	Perform reader and console initiation.
#RJRK	10.7.4	RJRKSTRT	MSRJE reader/console close processor	Close and deallocate a file/library member.
#RJRL	10.7.4	RJRLSTRT	MSRJE reader/console library statement processor	Process LIBRARY statement.
#RJRM	10.7.4	#RJRMAIN	MSRJE reader/console mainline	Acquire reader sessions to ICF as necessary, process MSRJE commands, and terminate MSRJE reader/console when requested.
#RJRN	10.7.4	#RJRINIT	MSRJE reader/console initiation setup	Initialize data areas and transfer to initiation.
#RJRR	10.7.4	RJRRSTRT	MSRJE reader/console readfile processor	Process READFILE statement.
#RJRT	10.7.4	#RJRTERM	MSRJE reader/console termination processor	Terminate MSRJE either normally or abnormally.
#RJRX	10.7.4	#RJRX	MSRJE reader/console syntax checker	Load and branch to the syntax checker, and syntax check MSRJE commands.
#RJSBC	—	'1000'X	MSRJE SNA bind check	Check bind images received from C/SNA and report acceptability.
#RJSET	—	'1000'X	Perform abnormal termination of the MSRJE SNA subsystem	Perform abnormal termination function.
#RJSNA	10.7.3		MSRJE SNA ICF subsystem	Perform interface functions between MSRJE device drivers and C/SNA.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#RJSXT	—	RJSLSAX RJSLSIX RJSLUCX RJSLTPX RJSLSRX RJSLSSX RJSLERX RJSLESX RJSINLX RJSABDX RJSENDX RJSCANX RJSINAX	MSRJE SNA SSP-ICF subsystem	Perform non-critical path interface functions between MSRJE subsystem and C/SNA.
#RJTA	10.3.1	#RJTA	MSRJE SNA protocol processor	Assign and initialize the control blocks and line buffer for a current location during enable.
#RJTB	—	#RJTB	MSRJE control block and buffer free	Free all control blocks assigned at enable time.
#RJTF	—	#RJTF	Forms set change processor	Process forms set manipulation requests.
#RPDD	—	#RPDD	Data dictionary existence test	Determine if data dictionary named in LDA exists; place response in LDA. Called by RPG and COBOL procedures.
#RPSPD	—	BEGIN	Spool queue print file delete	Delete the compiler listing that was held as a print file on the spool queue by the RPGONL, FORTC or COBONL procedure.
#RRA1	8.23.2.5	1000	Resource security check #1	Check user for authorization to use a file or library.
#RRA2	8.23.2.6	X'1000'	Resource security check #2	Check user authorization to attach to a task that may already have secure resources allocated to it.
#RRA3	8.23.2.7	X'0000'	Resource security check #3	Check user authorization to rename a file.
#RRA4	8.23.2.8	X'0000'	Resource security check #4	Check user for authorization to use a folder member.
#RWAB	—	#RWAB	Remote work station abnormal termination	Process a remote work station processor check or 2K-page failure within the remote work station task or SDLC.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#RWAS	9.6.1 9.6.2.1 9.6.2.2 9.6.2.3 9.6.3 9.6.4		RWS mainline task	Perform data management processing.
		#RWCS	RWS send DFC commands	
		#RWMG	RWS message handler	
		#RWPR	RWS printer PUT	
		#RWSC	RWS session control	
		#RWSS	RWS SNA SSCP-LU flow processor	
		#RWSV	Save screen on disk	
		#RWTS	RWS SNA	
#RWDD	9.6.2.1		RWS read data to disk	Write WSQS overflow data from display screen to disk.
#RWDL	9.6.1	#RWDL	Remote work station controller download processor	Issue and process all commands required to download 5294 Control Unit.
#RWER	9.6.4		RWS data flow control (DFC) command handler	Process SNA DFC commands on LU-to-LU flow.
#RWLK	9.6.4		RWS SDLC error handler	Process all error IOBs except open station, disconnect, and terminate.
#RWNR	9.6.2.3 9.6.4		RWS negative response handler	
#RWRL	9.6.1	#RWRL	Remote work station downline loader patch table search	Get patch table data for #RWDL and verify appropriate system type and code level.
#RWVF	9.6.3 9.6.4		RWS initialization/termination	
#RWVO	9.6.1 9.6.4		RWS initialization	
#RWVY	9.6.1 9.6.3		Process vary command	

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#SBAT	10.6.8	#SBAT	Abnormal termination exit	Process after serious Finance subsystem errors.
#SBDE	—	#SBDE	Data encryption/decryption subroutine	Encrypt or decrypt user data for Finance subsystem.
#SBFS	10.6.8	SBACxxxx	Finance subsystem	Interface between all System/36 functions and the Finance subsystem.
#SBIT	—	#SBIT	Diskette image transfer transient	Send diskette image to a 3601 communications controller.
#SBOP	10.3.1	#SBOP	SDLC line and task open routine	Perform enable-time processing or open functions for the subsystem. Allocate and initialize resources.
#SDAM	9.2.4	#SDAM	Abnormal termination for mainline	Perform initial abnormal termination for errors detected by SDLC mainline. Route terminate/detach IOB to #SDAT.
#SDAT	9.2.4	#SDAT	Abnormal termination for initialization/termination	Perform initial abnormal termination for errors detected by SDLC initialization or termination, and perform abnormal termination for all SDLC processes.
#SDIT	9.2.1.1 9.2.1.2		SDLC initialization/termination task	Process line open requests from SNA, handle open station and enable IOBs from the SNA, process all messages for SDLC, and complete processing for terminate/detach from SDLC.
#SDLC	9.2.2.2 9.2.1.1 9.2.1.2		Mainline SDLC subtask	
#SDRI	9.2.3 9.10.1.1 9.10.1.2		SNA-to-DLC interface loader	Assign an area of SQS and load the resident interface program. (The resident interface processes all IOBs from SNA for SDLC, X.25, or LAN.)
#SIVD	4.11.1	#SIVD	DDM NRD access	Get network resources directory (NRD) entry from #NRD.FLE file for requested remote file.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#SPALC	4.5.2	#SPALC	Spool allocate	Allocate spool intercept control blocks and spool file segments.
#SPATW	4.5.1 4.5.2	#SPATW	Spool writer attach	Attach spool writer.
#SPCLO	4.5.2	#SPCLO	Spool intercept close	Complete the intercepting of a print file so it is no longer associated with the creating task.
#SPDBR	4.5.3	#SPDBR	Spool writer data blanker	Blank ideographic spool print data.
#SPINT	4.5.2	#SPINT	Spool intercept	Intercept the print data (included in #DPDM).
#SPMIC	4.5.2 4.5.3	#SPMIC	Spool message handler	Handle spool intercept and spool writer messages.
#SPQMG	4.5.2 4.5.3	#SPQMG	Queue manager	Spool queue management.
#SPRT	4.5.2 4.3.1	—	Spool/printer data management router resident	If possible, process print request; otherwise, set up parameter list mapping for #DPDM, to simplify transfer SVC instruction to/from printer data management.
		PRT#DPDM		Set up registers for implicit mapping to DTF and user buffer for transfer to #DPDM
		PRTSPINT		Set up registers for implicit mapping to the intercept buffer when spool intercept transfers to #DPDM.
#SPWDB	4.5.1	#SPWDB	Writer descriptor block (WDB) builder	Build writer descriptor block (WDB) for spool.
#SPWRT	4.5.3	#SPWRT	Spool writer	Print entries from the spool file.
#STLOAD	7.1.0 7.1.1	#STLOAD	SYSTEST loader	Load programs from the diagnostic diskette into main storage to run under the SSP.
#ST01	7.6	$START	Test request main storage verification program	Prompt operator for test request options: display verification, printer verification, configuration data, ERAP, or link test. Route control based on option selected.
#ST02	7.6	$START	Test request display verification program	Prompt for display verification options and execute test selected.
#ST03	7.6	$START	Test request printer verification program	Prompt for printer verification options and execute test selected.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#ST04	7.6	$START	Test request configuration verification program	Display configuration data for online local and remote work stations (excludes native-attach printers).
#SVAT	6.2.11 6.2.12	#SVAT	Task Attach/Reorg/Detach	Allocate necessary resources to attach a new task to the system. Change task ORG point when task execution passes SSP to user, or when the task is to be detached. On EOJ, free all task resources.
#SVDMP	6.2.15	#SVDMP	Snap dump transient	Snap dump storage within limits or snap dump entire region.
#SVEJR	4.5.3	#SVEJR	EOJ purge of console queue for I/O messages	Check the console SYSLOG queue and the console matrix looking for any I/O messages associated with the task being canceled.
#SVERP	2.5 2.13	#SVERP	Error processing transient	Allow I/O devices other than the work stations to issue error messages to the system console. Route control for I/O error recovery transient.
#SVESP	2.14	#SVESP	Display station error router	Allow display station error messages to be sent to the system console. Route control for display station errors.
#SVINF	6.2.14	#SVINF	Information retrieval transient	Support the $INFO macro.
#SVML	2.14	#SVML	Error task mainline	Initialize I/O device error processing whenever an error post is received.
#SVNRY	2.13	#SVNRY	Display station error recovery for device not ready	Attempt recovery from operator error mode command reject exceptions.
#SVQLK	—	#SVQLK	Quick lock/release resident	Interlock or release the specified system resource.
#SVRD	2.13	#SVRD	Command reject ready routine	Retrieve command rejected records from a work station command reject file.
#SVTS	4.5.3	#SVTS	System time stamp	Place a time stamp in the history file and calculate elapsed time.
#SVTUB	1.1.2 1.1.3	SVTUBMNT	TUB build transient	Allocate and initialize work station work area (WSWA) and initialize TUB for the device being varied on.
#SVTX	—	#SVTX	Task work area extension transient	Provide recovery when the TWA runs out.
#SVWER	2.13	#SVWER	I/O error message handler	Allow I/O devices to issue error messages to the system console.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#SVWSR	2.5 2.11 2.13 2.15	#SVWSR	Display station error recovery	Attempt recovery from: hardware malfunctions, software errors, command reject exceptions, and process not-ready-to-ready post.
#TACLO	5.1.1	#TACLO	Tape data management close	Rewind or unload tape on close.
#TAADD	4.8	#TAADD	Tape data management add	Perform tape data management add function.
#TAEOV	4.8	#TAEOV	Tape data management end of volume	Perform tape data management end-of-volume processing.
#TAFND	4.8	#TAFND	Tape data management find	Perform tape data management find function.
#TAGET	4.8	#TAGET	Tape data management get	Perform tape data management get function.
#TAOPN	3.3.1	#TAOPN	Tape data management open	On open, build tape IOB and set up physical buffer.
#TAPUT	4.8	#TAPUT	Tape data management put	Perform tape data management put function.
#TARBK	4.8	#TARBK	Tape read block data management	Read a full or partial block of data from the hardware buffer into the physical buffer.
#TAWBK	4.8	#TARBK	Tape write block data management	Write a full or partial block of data to tape from the physical buffer.
#TMALST	8.35	#TMALST	FMS archive list module	Read a save file (archived member file) from diskette or tape and list the information on the SYSLIST device.
#TMARC	8.35	#TMARC	FMS archive member	Copy member(s) of a folder to a file on diskette or tape. If specified, delete member after archiving.
#TMBDA	4.10.4	#TMBDA	FMS build DTA extent	Build a DTA extent for the specified folder.
#TMBLD	1.1.7	#TMBLD	IPL FMS folder rebuild	Determine which folder must be rebuilt and call #TMRBD for each one.
#TMCLS	4.10.3	#TMCLS	FMS close member and library	Close folder or folder member(s).
#TMCPY	4.10.4	#TMCPY	FMS copy member	Copy a member from a folder to the same or another folder, creating a new or replacing an existing member.
#TMCRT	4.10.4	#TMCRT	FMS create folder	Create a new folder.
#TMDAR	4.10.3	#TMDAR	FMS delete associate records	Delete one, several, or all of the associate records in a member.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#TMDT	4.10.3	#TMDT	FMS delete folder	Delete some or all of the folder stream in a member.
#TMFAC	8.35	#TMFAC	FMS archive member	Copy a folder member to a disk file.
#TMFD	8.35	#TMFD	FMS save folder	Save a folder to a disk file.
#TMFI	8.35	#TMFI	FMS restore folder	Restore a folder from a disk file.
#TMFRT	8.35	#TMFRT	FMS retrieve member	Retrieve a folder member from a disk file.
#TMGAR	4.10.3	#TMGAR	FMS get associate records	Get associate record (miscellaneous folder information) from a folder.
#TMGDD	4.10.3	#TMGDD	FMS get data descriptors module	Retrieve one or more DDA fields.
#TMGIT	4.10.3	#TMGIT	FMS get folder information	Get information about the folder units in a member.
#TMGT	4.10.3	#TMGT	FMS get folder	Get folder units (DW/36-folder-stream format) for a member in a folder.
#TMIO	4.10.3	#TMIO	FMS I/O router	Route for specialized FMS I/O modules.
#TMIPL	1.1.4	#TMIPL	FMS IPL initialization	Create FMS system work spaces.
#TMLDD	4.10.3	#TMLDD	FMS list data descriptors module	List one or more DDA fields.
#TMLST	4.10.4	#TMLST	FMS list folder	List information about a folder or group of folders.
#TMMDT	4.10.4	#TMMDT	FMS delete member	Delete a member, delete the text portion of a member, or delete a member without deleting its entries in the physical directory.
#TMMRN	4.10.4	#TMMRN	FMS member rename	Rename a member within a folder.
#TMMV	8.35	#TMMV	FMS move folder	Move folder from one location to another on disk.
#TMOPN	4.10.3	#TMOPN	FMS open member and folder module	Open requested member and folder.
#TMPAR	4.10.3	#TMPAR	FMS put associate record	Put an associate record to a member in a folder.
#TMPDD	4.10.3	#TMPDD	FMS put data descriptors	Put data descriptors or a folder description to the DDA portion of a folder.
#TMPT	4.10.3	#TMPT	FMS put folder	Put DW/36 folder stream to a member in a folder.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#TMRAR	4.10.3	#TMRAR	FMS read associate record	Read an associate record (no previous file open required).
#TMRBD	1.1.7	#TMRBD	FMS folder rebuild	Rebuild folders.
#TMRORG	8.35	#TMRORG	FMS reorganize folder module	Reorganize a folder removing as much unused space as possible and optionally changing its size.
#TMRS	8.35	#TMRS	FMS restore folder module	Restore a folder from a diskette or tape file.
#TMRTV	8.35	#TMRTV	FMS retrieve archived member	Retrieve an archived folder member from a diskette or tape file.
#TMSDD	4.10.3	#TMSDD	FMS search data descriptors	Search one or more DDAs against search argument passed in parameter list.
#TMSMD	4.10.3	#TMSMD	FMS search additional data descriptors module	Retrieve search information collected on a previous search.
#TMSR	1.1.7 4.10.3	TMS00000	FMS common subroutines	Process subroutine calls for common FMS functions.
#TMSV	8.35 4.10.4	#TMSV	FMS save folder	Save folder(s) to a diskette or tape file.
#TMTRM	5.2.1	#TMTRM	FMS termination	Perform FMS task termination: dequeue DSC buffer, deallocate partially allocated DTAs, back-out changes to partially renamed member, unlock FMS resources, close folders and members, delete partial members, and dequeue and free FMS control blocks.
#TMXT	4.10.3	#TMXT	FMS extend data descriptor area	Build replacement DDA with additional space in areas specified by caller.
#TPMA	4.10.2	#TPMA	Office systems profiles processor	Perform user-specified operations on an office systems profile or profile folder: open profile and profile member, get, put, or delete component subprofile, close profile member and folder, and translate MIC to profile services return code.
#TTAC	4.10.1	#TTAC	DW/36 format to alternative character set processor	Convert input DW/36 text stream to alternative character set output stream for printing.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#TTBS	4.10.1	#TTBS	FFT-DCA to 5219 Printer format processor	Convert FFT-DCA data stream to FFT-DCA output stream for printing on the 5219 Printer.
#TTBW	4.10.1	#TTBW	Transmit WP-EBCDIC data stream	Send WP-EBCDIC data stream to 6670 Printer via batch BSC.
#TTCA	—	#TTCA	Release 1 and 2 text format to DW/36 format conversion aid	Put up screen to prompt for conversion aid requests.
#TTCD	—	#TTCD	Release 1 and 2 text format to DW/36 format conversion aid	Put up screen to prompt for document conversion.
#TTCJ	—	#TTCJ	Release 1 and 2 text format to DW/36 format conversion aid	Initialize to run conversion on job queue job.
#TTCV	—	#TTCV	Release 1 and 2 text format to DW/36 format conversion aid	Convert release 1 and 2 document to DW/36 document.
#TTGR	4.10.1	#TTGR	Source to DW/36 format processor	Convert input stream source and procedure members to DW/36 text format for internal (utility) processing.
#TTIM	4.10.1	#TTIM	FFT-DCA to DW/36 format processor	Convert FFT-DCA input stream source to DW/36 text format for internal (utility) processing.
#TTJM	4.10.1	#TTJM	DW/36 format to FFT-DCA format processor	Convert input DW/36 text stream to FFT-DCA-format output stream for transmission to another system or printing on the 5219 Printer.
#TTKR	4.10.1	#TTKR	RFT-DCA to DW/36 format processor	Convert RFT-DCA input stream source to DW/36 text format for internal (utility) processing.
#TTLR	4.10.1	#TTLR	DW/36 format to RFT-DCA format processor	Convert input DW/36 text stream to RFT-DCA-format output stream for transmission to another system.
#TTPP	4.10.1	#TTPP	Release 1 and 2 text paragraph to DW/36 paragraph conversion processor	Convert input text paragraph to DW/36 folder paragraph for internal (utility) processing.
#TTPR	4.10.1	#TTPR	Release 1 and 2 text format to DW/36 format conversion processor	Convert input text stream to DW/36 folder format for internal (utility) processing.
#TTRT	4.10.1	#TTRT	Office systems transforms root phase 1	Initialize for office systems text format transformation.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#TTR1	4.10.1	#TTR1	Office systems transforms root phase 2	Route for requested office systems text format transformation.
#TTSA	4.10.1	#TTSA	DW/36 format to SCS#1 format processor	Convert input DW/36 text stream to SCS#1 format output stream for printing on the 4214, 5224, or 5225 Printer.
#TTSS	4.10.1	#TTSS	DW/36 format to SCS#2 format processor	Convert input DW/36 text stream to SCS#2 format output stream for printing on the 3262 or 5256 Printer.
#TTWP	4.10.1	#TTWP	DW/36 format to WP/EBCDIC format processor	Convert input DW/36 text stream to WP/EBCDIC output stream for printing on the IBM 6670 Information Distributor.
#USYX	6.2.13	#USYX	Syntax checker	Scan input records, using appropriate syntax specification module, to verify statement syntax and to collect, convert, and substitute statement elements. **Note:** Although this module has only one entry point, parameter list controls enable a caller to select syntax checking functions as if there were two entry points.
#WDDA	4.4.1	#WDDA	Work station data management	Resident version of work station data management. Put logic.
#WDDB	4.4.1 4.4.2	#WDDB	Work station data management	Miscellaneous subroutines for work station data management.
#WDDC	4.4.1	#WDDC	Print op modifier	Transient handles print op modifier.
#WDDG	4.4.2	#WDDG	Work station data management GET routine	Handle status inquiry, invite terminal input, accept terminal input, get terminal input, and release terminal from program functions.
#WDDH	4.4.2	#WDDH	Work station data management routines	Handle process requests for Help and Print keys, key masking functions, and issue messages for Print key request functions.
#WDDK	4.4.2	#WDDK	Work station data management routines	Process requests for the Print key.
#WDDL	—	#WDDL	Work station data management translated routine	Process requests for two read operations.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#WDDP	4.4.2	#WDDP	Work station data management Print key routine	Print key task.
#WDDQ	4.4.1 4.4.2	#WDDQ	Work station data management GET routine	Handle acquire terminal, get terminal attribute functions, issue WSDM SYSLOG halts, stop invite, and retrieve MICs for formats.
#WDHH	—	#WDHH	Work station data management routine	Process high level requests for the Help key.
#WDIGC	4.4.1 4.4.2	#WDIGC	Work station data management IGC support	Scans data stream for IGC data.
#WDOPN	3.3.1	#WDOPN	Work station open routine	Open work station DTFs passed by common open.
#WDRK	—	#WDRK	Roll keys reversal	Assigns direction of roll-up and roll-down keys (evoked by ROLLKEYS procedure).
#WPAS	9.9.1	#WPAS	Display station pass-through source abnormal termination	
#WPAT	9.9.2	#WPAT	Display station pass-through target abnormal termination	
#WPER	9.9.2	#WPER	Display station pass-through error mapping	Perform display station pass-through error mapping for System/38 attached displays.
#WPIN	9.9.1	#WPIN	Display station pass-through source initialization	Perform initial processing for display station pass-through source.
#WPSI	9.9.2	#WPSI	Display station pass-through stop outstanding invites	
#WPTS	9.9.1	#WPTS	Display station pass-through source program	Display station pass-through source mainline processing
#WPTT	9.9.2	#WPTT	Display station pass-through target program	Display station pass-through target mainline processing
#WPVY	9.9.2	#WPVY	Display station pass-through vary on transient	
#XTAT	9.8.3	#XTAT	X.25 abnormal termination mainline	Perform mainline processing for X.25 abnormal termination.
#XTEH	9.8.2	#XTEH	X.25 error handler	Process virtual circuit, link, and hardware errors.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
#XTIT	9.8.1	#XTIT	X.25 initialization termination task	Perform mainline processing for initialization and termination of X.25 tasks.
#XTML	9.8.2	#XTLM	X.25 mainline task	Perform mainline functions for X.25 transmit and receive operations.
@BSCC	—	@BSCC	BSC translate module	Translate ASCII to EBCDIC or EBCDIC to ASCII.
@GSCA7	—	@GSC70	Decimal to binary conversion	Convert columns 13 through 16 of the card work area to binary.
@GSCA9	—	@GSCA9	Calculate length	Operate on the current card in the 39-byte work area.
@GSEG	—	@GSEG0	Automatic work file allocation routine	Allocate work file space and build the entries in the MVF table in COMMON.
@GSZB	—	#GSZB0	Decimal to hex conversion	Convert an unsigned 7-byte zoned decimal number to a 3-byte hex number.
@GSZC	—	#GSZC0	Hex to zone decimal conversion	Convert a 3-byte hex number to a byte decimal number and concatenate a sign to the result.
@GSZD	—	#GSZD0	4-byte hex divide routine	Devide two hex numbers in own work area and upon exit, put the quotient and the remainder in the user's areas.
@GSZE	—	#GSZE0	Bitohex	Convert 4-bit groups to their EBCDIC equivalent.
@GSZF	—	#GSZF0	Bit to bit	Convert 1 byte of memory to printable graphics, 1 bit at a time.
@GSZM	—	#GSZM0	4-byte hex multiply routine	Multiply two hex numbers in own work areas and upon exit, put result in user's product area.
@PRLD	8.23.2.1	X'5000'	Security reload check subroutine	Checks security is password security is active.

Module Name	Chart Number	Entry Point	Descriptive Name	Function
@PRUDM	8.23.1.1 8.23.1.2 8.23.1.3 8.23.1.4 8.23.1.5 8.23.1.6 8.23.1.10 8.23.1.11 8.23.1.12 8.23.1.13 8.23.1.16	(Varies with include module)	User identification file data management routine	Perform file access for user identification file utilities.
@RRDMT	8.23.1.7 8.23.1.8 8.23.1.9 8.23.1.10 8.23.1.11 8.23.1.14 8.23.1.15 8.23.1.16 8.23.2.1	(Varies with include module)	Resource security file data management routine	Perform file access for resource security file utilities.
IPTR	4.3.1	IPTR	Printer I/O Supervisor	Route print requests to spool intercept, RAM print transient, or SNA task via remote printer setup. **Note:** There is no control storage printer IOS function. This module interfaces to control storage printer IOCH.
SUBRSF	—	SUBRSF	RPG II special SMFLOG file access routine	Read variable-length records from the SMFLOG data file created by the SMF data collection program and convert the data to a form usable by an RPG II program.
SUBR30	—	SUBR30	RPG II interface to DES algorithm	Provide the interface between an RPG II subroutine parameter list and the input data stream required by #SBDE.
SUBR31	—	SUBR31	COBOL interface to DES algorithm	Provide the interface between a COBOL subroutine parameter list and the input data stream required by #SBDE.

Appendix A: Data Communications Line Protocols

This appendix contains protocol reference information that is intended for programming service representatives who already know fundamental data communications line protocol. It contains representative examples of line protocols for various System/36 data communications capabilities; these examples are intended to assist the PSR in problem solving situations. Included in this appendix is normal line protocol information for the following:

- BSC Data Communications Line Protocols
  - Batch and SSP-ICF — Figure A-1
  - BSCEL — Figure A-2
  - 3270 — Figure A-3
  - MSRJE — Figure A-4
- SNA Session Protocols
  - Upline Facility (SNUF) — Figure A-5
  - 3270 — Figure A-6
  - MSRJE — Figure A-7
  - Remote Work Station — Figure A-8
  - Finance — Figure A-9
  - Peer — Figure A-10
  - APPC — Figure A-11
  - APPN — Figure A-12
- X.25 Data Communications Protocols — Figure A-13

Note: Additional information on all protocols described in this appendix is contained in the *Data Areas* manual.

BSC DATA COMMUNICATIONS LINE PROTOCOLS

The data communications line protocols for batch BSC and the four SSP-ICF BSC subsystems are included in this section. (Because of their similarity, batch and SSP-ICF are grouped together.) These line sequences are typical of those exchanged between a System/36 and a remote system for the various line types and nodes labeled.

Batch and SSP-ICF

Figure A-1 shows protocols for the following batch BSC and SSP-ICF BSC operations:

- Transmitting: Autocall/Manual Call
- Transmitting: Autoanswer/Manual Answer
- Receiving: Autocall/Manual Call
- Receiving: Autoanswer/Manual Answer
- Transmitting: Point-to-Point, Nonswitched Line
- Receiving: Point-to-Point, Nonswitched Line
- Transmitting: Multipoint Tributary Line
- Receiving: Multipoint Tributary Line
- Transmitting: 3740 Multiple File Support
- Receiving: 3740 Multiple File Support
- Transmitting: 3780 Protocol Format
- Receiving: 3780 Protocol Format
- Transmitting: OFFICE/36 Device Support

Transmitting: Autocall/Manual Call (Point-to-Point, Switched Line)

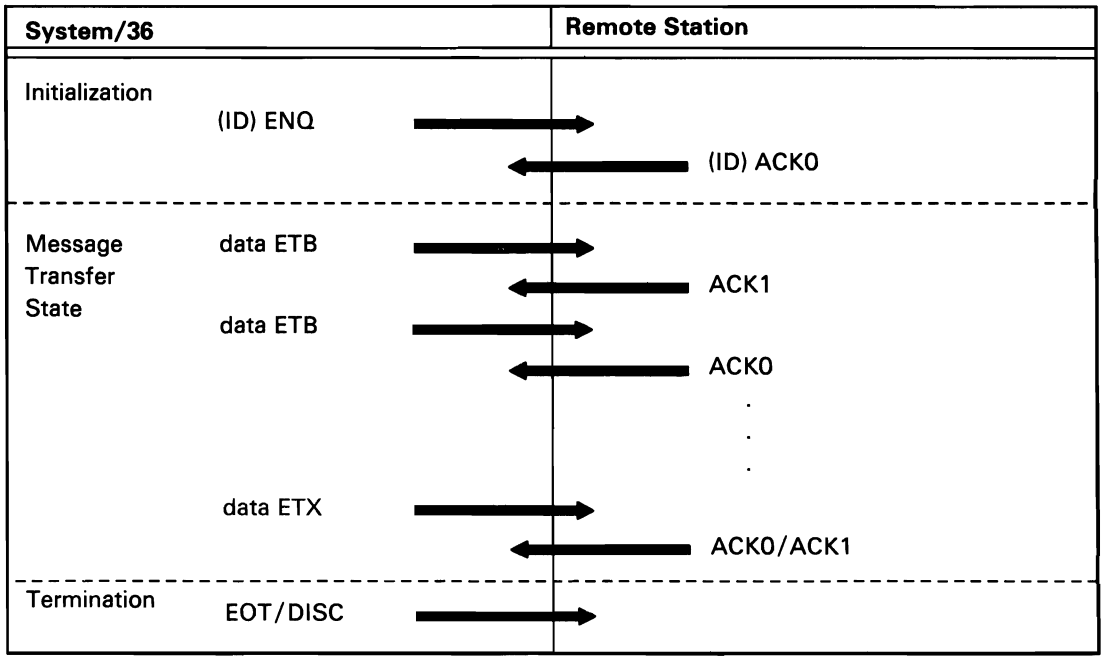


Figure A-1 (Part 1 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Transmitting: Autoanswer/Manual Answer
(Point-to-Point, Switched Line)

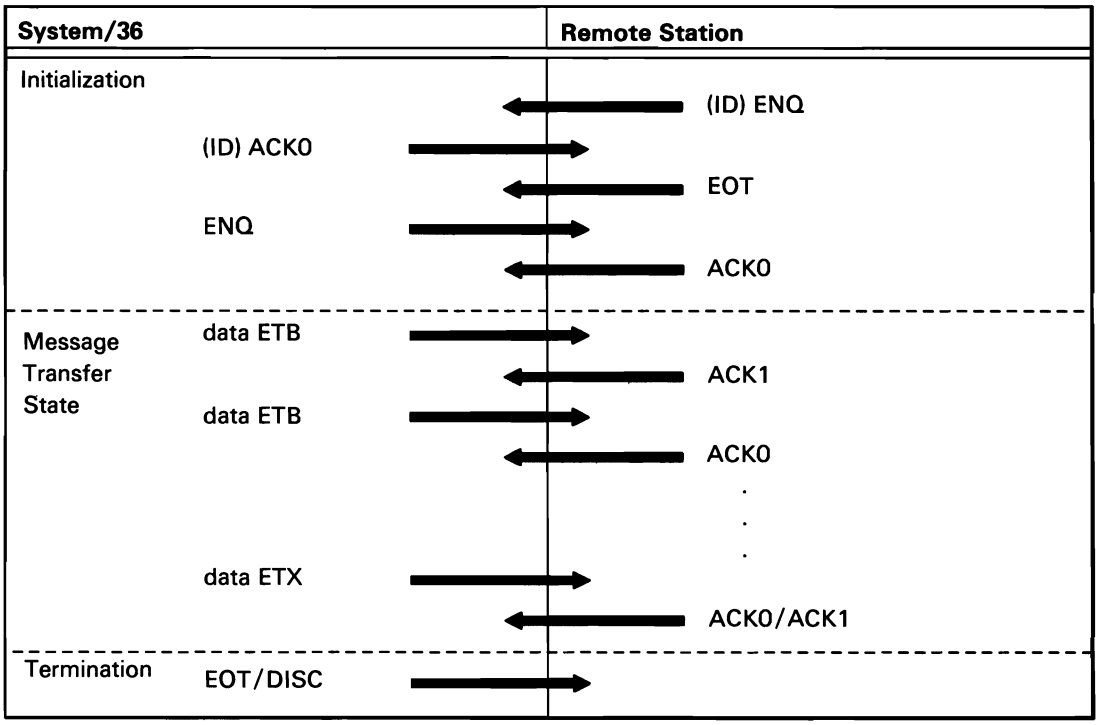


Figure A-1 (Part 2 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Receiving: Autocall/Manual Call (Point-to-Point, Switched Line)

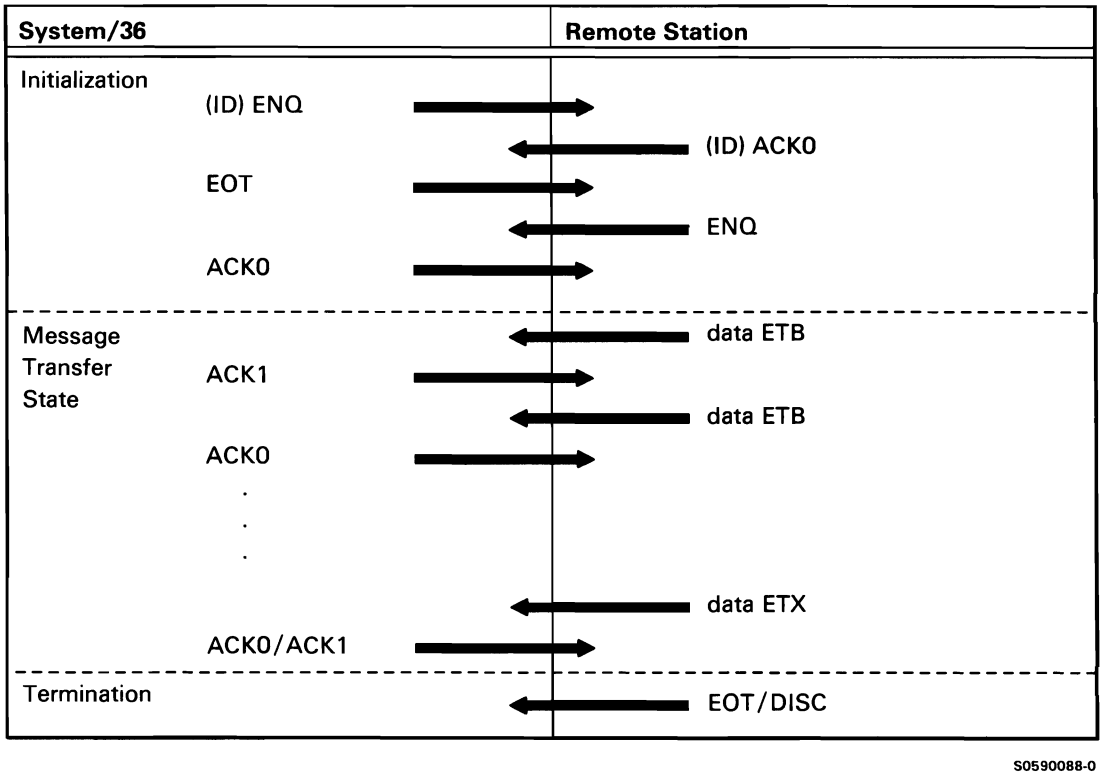
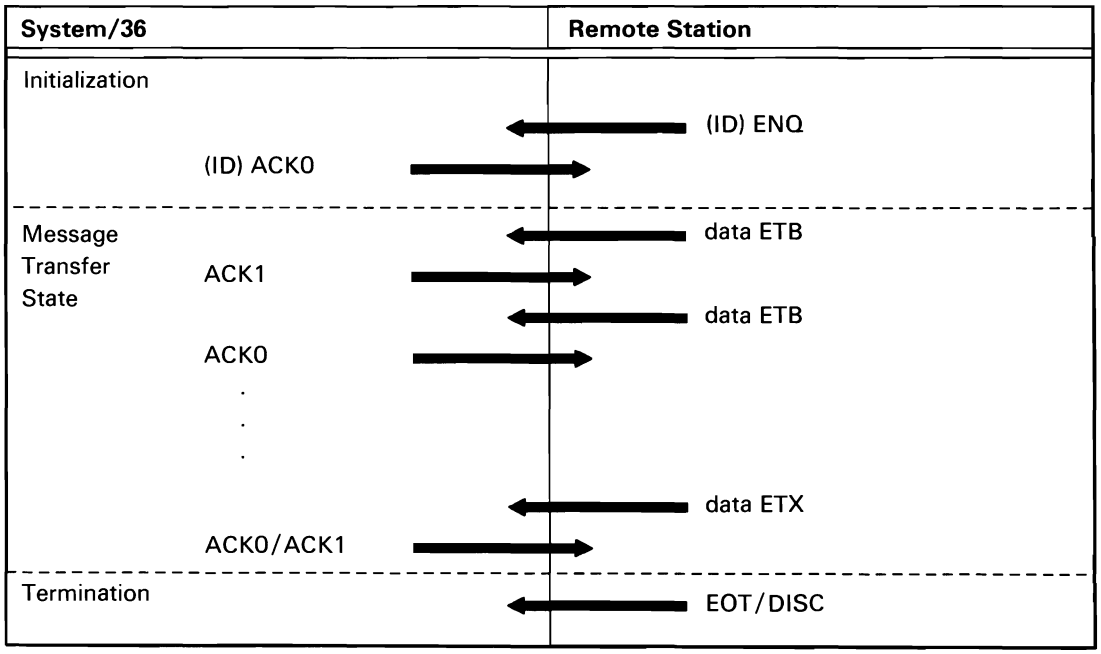


Figure A-1 (Part 3 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

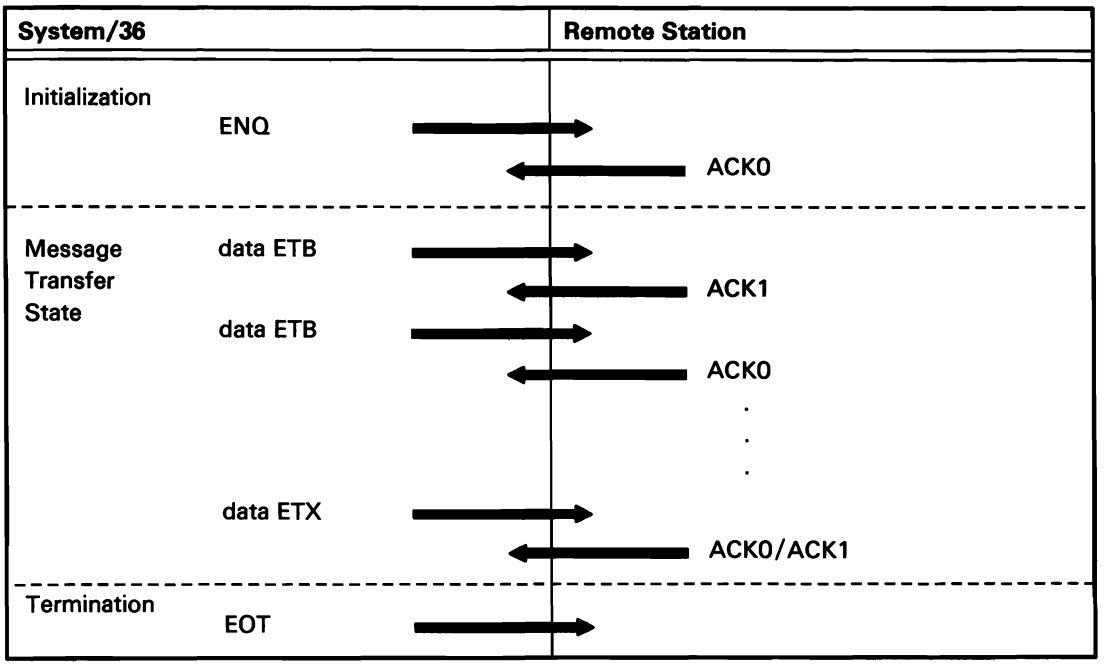
Receiving: Autoanswer/Manual Answer (Point-to-Point, Switched Line)



S0590089-0

Figure A-1 (Part 4 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Transmitting (Point-to-Point, Nonswitched Line)



Note: If ENQs are simultaneously issued by the System/36 and the remote station, contention will continue until the remote station sends ACK0 or the retry count is exceeded.

S0590090-1

Figure A-1 (Part 5 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Receiving (Point-to-Point, Nonswitched Line)

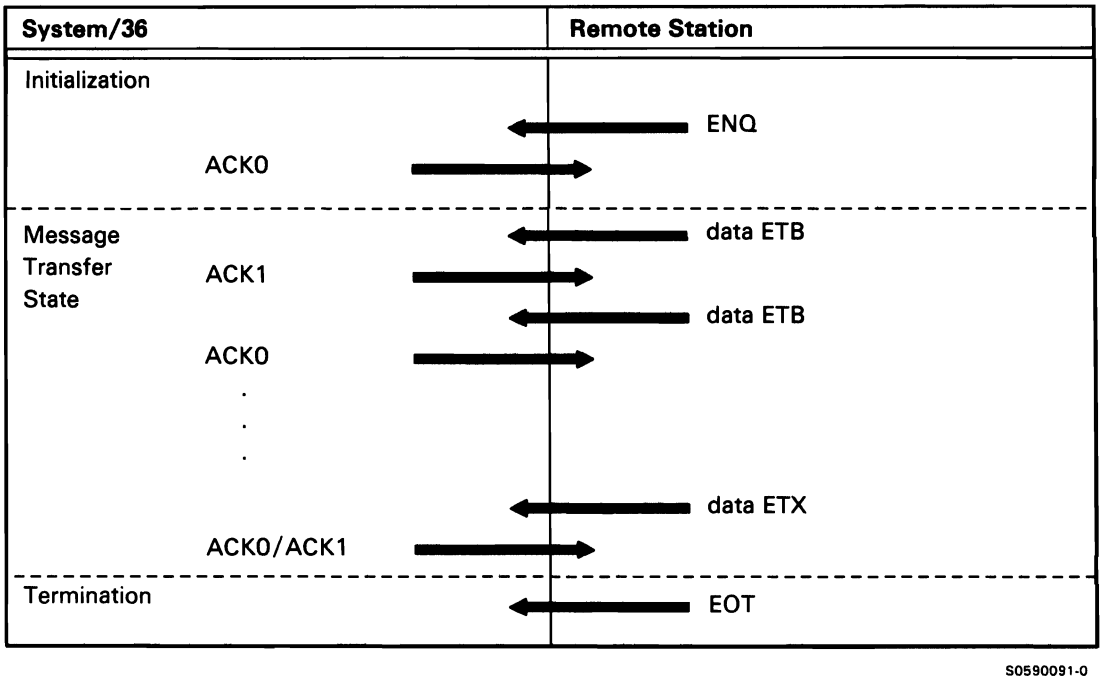


Figure A-1 (Part 6 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Transmitting (Multipoint Tributary Line)

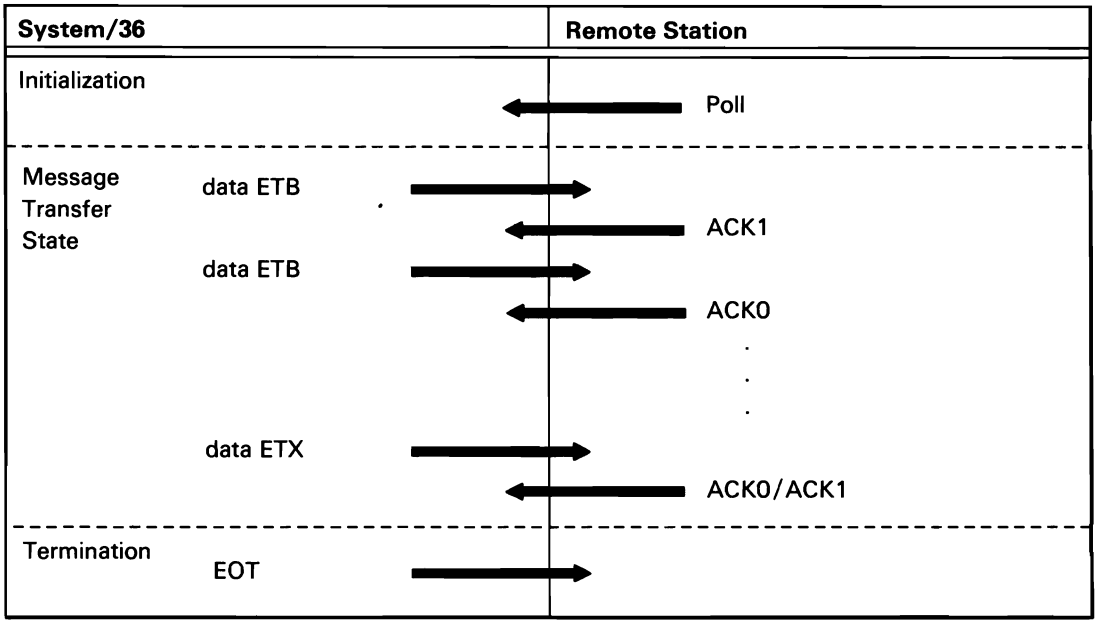


Figure A-1 (Part 7 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Receiving (Multipoint Tributary Line)

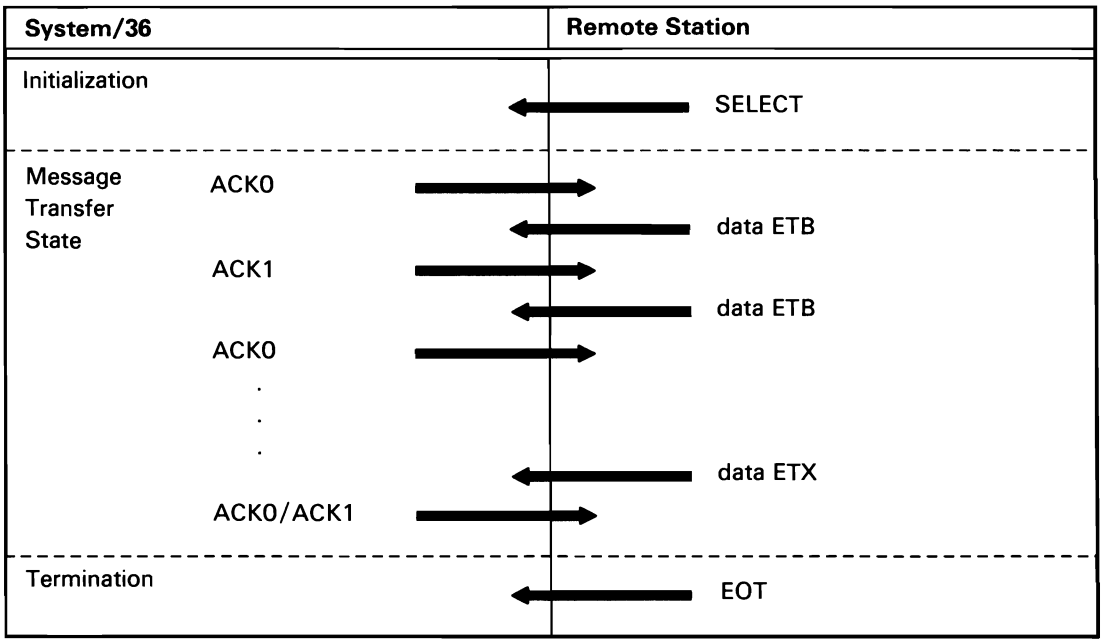


Figure A-1 (Part 8 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

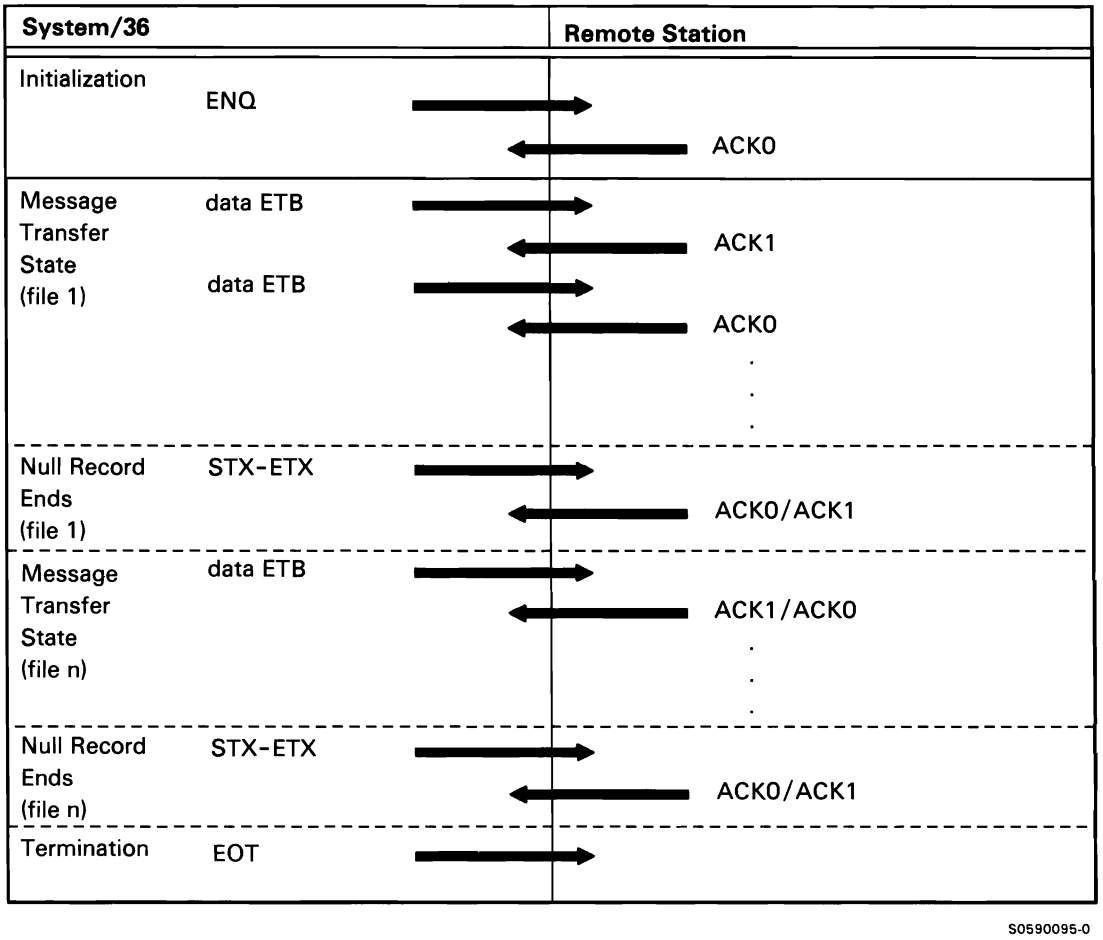


Figure A-1 (Part 9 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Receiving (3740 Multiple File Support)

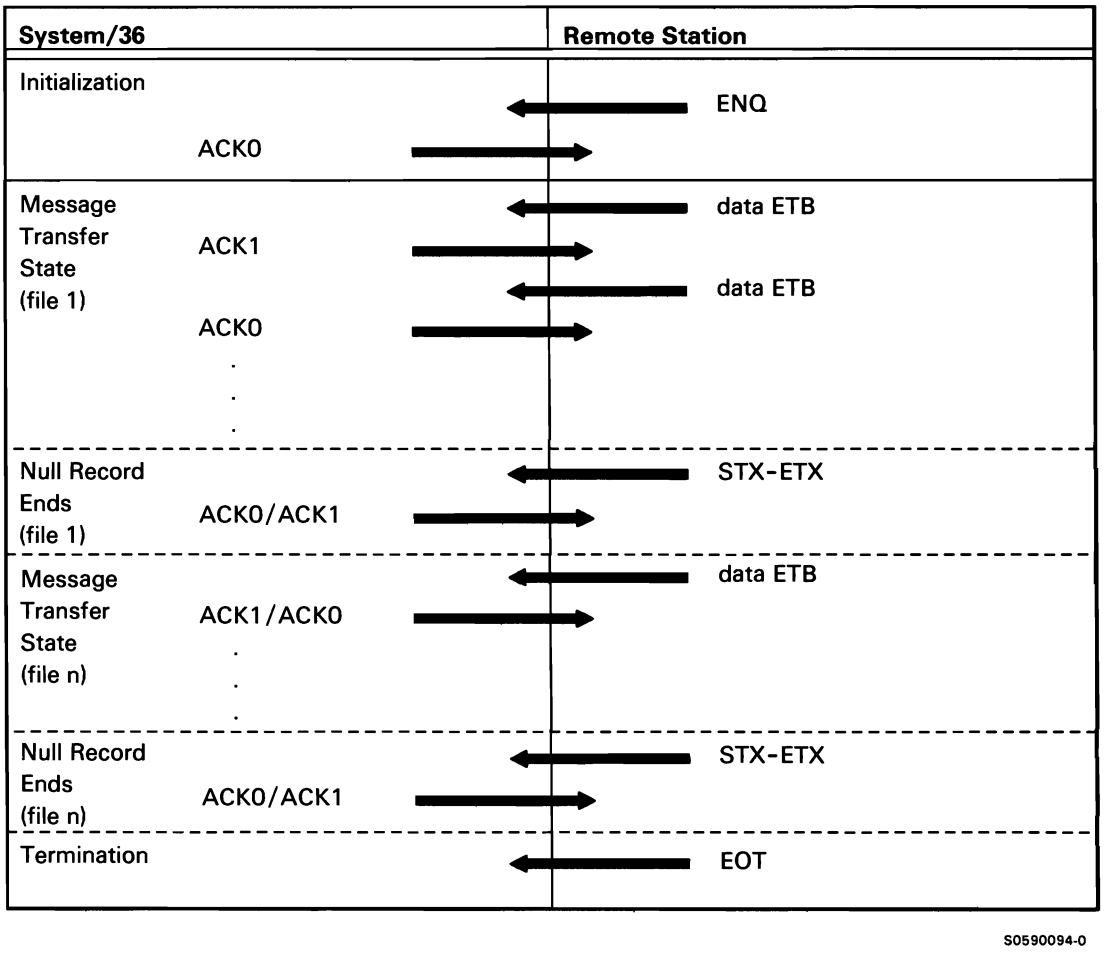
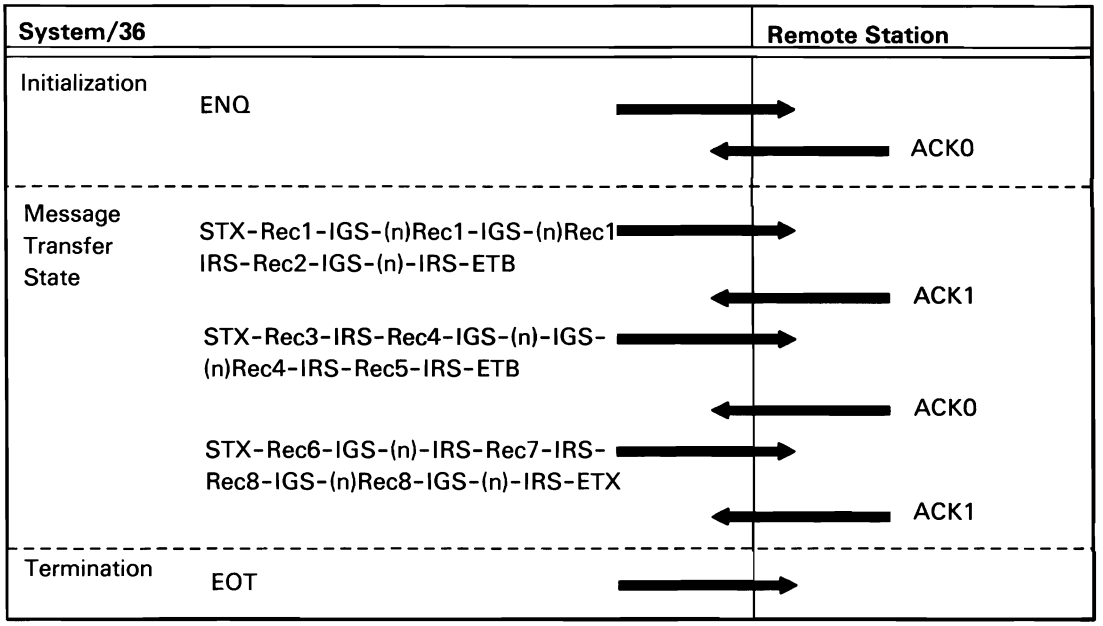


Figure A-1 (Part 10 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Transmitting: 3780 Protocol Format (Blank Compression/Expansion)



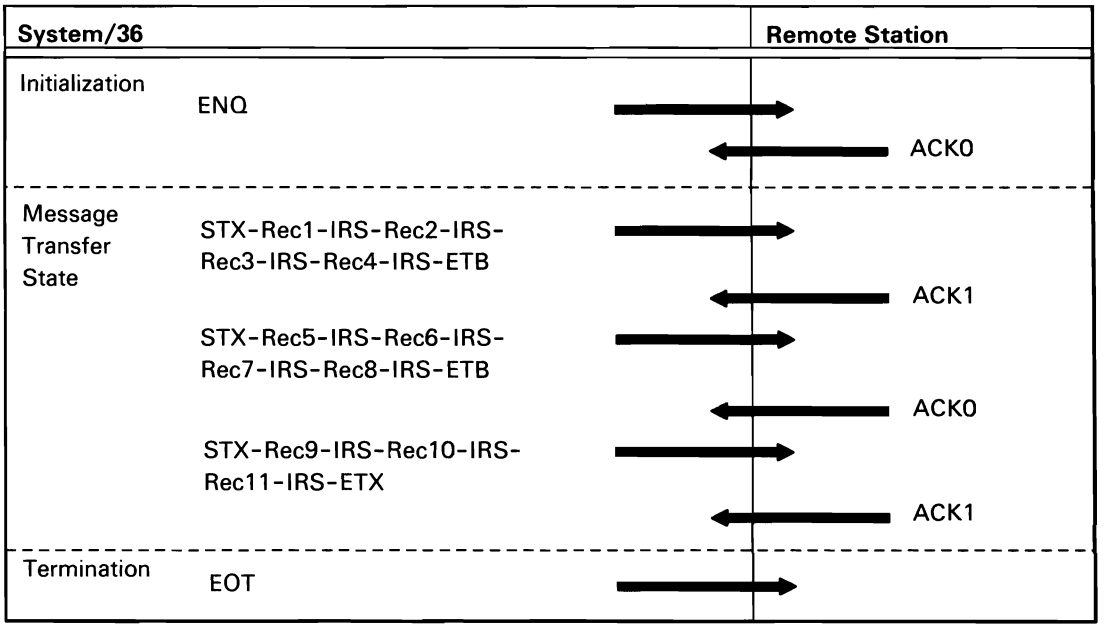
Notes:

1. *The (n) represents the number (1 byte) of compressed blanks. The maximum is 63. A hex 40 is always added to the count. For example, if 20 (hex 14) blanks are compressed, the count (n) is hex (54).*
2. *Records cannot span blocks.*
3. *Compression cannot be used with transparent output.*

S0590096-0

Figure A-1 (Part 11 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

Transmitting: 3780 Protocol Format (Blank Truncation)



Note: The trailing blanks at the end of each record are removed before transmitting the block.

S0590097-0

Figure A-1 (Part 12 of 13). Batch and SSP-ICF BSC Data Communications Line Protocols

This protocol is supported by batch BSC and by the SSP-ICF BSCEL subsystem.

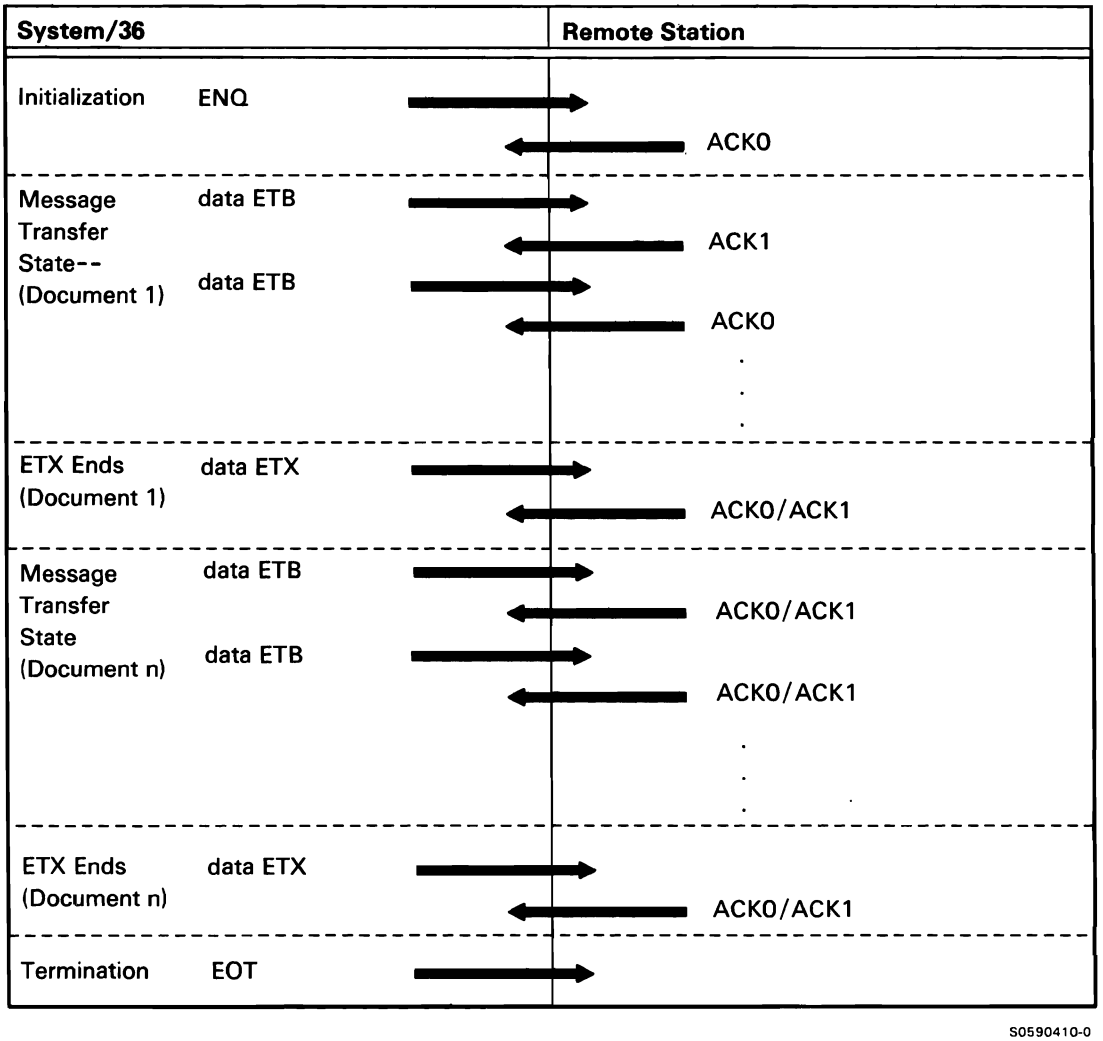


Figure A-1 (Part 13 of 13). Batch BSC (and BSCEL Subsystem) Data Communications Line Protocols

BSCEL Subsystem Office Product Device Support

Figure A-2 shows the line activity that takes place between the System/36 SSP-ICF BSCEL subsystem and the office product device.

Receiving

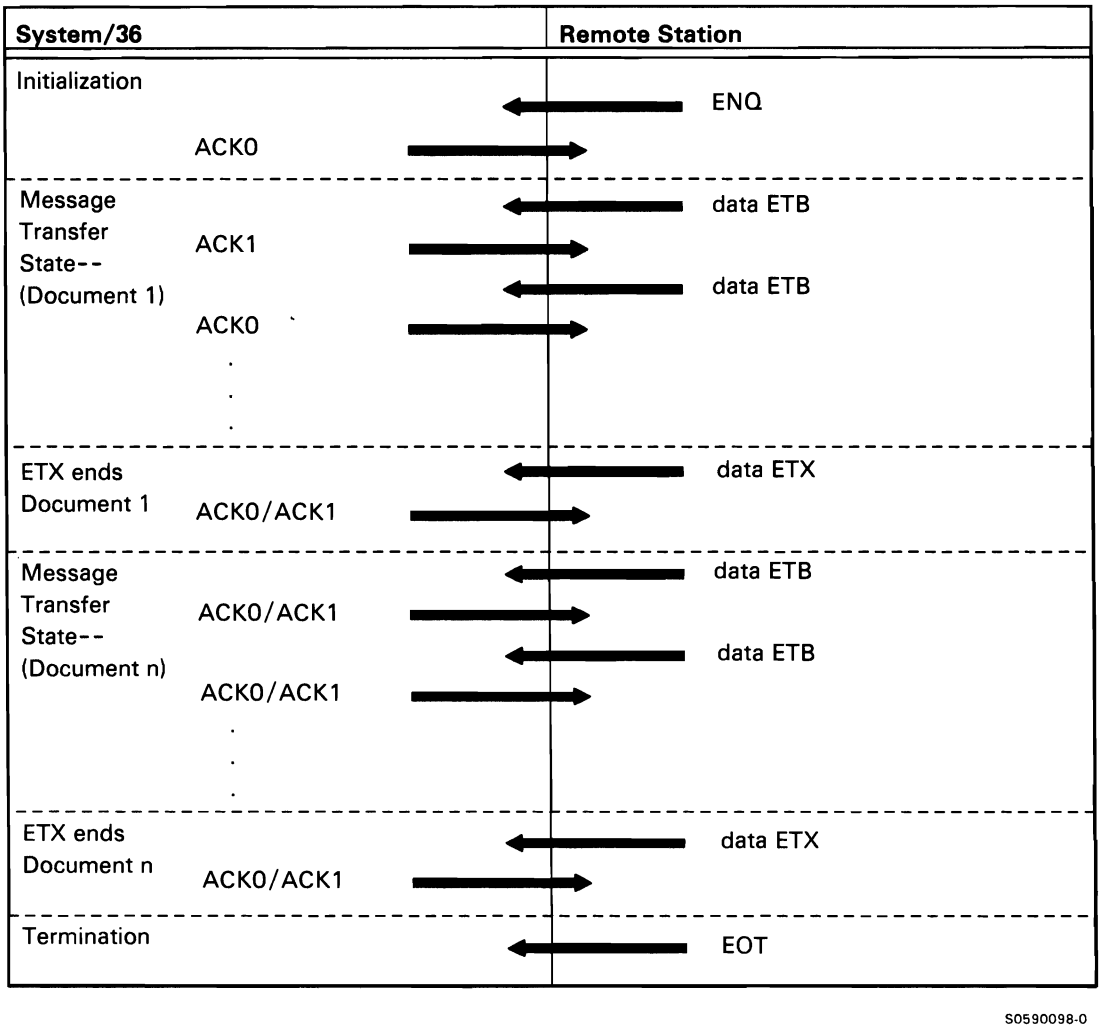


Figure A-2. SSP-ICF BSCEL Subsystem Line Protocols

BSC 3270

The following flow sequences show the line activity that takes place between a host system and a System/36 using BSC 3270 device emulation.

The four major areas of processing are:
- Device Initialization at Host
- Inbound Data Processing
- Outbound Data Processing
- Device Termination at Host

Device Initialization at the Host System, When a Work Station Is Powered On (Normal Device Initialization)

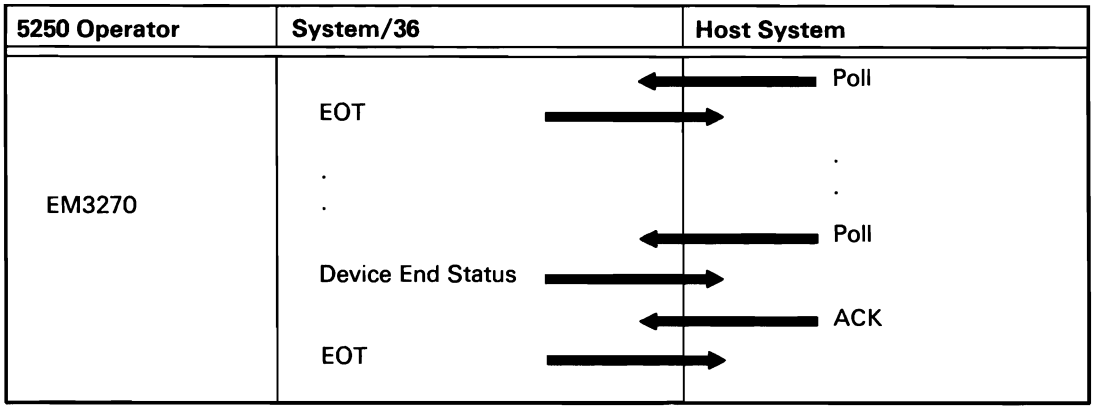


Figure A-3 (Part 1 of 4). BSC 3270 Data Communications Line Protocols

Inbound Data Processing (Data Coming to Host from System/36)

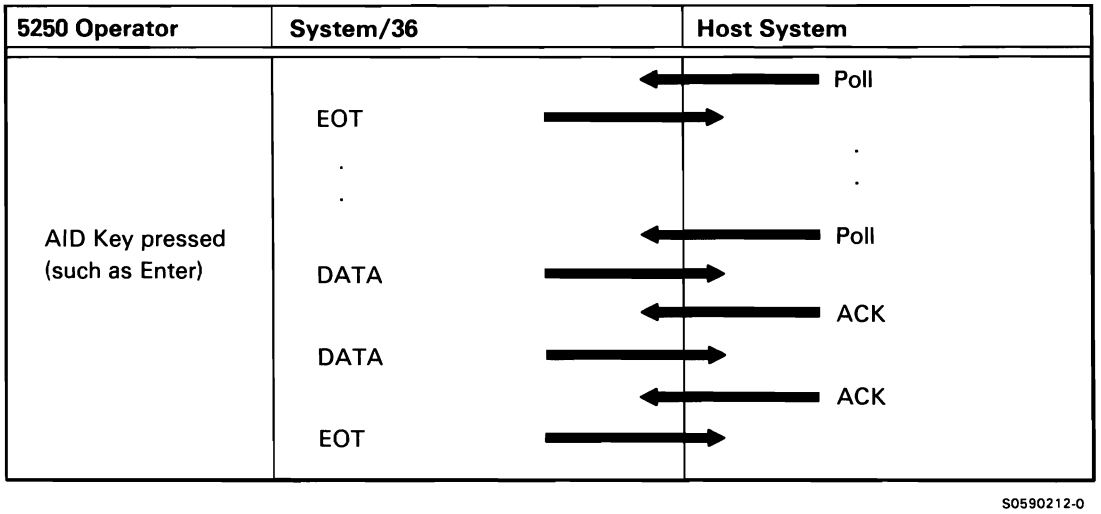


Figure A-3 (Part 2 of 4). BSC 3270 Data Communications Line Protocols

Outbound Data Processing (Data Sent from Host System to System/36)

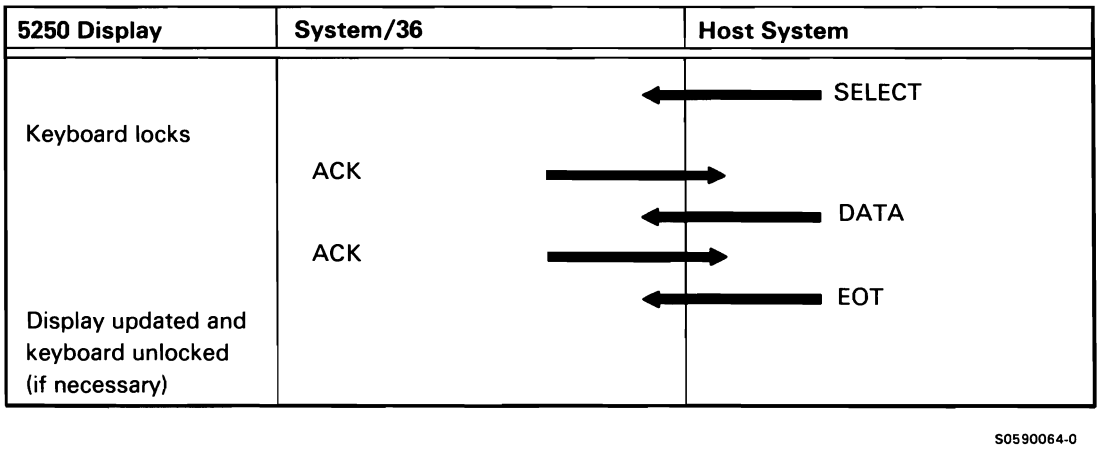


Figure A-3 (Part 3 of 4). BSC 3270 Data Communications Line Protocols

Device Termination at the Host System, As a Terminal Powering Off (Normal Device Termination)

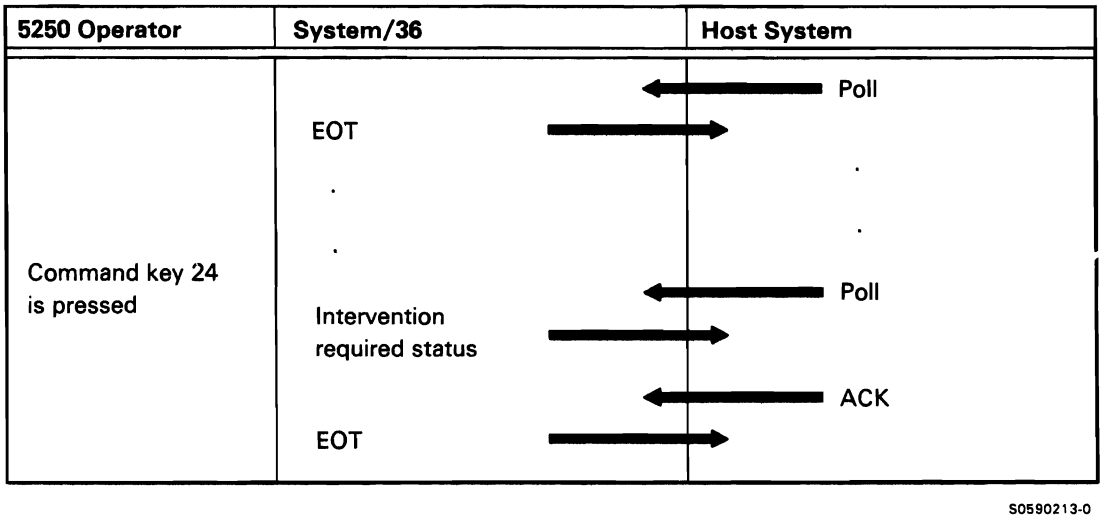


Figure A-3 (Part 4 of 4). BSC 3270 Data Communications Line Protocols

BSC MSRJE

Figure A-4 shows protocols for the following BSC MSRJE operations:

- Sign-On
- Data Transmission
- Sign-Off

Sign-On

System/36	Host System	Comment
ENQ →		System/36 sends ENQ to establish communication.
	← ACK0	Host system responds with ACK0 when ready.
SIGN-ON →		Sign-on sequence is sent to the host system.
	← ACK0	Host system receives sign-on and responds with ACK0.
ACK0 →		System/36 sends ACK0 because there is no data to send to the host system RD1 (see Note 1).
	← ACK0	Host system responds with ACK0 because there is nothing to send from the host System/36 (see Note 1).

S0590065-0

Figure A-4 (Part 1 of 4). BSC MSRJE Data Communications Line Protocols

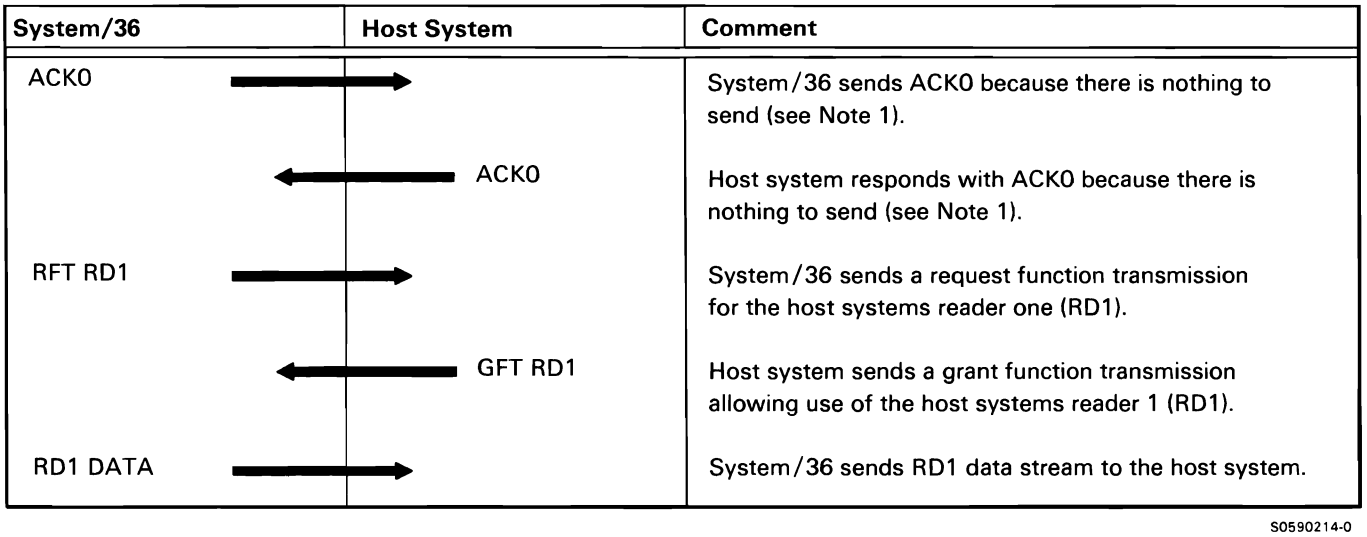
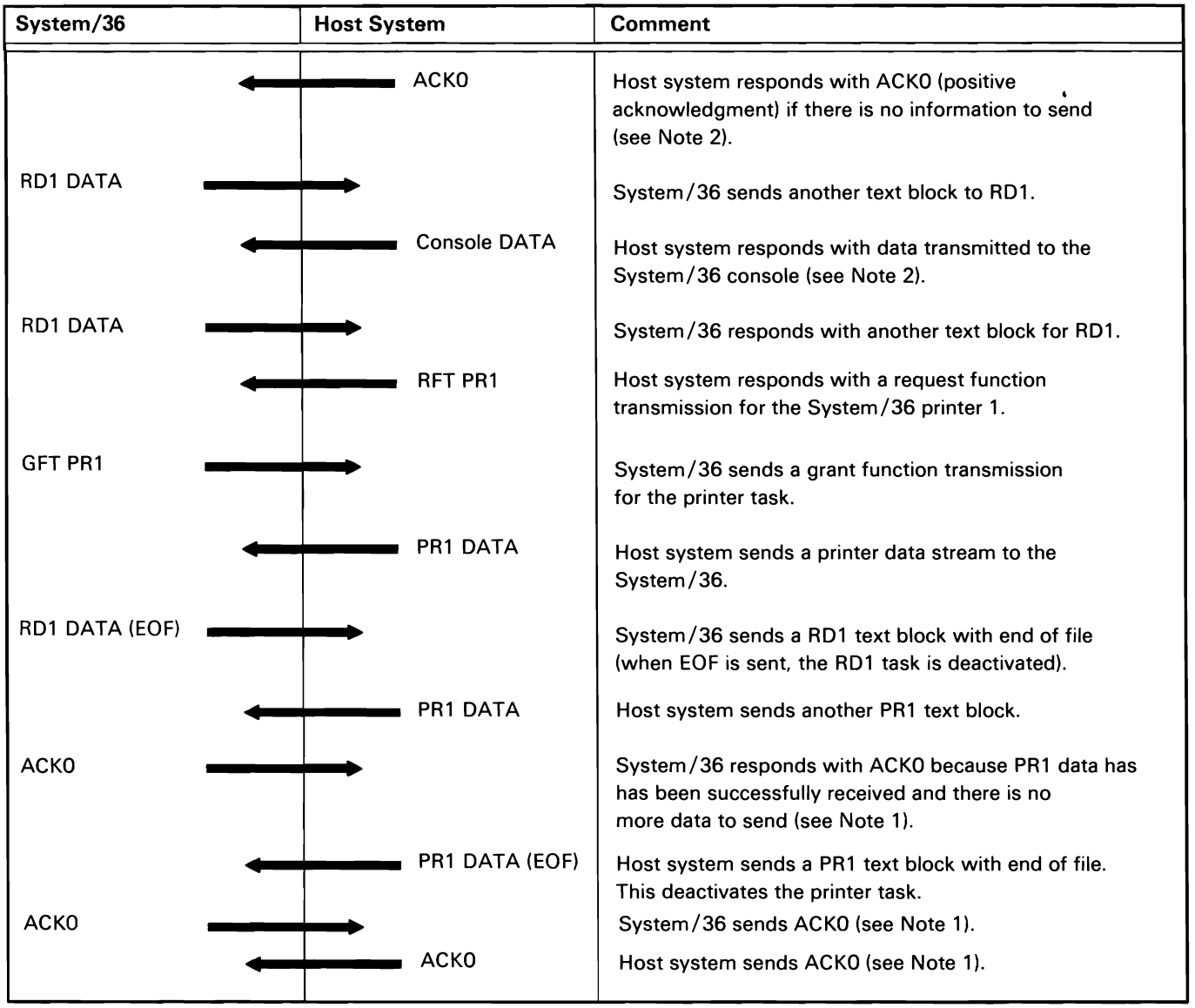
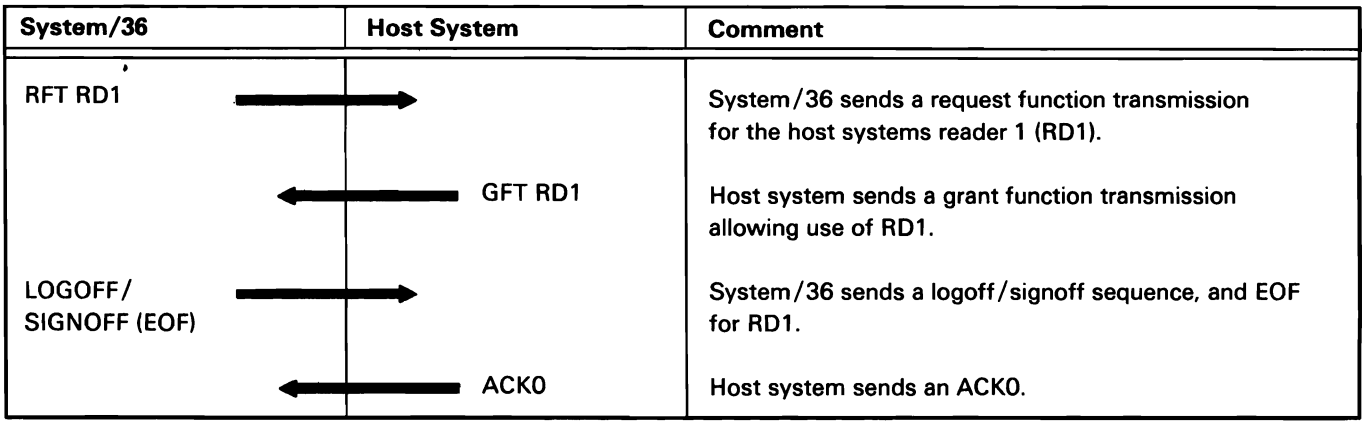


Figure A-4 (Part 2 of 4). BSC MSRJE Data Communications Line Protocols



S0590066-1

Figure A-4 (Part 3 of 4). BSC MSRJE Data Communications Line Protocols



Notes:

1. *If there is no information to be transferred, the line connection is maintained by sending and receiving ACK0s. This sequence continues until either the host system or System/36 has information to send.*

2. *If the host system receives a transmission successfully and has data ready to send, the data transmission is a positive acknowledgment (ACK0 is not required).*

S0590215-1

Figure A-4 (Part 4 of 4). BSC MSRJE Data Communications Line Protocols

SNA SESSION PROTOCOLS

This section contains typical line sequences for six SNA subsystems.

SNA Upline Facility (SNUF)

Figure A-5 shows protocols for the following SNA Upline Facility (SNUF) operations:
- Enable
- Acquire Session
- Evoke with Invite
- Release
- Abnormal End of Session
- Disable

Enable

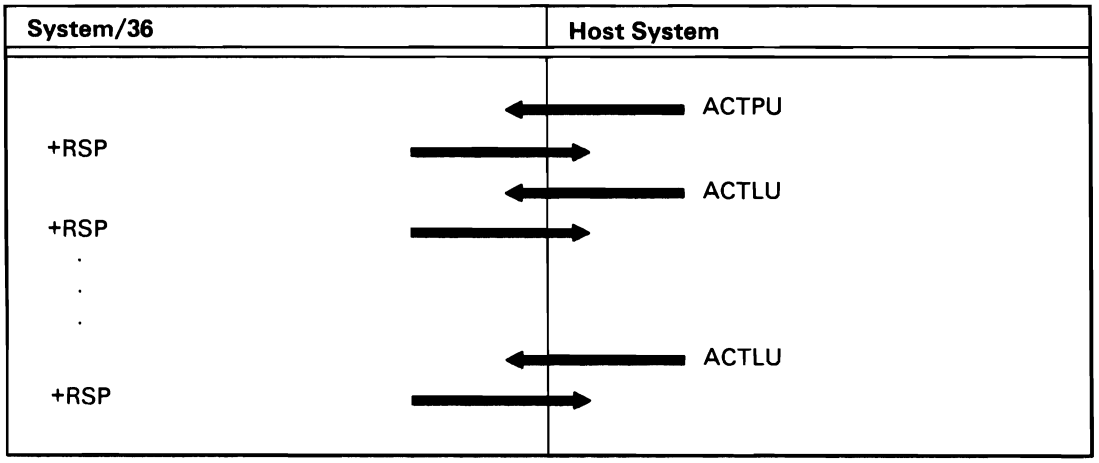


Figure A-5 (Part 1 of 7). SNUF Session Protocols

Acquire Session (One Communications Line Was Active before Acquire Operation)

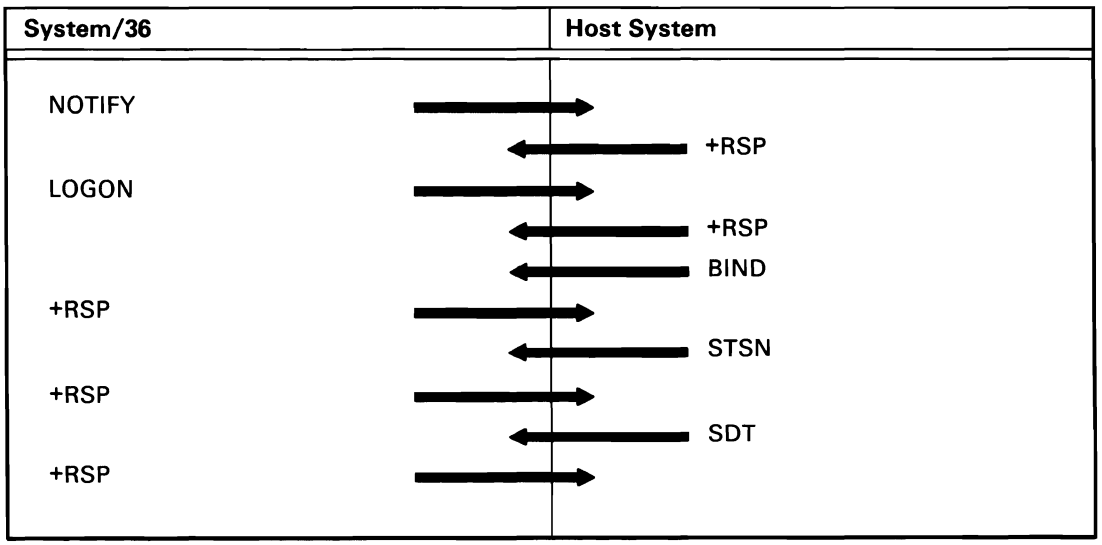


Figure A-5 (Part 2 of 7). SNUF Session Protocols

Acquire Session (before Host System Starts the Communications Line)

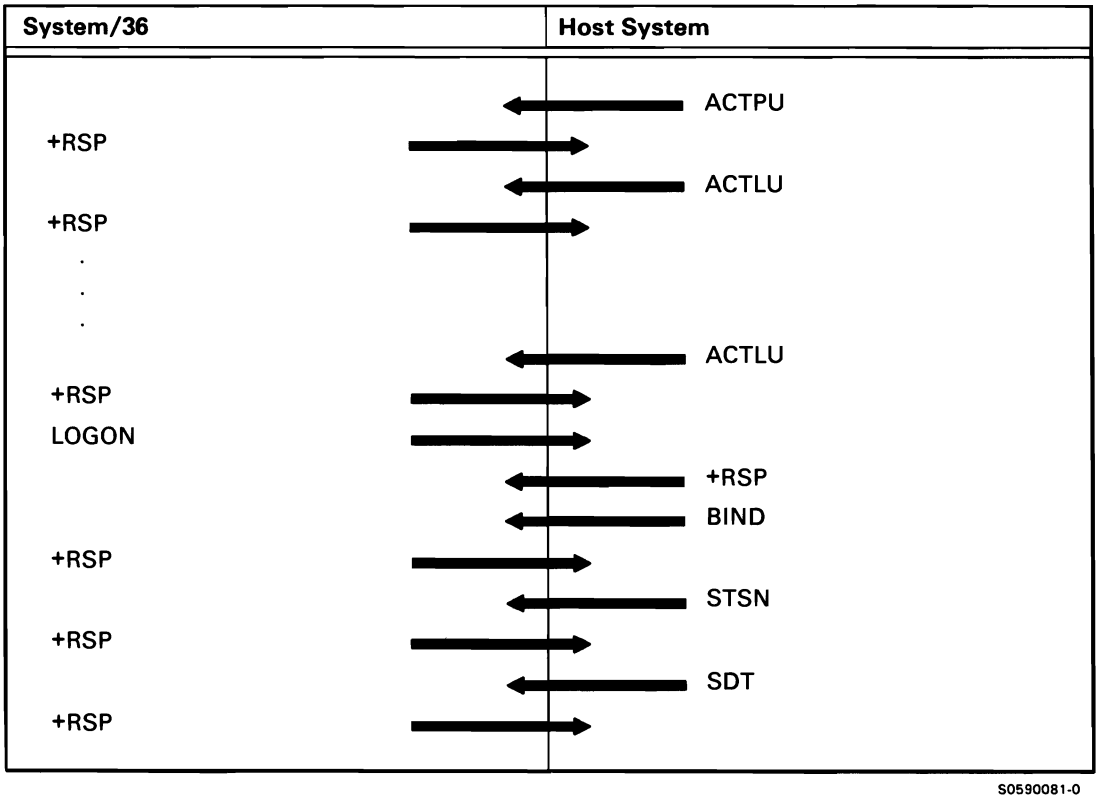


Figure A-5 (Part 3 of 7). SNUF Session Protocols

Evoke with Invite

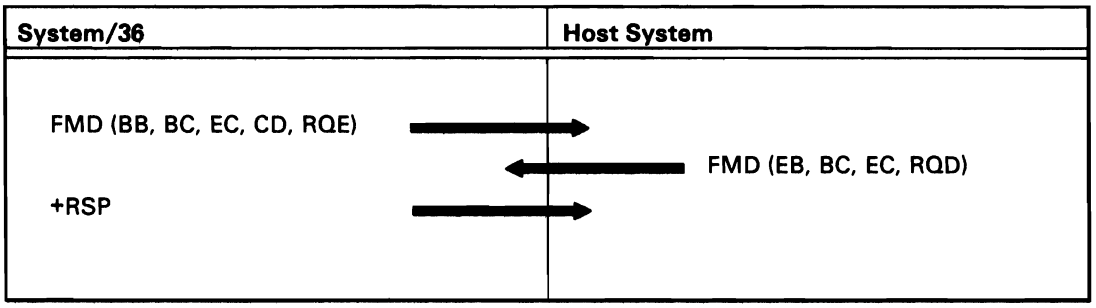


Figure A-5 (Part 4 of 7). SNUF Session Protocols

Release Operation

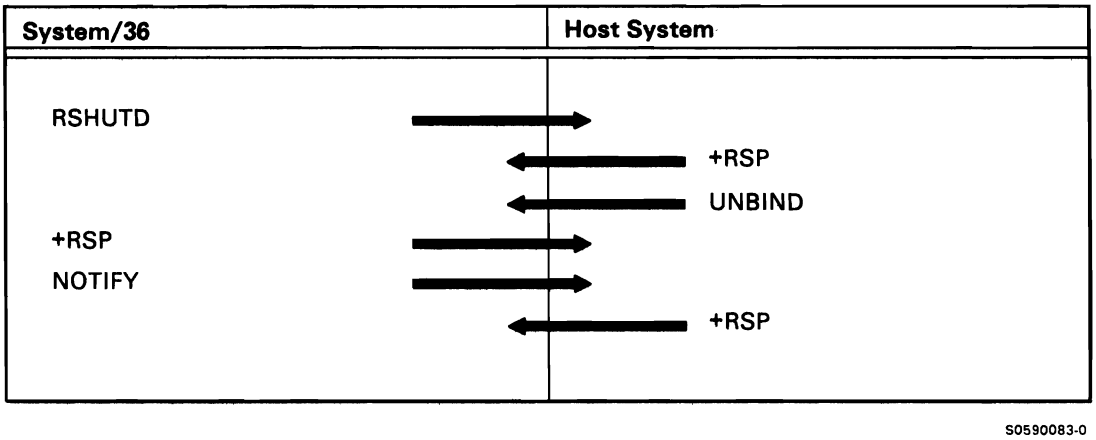


Figure A-5 (Part 5 of 7). SNUF Session Protocols

Abnormal End to a Session (via End-of-Session Operation)

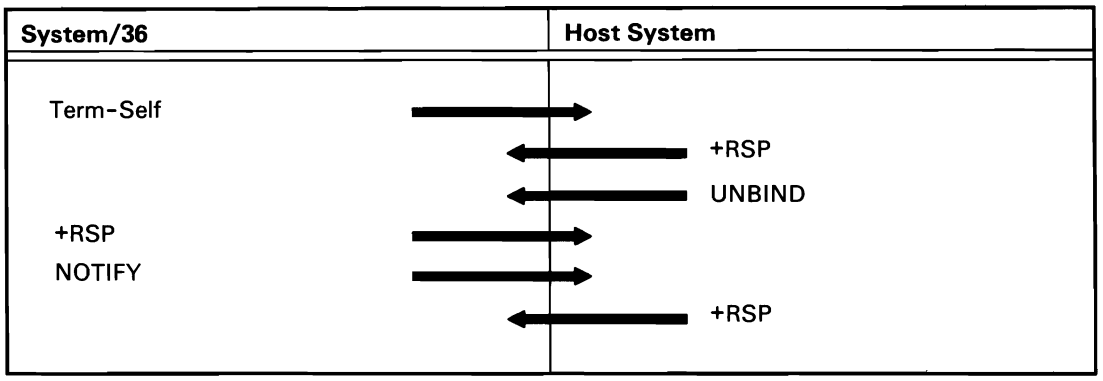


Figure A-5 (Part 6 of 7). SNUF Session Protocols

Disable

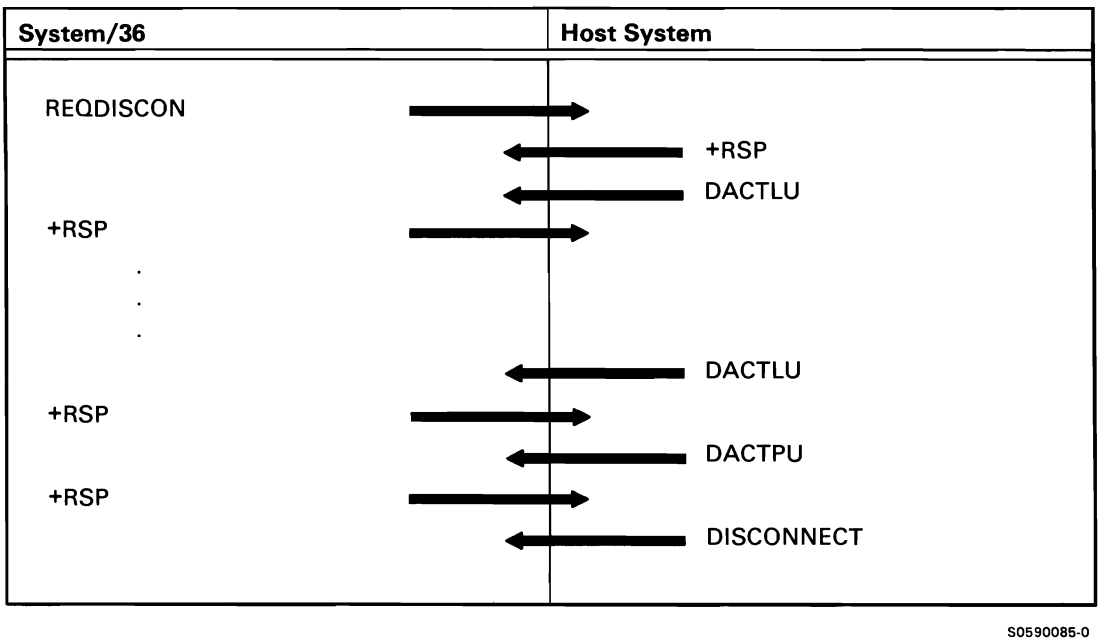


Figure A-5 (Part 7 of 7). SNUF Session Protocols

Figure A-6 shows flow sequences for line activity between a System/36 using SNA 3270 device emulation and the host system. The sequences are shown in logical order and emphasize the main host system and System/36 commands that are issued. The following operations are shown:

- Enable
- Session Initialization via System/36 Work Station
- Initialization by Host System
- Inbound Data
- Outbound Data
- Shut Down or Signal Command from Host System
- End of Session
- Disable Session

Enable

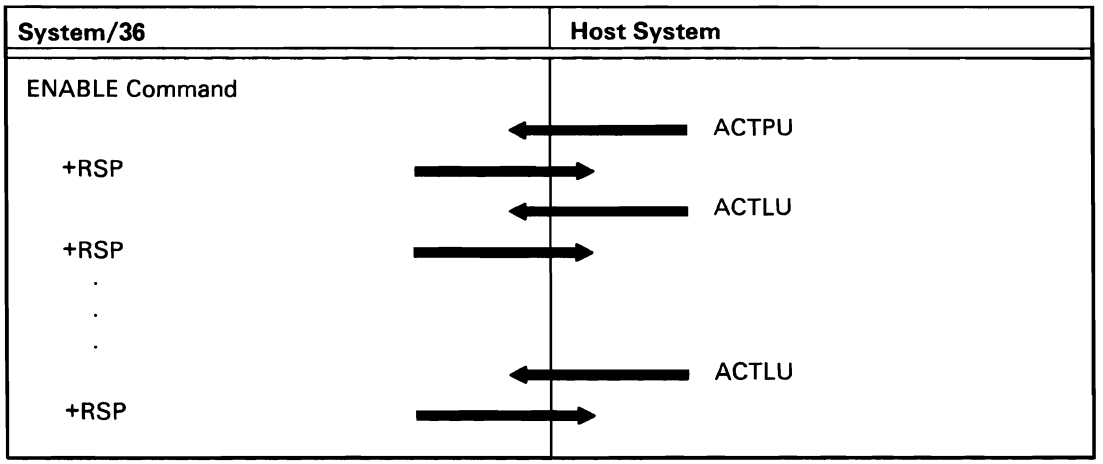


Figure A-6 (Part 1 of 8). SNA 3270 Session Protocols

Session Initialization via System/36 Work Station

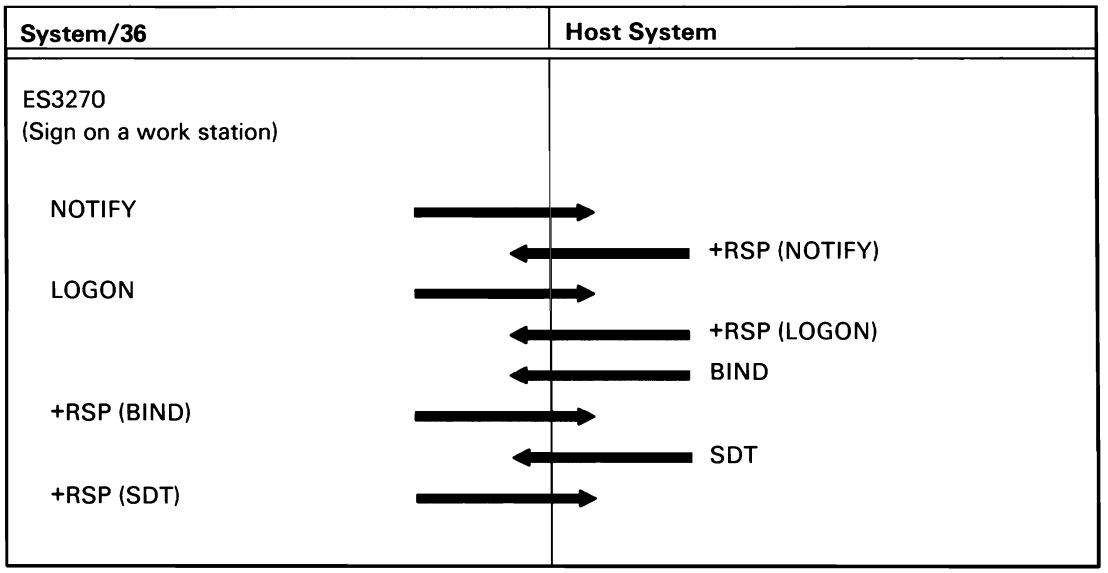


Figure A-6 (Part 2 of 8). SNA 3270 Session Protocols

Initialization by Host System

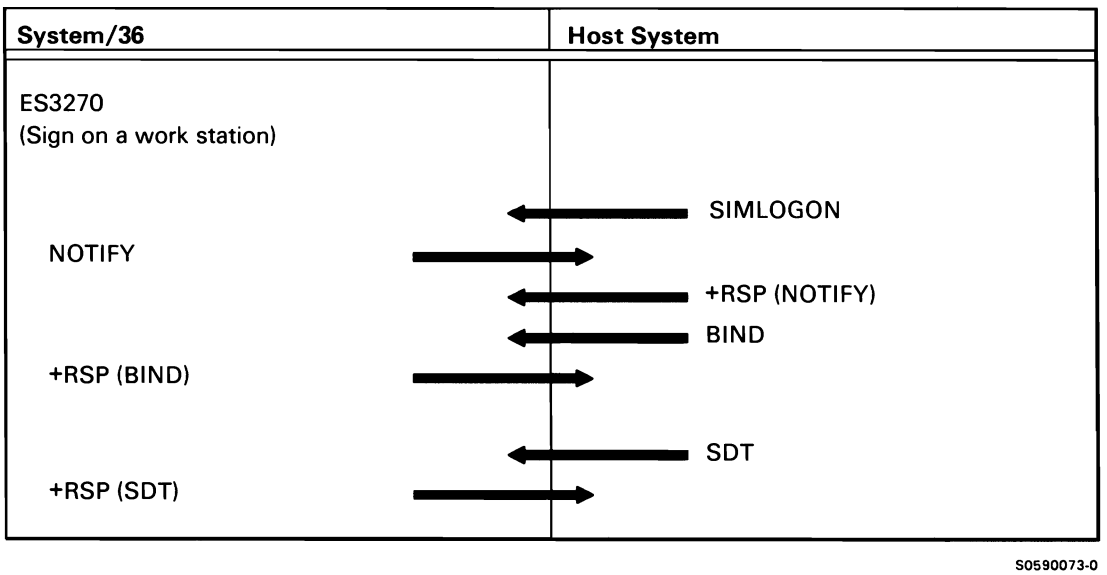


Figure A-6 (Part 3 of 8). SNA 3270 Session Protocols

Inbound Data—System/36 to Host System (LU-to-LU Normal)

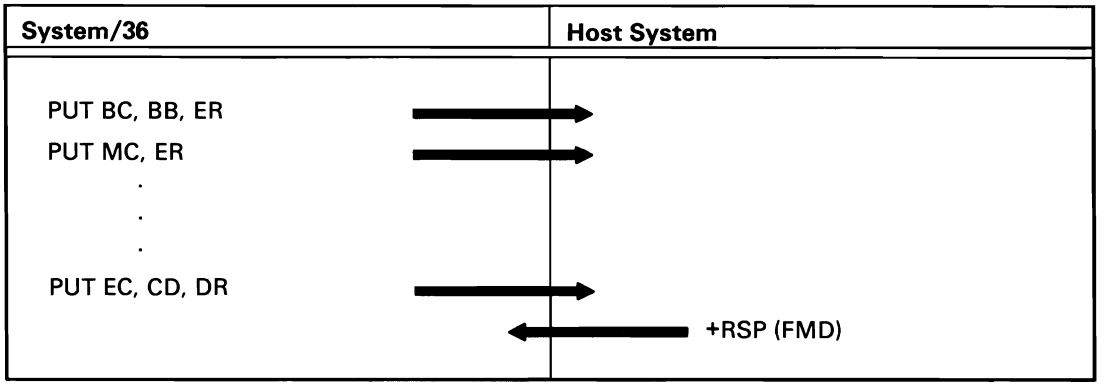


Figure A-6 (Part 4 of 8). SNA 3270 Session Protocols

Outbound Data—Host System to System/36 (LU-to-LU Normal)

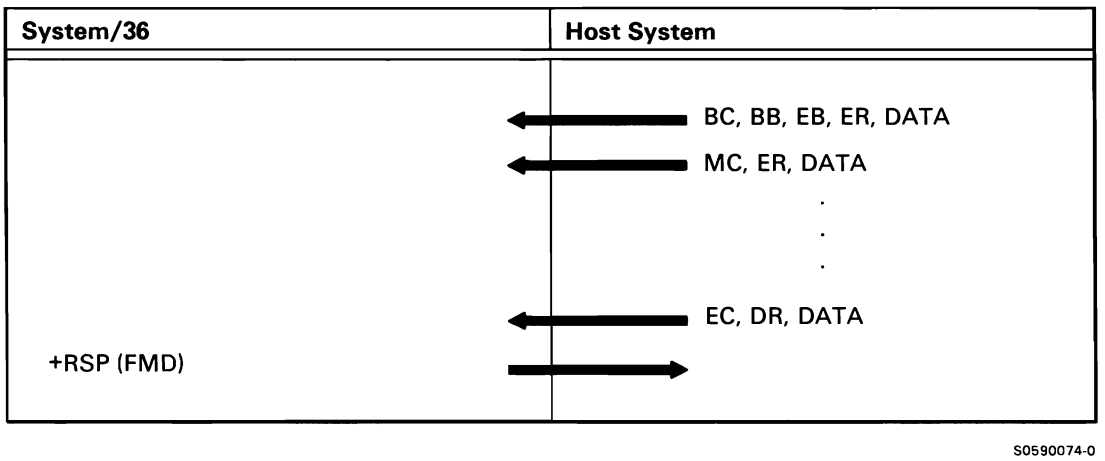


Figure A-6 (Part 5 of 8). SNA 3270 Session Protocols

Shut Down or Signal Command from Host System
(Expedited Requests)

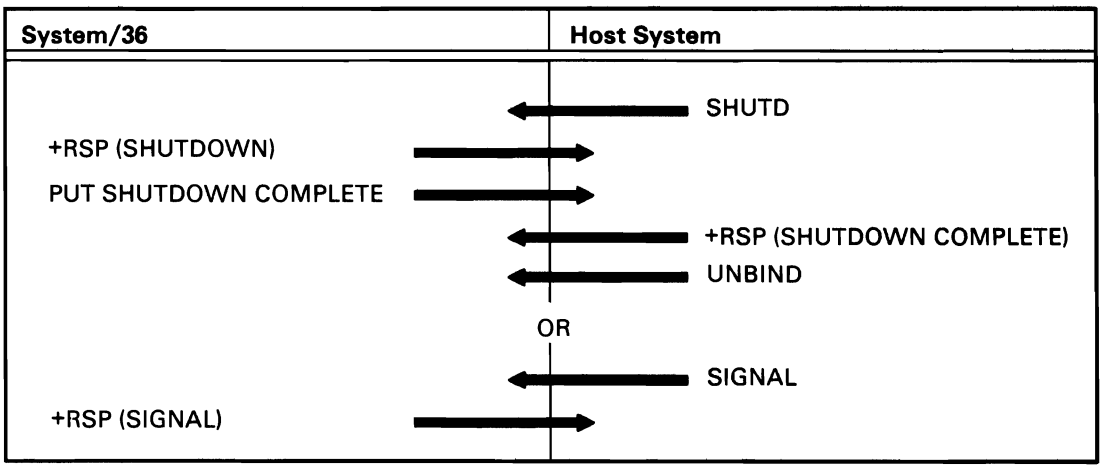


Figure A-6 (Part 6 of 8). SNA 3270 Session Protocols

End of Session (via Work Station Power Down)

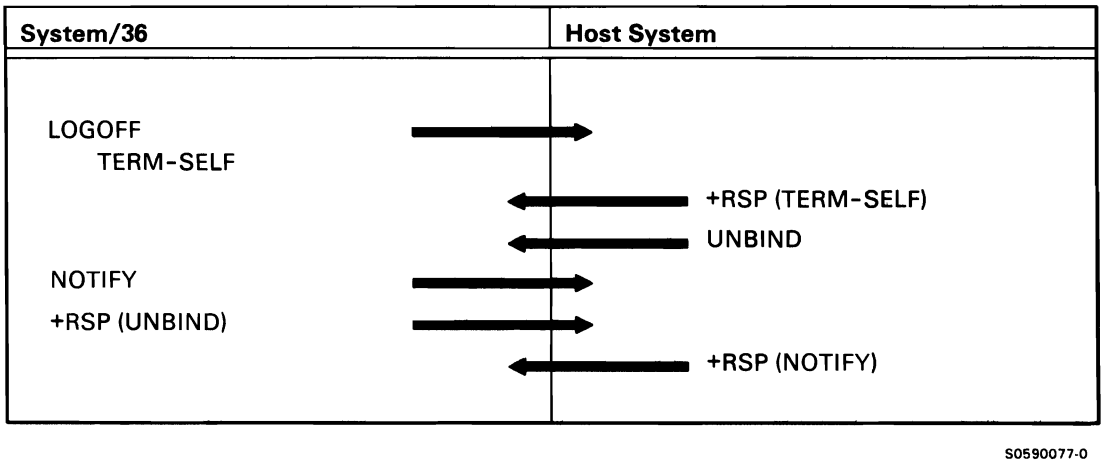


Figure A-6 (Part 7 of 8). SNA 3270 Session Protocols

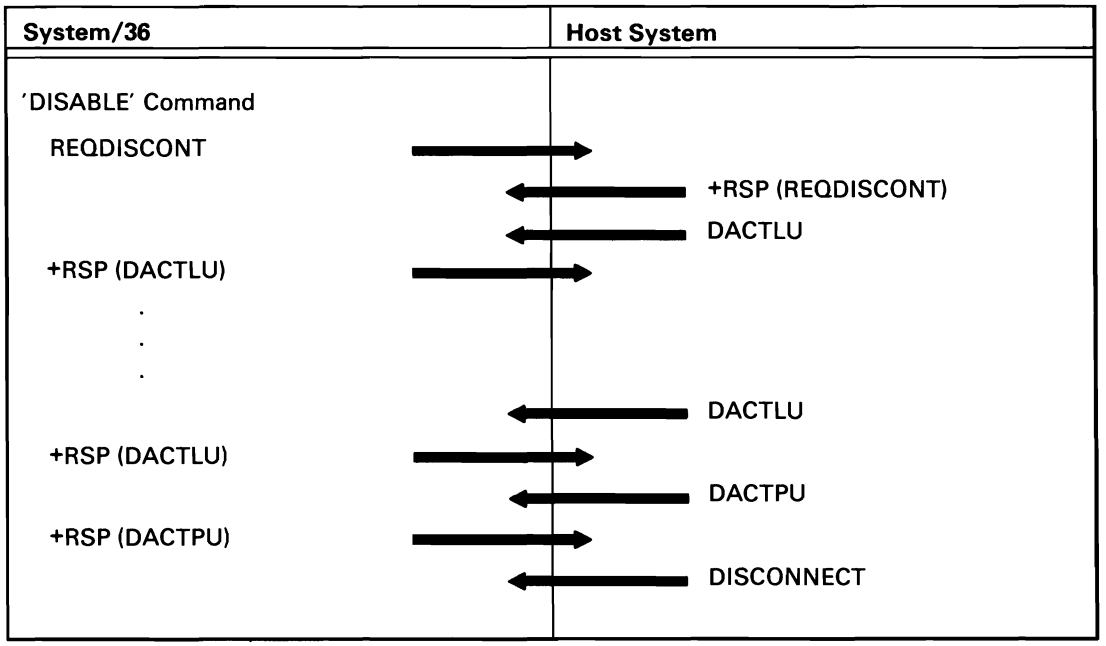


Figure A-6 (Part 8 of 8). SNA 3270 Session Protocols

SNA MSRJE

Figure A-7 shows the sequences generated when a System/36 user starts an RJE session with the host system (the MSRJE procedure command is entered at a display station). The following operations are shown:

- Initialization
- System/36-Initiated Bracket Reader
- System/36-Initiated Bracket Console
- Host System-Initiated Bracket
- System/36 Request Direction Change
- Host System Request Direction Change
- Host-Detected Error
- Host System Interrupts Outbound Data
- System/36 Interrupts Inbound Data
- System/36-Initiated Termination Sequence
- Host System-Initiated Termination Sequence

Initialization

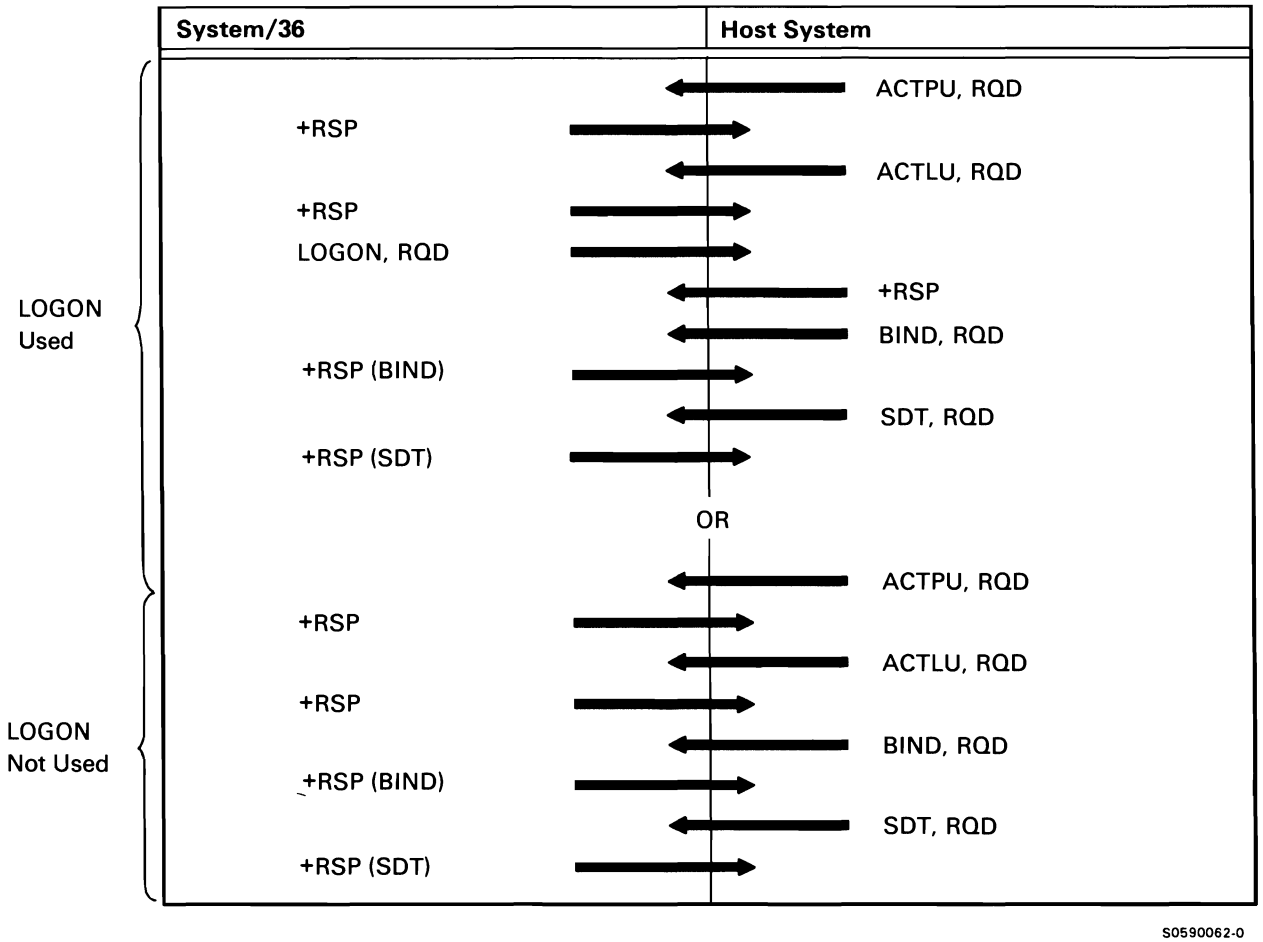


Figure A-7 (Part 1 of 13). SNA MSRJE Session Protocols

System/36-Initiated Bracket Reader

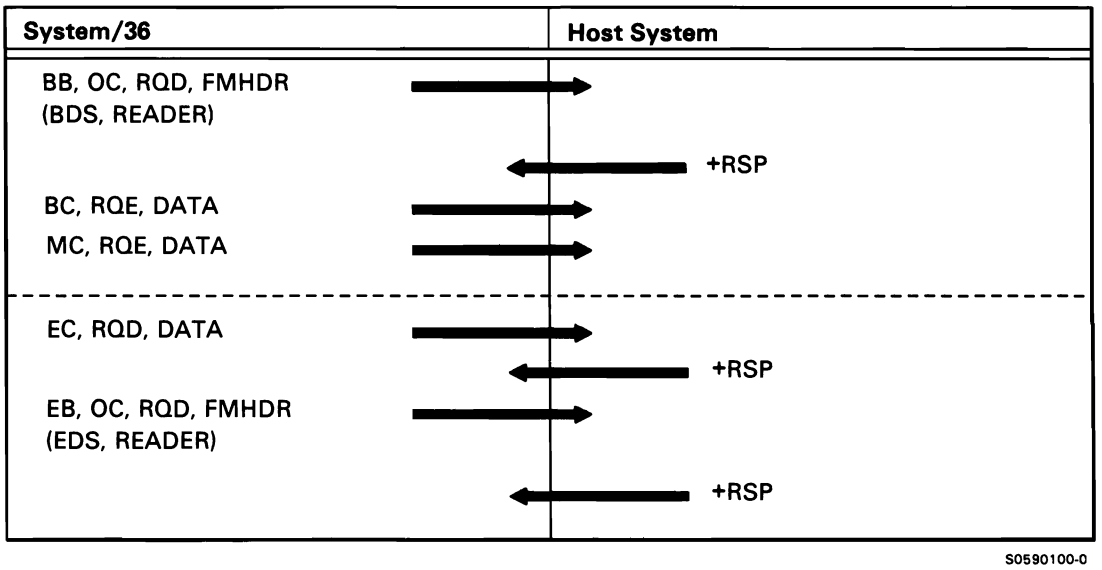


Figure A-7 (Part 2 of 13). SNA MSRJE Session Protocols

System/36-Initiated Bracket Console (MSRJE Console Enter Key)

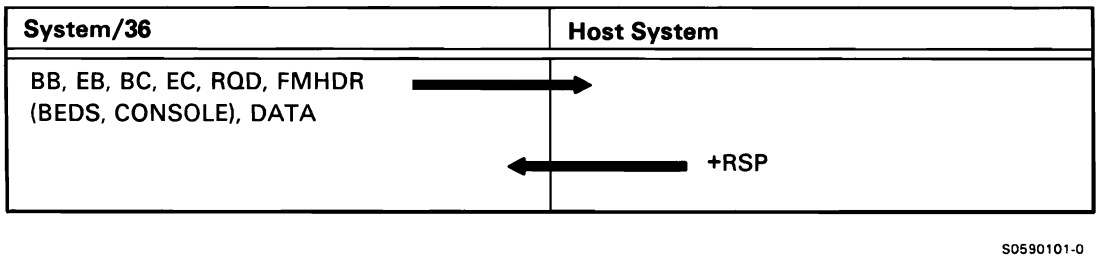


Figure A-7 (Part 3 of 13). SNA MSRJE Session Protocols

Host System-Initiated Bracket (Host Sends Data to System/36)

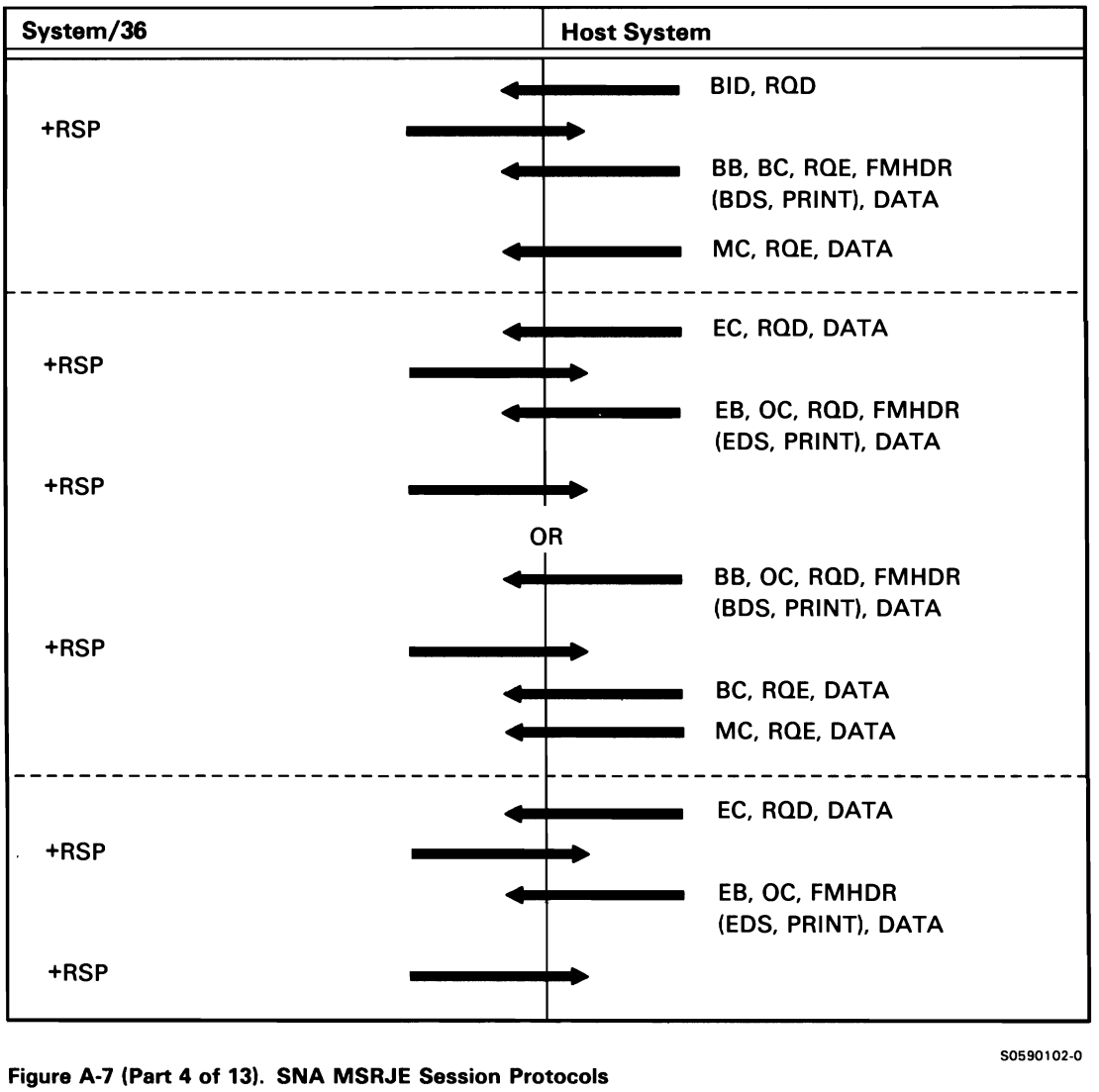


Figure A-7 (Part 4 of 13). SNA MSRJE Session Protocols

System/36 Requests Direction Change (System/36 Interrupts Outbound Data Flow to Send Console Data)

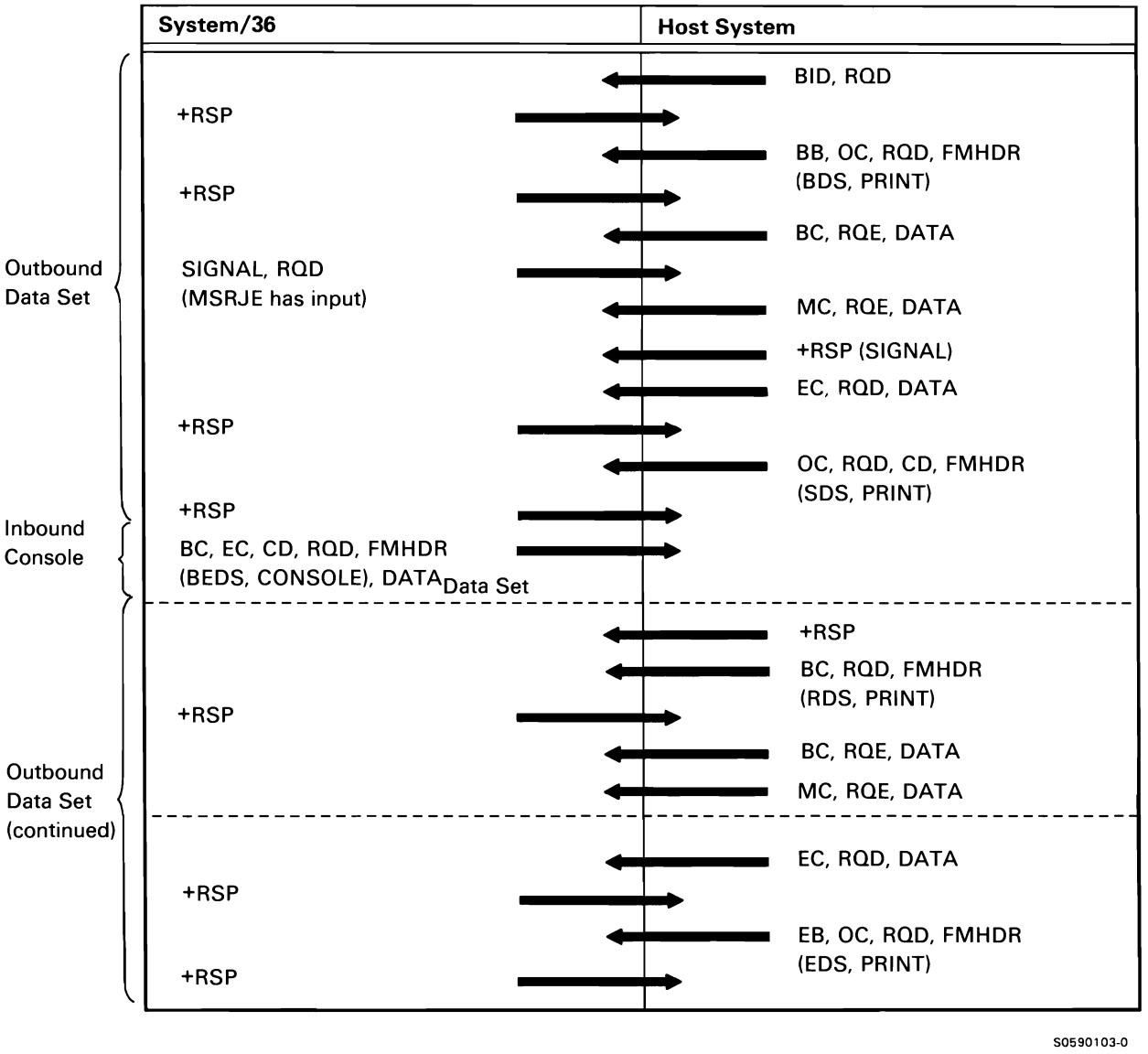


Figure A-7 (Part 5 of 13). SNA MSRJE Session Protocols

System/36 Requests Direction Change (System/36 Interrupts Outbound Data Flow to Send Reader Data)

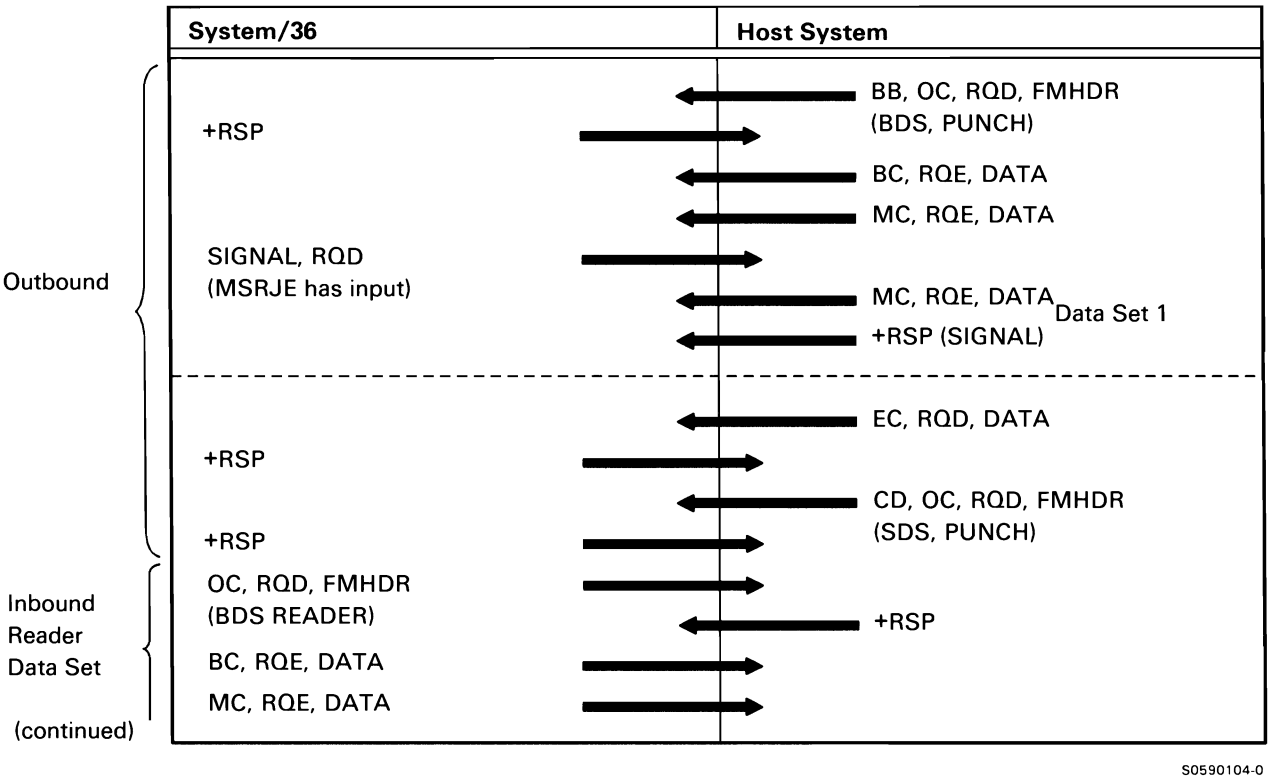


Figure A-7 (Part 6 of 13). SNA MSRJE Session Protocols

System/36 Requests Direction Change (System/36 Interrupts Reader Data to Send Console Data)

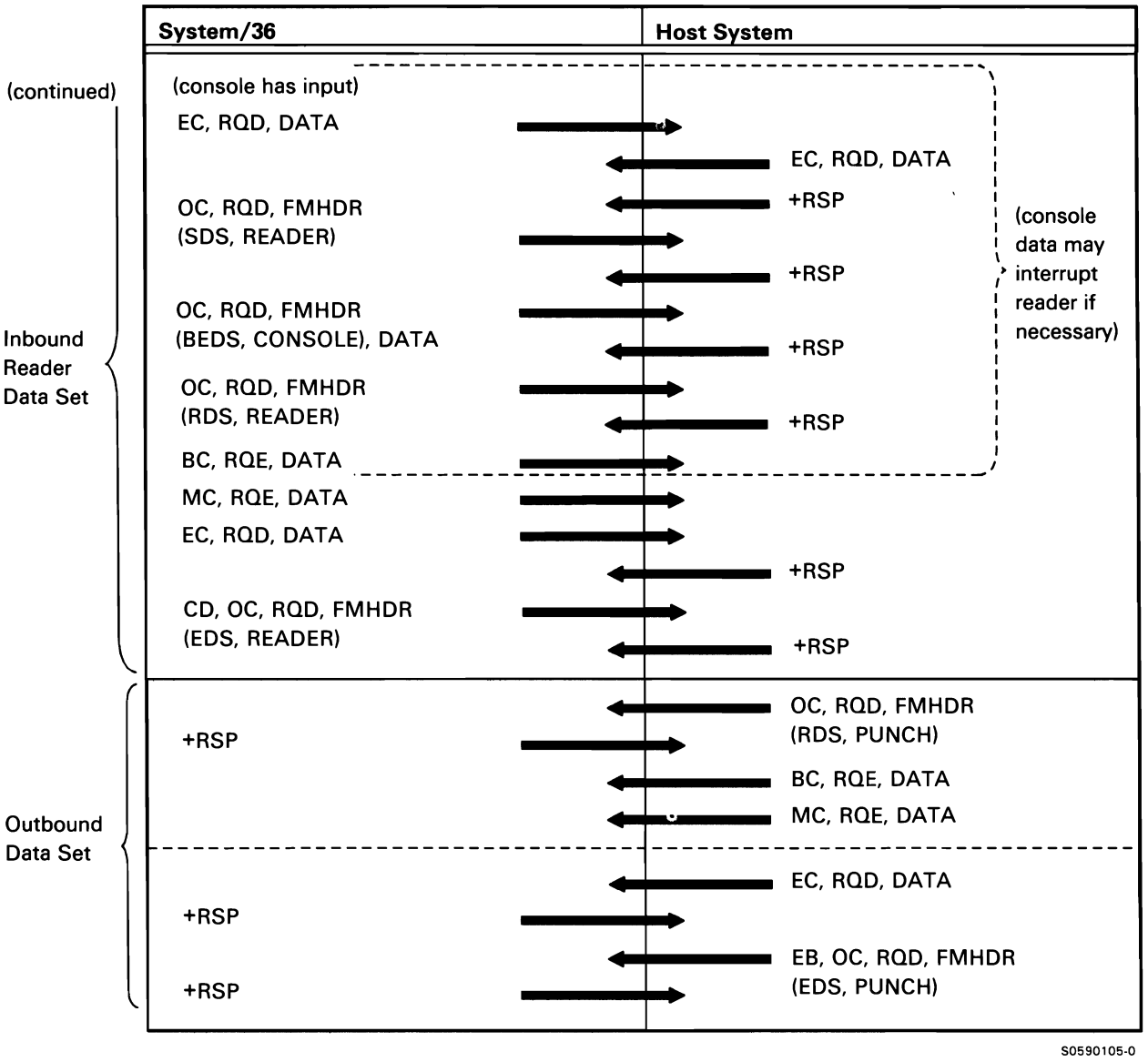


Figure A-7 (Part 7 of 13). SNA MSRJE Session Protocols

Host System Requests Direction Change (Host Interrupts Reader Data to Send Console Data)

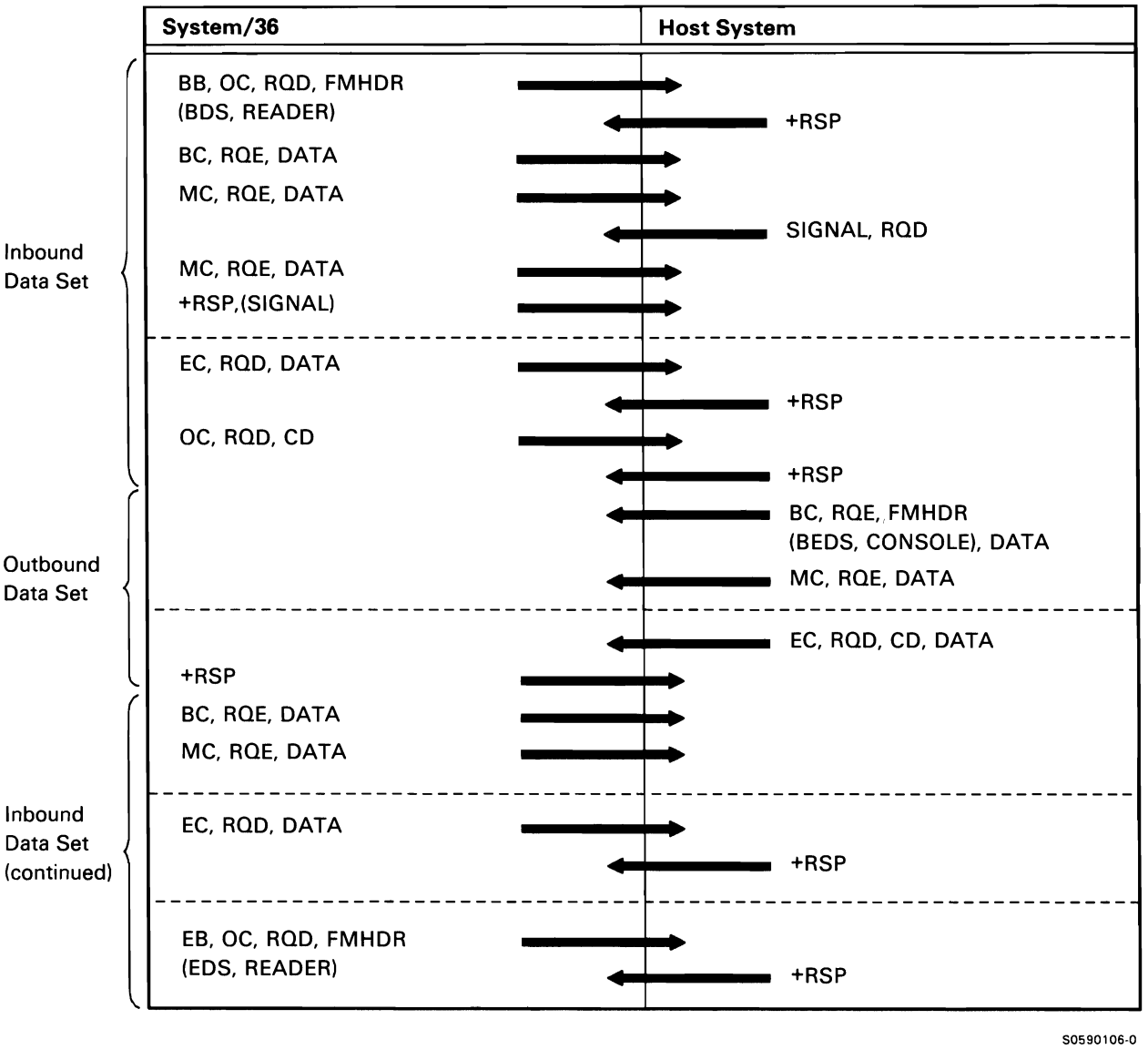


Figure A-7 (Part 8 of 13). SNA MSRJE Session Protocols

Host-Detected Error (Host Terminates Inbound Reader Data)

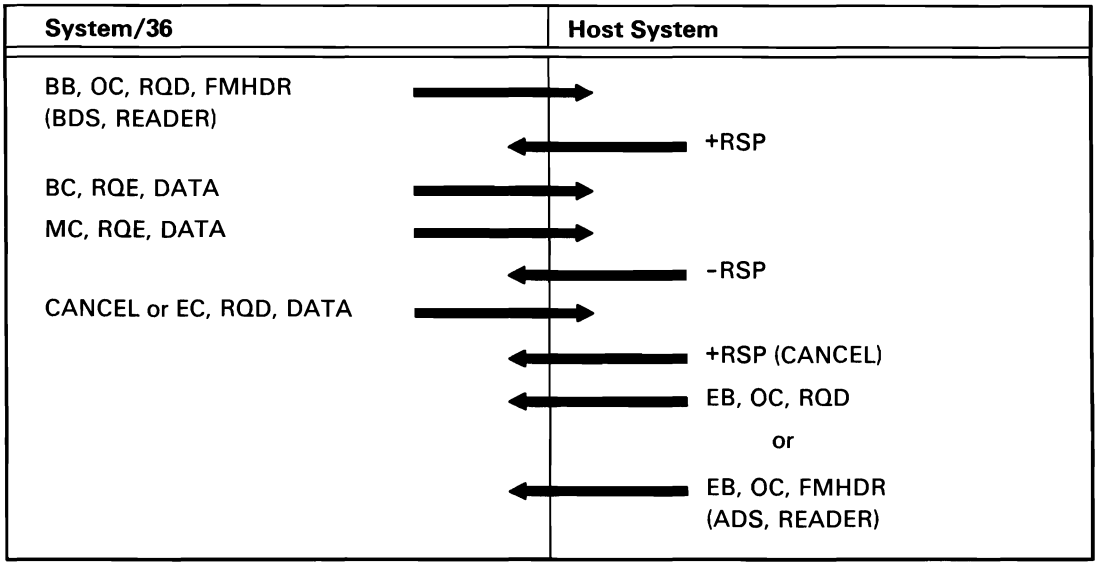


Figure A-7 (Part 9 of 13). SNA MSRJE Session Protocols

Host System Interrupts Outbound Data to Send Console Output Data

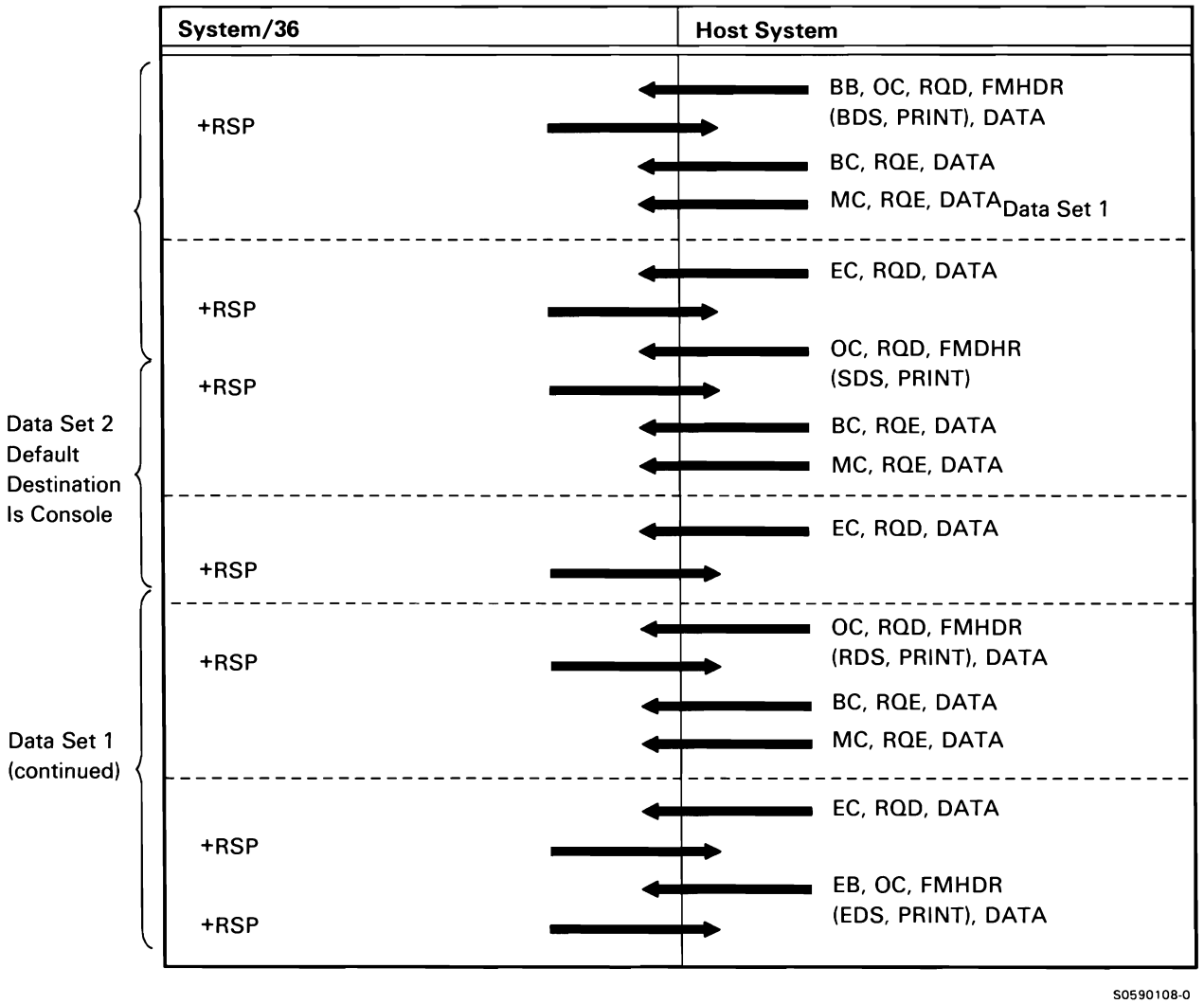


Figure A-7 (Part 10 of 13). SNA MSRJE Session Protocols

System/36 Interrupted Inbound Data (System/36 Interrupts Reader Data to Send Console Data)

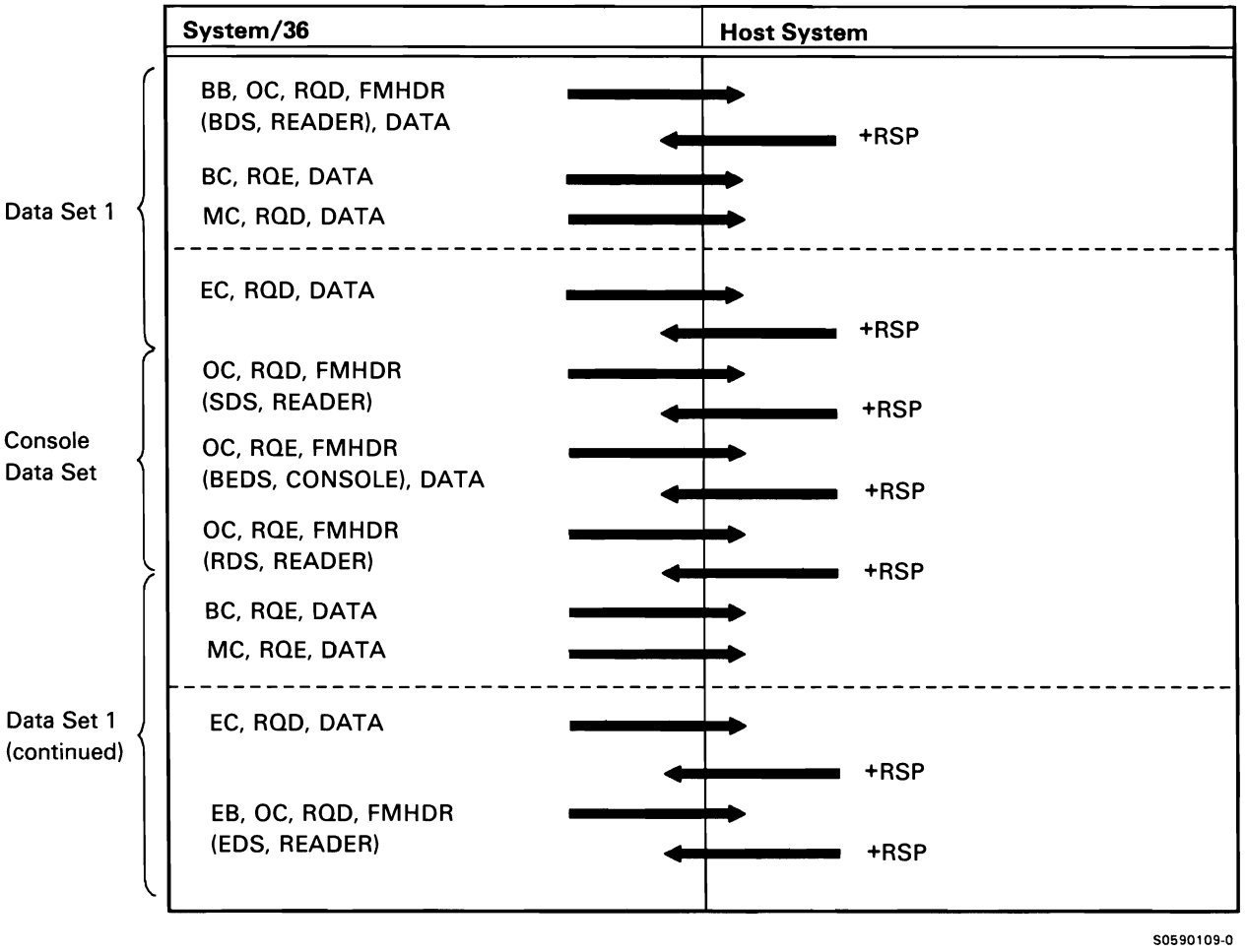


Figure A-7 (Part 11 of 13). SNA MSRJE Session Protocols

System/36 Initiated Termination Sequence (Generated When System/36 Ends a Session with the Host System—an END Statement Was Entered or a Configured Delay Timer Elapsed)

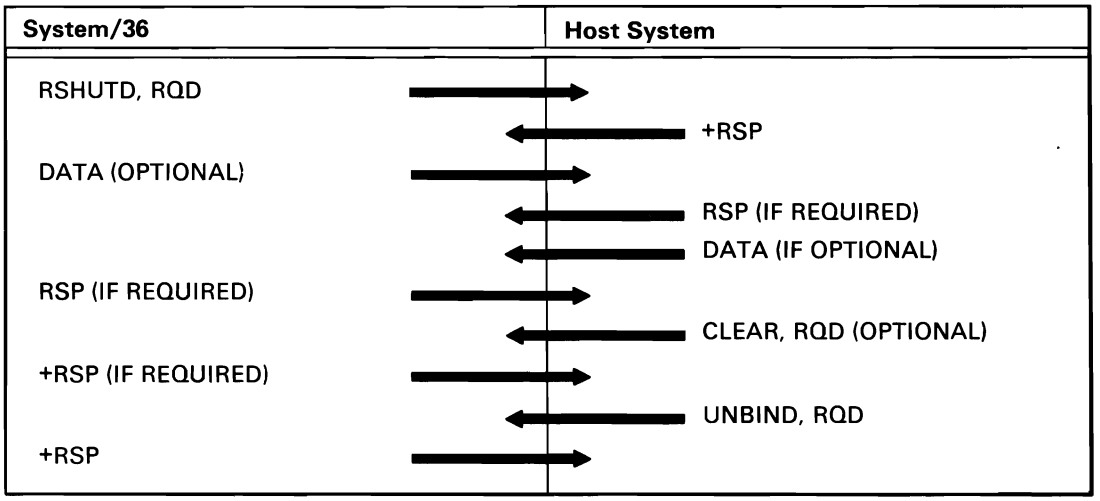


Figure A-7 (Part 12 of 13). SNA MSRJE Session Protocols

Host System Initiated Termination Sequence (Generated When Host System Ends an RJE Session with System/36)

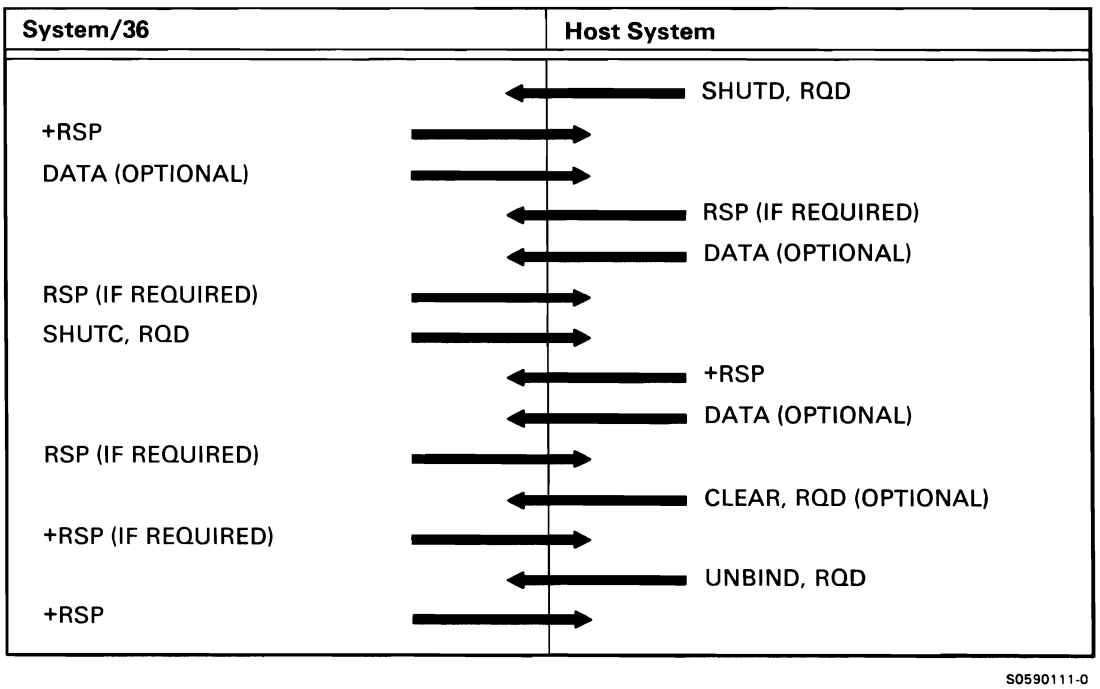


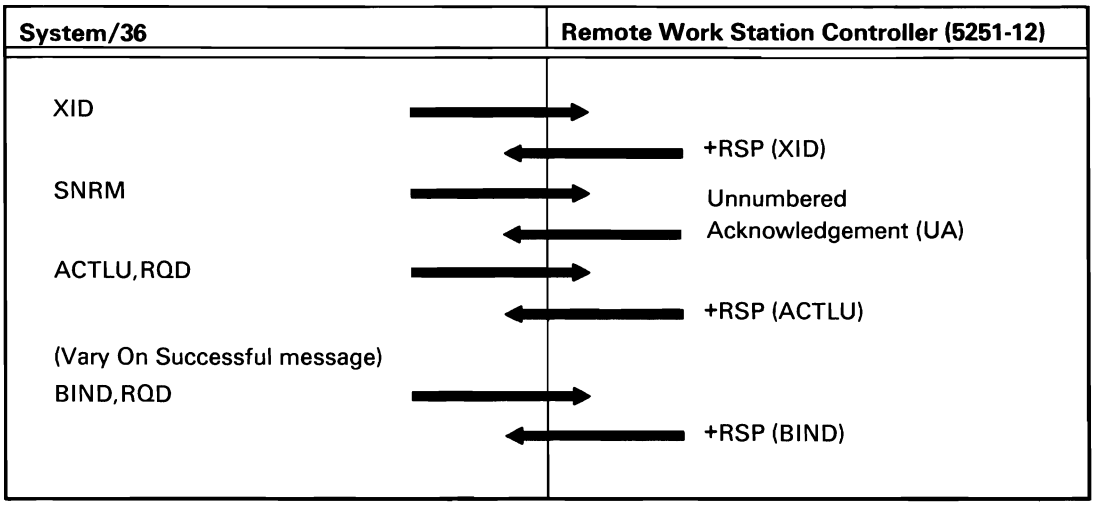
Figure A-7 (Part 13 of 13). SNA MSRJE Session Protocols

Remote Work Station

Figure A-8 shows the protocols for the following remote work station operations:

- Initialization
- Termination
- Put Data
- Get Data

Initialization Sequence (Initiated When Operator Enters Vary-On Command at System Console)



S0590067-0

Figure A-8 (Part 1 of 4). Remote Work Station Data Communications Line Protocols

Termination Sequence (Initiated When Operator Enters Vary-Off Command at System Console)

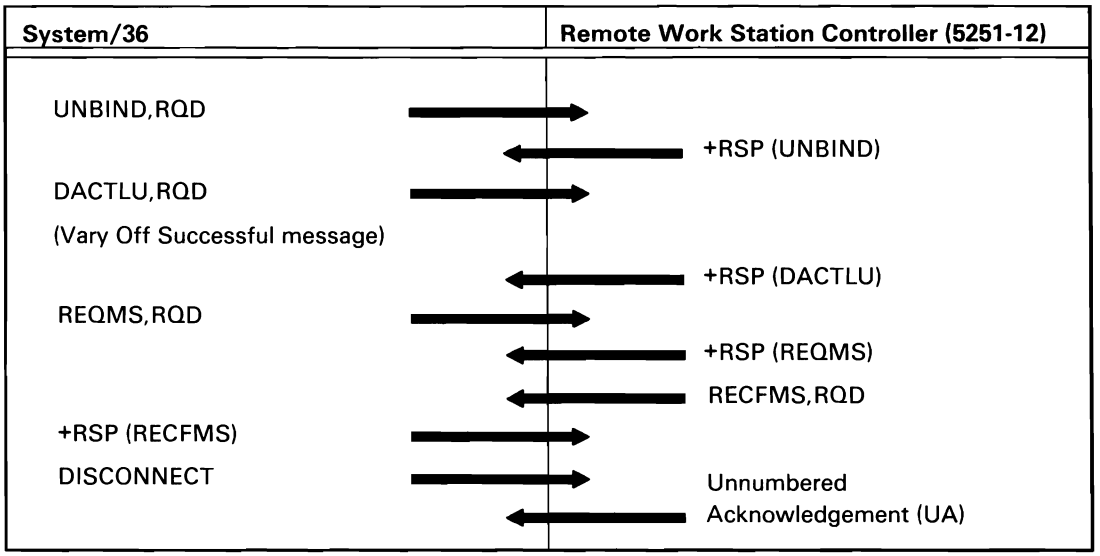


Figure A-8 (Part 2 of 4). Remote Work Station Data Communications Line Protocols

Put Data Sequence (Data Is Sent to Remote Work Station; Put Is Initiated When the TUB or PUB, Set Up for the Put, Is Posted to the Remote Work Station Task)

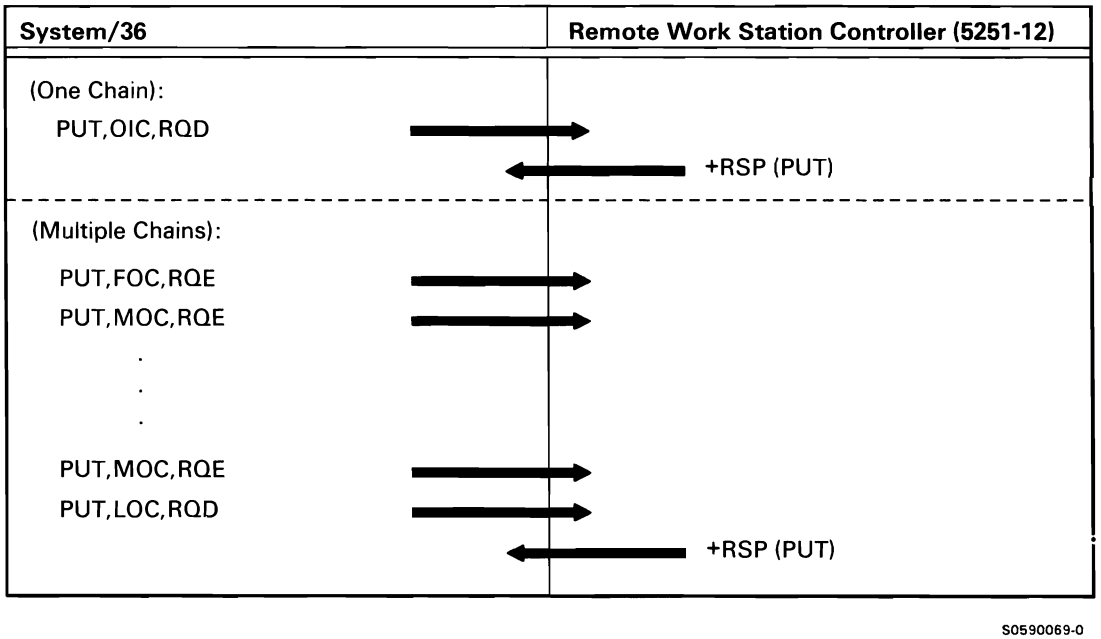
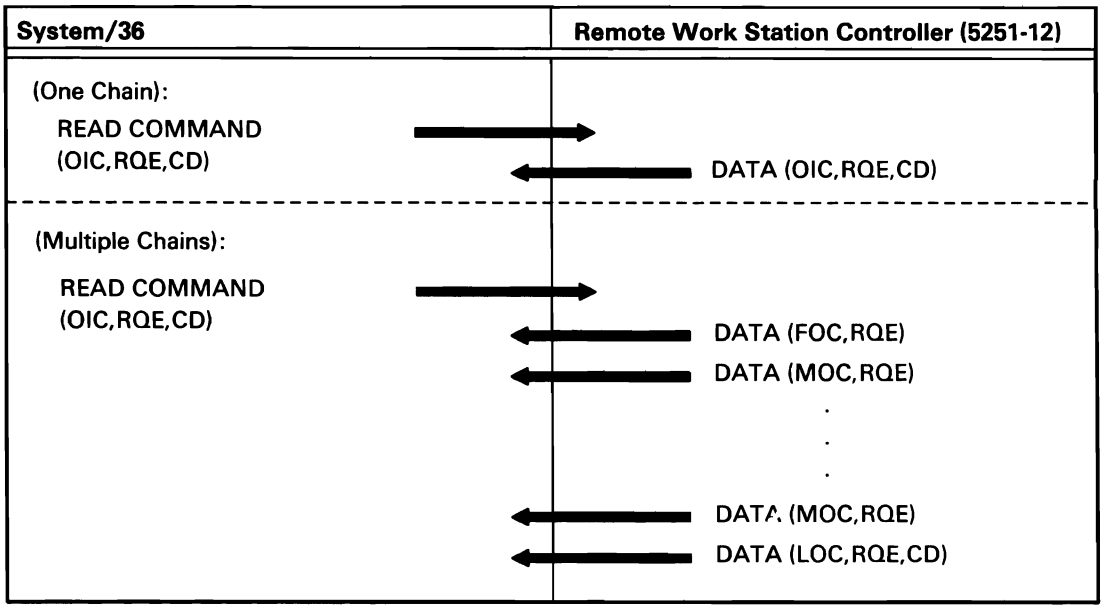


Figure A-8 (Part 3 of 4). Remote Work Station Data Communications Line Protocols

Get Data Sequence (an Explicit READ Command Must First be Sent; Get Is Initiated When TUB, Set Up for the Get, Is Posted to the Remote Work Station Task)



S0590070-0

Figure A-8 (Part 4 of 4). Remote Work Station Data Communications Line Protocols

Finance

Figure A-9 shows the flow sequences generated when a location is enabled to a 3601/4701 communications controller to communicate with an operational diskette. The controller has N logical work stations configured. For this example, logical work stations 2 and 5 are configured for auto-sign-on in the controller; the remainder are not. The following operations are shown:

- Enable: Operational Diskette
- Enable: To 3601/4701 Communications Controller
- Disable: To 3601/4701 Communications Controller
- Sign-On
- Sign-On Response
- Enable: To 3694 Document Processor
- Disable: To 3694 Document Processor

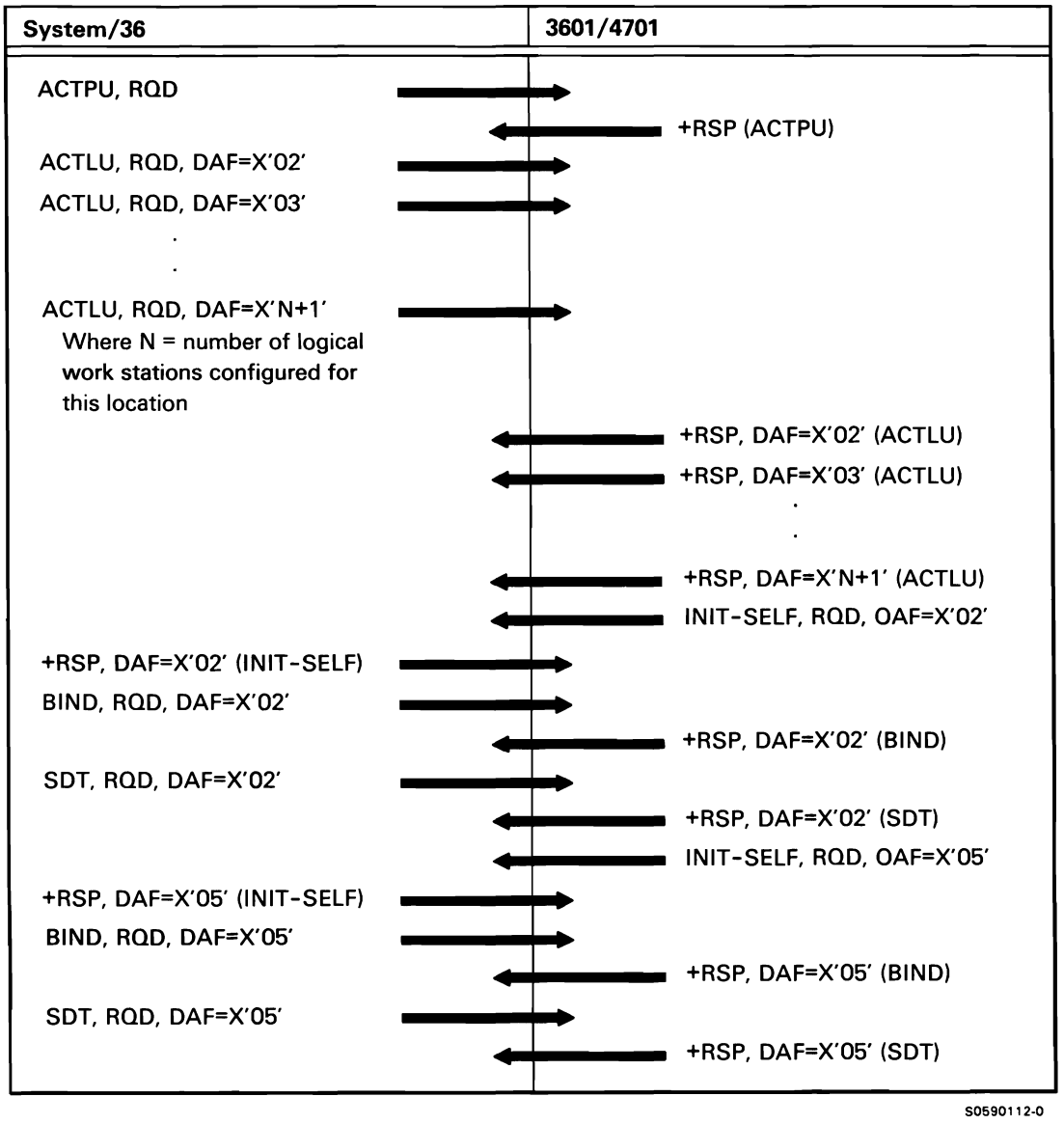


Figure A-9 (Part 1 of 8). Finance Data Communications Line Protocols

Enable: To 3601/4701 Communications Controller with System Monitor Specified

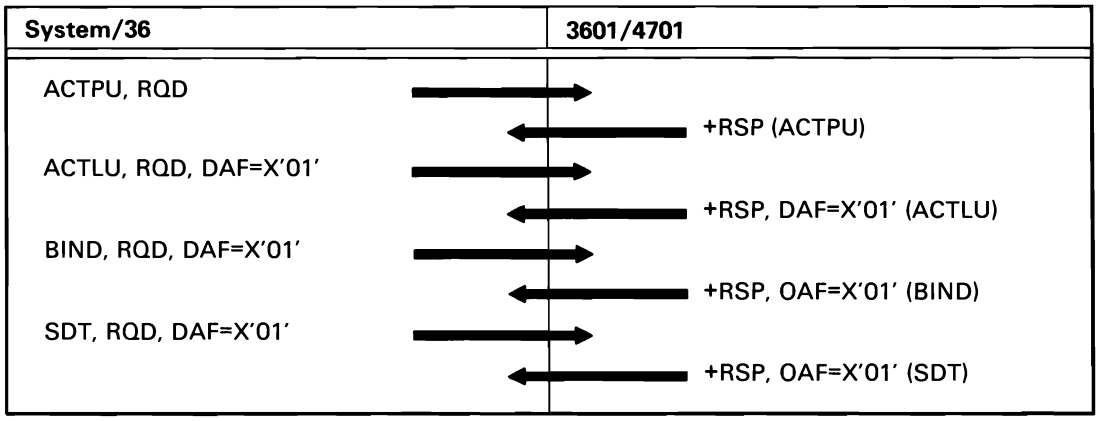


Figure A-9 (Part 2 of 8). Finance Data Communications Line Protocols

Disable: To 3601/4701 Communications Controller Communicating with System Monitor Session

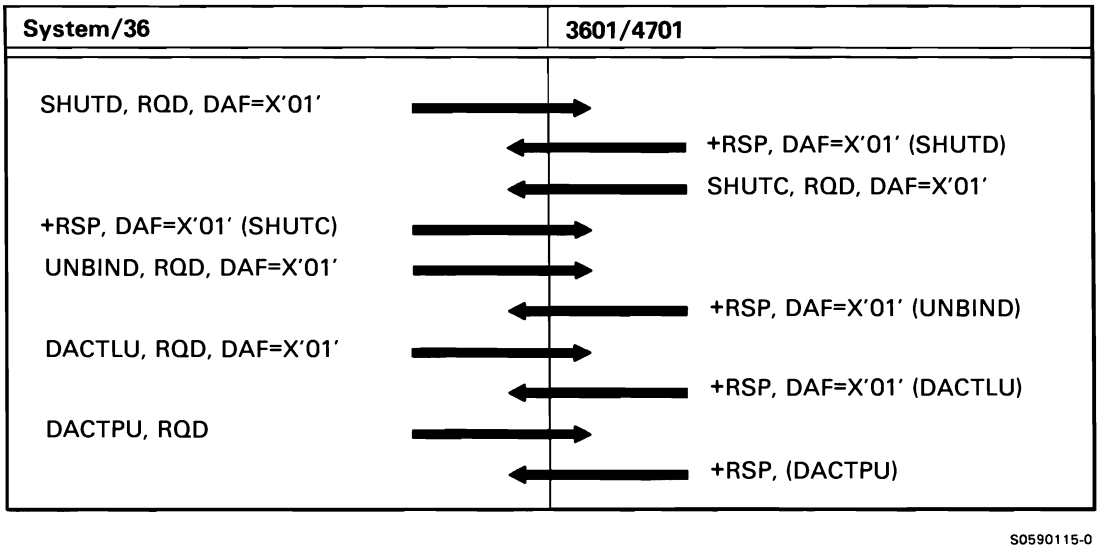
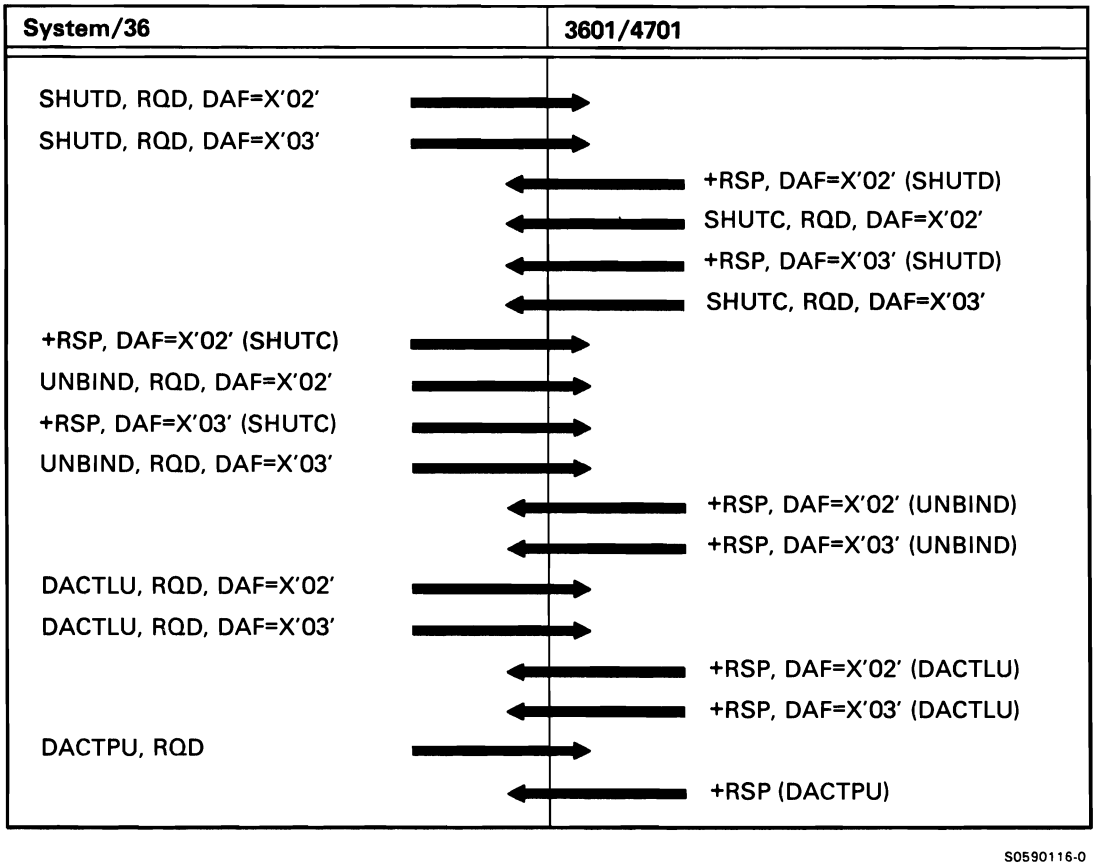


Figure A-9 (Part 3 of 8). Finance Data Communications Line Protocols

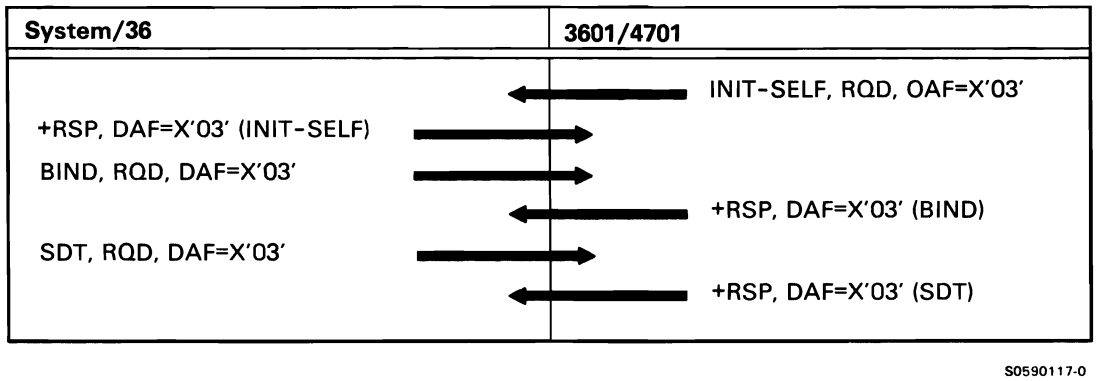
Disable: To 3601/4701 Communications Controller Controlling Teller Work Stations (in This Example, Two Work Stations Are Configured and Operational)



S0590116-0

Figure A-9 (Part 4 of 8). Finance Data Communications Line Protocols

Sign-On: A 3601/4701 Operator Signs On a Teller Work Station (in This Example, the Teller Is at Logical Work Station 3)



S0590117-0

Figure A-9 (Part 5 of 8). Finance Data Communications Line Protocols

Sign-On Response (in This Example, an Operator at Logical Work Station 3 Does a Transaction)

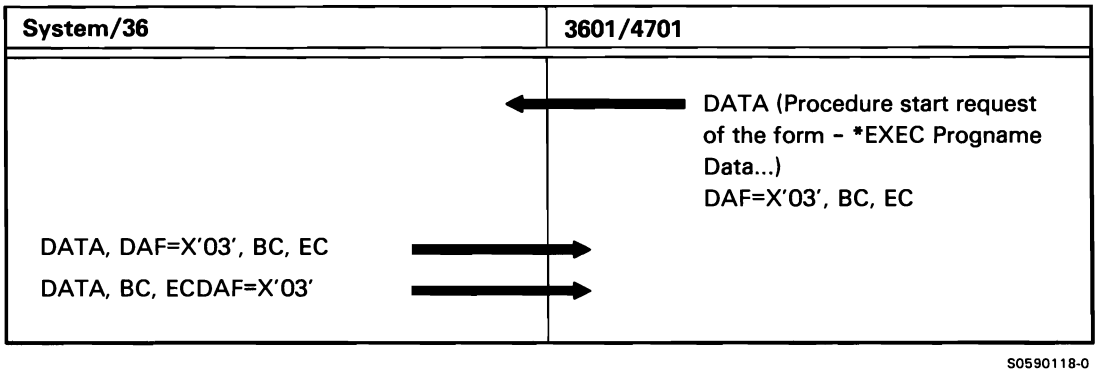


Figure A-9 (Part 6 of 8). Finance Data Communications Line Protocols

Enable to 3694 Document Processor

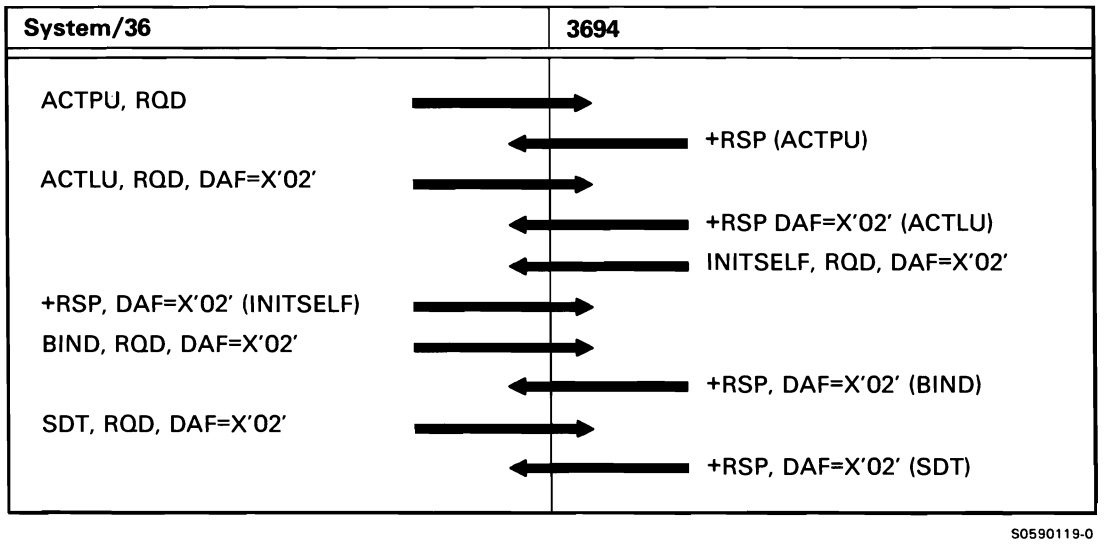


Figure A-9 (Part 7 of 8). Finance Data Communications Line Protocols

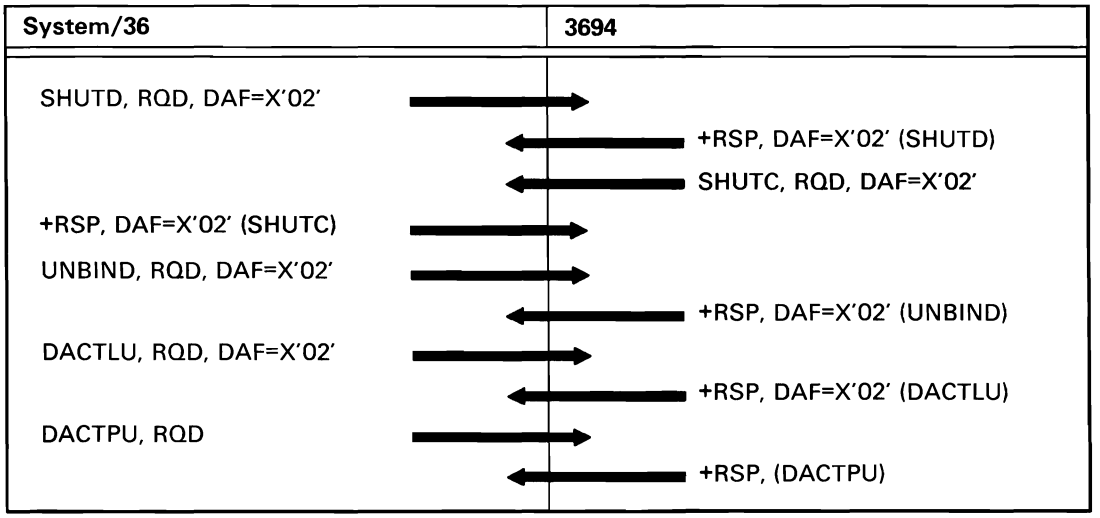


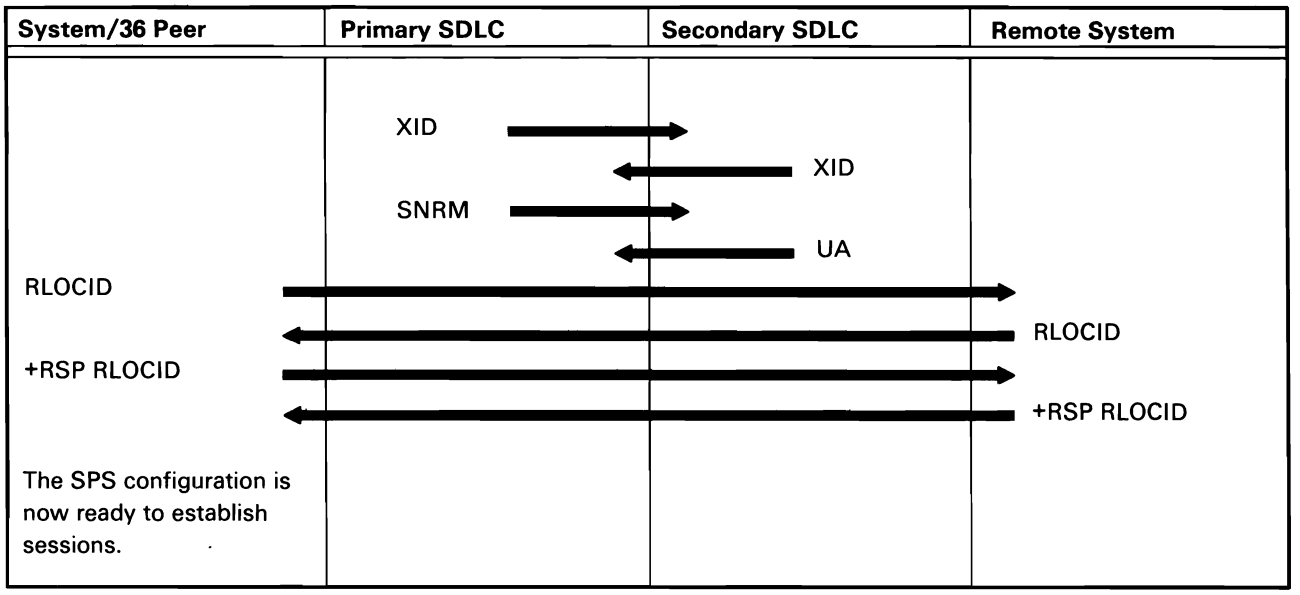
Figure A-9 (Part 8 of 8). Finance Data Communications Line Protocols

SNA Peer Subsystem

Figure A-10 shows the protocol flow generated when line and session activation/deactivation takes place between the System/36 Peer subsystem support and the remote system. The line activation/deactivation portions show two types of flow: protocol flow between the two SDLCs and flow between the two Peer subsystems.

The following Peer operations are shown:
- Enable Primary: Line Activation
- Enable Primary with Remote Rejection of RLOCID
- Enable Secondary: Line Activation
- Enable Secondary with Remote Rejection of RLOCID
- Disable Primary: Line Deactivation
- Disable Secondary: Line Deactivation
- Disable Remote Secondary: Line Deactivation
- Disable Remote Primary: Line Deactivation
- Establish Peer Configuration Session: Session Activation
- Establish Remote System Session: Session Activation
- Peer Terminates a Session It Created
- Peer Terminates a Remotely-Created Session
- Remote System Terminates a Session It Created
- Remote System Terminates a Peer-Created Session
- Acquire
- Evoke
- Evoke End of Transaction
- Evoke Then-Get/Invite
- Put
- Put Then-Get/Invite
- Put End of Group
- Put End of Transaction
- Put Fail
- Get/Invite/Accept-Input
- Local Half-Session in Transmit Mode Must Switch to Receive
- Request Change Direction
- Release
- End of Session

Enable Primary: Line Activation

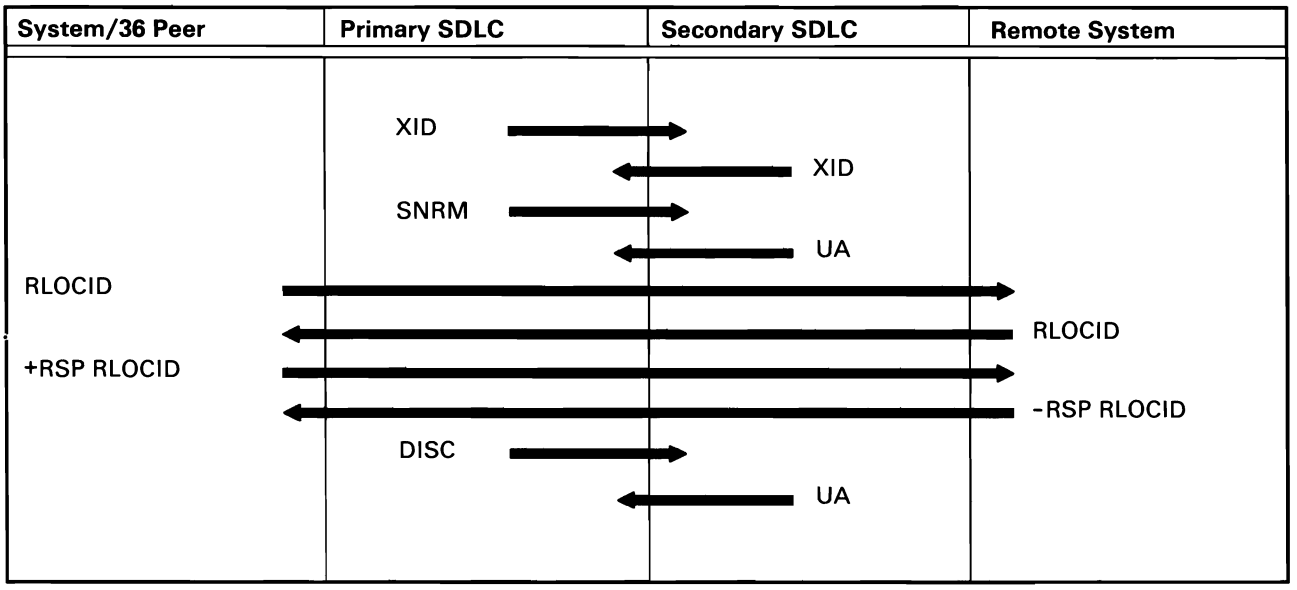


Note: XID and XID response flow are only on System/36 Peer leased line configurations.

S0590181-1

Figure A-10 (Part 1 of 31). Peer Subsystem Activation/Deactivation

Enable Primary with Remote Rejection of RLOCID: Line Activation

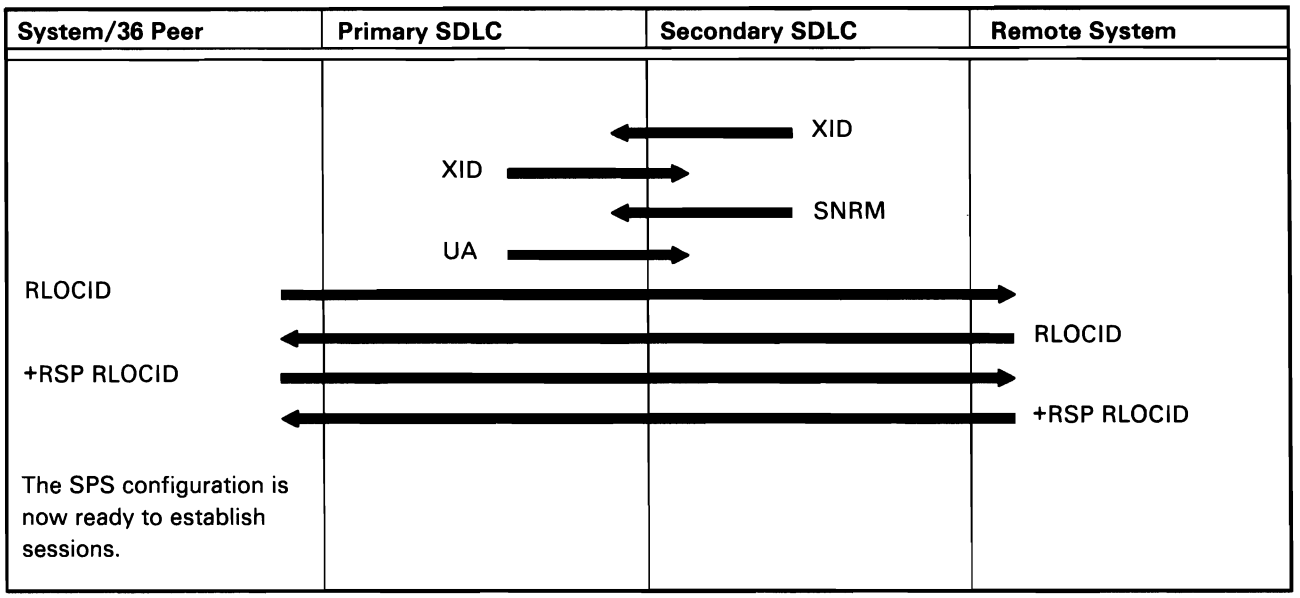


Note: XID and XID response flow are only on System/36 Peer leased line configurations.

S0590182-1

Figure A-10 (Part 2 of 31). Peer Subsystem Activation/Deactivation

Enable Secondary: Line Activation

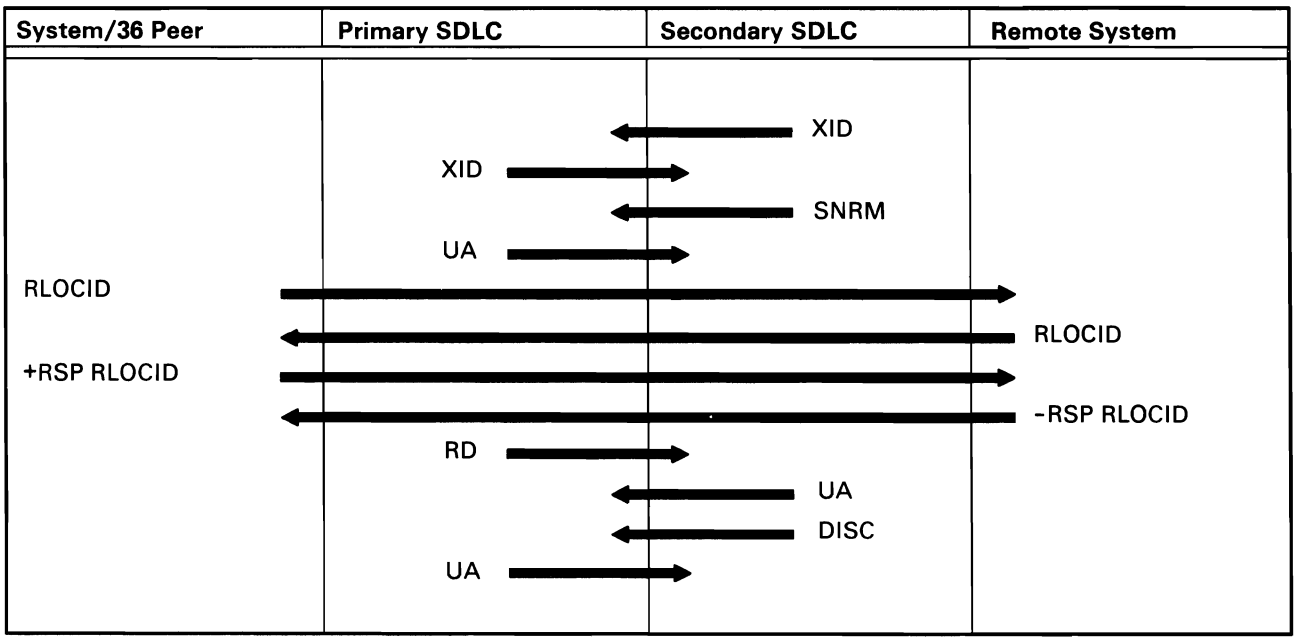


Note: XID and XID response flow are only on System/36 Peer leased line configurations.

S0590183-1

Figure A-10 (Part 3 of 31). Peer Subsystem Activation/Deactivation

Enable Secondary with Remote Rejection of RLOCID: Line Activation



Note: XID and XID response flow are only on System/36 Peer leased line configurations.

S0590184-1

Figure A-10 (Part 4 of 31). Peer Subsystem Activation/Deactivation

Disable Primary System: Line Deactivation

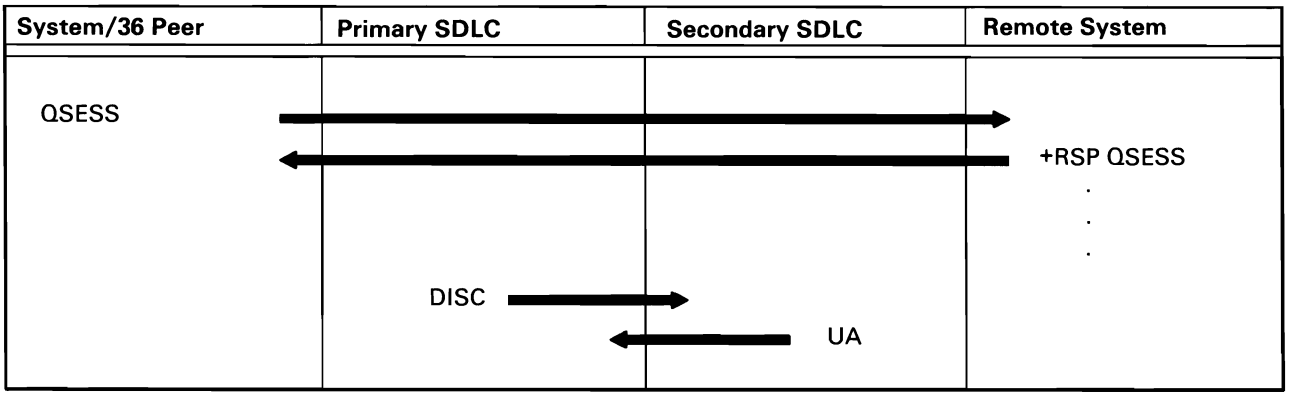


Figure A-10 (Part 5 of 31). Peer Subsystem Activation/Deactivation

Disable Secondary Configuration: Line Deactivation

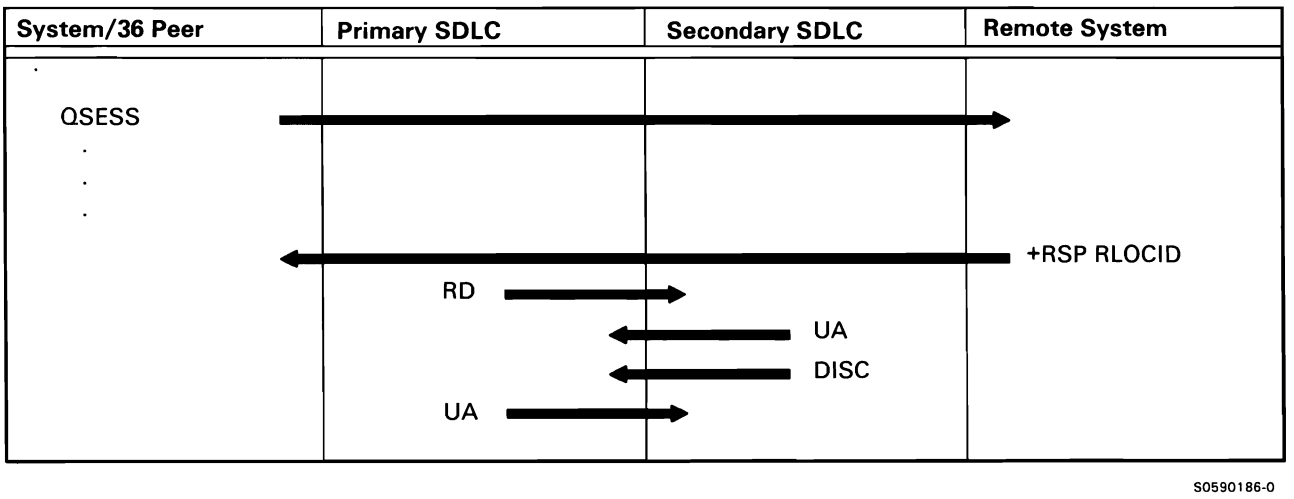


Figure A-10 (Part 6 of 31). Peer Subsystem Activation/Deactivation

Disable Remote Secondary Location: Line Deactivation

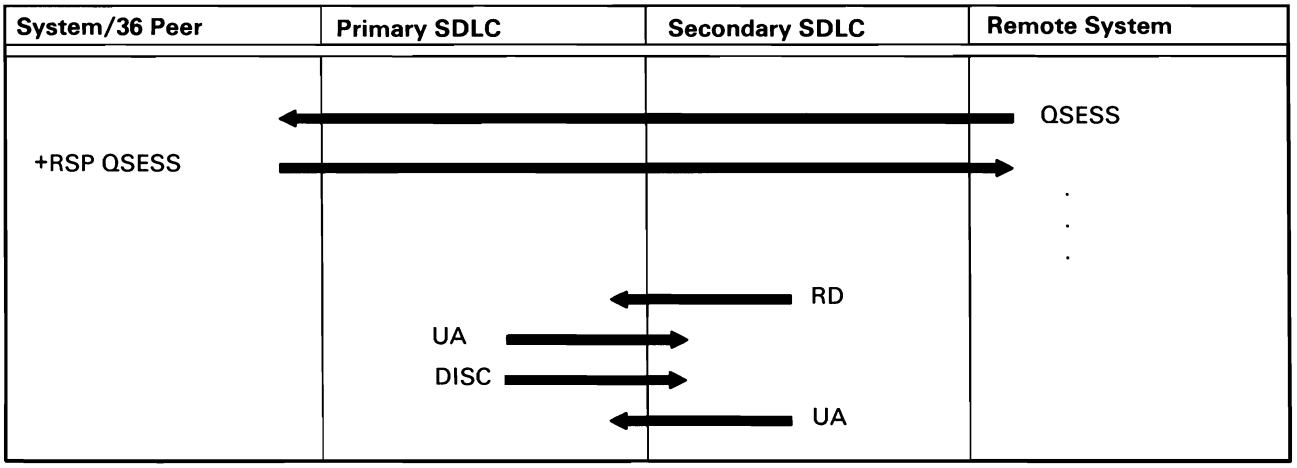


Figure A-10 (Part 7 of 31). Peer Subsystem Activation/Deactivation

Disable Remote Primary Location: Line Deactivation

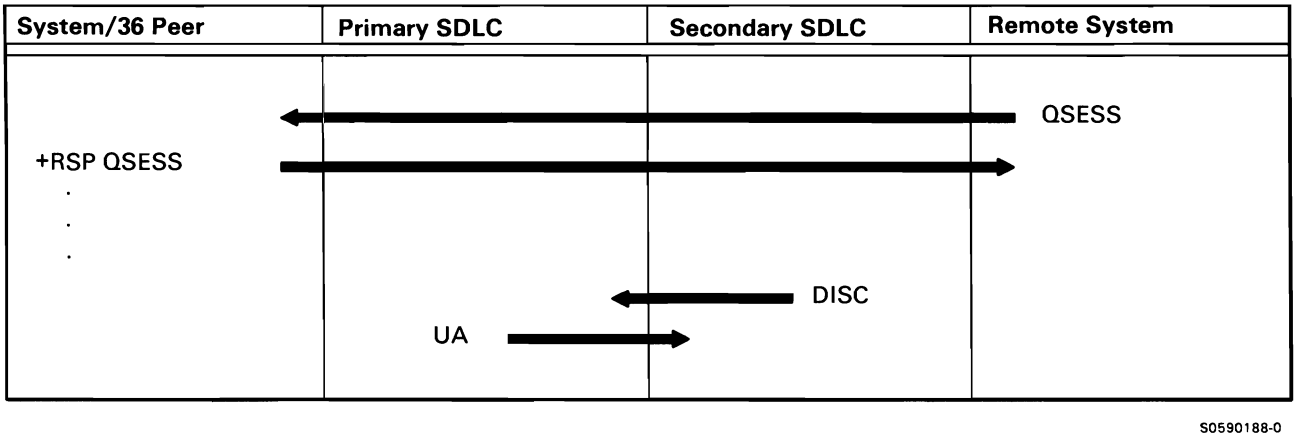


Figure A-10 (Part 8 of 31). Peer Subsystem Activation/Deactivation

Establish Peer Configuration Session: Session Activation

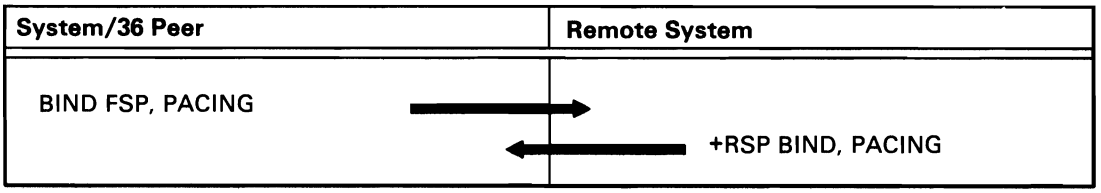


Figure A-10 (Part 9 of 31). Peer Subsystem Activation/Deactivation

Establish Remote System Session: Session Activation

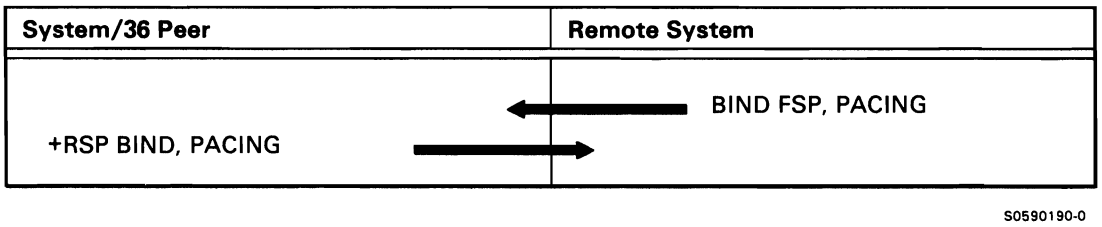


Figure A-10 (Part 10 of 31). Peer Subsystem Activation/Deactivation

Peer Session Deactivation

Peer Subsystem Terminates Session It Created

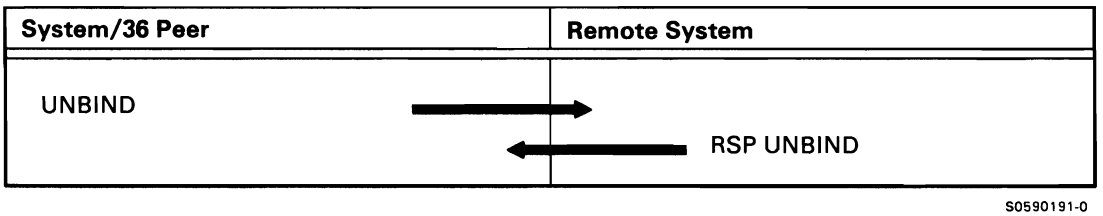


Figure A-10 (Part 11 of 31). Peer Subsystem Activation/Deactivation

Peer Subsystem Terminates Remotely-Created Session

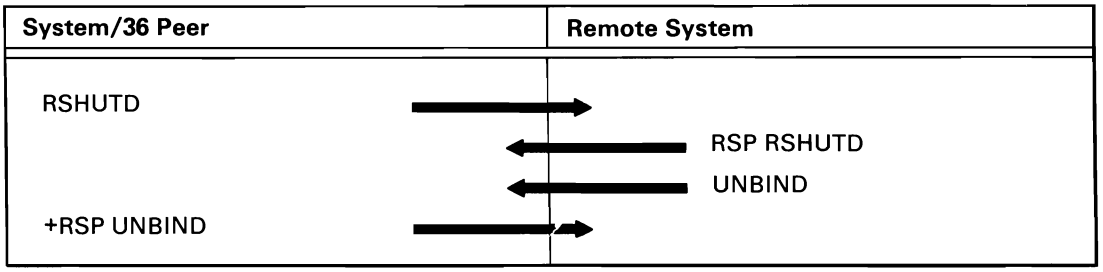


Figure A-10 (Part 12 of 31). Peer Subsystem Activation/Deactivation

Remote System Terminates a Session It Created

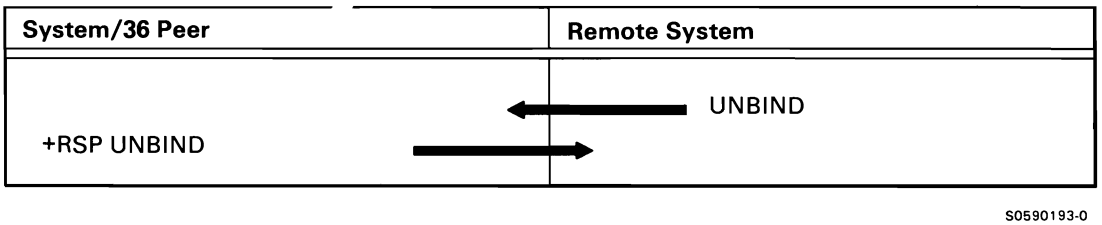


Figure A-10 (Part 13 of 31). Peer Subsystem Activation/Deactivation

Remote System Terminates a Peer-Created Session

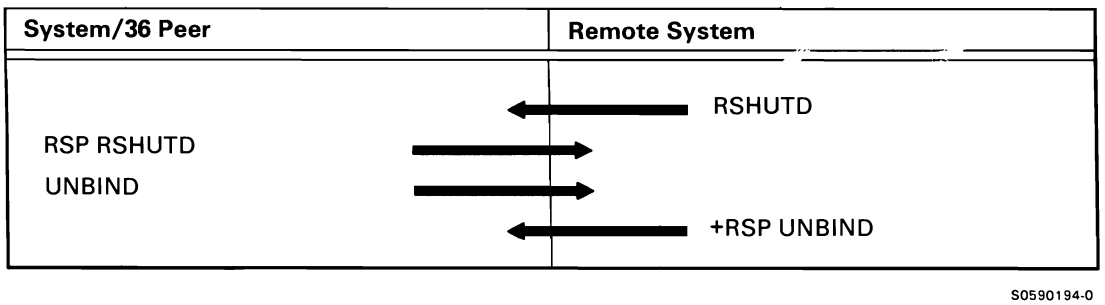


Figure A-10 (Part 14 of 31). Peer Subsystem Activation/Deactivation

SSP-ICF Peer Operations

Acquire (ACQ)

If preestablished sessions are available, no line flow occurs. If preestablished sessions are not available, the following line flow takes place:

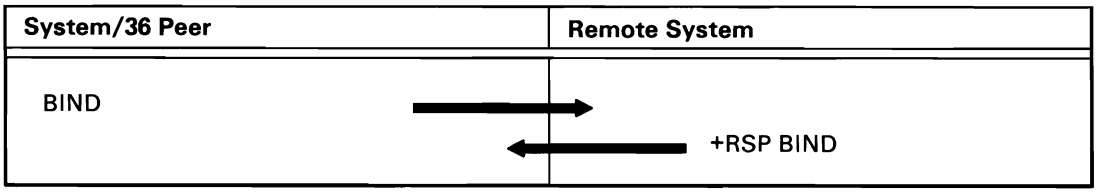


Figure A-10 (Part 15 of 31). Peer Subsystem Activation/Deactivation

Evoke (EVK)

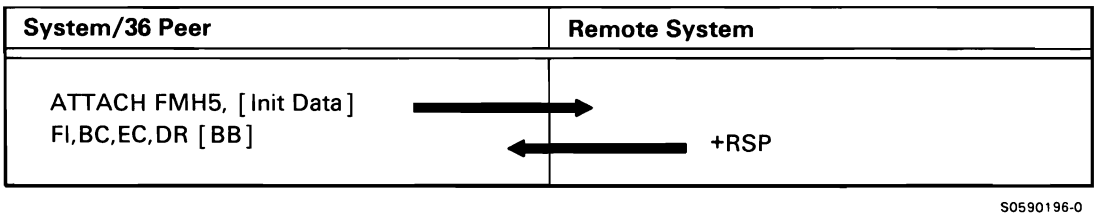


Figure A-10 (Part 16 of 31). Peer Subsystem Activation/Deactivation

Evoke End of Transaction (EVE)

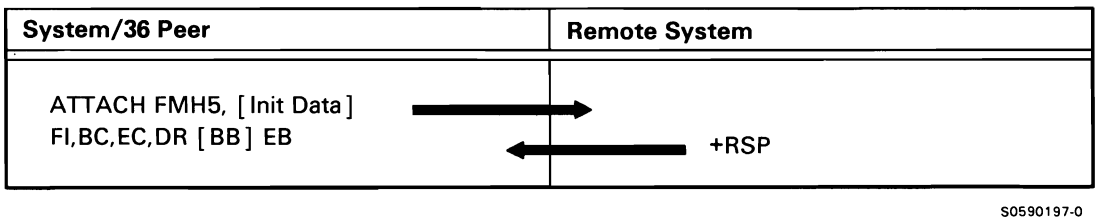


Figure A-10 (Part 17 of 31). Peer Subsystem Activation/Deactivation

Evoke Then-Get/Invite (EVG/EVI)

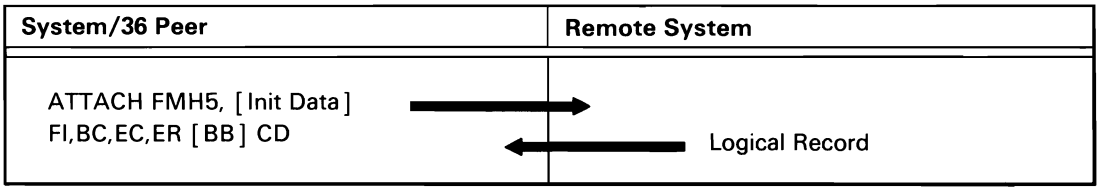


Figure A-10 (Part 18 of 31). Peer Subsystem Activation/Deactivation

Put

Note: For all put operations, BC is sent only if the put operation is issued while the half-session is between chains.

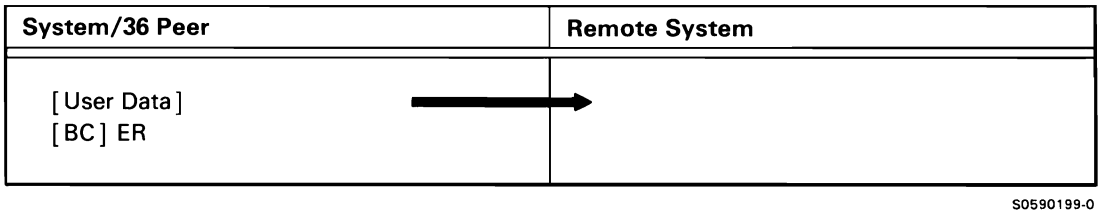


Figure A-10 (Part 19 of 31). Peer Subsystem Activation/Deactivation

Put Then-Get/Invite (PTG/PTI)

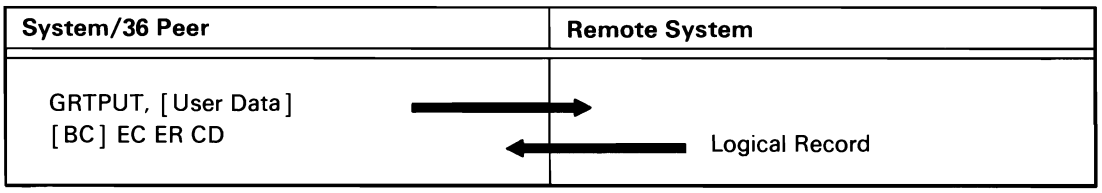


Figure A-10 (Part 20 of 31). Peer Subsystem Activation/Deactivation

Put End of Group

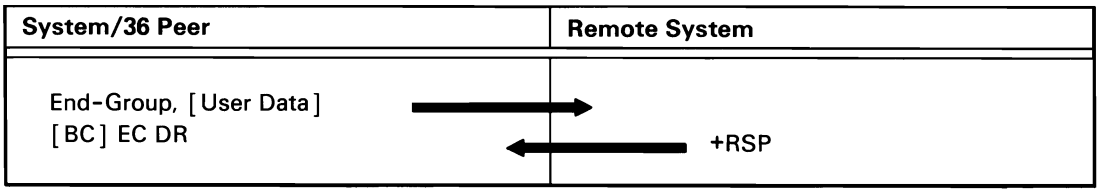


Figure A-10 (Part 21 of 31). Peer Subsystem Activation/Deactivation

Put End of Transaction (PEX)

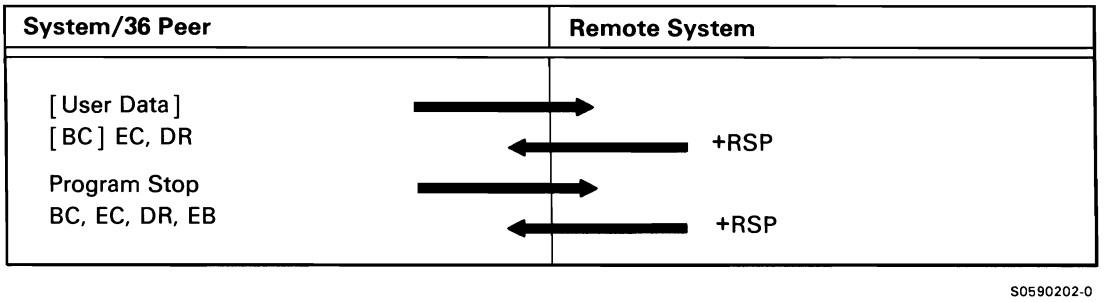


Figure A-10 (Part 22 of 31). Peer Subsystem Activation/Deactivation

Put Fail (Local Half-Session in Transmit Mode)

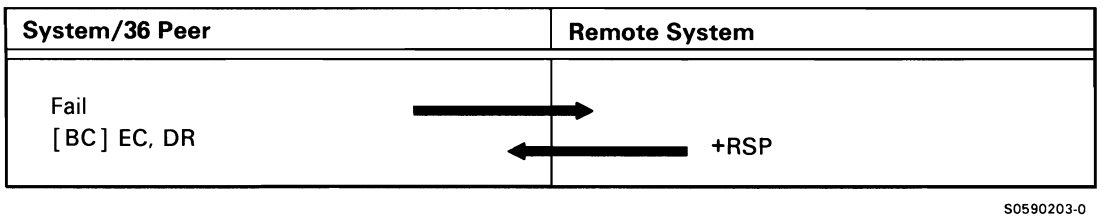


Figure A-10 (Part 23 of 31). Peer Subsystem Activation/Deactivation

Put Fail (Local Half-Session in Receive Mode)

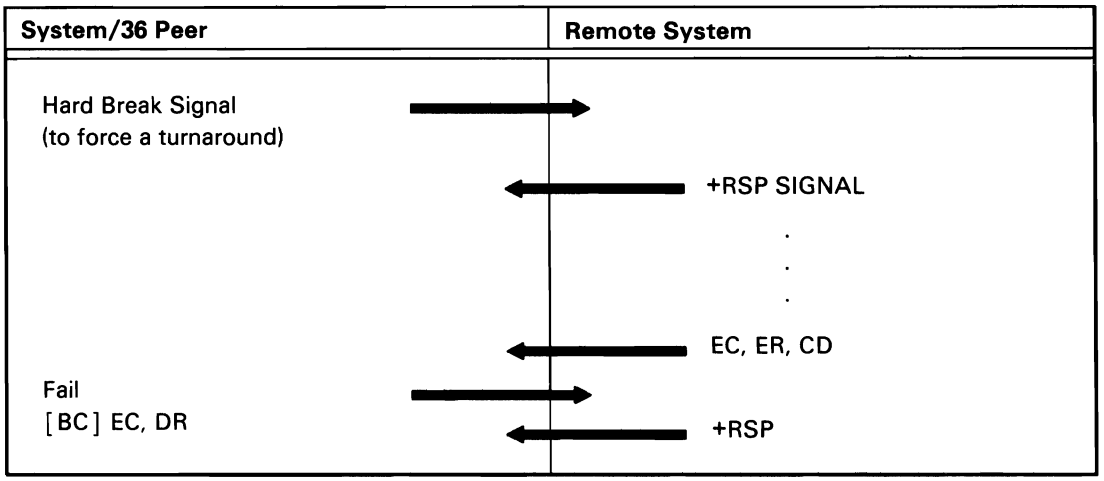


Figure A-10 (Part 24 of 31). Peer Subsystem Activation/Deactivation

Get/Invite/Accept-Input (GET)

If the local half-session is in receive mode:

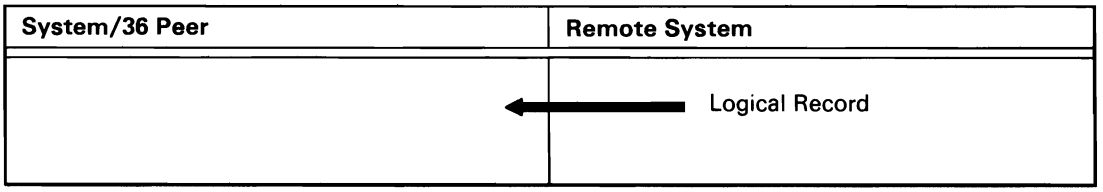


Figure A-10 (Part 25 of 31). Peer Subsystem Activation/Deactivation

If the local half-session is in transmit mode and wants to receive:

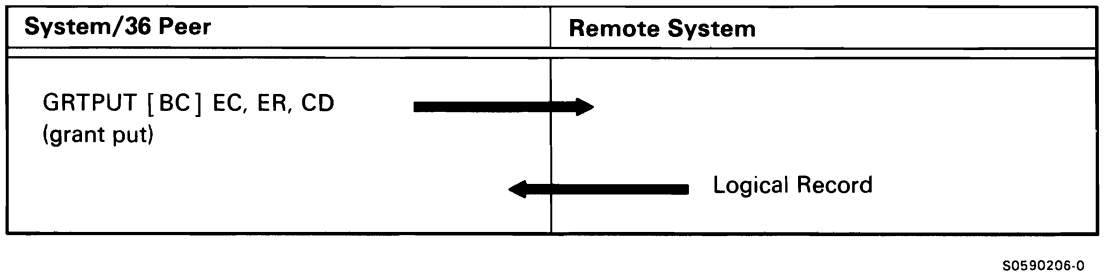


Figure A-10 (Part 26 of 31). Peer Subsystem Activation/Deactivation

Request Change Direction–Get/Invite (RCDG/RCDI)

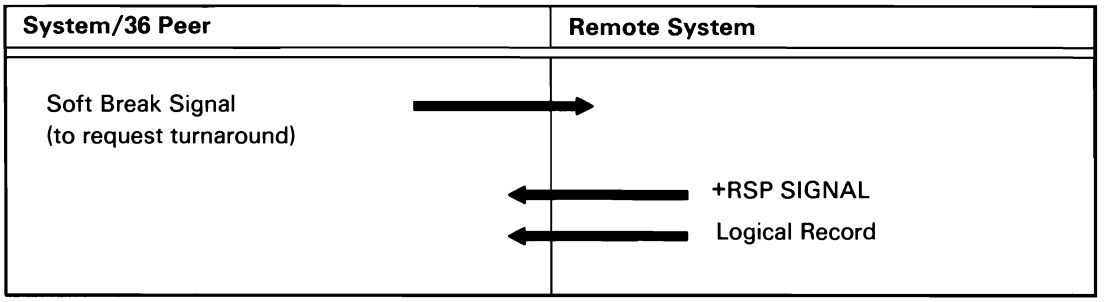


Figure A-10 (Part 27 of 31). Peer Subsystem Activation/Deactivation

Release (REL)

If the session had been preestablished, no line flow occurs. If the session had not been preestablished, the following flow occurs:

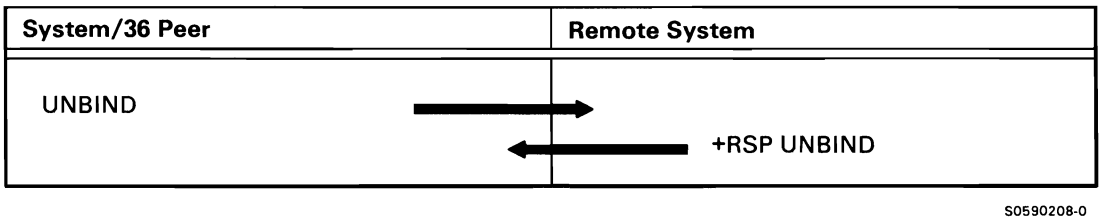


Figure A-10 (Part 28 of 31). Peer Subsystem Activation/Deactivation

End of Session (EOS)

If the local half-session is in transmit mode:

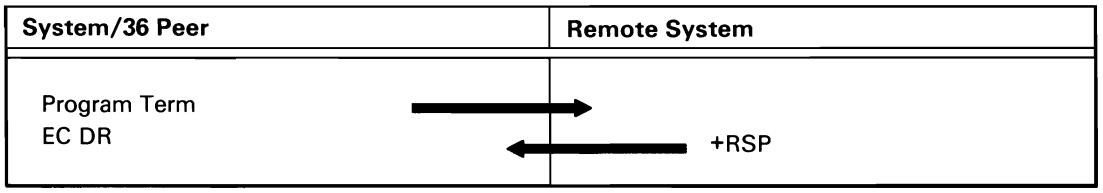


Figure A-10 (Part 29 of 31). Peer Subsystem Activation/Deactivation

If the local half-session is in receive mode:

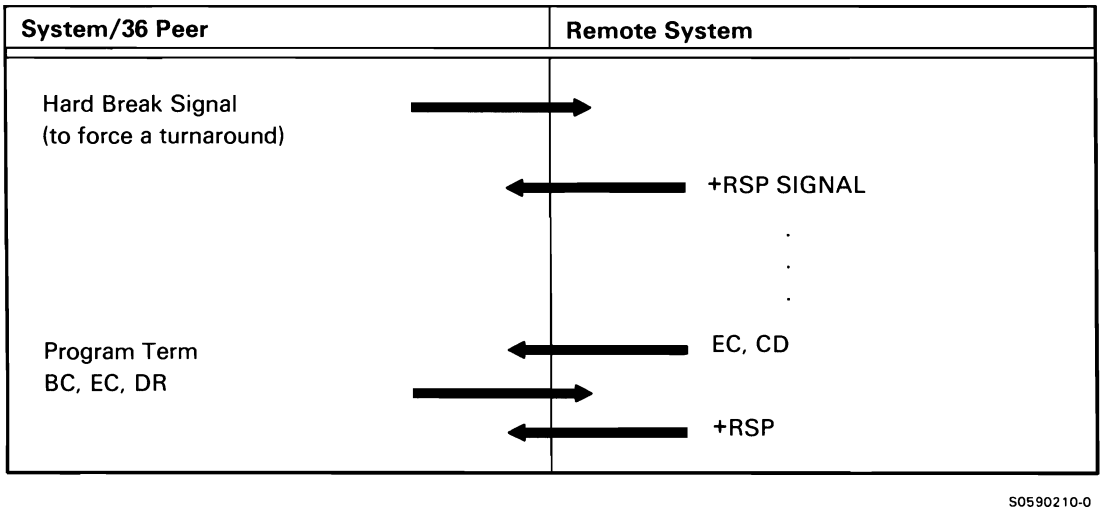


Figure A-10 (Part 30 of 31). Peer Subsystem Activation/Deactivation

If the session had been preestablished, no control flow occurs. If the session had not been preestablished, the following occurs:

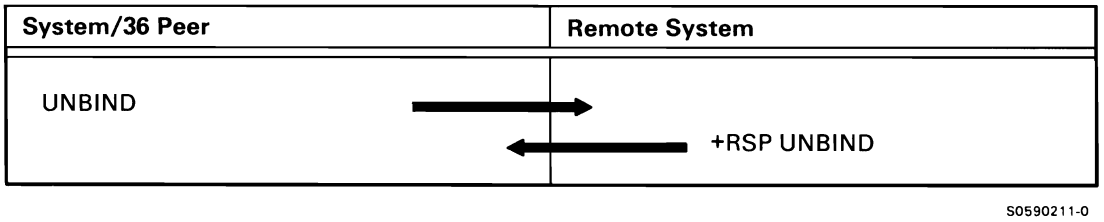


Figure A-10 (Part 31 of 31). Peer Subsystem Activation/Deactivation

APPC Subsystem

Figure A-11 shows the protocol flow generated when line and session activation/deactivation and data transmission take place between the System/36 APPC subsystem and the remote system.

The following APPC operations are shown:
- Enable: Peer Network with Remote Configured as Primary
- Enable: Host-Type Network
- STARTGRP Command
- Session Initiation/Allocation
- Transaction Initiation
- Input/Output
- Session Deallocation/Termination
- STOPGRP Command
- Disable

Enable: Peer Network with Remote Configured as Primary

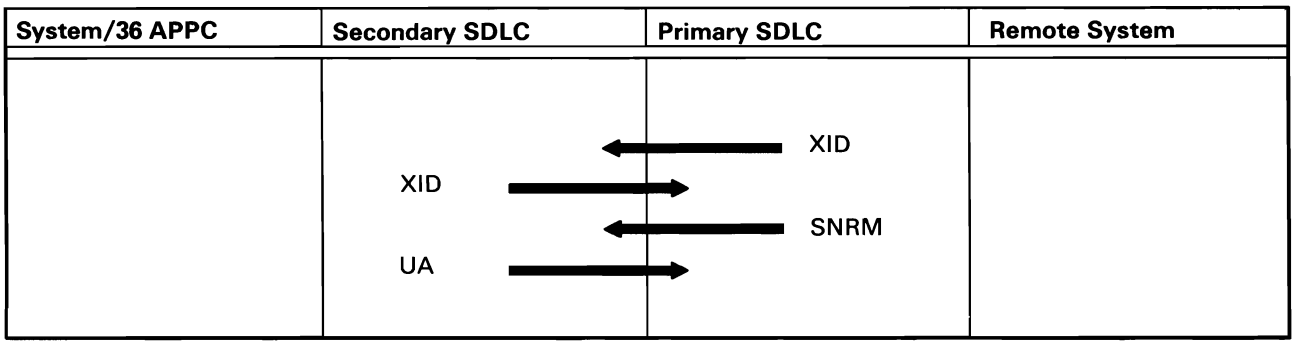


Figure A-11 (Part 1 of 10). APPC Subsystem Enable

Enable: Host-Type Network

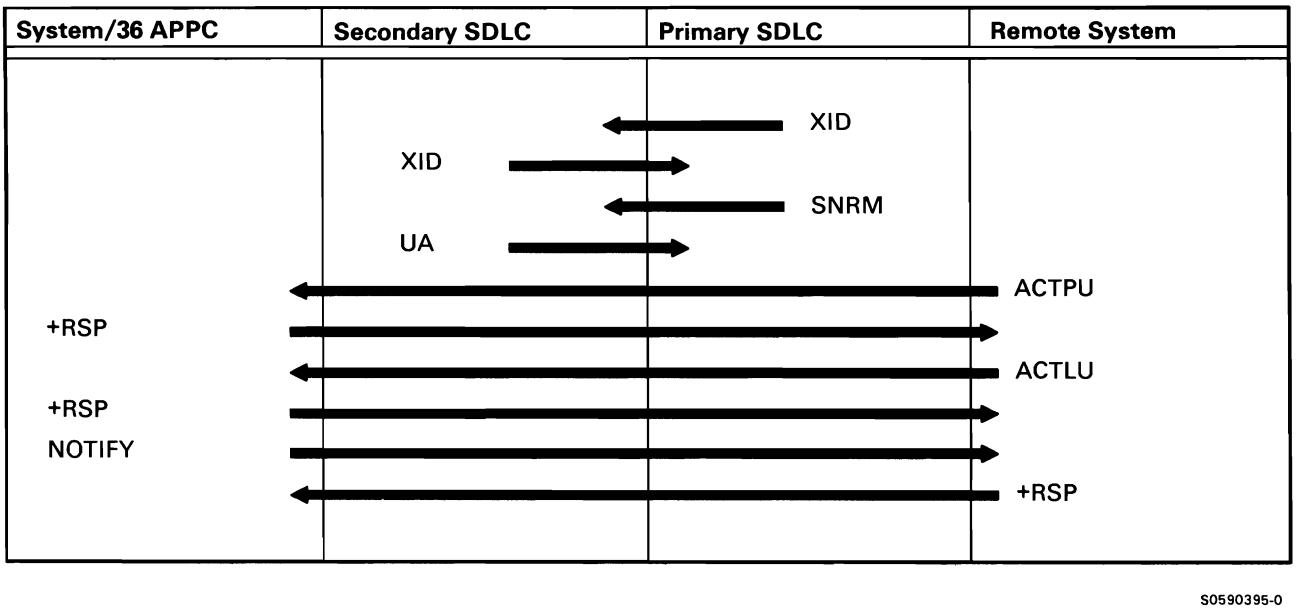
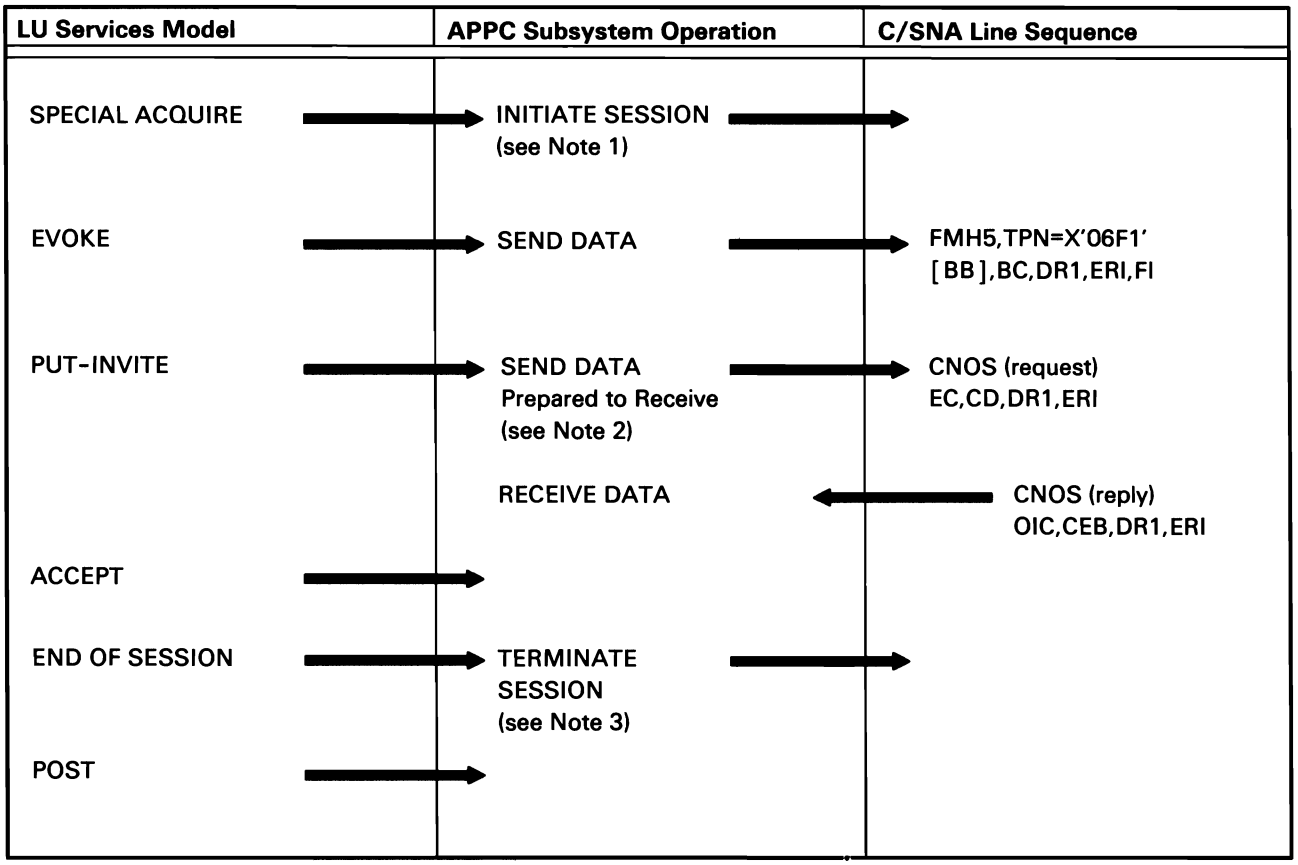


Figure A-11 (Part 2 of 10). APPC Subsystem Enable



Notes:

1. *See **Session Initiation/Allocation** for detail line flows concerning the acquire operations.*

2. *Parallel session connections only. For single-session connections the CNOS exchange is not performed; only the post to the subsystem is performed.*

3. *See **Session Deallocation/Termination** detail line flows concerning the end-of-session operations.*

S0590396-2

Figure A-11 (Part 3 of 10). APPC Subsystem STARTGRP Command

If preestablished sessions are available, no line flow occurs. If preestablished sessions are not available, the following line flow takes place.

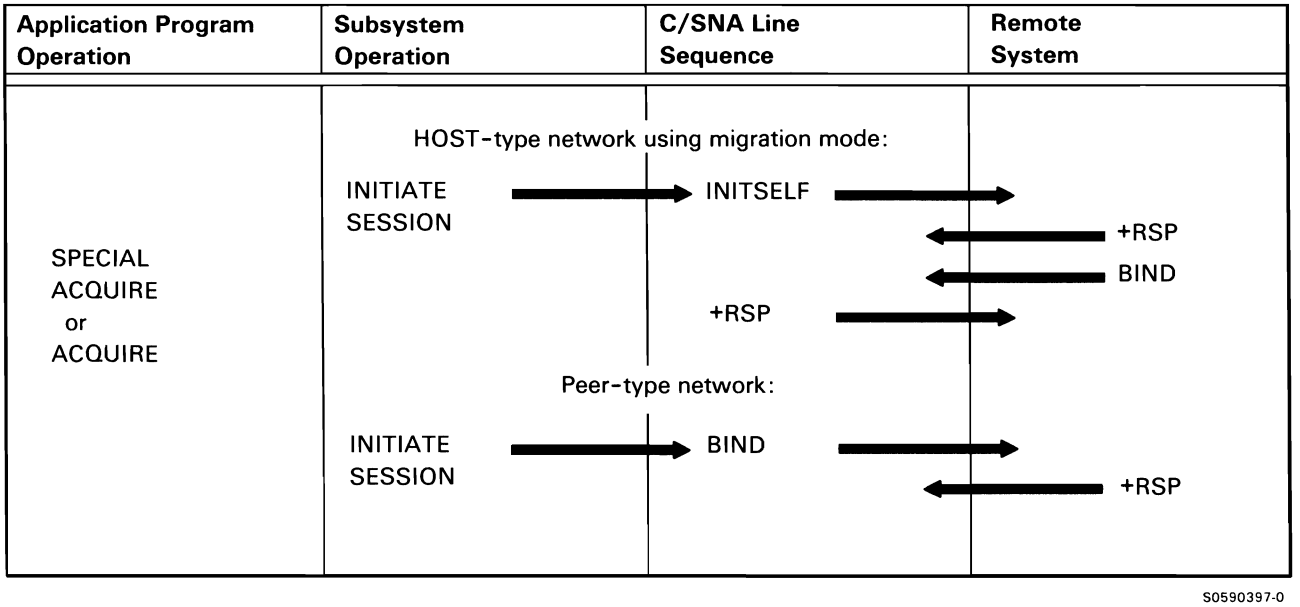
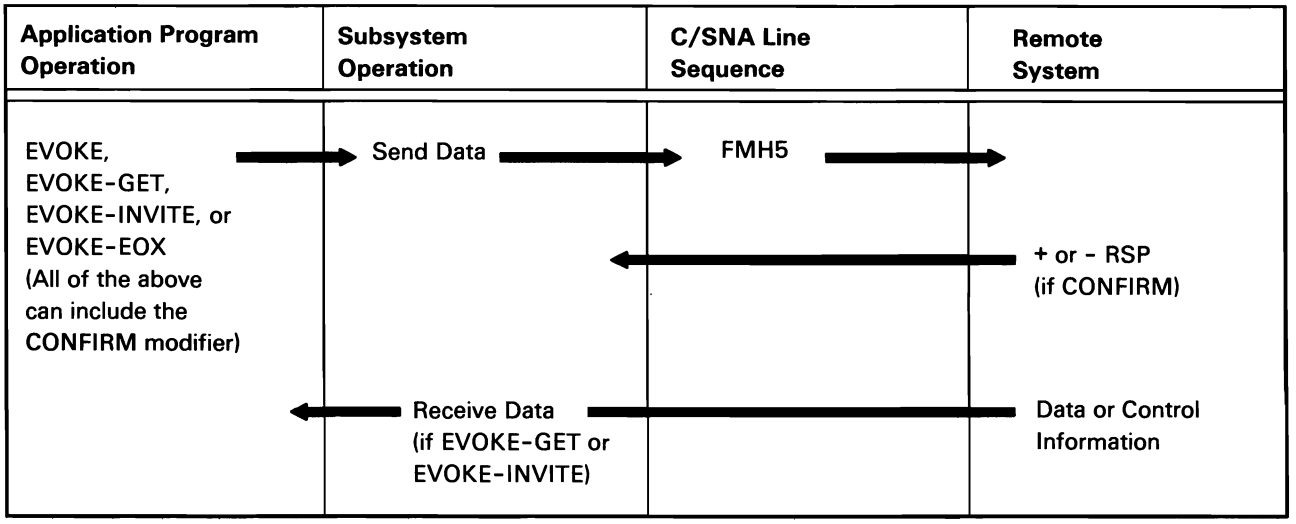


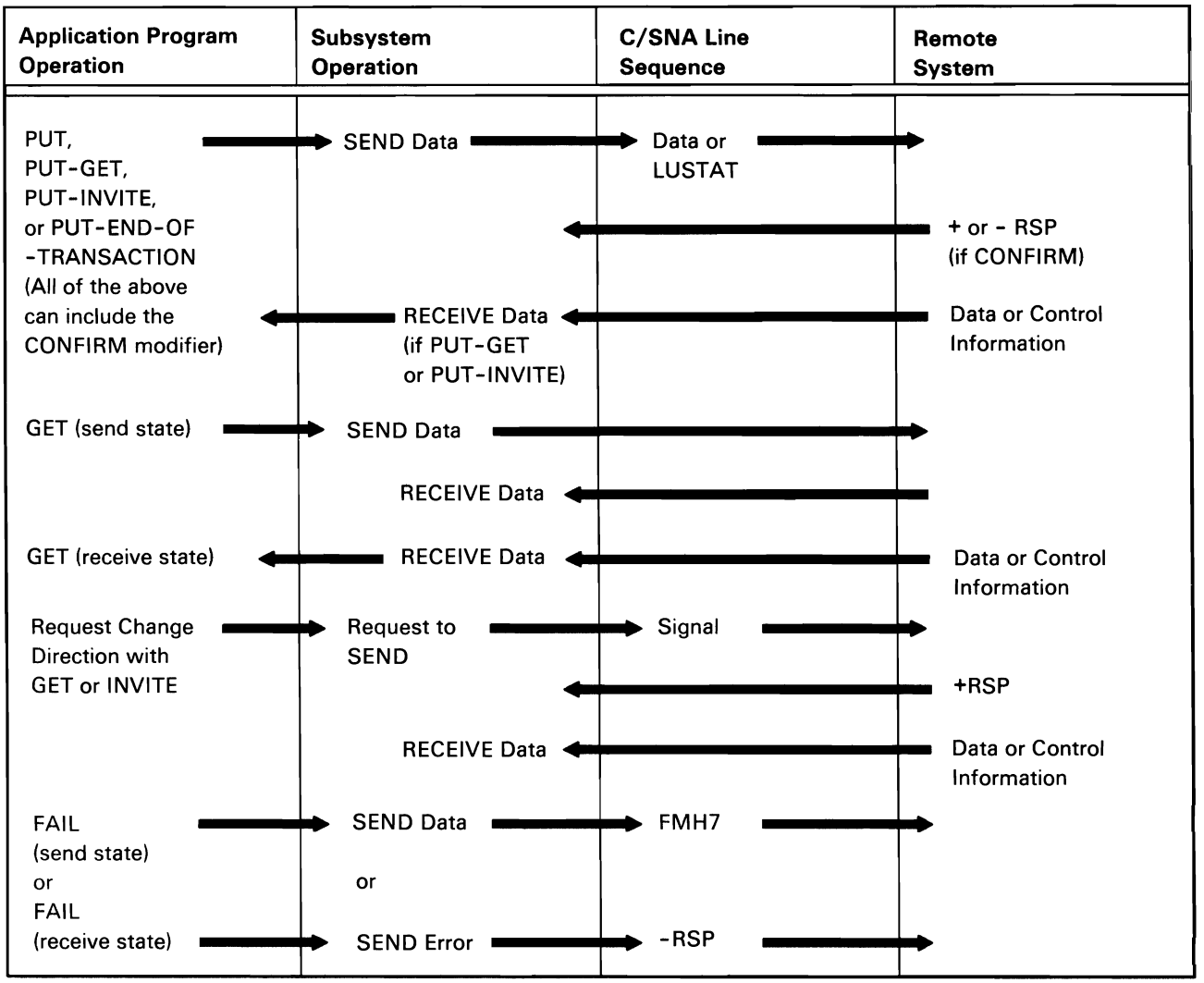
Figure A-11 (Part 4 of 10). APPC Subsystem Session Initiation/Allocation



Note: The +RSP to confirm is not required if GET was specified. Instead, the data or control information is treated as a +RSP acknowledgment.

S0590413-2

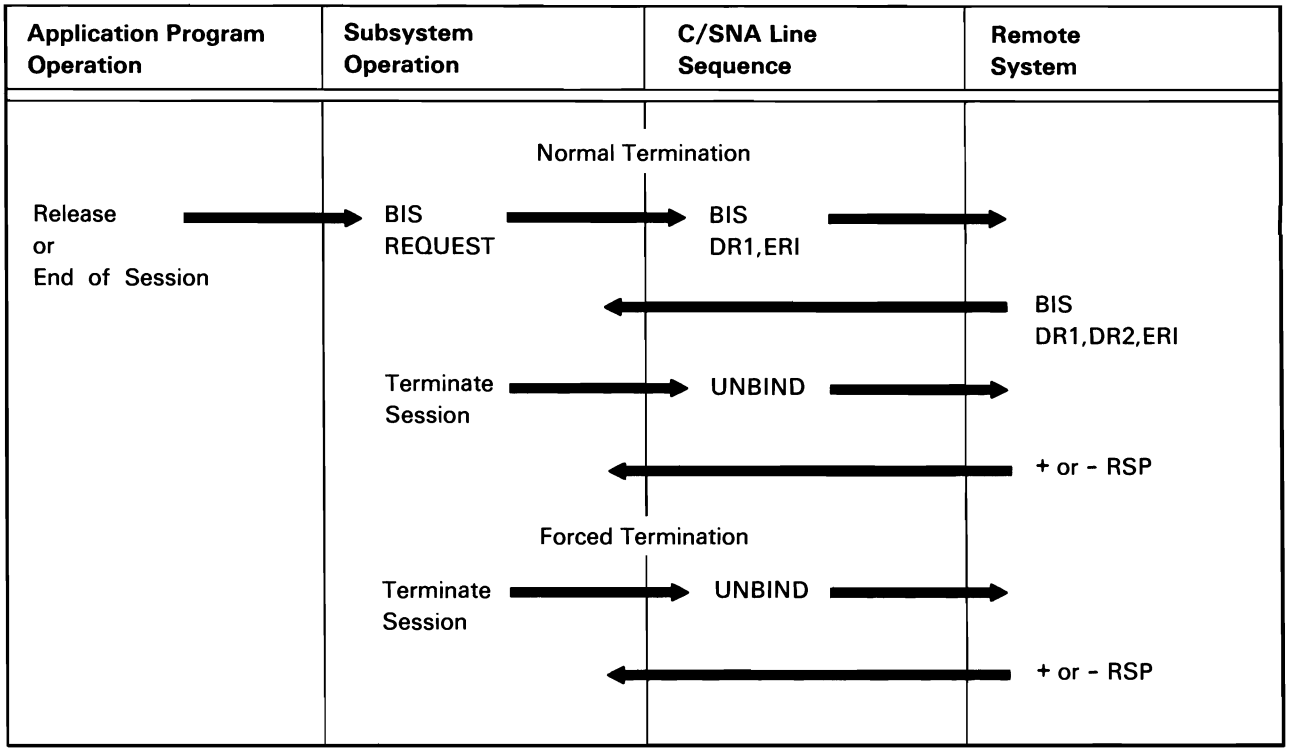
Figure A-11 (Part 5 of 10). APPC Subsystem Transaction Initiation



Note: The +RSP is not required if GET was specified. Instead, the data or control information is treated as a +RSP acknowledgment.

S0590414-2

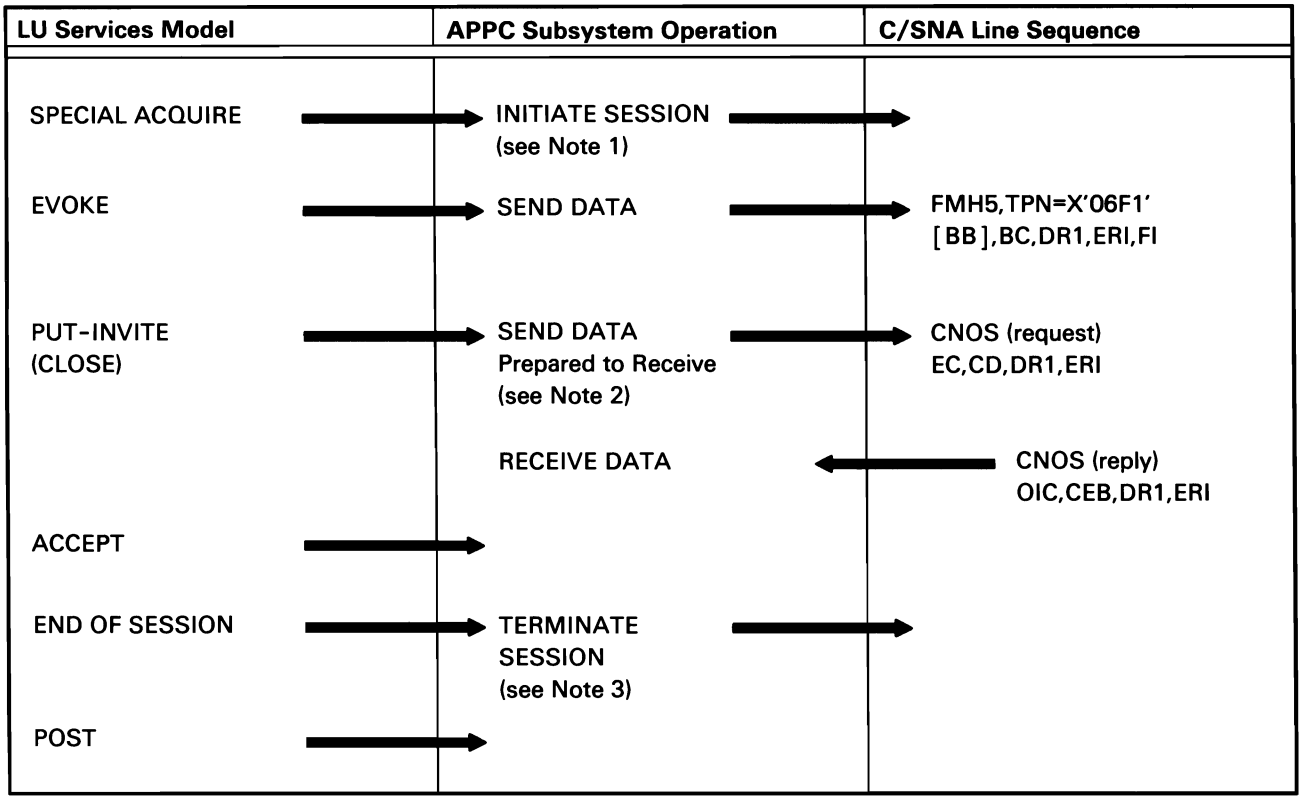
Figure A-11 (Part 6 of 10). APPC Subsystem Input/Output



Note: If the session had been preestablished, the session remains bound.

S0590415-1

Figure A-11 (Part 7 of 10). APPC Subsystem Session Deallocation/Termination



Notes:

1. *See* ***Session Initiation/Allocation*** *for detail line flows concerning the acquire operations.*

2. *Parallel session connections only. For single-session connections the CNOS exchange is not performed; only the post to the subsystem is performed.*

3. *See* ***Session Deallocation/Termination*** *detail line flows concerning the end-of-session operations.*

S0590029-2

Figure A-11 (Part 8 of 10). APPC Subsystem STOPGRP Command

Disable: Peer Network with Remote Configured as Primary

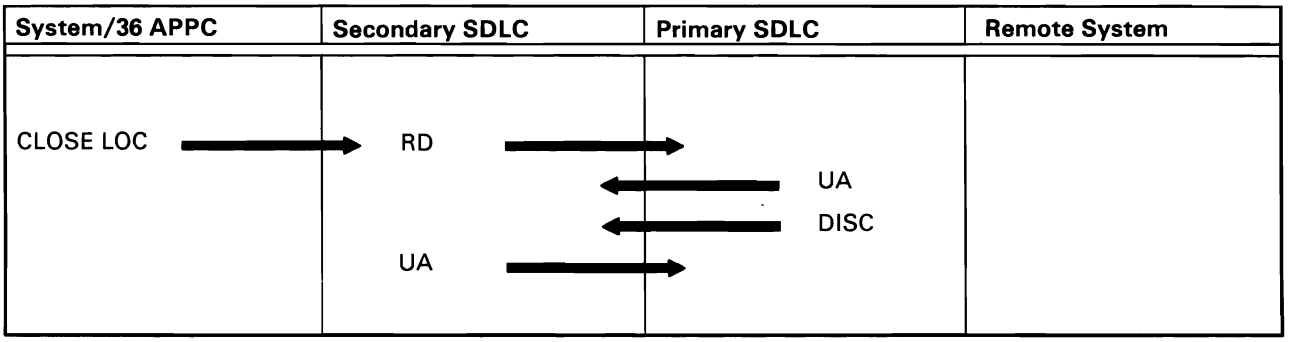


Figure A-11 (Part 9 of 10). APPC Subsystem Disable

Disable: Host-Type Network

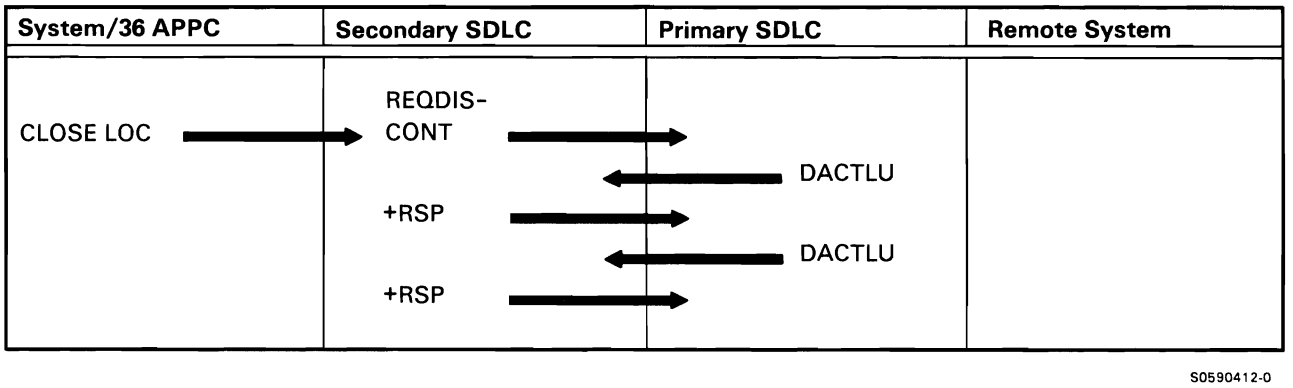


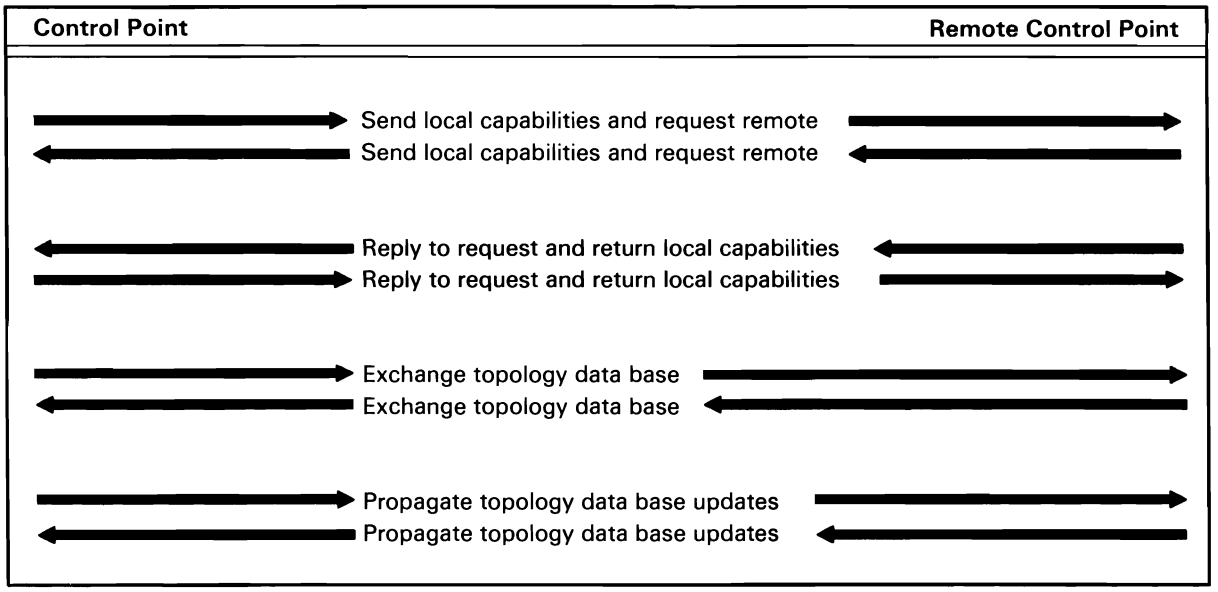
Figure A-11 (Part 10 of 10). APPC Subsystem Disable

APPN Subsystem

Figure A-12 shows the protocol flow between the control point modules within the APPN subsystem. Since these control point modules communicate using a modified APPC subsystem, the lower level protocol flows are the same as those for APPC.

The following APPN operations are shown:
- Enable: Between Two APPN Nodes with Both Enabled as Networking Nodes
- Topology Update: Between Two APPN Nodes
- Session Request: If Directory Services and Network Search Are Required

Enable: Between Two APPN Nodes with Both Enabled as Networking Nodes

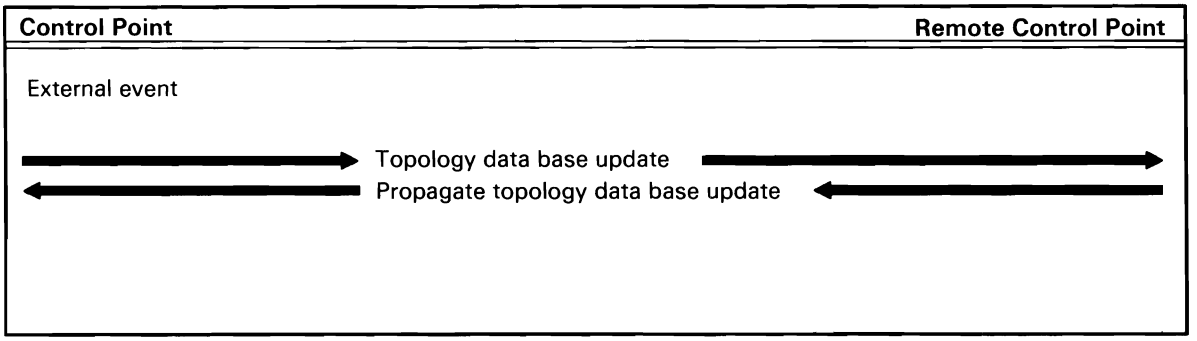


Note: This sequence will occur whenever a line is enabled on the CPSVCMG session between two networking nodes on the line and/or whenever the CPSVCMG session fails and is brought back up.

S0590490-0

Figure A-12 (Part 1 of 3). APPN Subsystem Enable

Topology Update: Between Two APPN Nodes

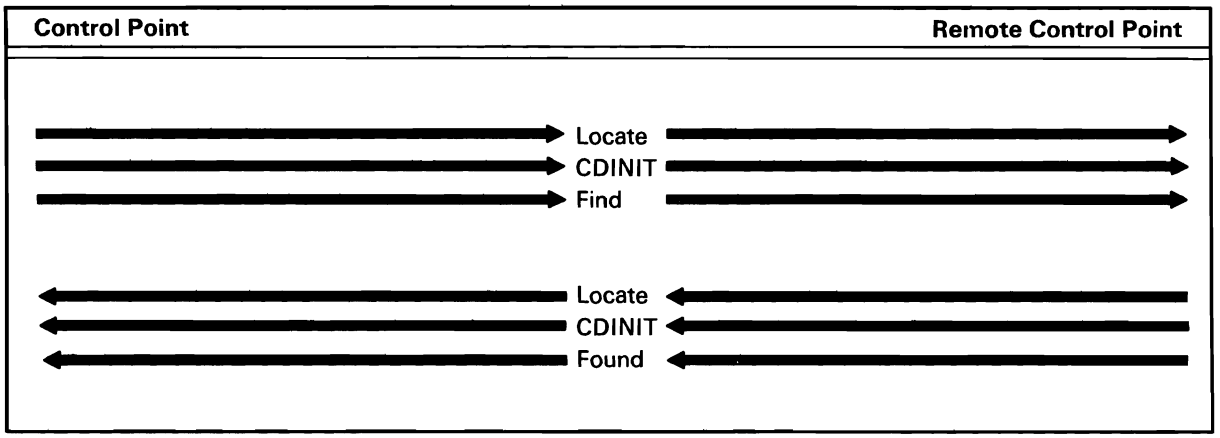


Note: External event is any action that affects the networks topology such as a node or link changing status, nodes becoming congested for routing, and disable pending. These will cause updates to flow throughout all the attached and communicating network nodes.

S0590491-0

Figure A-12 (Part 2 of 3). APPN Subsystem Topology Update

Session Request: If Directory Services and Network Search Are Required



Note: If directory services has previous knowledge of the requested location, a directed search will be sent. This is similar to a BIND flowing where the route is specified. If directory services has no knowledge of the requested location, the originating nodes and all nodes that receive the search request will send a search request to all of their neighbors (unless they are the requested location).

S0590492-0

Figure A-12 (Part 3 of 3). APPN Subsystem Session Request

X.25 Line Protocols

Figure A-13 shows the protocol flow within the X.25 link.

The following X.25 operations are shown:
- Establishing the X.25 Link
- Terminating the X.25 Link
- Opening an X.25 Virtual Circuit
- Closing an X.25 Virtual Circuit
- Transferring Data in an X.25 Network

Establishing an X.25 Link

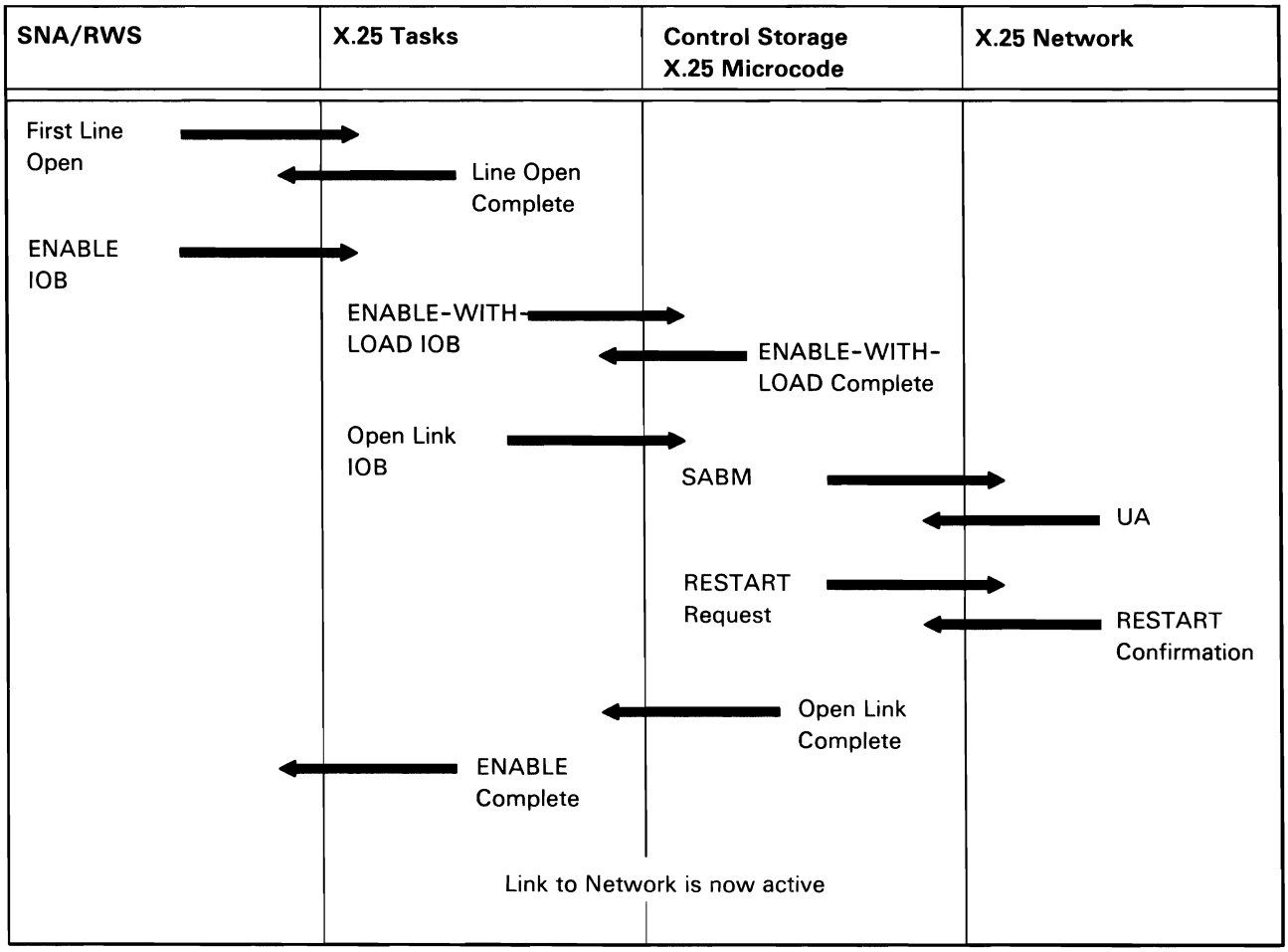


Figure A-13 (Part 1 of 7). X.25 Link Initialization

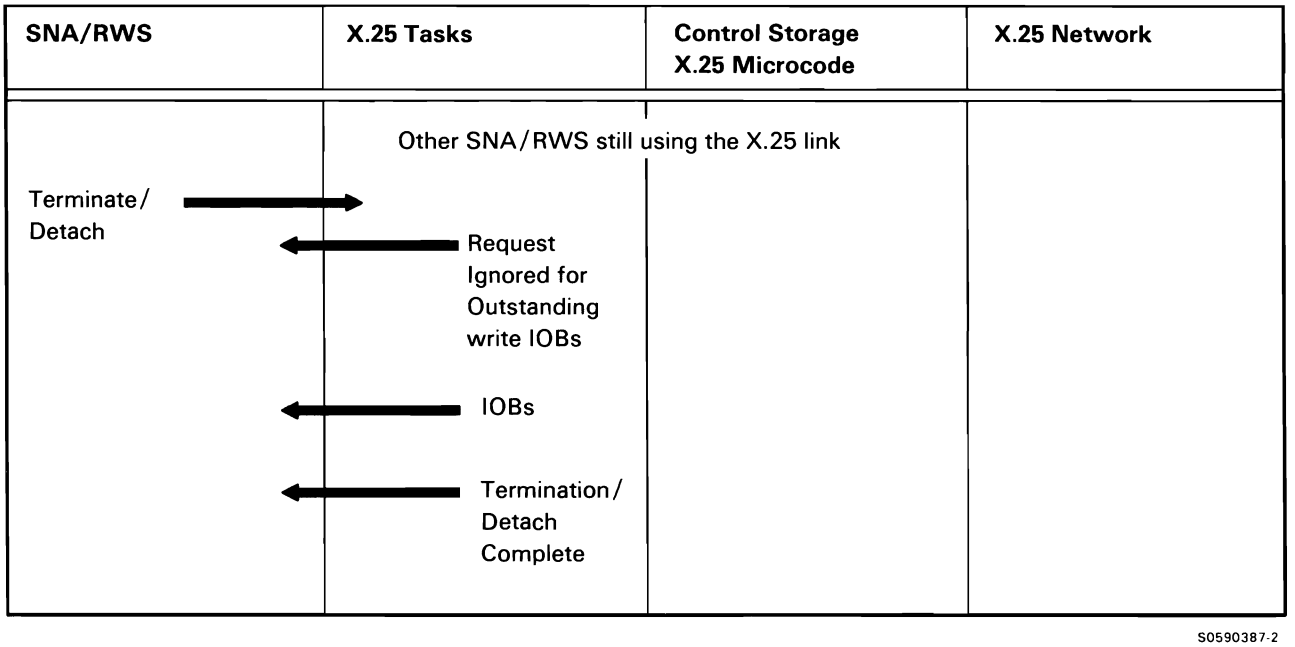


Figure A-13 (Part 2 of 7). X.25 Link Termination

Establishing an X.25 Virtual Circuit in Response to an Incoming Call

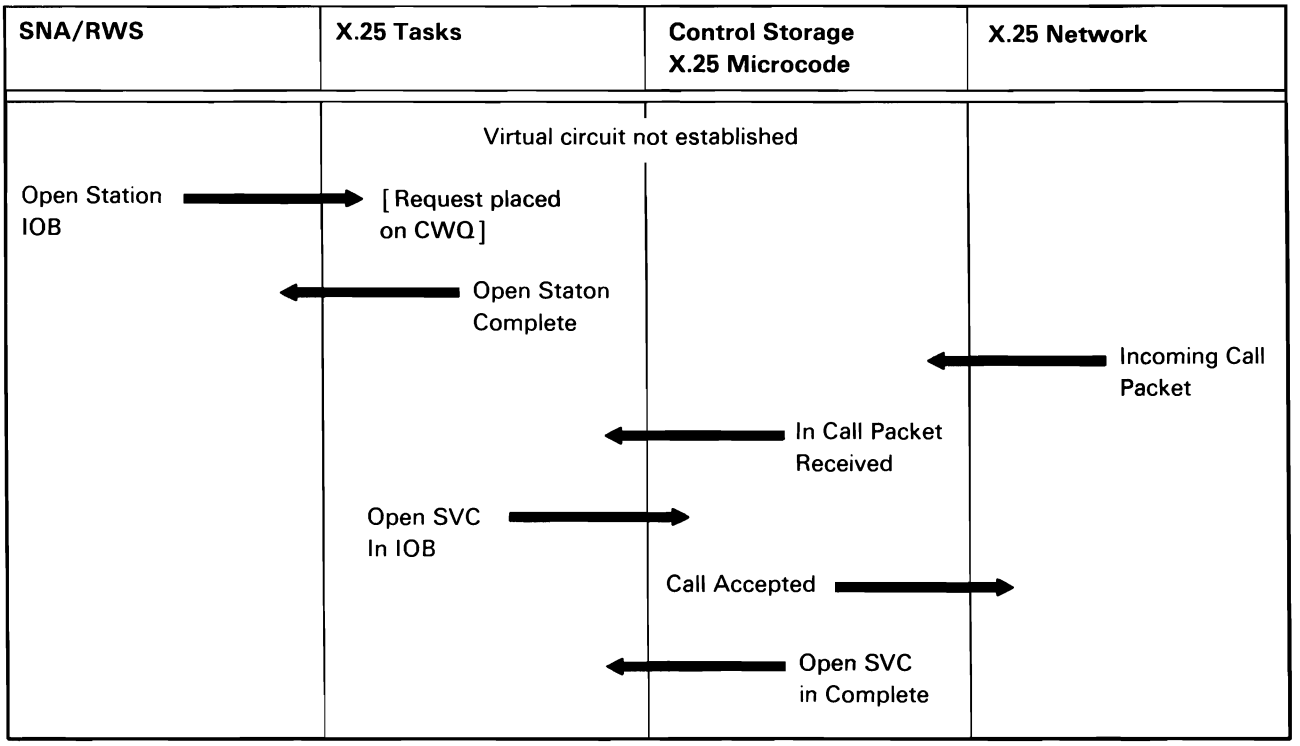


Figure A-13 (Part 3 of 7). Opening an X.25 Virtual Circuit

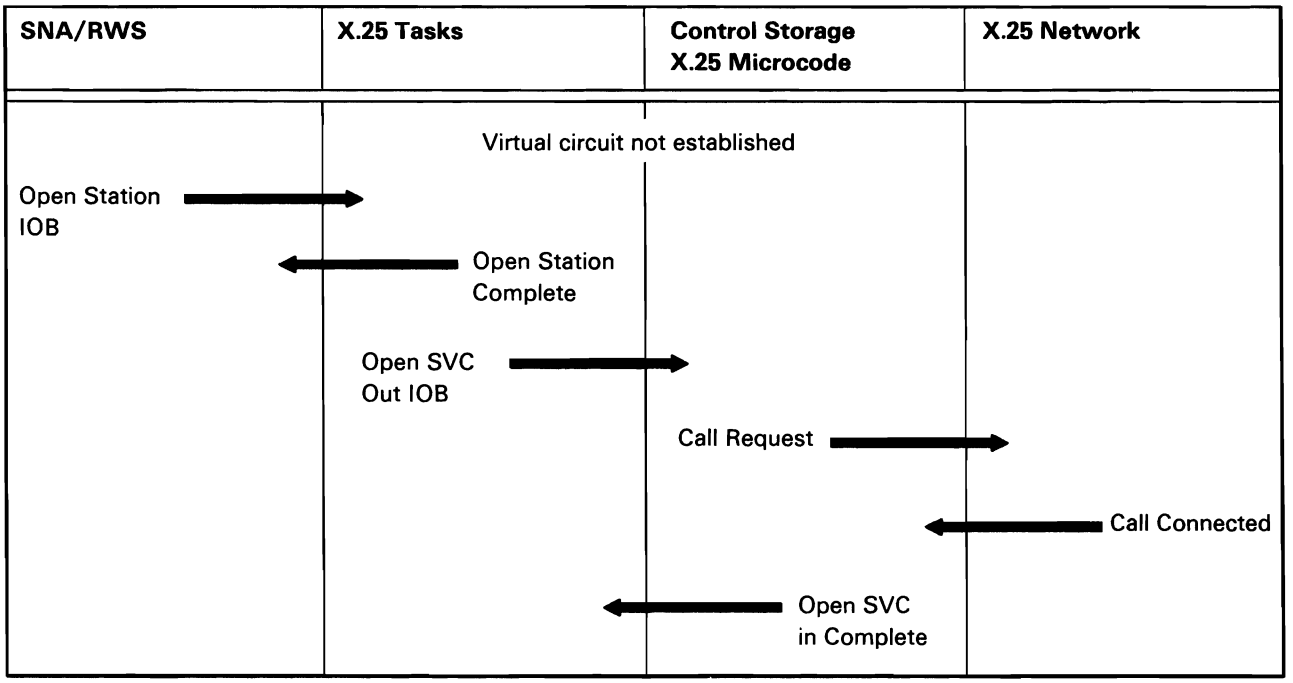


Figure A-13 (Part 4 of 7). Opening an X.25 Virtual Circuit

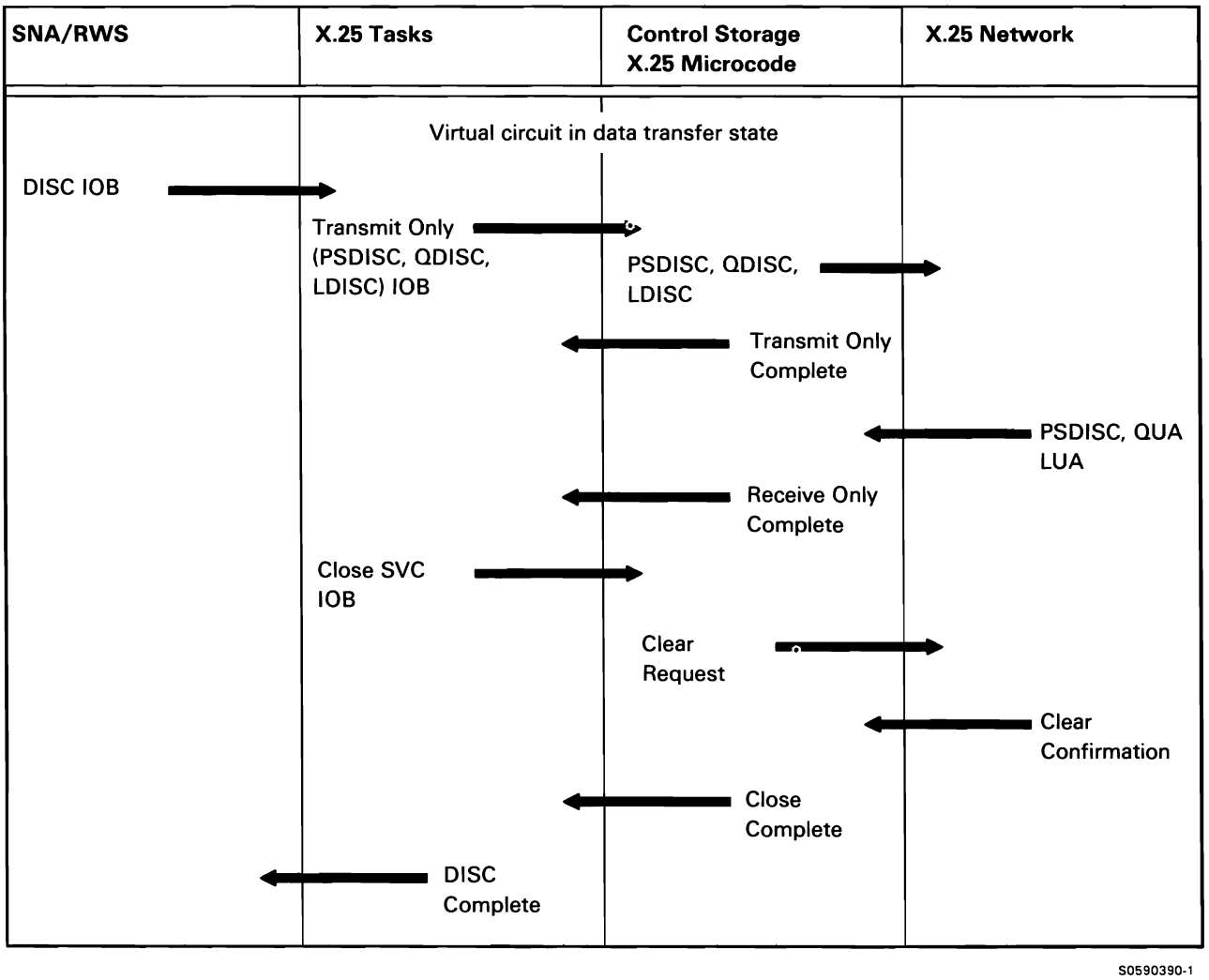


Figure A-13 (Part 5 of 7). Closing an X.25 Virtual Circuit

Transmitting Data in an X.25 Link

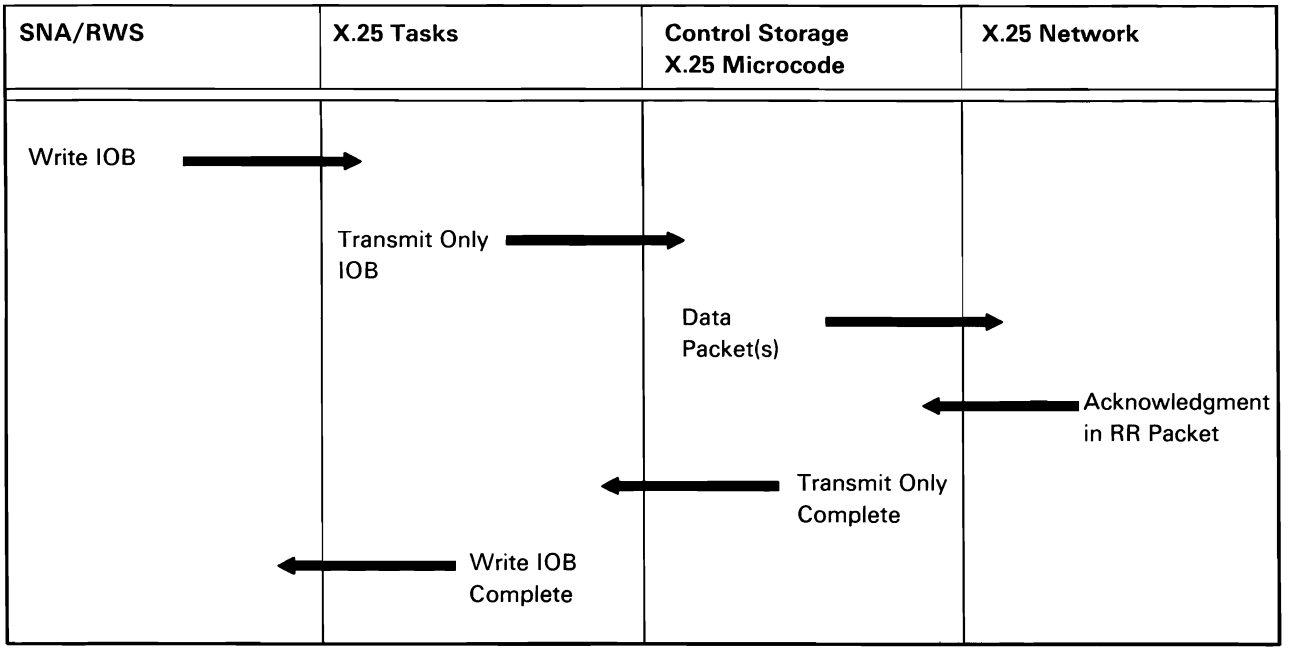


Figure A-13 (Part 6 of 7). X.25 Virtual Circuit Data Transfer

Receiving Data in an X.25 Link

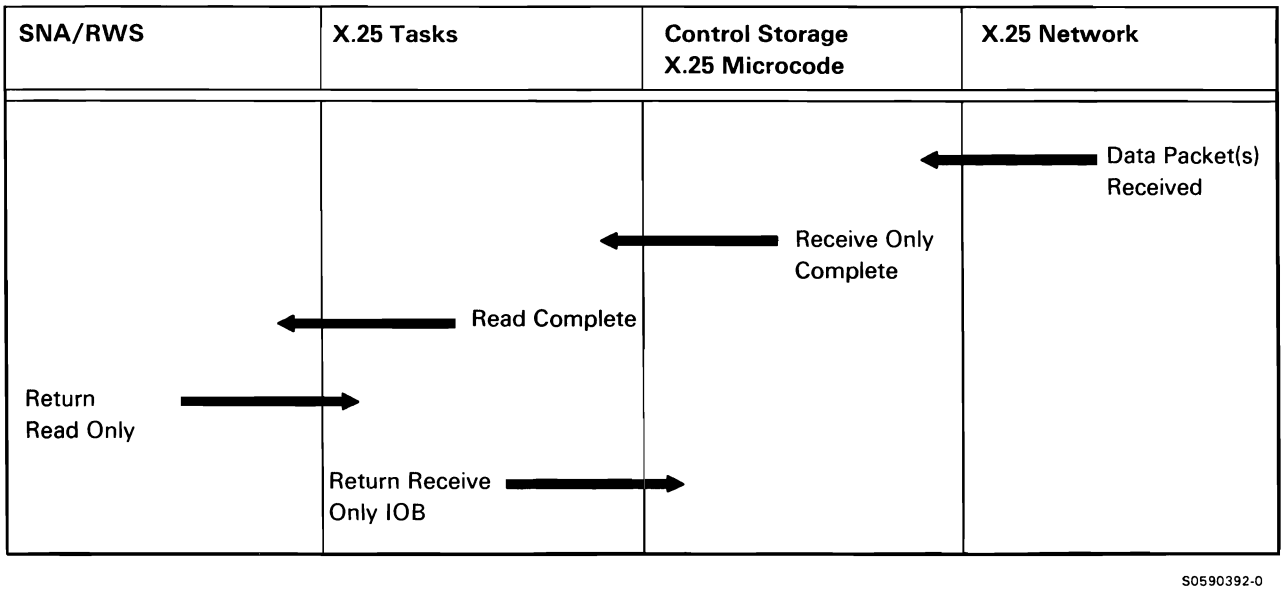


Figure A-13 (Part 7 of 7). X.25 Virtual Circuit Data Transfer

Appendix B: Programming Language Directory

This appendix provides brief descriptions of compiler modules and routines, and execution-time subroutines of the following programming languages:

- Assembler
- BASIC
- COBOL
- FORTRAN
- RPG II

Use this directory when diagnosing problems relating to IBM language compilers and object code generated by them. The compiler information is provided for use in analyzing compile-time diagnostics and program dumps.

The execution-time information consists of descriptions of subroutines built into user object programs. The subroutines provide specialized functions such as data conversion. For example, RPG II uses a multiply subroutine to execute the MULT calculation specification. The subroutines are selected by the preassemble and assemble phases of the compiler and are included in the object program overlay segments by the overlay phases.

This directory is organized by language. Each language, with the exception of BASIC, contains a directory for compiler modules and subroutines and a directory for execution time subroutines. (Because the BASIC language is interactive, the module descriptions are not organized by compile-time/execution-time types.) Each directory entry contains the module/routine name and a brief description of the module/routine's function.

ASSEMBLER LANGUAGE COMPILER MODULES

Module Name	Descriptive Name	Function
#ASSEM (#ASROT, #ASIN0)	Assembler initialization	Phase 05: Perform assembler initialization. Process OPTIONS, HEADERS, and input format (ICTL) control. #ASROT remains in storage and contains the work file DTFs and the common area.
#ASCM0	Compression	Phase 10: Process all the source statements for the control statements. Build the intermediate text file and decode and test the source statements. Convert source names to 6-byte text record names.
#ASSB0	Symbol table build	Phase 20: Assign the location counter values and build the symbol table.
#ASSF0	Symbol table overflow	Phase 21: Test the intermediate text from the statement following the overflow to the end of the text for symbols that have been previously defined. Resolve all references to symbols that are defined in the present symbol table.
#ASSS0	Symbol substitution	Phase 22: Assign the symbol value and attribute from the last (or only) symbol table to that symbol in the intermediate text term records. Build an ESL table for module name and EXTRN and ENTRY statements. Convert 6-byte ESL text record names to original source names to be stored in the ESL table.
#ASPE0 (#ASPRC, #ASPE1)	ESL output	Phase 29: Print the OPTIONS and HEADERS records and the prolog. Convert 6-byte title text names to original source name. Sort and print the ESL listing if LIST is specified. #ASPRC remains in storage and contains the CAM macro.
#ASPS0	Source/object output	Phase 30: Generate object code for machine instructions and DCs. Put out object code. Build printer image. Print source statements and object code generated. Perform assembly-time operations requested by assembler statements. Process machine instructions and assembler instructions.
#ASPD0	Print diagnostics	Phase 35: Read the error records and print diagnostic messages. Print the error summary statements.
#ASBX0	Build XREF file	Phase 40: Read the work file and create the cross-reference sort file. Initialize working storage, print area, and listing header for Phase 41.
#ASSX0	Merge and list XREF	Phase 41: Merge the XREF sort file. Generate the XREF listing. Print the error summary statements. Fetch overlay linkage editor if object output is required. Convert 6-byte text record names to original source names for XREF listing.
#MPXD	Macro processor driver	Phase 00: Read source statements from $SOURCE. Write assembler statements to $ASMINPT.
#MPX1	Overlay router	Phase 01: Transfer control to the proper overlay. Check for headers, prototypes, global, local, table definitions and call required phase.
#MPX2	Statement processor	Phase 02: Process MACRO, GBLA, LCLA, GBLB, LCLB, GBLC, LCLC, Prototype and macroinstruction.
#MPX3	Statement processor	Phase 03: Process AIF, AGO, SETB, MNOTE, MEXIT, MEND.
#MPX4	Statement processor	Phase 04: Process SETA, SETC.

BASIC Directory

Module Name	Descriptive Name	Function
#BLCHR	Chain processor	Set up execution of a new program by processing chain statement.
#BLCLR	Close	Process the close statement.
#BLCMR	Communications I/O	Performs communications file input/output operations.
#BLCPE	Command parser	Parse BASIC commands and invoke appropriate command processor.
#BLCSE	COLSEQ processor	Process the COLSEQ command.
#BLDAR	Data statement handler	Process the data statement.
#BLDEE	DEPARSER	Deparse (convert) intermediate text back to source statements.
#BLDIR	Library member I/O	Performs library member input/output operations.
#BLDSE	Edit-time display station manager	Interface with SSP WSDM during BASIC edit.
#BLEDE	Editor	Perform line change operations.
#BLEER	Run-time error exit handler	Process run-time errors.
#BLERE	BASICP error handler	Handles BASICP run-mode errors.
#BLEXE	Expression parser	Convert expressions (contained in BITBF) to IT for the statement parser (#BLPAE).
#BLFOE	Form statement parser	Convert form statements to IT for the statement parser (#BLPAE).
#BLFPR	Form processor	Process the run-time FORM statement.
#BLFRE	Free	Process the FREE command.
#BLFSR	Full screen processor	Process INPUT and PRINT FIELDS statements.
#BLGSR	Unformatted READ/GET I/O handler	Process unformatted READ or stream file GET requests.
#BLGTE	Get token	Get tokens for the statement and command parsers.
#BLHPE	Help mode processor	Perform help functions.
#BLICR	Interactive compiler	Convert intermediate text to BASIC instruction set code that can be executed by the control storage BASIC instruction set emulator.
#BLIFR	Find intrinsic function	Locate and create a table entry for a new intrinsic function.
#BLIMR	Image statement handler	Process print image statement.
#BLINC	Normal run-mode initialization and nucleus functions	Initialize BASIC for normal run mode and provide common nucleus functions.
#BLIOR	Run-time common I/O handler	Perform general I/O dispatching functions.
#BLIPR	INPUT and LINPUT	Set up I/O for BASIC INPUT and LINPUT statements.
#BLLDE	LOAD command processor	Process the BASIC LOAD and CLEAR commands.

Module Name	Descriptive Name	Function
#BLLIE	LIST command processor	Process the LIST command.
#BLLPE	LISTP command processor	Process the LISTP command.
#BLLSE	Source loader	Load lines of source into the parse buffer.
#BLMAR	Matrix assign	Perform matrix assign operation.
#BLMGE	MERGE command processor	Process the MERGE command.
#BLMIC	MRT run-mode initialization and nucleus functions	Initialize BASIC for MRT run mode and provide common nucleus functions.
#BLOPR	Open disk files	Process the OPEN statement for disk files, library members, and the local data area.
#BLPAE	Statement parser	Parse BASIC source statement, build the intermediate text, and invoke the editor.
#BLPFR	Print using form	Process control specifications from the using statement.
#BLPIC	BASICP run-mode intialization and nucleus functions	Initialize BASIC for PROC run mode and provide common nucleus functions.
#BLPOR	Open printer files	Process the printer OPEN statement.
#BLPSR	Unformatted put I/O handler	Process unformatted write and stream file PUT I/O operations.
#BLRER	Run-time error handler	Handle run-time warning error or trace conditions.
#BLRFR	Read using form	Process read using statement.
#BLRNE	Renumber	Renumber source statements.
#BLRUE	BASIC run/go command processor	Switch the BASIC program product from edit mode to run mode.
#BLSCR	Special run-mode chain processor	Load and initialize the user library SUBR member, and interface with the compiler.
#BLSEE	BASICS error handler	Handle BASICS edit-mode errors and save the subroutine.
#BLSIC	Special run-mode initialization	Initialize BASIC for special run mode and provide common nucleus functions.
#BLSLS	Source load	Loads source for BASICS.
#BLSNC	BASICS edit-mode initialization and nucleus functions	Initialize BASIC for source edit mode and provide common nucleus functions.
#BLSSE	Save source	Write source to a library member.
#BLSTR	ASCII collate table	The index is the equivalent EBCDIC character.
#BLSVE	SAVE command processor	Process SAVE and REPLACE commands.
#BLSXR	Special run-mode termination	Interface with SSP SYSLOG for special run-mode termination.
#BLUPR	Unformatted print	Process unformatted PRINT statement.
#BLWAR	WAITIO handler	Handle the WAITIO statement for work stations and ICF.

Module Name	Descriptive Name	Function
#BLWOR	Open work station files	Process the OPEN statement for work station and communications sessions.
#BLXRE	XREF handler	Provide a cross reference of a BASIC program.
#BLXTR	Normal run-mode termination	Save symbol table on disk and terminate normal run mode.
#BL00(L,S)	LOG(X), EXP(X) Exponentiation function	Return (LOG(X), EXP(X), or ***X.
#BL01(L,S)	RND intrinsic function	Generate a random number, using RANDOMIZE.
#BL02	CHR$(X) intrinsic function	Return the character equivalent of decimal number X.
#BL02A	CHR$(X) intrinsic function for a given collating sequence	Return the character equivalent from the given collating sequence of decimal number X.
#BL03	DATE$ intrinsic function	Return the current date.
#BL04	FILE$(X) intrinsic function	Return the number of the file X.
#BL05	LPAD$(A$,X) intrinsic function	Left pad character string A$ with blanks to create a character string of length X.
#BL06	LTRM$(A$) intrinsic function	Remove the leading blanks from character string A$.
#BL07	LWRC$(A$) intrinsic function	Convert uppercase letters in string A$ to lowercase.
#BL08	RPAD$(A$,X) intrinsic function	Right pad character string A$ with blanks to create a character string of length X.
#BL09	RPT$(A$,X) intrinsic function	Repeat character string A$, X times.
#BL0A	RTRM$($A) intrinsic function	Remove the trailing blanks from character string A$.
#BL0B	SREP$(A$,X,B$,C$) intrinsic function	Return character string A$ with all occurrences of B$ replaced by C$, starting at position X.
#BL0C	STOP$ intrinsic function	Return Y if stop system has been requested, N if stop system has not been requested.
#BL0D	STR$(X) intrinsic function	Return a character representation of the number X.
#BL0E	TIME$ intrinsic function	Return the time of day in the form, HH:MM:SS.
#BL0F	UPRC$(A$) intrinsic function	Convert lowercase letters in character string A$ to uppercase.

Module Name	Descriptive Name	Function
#BL10	UPSI$(A$) intrinsic function	Return the UPSI switches as character strings of ones and zeros, as requested by character string A$.
#BL11	WSID$	Return the work station ID of the current work station or session.
#BL12	ABS(X) intrinsic function	Return the absolute value of X.
#BL13(L,S)	ATN(X) intrinsic function	Return the arctangent (in radians) of X.
#BL14	CEIL(X) intrinsic function	Return the smallest integer not less than X.
#BL15	CMDKEY intrinsic function	Return the numeric value of the last enter or command key used.
#BL16	CNT intrinsic function	Return the number of items successfully processed by the last input or output statement.
#BL17	CODE intrinsic function	Return the numeric representation of the UPSI switches.
#BL18(L,S)	COS(X) and SIN(X) intrinsic functions	Return the sine or cosine of X.
#BL19	DATE intrinsic function	Return the current date as a number in the form YYDDD.
#BL1A(L,S)	DEG(X) intrinsic function	Return the number of degrees in X radians.
#BL1B	ERR intrinsic function	Return the number of the error when an error occurs during program execution.
#BL1D	FILE(X) intrinsic function	Return the status (not opened, last operation successful, EOF during input, EOF during output) for file N.
#BL1E	FILENUM intrinsic function	Return the file reference number of the last file in which an error occurred.
#BL1F	FP(X) intrinsic function	Return the fractional part of X.
#BL20(L,S)	INF intrinsic function	Return the largest positive number representable in BASIC.
#BL21	INT(X) intrinsic function	Return the largest integer not greater than X.
#BL22	IP(X) intrinsic function	Return the integer part of X.
#BL23	KLN(X,Y) intrinsic function	Return the key length of Y-key part of file X.
#BL24	KPS(X,Y) intrinsic function	Return the key position of Y-key part of file X.
#BL25	LEN(A$) intrinsic function	Return the length of the string A$.
#BL26	LINE intrinsic function	Return the line number of the last statement with an error.
#BL28	MAX(X1,X2,...Xn) intrinsic function	Return the maximum value of X1, X2,...Xn.
#BL29	MIN(X1,X2,...Xn) intrinsic function	Return the minimum value of X1, X2,...Xn.

Module Name	Descriptive Name	Function
#BL2A	MOD(X,Y) intrinsic function	Return X modulo Y.
#BL2B	ORD(A$) intrinsic function	Return the ordinal position of A$ in the character set.
#BL2BA	ORD(A$) intrinsic function for a given collating sequence	Return the ordinal position of A$ in the character for the given collating sequence.
#BL2C(L,S)	PI intrinsic function	Return the value of pi.
#BL2D	POS(A$,B$,X) intrinsic function	Return the position of the first occurrence of B$ within A$ starting at position X.
#BL2E(L,S)	RAD(X) intrinsic function	Return the number of radian in X degrees.
#BL2F	REC(X) intrinsic function	Return the record number of the last record processed in file X.
#BL30	REM(X,Y) intrinsic function	Return the remainder of X/Y.
#BL31	RLN(X) intrinsic function	Return the record length of file X.
#BL32(L,S)	RND(X) intrinsic function	Return a pseudo-random number between 0 and 1, using X as the seed value.
#BL33(L,S)	SGN(X) intrinsic function	Return the sign X.
#BL35(L,S)	SQR(X) intrinsic function	Return the square root of X.
#BL36(L,S)	SRCH(A,X,N) intrinsic function	Return the position within the array A of X starting with element N.
#BL37(L,S)	SUM(A) intrinsic function	Sum all elements in the array A.
#BL38(L,S)	TAN(X) intrinsic function	Return the tangent value of X radians.
#BL39	TIME intrinsic function	Return the number of seconds since midnight.
#BL3A	UDIM(A,N) intrinsic function	Return the upper subscript of the Nth dimension of array A.
#BL3B	UPSI intrinsic function	Return the number represented by a binary character string of ones and zeros.
#BL3C	VAL(X$) intrinsic function	Return the number represented by X$.
#BL3D(L,S)	MAT A = AIDX(B)	Set elements of array A to indexes of ascending values of array B.
#BL3E	MAT A = CON	Set every element in array A equal to 1.
#BL3F(L,S)	MAT A = DIDX(B) intrinsic function	Set elements of array A to indexes of descending values of array B.
#BL40	MAT A = ZER	Set all elements of array A equal to 0.
#BL41(L,S)	ROUND(X,N) intrinsic function	Round X to the Nth digit.
#BL42	SRCH(A$,X$,N) intrinsic function	Return the position within array A$ or X$, starting with element N.
#BL43	UDIM(A$,N) intrinsic function	Return the upper subscript of the Nth dimension of array A$.
#BL44	MAT A =(X) or (X)*CON	Set every element of array A equal to X.
#BL45	MAT A$ = (X$)	Set every element of array A$ equal to X$.
#BL46	MAT A=B	Set each element of array A equal to the corresponding element B.

Module Name	Descriptive Name	Function
#BL47	MAT A$=B$	Set each element of array A$ equal to the corresponding element of B$.
#BL48	MAT A=B+C MAT A=B-C	Set each element of array A equal to the sum or difference of the corresponding elements of arrays B and C.
#BL4A	MAT A=(X)*B	Set each element of array A equal to X times the corresponding element of array B.
#BL4B	MAT A = AIDX(B$)	Set elements of array A to indexes of ascending values of array B$.
#BL4C	MAT A = DIDX(B$)	Set elements of array A to indexes of descending values of array B$.
#BL4D	CNVRT$(A$,X) intrinsic function	Convert X to a character with the form shown in the form specification (A$).
#BL4E	HEX$(A$) intrinsic function	Convert A$ to its hexadecimal equivalent.
#BL4F	PIC$(A$) intrinsic function	Return the value of the floating currency symbol or change the value of the floating currency symbol to A$.
#BL50	MSG$(X,A$) intrinsic function	Return the message with MIC value X from message member A$.
#BL51	PROCIN intrinsic function	Return 0 if input is from keyboard; return 1 if input is from a BASIC procedure.
#BL52	PROCLVL intrinsic function	Return the number of BASIC procedures currently active.
#BL53	RETCODE$ intrinsic function	Return the value of the SSP-ICF return code for the last read or write to a work station or session.
#BL54	TIMER(A$) intrinsic function	Set the system timer for a period of time given by the value of A$, in the form hhmmss; return 0 if successful, 1 if unsuccessful.
#BL55	CURROW	Return the row position of the cursor when the last function key was pressed.
#BL56	CURCOL	Return the column position of the cursor when the last function key was pressed.
#BL58	USERID$	Return a character string containing the user ID associated with the current program.

COBOL Directory

COBOL COMPILER MODULES

Compiler Module	Descriptive Name	Function
#CB00	ROOT phase	Main controlling phase for the COBOL compiler.
#CB01	TEXT phase	Process source statements within the Identification, Environment, Data, and Procedure Divisions; process IGC literal.
#CB02	DTA1 phase	Begin the processing of the Data Division, allocating space and displacements for level items and producing B-text for all statement numbers, errors, and edit pictures within this division.
#CB03	DTA2 phase	Examine the syntax of the Identification and Environment Divisions.
#CB04	DLIT phase	Generate literals and edit masks for the Data Division.
#CB05	PRO1 phase	Begin processing of Procedure Division of B-text and perform special processing (reordering) of selected Procedure Division verbs.
#CB06	PRO2 phase	Further reduce Procedure Division text, make entries in the Item Table for procedure names, and compute DMAX values for arithmetic statements.
#CB07	PRO3 phase	Further reduce the Procedure Division B-text. Convert arithmetic and conditional expressions to reversed Polish strings and relational strings, respectively.
#CB08	GEN1 phase	Process various Procedure Division statements and produce inline code or subroutine calls for address computation of subscripted, indexed, or linkage Section data names in all Procedure Division statements.
#CB09	GEN2 phase	Process various Procedure Division statements.
#CB10	GEN3 phase	Process all arithmetic statements and relational statements.
#CB11	GEN4 phase	Complete the transformation of the Procedure Division to C-text. Optimize all code generated by the GEN1, GEN2, and GEN3 phases and eliminate all extraneous elements.
#CB12	PLIT phase	Process Procedure Division literals and external references.
#CB13	OBJ phase	Produce the object module from the external reference list, the Data and Procedure Division C-text, and the Literal text.
#CB14	MAP phase	Optionally provide Data Division and Procedure Division maps.
#CB15	ERR phase	Produce a listing of diagnostic messages and, optionally, a diagnosed source file.
#CB16	END phase	Terminate compilation and pass control to the system or to the overlay linkage editor.
#CB50	COPY phase	Process compiler control and COPY statements, and initiate diagnosed source file processing, if requested; process IGC literal in copy statement and copied text.
#CB51	XREF phase	Generate cross-reference listings for data names and procedure names.
#CB54	DTA3 phase	Examine the syntax in the Data Division and complete the Item Table entries for files.
#CB55	PROA phase	Scan all A-text for the Procedure Division, generating any error messages that are necessary in B-text, and reduce Procedure Division A-text to B-text.
#CB58	GENA phase	Process various Procedure Division statements, primarily program control verbs and low-level I/O verbs.
#CB99	DBUG phase	Process CDEBUG process option for compilation.

COBOL EXECUTION-TIME SUBROUTINES

Subroutine Name	Descriptive Name	Function
@CB000	—	Store arithmetic result in case of GIVING option, or IF or COMPUTE statement.
@CB010	—	Test for sign condition and ON SIZE ERROR during addition or subtraction, and compare numeric fields.
@CB020	—	Multiplication.
@CB030	—	Division.
@CB040	—	Exponentiation.
@CB050	—	Process ndd.
@CB060	—	Called by @CB050 to convert zoned decimal arithmetic result to binary or binary data item to zoned decimal.
@CB070	—	Called by @CB050 to convert zoned decimal arithmetic result to packed decimal, or packed decimal item to zoned decimal.
@CB080	—	Called by CB050 to edit arithmetic result being stored in numeric edited field.
@CB100	—	Move numeric and alphanumeric fields.
@CB110	—	Move numeric and alphanumeric fields to edited fields.
@CB120	—	Move Boolean data items.
@CB150	—	Resolve addresses for indexed items.
@CB200	—	Compare alphabetic and alphanumeric operands.
@CB210	—	Compare index names.
@CB220	—	Test for numeric field.
@CB230	—	Test identifier for alphabetic characters (A to Z and blank).
@CB240	—	Test system switches represented by UPSI-n special names.
@CB250	—	Maintain a truth-value stack.
@CB260	—	Compare alphabetic and alphanumeric fields and operands.
@CB270	—	Compare Boolean data.
@CB290	—	EBCDIC to ASCII translate table; used when ASCII collating sequence is specified.
@CB300	—	Read consecutively from disk file, and write records consecutively to disk file.
@CB310	—	Write, read, update, and locate records to direct file on disk.
@CB320	—	Locate record, read sequentially, and write sequentially from indexed file on disk. Also update and delete records sequentially on indexed file.
@CB330	—	Randomly read, write, update, and delete an indexed file.
@CB340	—	Set file status in COBOL program.
@CB350	—	Determine if there is a USE procedure to be called.
@CB360	—	Perform transaction file I/O operations.
@CB400	—	Implement OPEN statement.
@CB410	—	Implement CLOSE statement.

Subroutine Name	Descriptive Name	Function
@CB420	—	Perform the STOP 'literal' function, and perform the DISPLAY on the SYSLIST, REQUESTOR, and CONSOLE functions.
@CB430	—	ACCEPT input from the input stream, the requester's work station, and the operator's console until the identifier is filled.
@CB440	—	Implement WRITE statement for printer files.
@CB480	—	Perform ACCEPT/DISPLAY LOCAL DATA AREA.
@CB500	—	Perform the GO TO . . . DEPENDING ON 'identifier' in segmented and unsegmented programs.
@CB510	—	Implement TRACE statement and save address of program's READY/RESET switch tested by @CB510.
@CB520	—	Implement STOP RUN statement.
@CB530	—	Perform the ACCEPT FROM DATE, ACCEPT FROM DAY, and ACCEPT FROM TIME functions.
@CB540	—	Control all entry to independent segments, and perform branching between independent segments.
@CB550	—	Handle checkpoint processing for files described with RERUN clause.
@CB570	—	Perform the SET index 1 UP BY, DOWN BY, and TO 'identifier' functions, and the SET index 1 TO index 2 and SET 'identifier' TO index 2 functions.
@CB580	—	Calculate address of subscripted data items, and current length of a group item that contains a variable-length table.
@CB590	—	Implement EXHIBIT statement.
@CB600	—	Issue object-time message and allow user to select step termination or job termination, and move record when request made for input operation on locate-mode file.
@CB620	—	Contain standard initialization code for all COBOL-generated programs.
@CB650	—	Initialize execution of STRING operation and pass description of output field and optional pointer field. Also pass delimiter description, and concatenate a string input field to the output field.
@CB660	—	Initialize execution of UNSTRING operation, pass description and addresses of parameters.
@CB680	—	Initialize, select tallying or replacing, and pass target identifier description.
@CB700	—	Build specifications for sort and merge operations.
@CB900	—	Set up the DEBUG ITEM special register and invoke the USE procedure.
CBSTOP	—	Test system status to see if system shutdown is in process.
CBEMCR		Interface with SUBR25 for enhanced support of the 1255 MCR.
CBFTOD		Provide time of day in timer units of 8.192 milliseconds.
CBINST		Insert shift-out and shift-in characters in the first and last position for IGC processing.
CBMICR		Interface with SUBR08 for support of the 1255 MCR.
CBREMV		Remove shift-out and shift-in characters from the first and last position for IGC processing.

FORTRAN Directory

FORTRAN COMPILER MODULES

Compile-Time Module	Descriptive Name	Function
#FORT	Root phase	Perform initialization, load processing phases sequentially, provide system interface, and build the FORTRAN communications area.
#FO00D	Debug module	Control the operation of the console debugging capability.
#FO01	Phase 1	Read control records and source statements, process options, check for statement continuations, list source program, process FORMAT statements, and process logical IF statements.
#FO02	Phase 2	Determine statement type, and insert statement terminators.
#FO03	Phase 3	Check validity of logical IF, check FORMAT statement, and process AT and TRACE statements.
#FO04	Phase 4	Set FORMAT usage, pack statements, process PROGRAM statements, and check first character of unit number for BACKSPACE, END FILE, REWIND.
#FO05	Phase 5	Process subprogram, IMPLICIT, and specification statements; check statement numbers; remove unnumbered CONTINUE statements; and process statement numbers.
#FO06	Phase 6	Process COMMON and GLOBAL variables, process SUBROUTINE and FUNCTION dummy arguments, check for type FUNCTION statements, process IMPLICIT statements, and process GENERIC.
#FO07	Phase 7	Process DIMENSION statements; process REAL, INTEGER, LOGICAL, and EXTERNAL variables; and identify variable modes.
#FO08	Phase 8	Process real constants, and convert real constants to single or double precision format.
#FO09	Phase 9	Process DEFINE FILE, BACKSPACE, END FILE, and REWIND statements.
#FO10	Phase 10	Process integer constants and variables, ASF definitions, and subscripts.
#FO11	Phase 11	Check syntax of DEBUG statement, create DED entries for INIT and SUBCHK variables, and convert FORTRAN subroutine names to internal routine names.
#FO12	Phase 12	Process DATA statements.
#FO13	Phase 13	Process FORMAT statements.
#FO14	Phase 14	Process subscript constants.
#FO15	Phase 15	Process IF, CALL, and statement function statements, and check record numbers in READ/WRITE statements.
#FO16	Phase 16	Process processor-supplied and GENERIC functions.
#FO17	Phase 17	Process READ, WRITE, FIND, GOTO, and EQUIVALENCE statements.
#FO18	Phase 18	Process CONTINUE, STOP, PAUSE, and END statements, and implied DO-loops in READ/WRITE statements.
#FO19	Phase 19	Process DO statements and subscripts, identify required library modules for DO statements, and generate scientific instructions for DO statements.

Compile-Time Module	Descriptive Name	Function
#FO20	Phase 20	Convert FIND, READ, WRITE, IF, GOTO, CALL statement function, and assignment statements into modified Polish notation; and process arithmetic and logical operations, and arguments for subroutine calls.
#FO21	Phase 21	Process generated temporary markers for subscripts, and position subscript information in the statement string.
#FO22	Phase 22	Generate scientific instructions for subscripts and for RETURN, GOTO, and computed GOTO.
#FO23	Phase 23	Generate scientific instructions for I/O statements.
#FO24	Phase 24	Generate scientific instructions for expressions in ASF, IF, CALL, assignment, and direct access I/O statements.
#FO25	Phase 25	Process CALL, IF, ASF, assignment, I/O statements, and I/O lists; identify required library modules; generate scientific instructions for IF statements and IF blocks; and generate scientific instructions for CALL statements and CALL blocks.
#FO26	Phase 26	Process DEBUG options in TRACE, DO, and labeled statements.
#FO27	Phase 27	Assign storage addresses to variables, calculate storage required for constants and blocks, and print variable map.
#FO28	Phase 28	Build unit table.
#FO29	Phase 29	Check symbol table for errors, and print all error messages.
#FO31	Phase 31	Create save areas, prologues, and epilogs, and allocate storage for text string.
#FO32	Phase 32	Process statement allocations, and print label MAP.
#FO33	Phase 33	Allocate and open $WORK, create OPTNS record, and create ESLs.
#FO34	Phase 34	Create load IAR instruction, and create text records for DATA statements and constants.
#FO35	Phase 35	Create text for all blocks.
#FO36	Phase 36	Create text records for DTFs.
#FO37	Phase 37	Create text records for all FORMAT statements and literal constants in FORMAT statements.
#FO38	Phase 38	Create text records for all executable code, DO blocks, unit table, END and /* records, close $WORK file, and invoke the overlay linkage editor.
#FO59	Phase 59	Compute I/O storage requirements, and identify required data management modules.
#FO78	Phase 78	Rearrange statement string.

FORTRAN EXECUTION-TIME SUBROUTINES

Execution-Time Subroutine	Descriptive Name	Function
@FOA1D	Commercial subroutine	Convert A1 to decimal format.
@FOB1	I/O control	Contain DED operators, list references, and debug references.
@FOB2	I/O control	Control formatted I/O.
@FOB3	I/O control	Control unformatted I/O.
@FOB4	I/O control	Control list-directed I/O.
@FOB8	I/O control	Fill I/O buffer with blanks for formatted and list-directed I/O.
@FOB9	I/O control	Contain buffer control blocks for formatted, unformatted, and list-directed I/O.
@FOBA	I/O control	For list-directed I/O, check that data will fit in available buffer space.
@FOBB	I/O control	Handle any program terminal error.
@FOBUG	Service subprogram	A dynamic execution-time debug subprogram.
@FOC0	Conversion	Handle all conversions for the A-format code data on input.
@FOC1	Conversion	Handle all conversions for the A-format code data on output.
@FOC2	Conversion	Handle all conversions for the I-format code data on input.
@FOC3	Conversion	Handle all conversions for the I-format code data on output.
@FOC4	Conversion	Handle all conversions for the E-, D-, and F-format code data on input.
@FOC5	Conversion	Handle all conversions for the E-, D-, and F-format code data on output.
@FOC6	Conversion	Handle all conversions for the L-format data on output.
@FOC7	Conversion	Handle all conversions for the Z-format code data on output.
@FOC9	Conversion	Determine the length of a variable.
@FOCA	Conversion	Convert a binary number to zoned decimal.
@FOCE	Conversion	Handle all conversions for the L-format data on input.
@FOCF	Conversion	Handle all conversions for zoned decimal data to REAL*4 or REAL*8.
@FOCG	Conversion	Power of 5 table, multiplication, and division routines for conversion.
@FOCH	Conversion	Handle all conversions of REAL*4 or REAL*8 to zoned decimal.
@FOCMD	Service subprogram	Test whether a command key has been pressed.
@FODCK	Service subprogram	Divide check test.
@FODEC	Commercial subroutine	Convert decimal to A1 format.
@FODIV	Commercial subroutine	Divide zoned decimal fields.
@FODPK	Commercial subroutine	Convert zoned decimal to packed decimal fields.
@FODSW	Service subprogram	Test address/data switches on CE panel, and test data switch indicators.

Execution-Time Subroutine	Descriptive Name	Function
@FODSY	Service subprogram	Handle full screen information.
@FOD0	Debug	Control DEBUG facilities input and output.
@FOD2	Debug	Handle TRACE output.
@FOD3	Debug	Handle SUBCHK output.
@FOD5	Debug	Handle INIT output.
@FOD6	Debug	Handle SUBTRACE output.
@FOD7	Debug	Handle traceback errors.
@FOD8	Debug	Write DEBUG messages.
@FODWN	Service subprogram	Test system shutdown indicator.
@FOE0	Scientific instruction set interface	Interpret interface.
@FOEDT	Commercial subroutine	Edit a field with a mask field.
@FOFIL	Commercial subroutine	Fill an area with a specified value.
@FOFST	Service subprogram	Indicate errors in library routine or I/O conversions.
@FOGET	Commercial subroutine	Convert a numeric field in A1 format to a REAL*8 value.
@FOIBT	Commercial subroutine	Test for bit on or bit off.
@FOICO	Commercial subroutine	Compare two zoned decimal fields.
@FOINQ	Service subprogram	Invoke ROLLOUT/ROLLIN.
@FO10	I/O interface	Execute the scientific instruction I/O operator.
@FO12	I/O interface	Card I/O interface.
@FOI24	Commercial subroutine	Determine whether an integer is INTEGER*2 or INTEGER*4.
@FO13	I/O interface	Printer I/O interface.
@FO16	I/O interface	Direct access disk interface.
@FO17	I/O interface	Sequential disk interface.
@FO18	I/O interface	Sequential disk BACKSPACE interface.
@FO19	I/O interface	Sequential disk END FILE interface.
@FOIA	I/O interface	Sequential disk REWIND interface.
@FOIB	I/O interface	Close DTF open in UNITAB.
@FOIC	I/O interface	Print DEBUG and errors.
@FOID	I/O interface	Display screen interface.
@FOM1	Math library	Handle SIN/COS.
@FOM2	Math library	Handle DSIN/DCOS.
@FOM5	Math library	Handle LOG_2.

Execution-Time Subroutine	Descriptive Name	Function
@FOM6	Math library	Handle $DLOG_2$.
@FOM7	Math library	Handle INTEGER**INTEGER exponentiation.
@FOM8	Math library	Handle REAL**INTEGER exponentiation.
@FOM9	Math library	Handle REAL*8**INTEGER exponentiation.
@FOMA	Math library	Handle REAL**REAL exponentiation.
@FOMB	Math library	Handle REAL*8**REAL*8 exponentiation.
@FOMC	Math library	Handle 2**REAL exponentiation.
@FOMD	Math library	Handle 2**REAL*8 exponentiation.
@FOME	Math library	Handle ARCTAN.
@FOMF	Math library	Handle DTAN.
@FOMG	Math library	Handle SQRT.
@FOMH	Math library	Handle DSQRT.
@FOMI	Math library	Handle ALOG.
@FOMJ	Math library	Handle DLOG.
@FOMK	Math library	Handle ALOG10.
@FOML	Math library	Handle DLOG10.
@FOMM	Math library	Handle EXP.
@FOMN	Math library	Handle DEXP.
@FOMO	Math library	Handle TANH.
@FOMOV	Commercial subroutine	Move data from one area to another.
@FOMP	Math library	Handle DTANH.
@FOMPY	Commercial subroutine	Multiply two zoned decimal fields.
@FOMQ	Math library	Handle DMOD.
@FOMSG	Service subprogram	Retrieve user message.
@FOOVF	Service subprogram	Indicate exponent overflow or underflow.
@FOPCK	Commercial subroutine	Convert A1 to A2 format.
@FOPUN	Commercial subroutine	Punch a card.
@FOPUT	Commercial subroutine	Convert a REAL*8 value to A1 format.
@FORED	Commercial subroutine	Read a card.
@FOSIG	Commercial subroutine	Test or modify a sign.
@FOSKY	Service subprogram	Enable command keys.
@FOSLE	Service subprogram	Set four internal switches.
@FOSLT	Service subprogram	Test the internal switches.

Execution-Time Subroutine	Descriptive Name	Function
@FOSV0	Service subprogram	Handle FORTRAN INVOKE statement.
@FOSV1	Service subprogram	Handle DUMP and PDUMP.
@FOSYN	I/O interface	Handle input from the SYSIN device.
@FOS0	Commercial subroutine	Handle print for commercial subroutines.
@FOS10	Commercial subroutine	Check unit table.
@FOS20	Commercial subroutine	Handle SKIP and SPACE.
@FOS30	Commercial subroutine	Shift the element of an array.
@FOS35	Commercial subroutine	Set indicated bit to 0 or 1.
@FOS40	Commercial subroutine	Handle KEYBD and TYPER.
@FOS60	Commercial subroutine	Process parameters.
@FOS70	Commercial subroutine	Handle zoned decimal ADD and SUBTRACT.
@FOS80	Commercial subroutine	Process commercial subroutine arguments.
@FOS90	Commercial subroutine	Handle commercial subroutine ADD and SUBTRACT.
@FOSA0	Commercial subroutine	Handle NCOMP/LCOMP.
@FOTMD	Service subprogram	Return date and time of day.
@FOUNP	Commercial subroutine	Convert packed decimal to zoned decimal.
@FOUPC	Commercial subroutine	Convert A2 to A2 format.
@FOV0	Invoked execution	Handle the absolute value function IABS.
@FOV1	Invoked execution	Handle the absolute value function ABS.
@FOV2	Invoked execution	Handle the maximum value function MAX0.
@FOV3	Invoked execution	Handle the minimum value function MIN0.
@FOV4	Invoked execution	Handle the maximum value function MAX1.
@FOV5	Invoked execution	Handle the minimum value function AMIN1.
@FOV6	Invoked execution	Handle the maximum value function DMAX1.
@FOV7	Invoked execution	Handle the minimum value function DMIN1.
@FOV8	Invoked execution	Handle the truncation function AINTM and AINT.
@FOVD	Invoked execution	Handle the sign changes for floating-point numbers.
@FOVE	Invoked execution	Handle the change of sign function SIGN.
@FOVF	Invoked execution	Handle the change of sign function ISIGN.
@FOVI	Invoked execution	Handle the sign changes for INTEGER numbers.
@FOVK	Invoked execution	Handle the positive difference function DIM.
@FOVL	Invoked execution	Handle the positive difference function IDIM.
@FOVM	Invoked execution	Handle the STOP n statement.

Execution-Time Subroutine	Descriptive Name	Function
@FOVN	Invoked execution	Handle the PAUSE n statement.
@FOVP	Invoked execution	Handle the STOP EOJ statement.
@FOVQ	Invoked execution	Handle the precision decrease function SNGL.
@FOVR	Invoked execution	Handle the precision increase function DBLE.
@FOVS	Invoked execution	Handle the modulo arithmetic function AMOD.
@FOVT	Invoked execution	Handle the modulo arithmetic function AMOD.
@FOVU	Invoked execution	Handle the maximum value function AMAX0.
@FOVV	Invoked execution	Handle the maximum value function MAX1.
@FOVW	Invoked execution	Handle the minimum value function AMIN0.
@FOVX	Invoked execution	Handle the minimum value function MIN1.
@FOWHL	Commercial subroutine	Convert a REAL*8 value to a whole number.
@FOZNE	Commercial subroutine	Test and modify zones.

RPG Directory

RPG COMPILER MODULES

Module Name	Descriptive Name	Function
#AUTO	Copy function	Initialize the common area and read and list the source.
#AU00A	Catalog source program	Create a library member of the generated source program.
#AU00B	Diagnostic root phase	Retrieve and print the error messages and associated headings.
#AU00C	Output Specification Generator	Generate output specifications for heading *AUTO lines.
#AU00D	Output Specification Generator	Generate output specifications for detail/total *AUTO lines.
#AU00E	Output Specification Generator	Generate output specifications for detail/total *AUTO lines.
#AU002	Source sort	Sort source into compiler order.
#AU003	File table build	Build file table for printer files.
#AU004	Heading diagnostics	Diagnose heading *AUTO lines.
#AU005	Detail/total diagnostics	Diagnose detail/total *AUTO lines.
#AU006	Field-end position	Build field-end positions for detail/total *AUTO lines.
#AU007	Calculation Specifications Generator	Generate calculation specifications for detail/total *AUTO lines.
#AU008	Root phase processor	Perform root phase functions for #AU00C, #AU00D, and #AU00E.
#AU009	RPG compiler input	Build input for the RPG II compiler.
#RPEA	RPG file description compression	Process file description specifications.
#RPEB	RPG extension compression	Process extension specifications.
#RPEC	RPG line counter compression	Process line counter specifications.
#RPEE	RPG telecommunications compression	Process telecommunications specifications.
#RPEI	RPG input compression	Process input specifications.
#RPEK	RPG calculation compression	Process calculation specifications.
#RPEO	RPG output compression	Process output specifications.
#RPEW	RPG postcompression and initialization	Perform compression cleanup.
#RPFA	RPG copy tables into compressions	Process tables, arrays, entry points, and module names.

Module Name	Descriptive Name	Function
#RPG	RPG compiler initialization	Process control specifications.
#RPGF	RPG assign and check indicators	Assign and check indicators.
#RPGG	RPG check calculation specifications 2	Check calculation specifications, and compress and set specifications.
#RPGH	RPG assign file name and IOCB	Process file input/output table and IOCB addresses.
#RPGI	RPG assign file name and DTF	Process file input/output table and DTF addresses.
#RPGK	RPG check telecommunications specifications 1	Check errors in file name, ITBs, station IDs, and ASCII, and build and update tables.
#RPGL	RPG check telecommunications specifications 2	Check for errors in fields.
#RPGN	RPG build DTFs	Build IOCBs and preopen DTFs.
#RPGP	RPG build extension, WSID table, and TDATA for WORKSTN	Generate WSID table, TDATA hold areas, space for FORMAT directory, and TIND save and restore logic.
#RPGT	RPG assign I/O areas (nondisk devices: CONSOLE, CRT, PRINTER, BSCA, KEYBORD, SPECIAL)	Assign shared I/O areas, IOBs, and I/O buffer; generate object code and place addresses into IOCB and DTF.
#RPGU	RPG build name table	Build name table of compile-time and object-time tables/arrays and data structures.
#RPGV	RPG check table/array	Check elements in table/array and write and delete data structure entries from name table.
#RPGW	RPG check name table	Check and set bits on in table.
#RPGX	RPG print name table	—
#RPGY	RPG build compile-time tables/arrays	Build compile-time symbol table of compile-time table/array entries and DTT records.
#RPGZ	RPG compile-time table/array code	Generate object code blocks.

Module Name	Descriptive Name	Function
#RPHA	RPG build symbol table	Build symbol table of field names.
#RPHB	RPG build symbol table (DEFN processing)	Add DEFN field names to symbol table.
#RPHC	RPG check symbol table	Set bits on and assign 2-byte addresses.
#RPHD	RPG print symbol table	Scan table and compressions for unreferenced and undefined names and write names.
#RPHQ	RPG assign alternate collating sequence and file translation tables	Generate object code and place addresses in IOCB AND root segment.
#RPHS	RPG check input specifications 2	
#RPHT	RPG assign field hold areas and edit patterns	Assign holds areas for control and match fields, and assign and check edit code patterns.
#RPHU	Assign limits file hold areas	
#RPHW	RPG initialize allowable formats for CONSOLE device	
#RPIC	RPG postcompression initialization	
#RPIG	RPG check file description continuation records	
#RPJA	RPG check input specifications 1	
#RPJE	RPG check calculation specifications 1	
#RPJG	RPG check calculation specifications 3	
#RPJJ	RPG control field and match field move optimization	
#RPJK	RPG check file and table/array	
#RPJL	RPG check calculation specifications 4	
#RPJM	RPG check output specifications 1	
#RPJN	RPG check output specifications 2	
#RPJO	RPG check output specifications 3	
#RPJP	RPG check input specifications 3	
#RPJU	RPG check compile-time tables/arrays	
#RPJW	RPG check file description specifications 1	
#RPJX	RPG check file description specifications 2	
#RPJY	RPG check extension and line counter specifications	
#RPJZ	RPG check file description specifications 3	
#RPKA	RPG error sort, print, and error message list	

Module Name	Descriptive Name	Function
#RPKB	RPG diagnosed source file build	
#RPLB	RPG output indicator optimization	
#RPLG	DTF parameter assign 1	
#RPLJ	RPG preassemble calculations 3	
#RPLN	RPG constant, literal, edit word assign I	
#RPLR	RPG constant, literal, edit word, DTF parameter assign II	
#RPLV	RPG output fields move optimization	
#RPLZ	RPG output indicator testing	
#RPMA	RPG output object code block length	
#RPMB	RPG preassemble calculations 1	
#RPMH	RPG preassemble calculations 4	
#RPMI	RPG preassemble indicator optimization	
#RPMK	RPG build segment list	
#RPMM	RPG output segment list entries build 1	
#RPMN	RPG output segment list entries build 2	
#RPMP	RPG build calculation segment list	
#RPMQ	RPG complete calculation segment list	
#RPPA	RPG generate IPCR	
#RPPB	RPG generate OPCR	
#RPPC	RPG input mainline code	
#RPPE	RPG build input record recognition code	
#RPPG	RPG file selection and match field extraction code	
#RPPJ	RPG control field extraction code	
#RPPL	RPG lookahead field extraction code	
#RPPM	RPG input field extraction code	
#RPPN	RPG chain and read files code	
#RPPO	RPG LR and overflow control mainline code	
#RPPS	RPG output record code	
#RPPU	RPG output fields 1 code	
#RPPV	RPG output fields 2 code	
#RPPW	RPG output fields 3 code	
#RPQA	RPG operation independent calculations code	

Module Name	Descriptive Name	Function
#RPQB	RPG calculations code 3	
#RPQC	RPG calculations code 1	
#RPQD	RPG calculations code 4	
#RPQE	RPG calculations code 5	
#RPQF	RPG calculations code 6	
#RPQG	RPG calculations code 7	
#RPQH	RPG calculations code 2	
#RPQK	RPG calculations code 8	
#RPQL	RPG calculations code 9	
#RPQT	RPG calculations code 10	
#RPQV	RPG calculations code 11	
#RPRA	RPG initialization and close code	
#RPRC	RPG table load/table dump code	
#RPRW	RPG chain and read files move field code	
#RPRX	RPG initialize segment list	
#RPRY	RPG library of subroutines code	
#RPRZ	Form R-dot RPG code from segment list and object code	
#RPZY	RPG dump control facility	
DRGCFI	RPG find item in compression (#RPIC)	
DRGCZA	RPG open a compression area (#RPG)	
DRGCZC	RPG write a compression (#RPG)	
DRGCZE	RPG close a compression area (#RPG)	
DRGCZG	RPG open a compression block (#RPIC)	
DRGCZH	RPG get next compression (#RPIC)	
DRGCZK	RPG close a compression block (#RPIC)	
DRGCZM	RPG write an object code block (#RPIC)	
DRGCZN	RPG get next source record (#RPG)	
DRGCZP	RPG printer control (#RPG)	
DRGCZZ	RPG call next compiler phase (#RPG)	
PIOCS	RPG disk control (#RPG)	

Subroutine Name	Descriptive Name	Function
$$SYRP	RPG II halt processor	Called by the supervisor whenever a halt is requested by an RPG II compiler phase or RPG II object program. This is a transient. When an error occurs, $$SYRP is called and sets up a parameter list to be passed to the system log transient in the supervisor. For a description of errors, error codes, and console displays, see *IBM System/36 RPG II Messages Manual,* SC21-7940.
#RPSPD	Delete held print files	Called by the RPGONL procedure if the user does not want to see the RPG II compiler listing. This module deletes the spool queue entry for the compiler listing.
#RPDD	Data dictionary test	Called by RPGC, AUTO, and RPGONL procedures to determine if a data dictionary exists.
@PGAA	Array index	Retrieve the address of the nth field in an array in which n is the value of the index in an input, calculation, or output specification. This subroutine is used when a specified element in an array is required.
@PGAB	File translation	Replace characters in an entry in the translation table. The first character of each 2-byte element in the table is compared against the input data in the input buffer. When a comparison is found, the second byte of the translation table is replaced by the first byte. For output, the first byte of the translation table is replaced by the second byte.
@PGAC	Square root	Find the square root of any zoned decimal number with or without decimal places by finding the number of significant digits in the source and squaring successively smaller numbers until the exact root is found. The result is automatically half adjusted.
@PGAS	Array sort	Sort the array specified into ascending or descending order, depending on how the array was defined.
@PGBB	BSCA logic for transmit and receive	Place the table element address into the BSCA preopen DTF, if the table element address was specified for the dial number or station IDs on the telecommunications specifications. Translate station IDs using the translation table, if file translation was specified for the BSCA file.
@PGBG	BSCA logic for retrieve	Place the table element address into the BSCA preopen DTF, if the table element address was specified for the dial number or station IDs on the telecommunications specifications. Translate station IDs to ASCII by using the translation table, if ASCII was specified.
@PGBI	Convert to binary	Convert a decimal number to its binary equivalent.
@PGBO	Convert to decimal	Convert a binary number to its decimal equivalent. (The length will be either hex 04 or hex 09, depending on the size of the field specified in the extension or input specifications. If the size is 2 bytes binary, the length will be hex 09. If the number is larger than the number that can be contained in 4 bytes, only the overflow portion is shown.)

Subroutine Name	Descriptive Name	Function
@PGBP	BSCA logic for transmit	Replace the table element address into the BSCA preopen DTF if the table element address was specified for the dial number of station IDs. Translate station IDs to ASCII by using the translation table, if ASCII was specified.
@PGBS	BSCA SVC data management router	Call the data management router to get or put to the BSCA file.
@PGCI	Unpack	Convert a packed decimal field with a maximum length of 8 bytes to a signed unpacked decimal field up to 15 bytes long.
@PGCO	Pack	Convert an unpacked field up to 15 bytes long into a packed field with a maximum length of 8 bytes. The sign is maintained.
@PGCR	KEYBORD/CRT data management	Determine the type of operation requested and builds the appropriate 5250 text stream. Issue the I/O request to work station data management. Check the validity of the command keys that are pressed with the SET operation. Position the keyed data in the RPG field for the KEY operation.
@PGDC	DEBUG	Produce either one or two records, depending upon the parameters passed by the object code generated in the #RPQV compiler phase. One record contains a list of all indicators that are on at the time the DEBUG operation code was encountered. The second record shows the contents of any one field, array, array element, or table element.
@PGDI	Alternate collating sequence	Compare two alphameric field when one or both are not in normal collating sequence.
@PGDM	SVC data management router	Call the data management router to get or put to all files other than BSCA.
@PGFI	Load object tables	Load any preexecution-time table/array and its alternate into storage and perform sequence checking.
@PGFO	Dump object tables	Dump object-time table/array and its alternate onto the appropriate device at last record (LR) time, if the extension specification for this table/array has a to-filename entry specified.
@PGIC	Divide	Divide two numbers in the prime work area of ROCA and leave the quotient and remainder in the prime work area in case an MVR operation code follows.
@PGLC	LOKUP	Compare a field or search argument against the entries in a given table or array, until a specified setting of the condition register results, and return the address of the table/array element that caused the register to be set.
@PGMA	Move array	Move data from a field to an array, from an array to a field, or from an array to an array. If an array if variably indexed, this subroutine calculates the address of the indexed element. Data is moved from factor 2 to the result field; the length of the move is the shorter of the lengths of factor 2 and the result field. If factor 2 is a figurative constant, it fills the array from the referenced array element to the last element in the array.

Subroutine Name	Descriptive Name	Function
@PGMC	Multiply	Multiply the two integers in the prime work area in ROCA.
@PGRI	Indicators	Set specified resulting indicators on or off as required by the object program.
@PGTA	CONSOLE data management	Determine and display the current record format on the display station display screen. Process the CONSOLE record keyed by the display station operator.
@PGTC	Test zone	Test the zone of a byte loaded into the prime work area and sets the condition register.
@PGTD	Work station return code update	Update *STATUS and data managment return codes on input, output, NEXT, ACQ, and REL operations.
@PGTI	Work station input processing	Obtain an input record for the object program from the WORKSTN file. Scan the work station ID table to establish control information for the WORKSTN file. Save data and indicators for the currently active work station. Process errors encountered by a work station. If a WORKSTN INFSR error/exception subroutine is specified, exit to the subroutine when an error condition occurs or when a function key is pressed.
@PGTL	CONSOLE limits data management	Display the limits prompt on the display station display screen. Process the low and high limits entered by the display station operator.
@PGTO	Work station output processing	Invoke work station data management to perform a put or release operation. Schedule an invite input operation for multiple work station files or for requestors. Initialize a work station that has not undergone startup input processing. If a WORKSTN INFSR error/exception subroutine is specified, exit to the subroutine when an error condition occurs.
@PGTP	Work station allocation (ACQ)	Invoke work station data management to allocate (acquire) a work station for the object program. Create a new work station identification table entry. If a WORKSTN INFSR error/exception subroutine is specified, exit to the subroutine when an error condition occurs.
@PGTR	Work station deallocation (REL)	Deallocate (releases) a work station from the object program. Halt if entry for work station to be released cannot be found. Delete work station identification table entry. If a WORKSTN INFSR error/exception subroutine is specified, exit to the subroutine when an error condition occurs.
@PGTS	Work station scan	Scan work station ID table for one of these types of entries: • A specified work station ID entry • Any work station ID entry • A null entry
@PGTT	Work station retrieve display station information post	Invoke work station data management to retrieve display station status information. The status information is display station size (1920 or 3564).

Subroutine Name	Descriptive Name	Function
@PGTU	Select next input work station	Select a work station (of a multiple work station file) from which to receive the next input record. If a WORKSTN INFSR error/exception subroutine is specified, exit to the subroutine when an error condition occurs.
@PG01	Read SYSIN	Read one 120- or 512-byte record each time the subroutine is called. Set the end-of-file indicator when the END statement is encountered.
@PG20	Get/put external indicators	Swap the external indicators between the specified display station's local communication area and the ROCA indicator table. Return results in the RPG field specified in the RLABL parameter list.
@PG21	Get/put local communications area	Swap data between the local communication area for a specified display station and an RPG II field or data structure. Return results in the RPG II field specified in the RLABL parameter list.
@PG22	Read and update work station utility transaction file	For the first read or when a restart option, read the first logical record from the chain specified by the parameter list array. For a normal get operation, read the next logical record in the chain and, if last record, return a last record indication to RPG. When the close option is specified in the special DTF, close the WSU transaction file and return to RPG. When an error condition is found, turn on appropriate indication in the parameter list array and return to RPG.
@PG23	Message retrieve subroutine	Retrieve messages from user-specified message member. The maximum length of a level-1 message is 75 characters; the maximum length of a level-2 message is 225 characters.
@PG40	Remove S/O and S/I IGC control characters	Move the from-field to the to-field and remove the shift-in and shift-out characters from the first and last positions of the from-field. Set a return code indicating the status of the move.
@PG41	Insert S/O and S/I IGC control characters	Move the from-field to the to-field and insert the shift-in and shift-out characters from the first and last positions of the from-field. Set a return code indicating the status of the move.
@PG95	Inline inquiry support	Turn on an indicator passed in the parameter list if an inquiry request if pending. Turn off the inquiry request bit in system communications area before returning.

Appendix C: Program Product Utilities Directory

This appendix provides brief descriptions of program product utilities modules. It contains module/routine descriptions for modules in the the following program product utilities:

- Data file utility (DFU)
- Screen design aid (SDA)
- Source entry utility (SEU)
- Work station utility (WSU)
- Ideographic sort utility
- Ideographic character generator utility
- Query/36
- PS/36
- DW/36
- Development Support Utility (DSU)

Use this appendix when diagnosing problems relating to the IBM program product utilities and any code generated by utilities within it.

All modules listed in this appendix, with the exception of WSU modules, are execution-time modules. The modules listed under *WSU compile-time modules* are the modules that comprise the WSU compiler; the modules listed under *WSU execution-time modules* are those that are either part of the user's WSU object program or are available to the user's object program.

Data File Utility (DFU) Program Product Directory

Module Name	Descriptive Name	Function
#DFDE	DFU enter/update mainline–entry and insert modes	Prompt operator for new record data in sequential or direct file. Build record and write to file.
#DFDI	DFU diagnose specifications	Diagnose DFU specifications.
#DFDU	DFU enter/update mainline–update mode	Prompt for the review or edit of the previous record processed in a sequential or direct file.
#DFEI	DFU enter/update mainline–entry and insert modes	Prompt operator for key and data fields for a new record in an indexed file. Build record and write to file.
#DFEU	DFU–enter/update mainline subroutines	Retrieve the first enter/update mainline driver phase. Contains the following subroutines: update and display batch and total accumulators; allocate printer; print new, updated or deleted records; and process operator requested end of job.
#DFEX	DFU initialize enter/update/inquiry/list execution	Read and process DFU control statements; allocate new file for enter function; initialize job execution area (COMMON) for update, inquiry, and list functions; route control to the execution module to process the file.
#DFFB	DFU create format in DFU work file	Create the file, record, and field information sectors for the DFU format and write them in the work file.
#DFIN	DFU inquiry execution	Process user requests interactively in displaying indexed sequential or direct data file records at the display station.
#DFLS	DFU list execution	List the data file in the format specified by the DFU program.
#DFMP	DFU build attributes specifications	Find and retrieve records from the source member containing the file definition and input specifications. Allocate and open temporary data file for the diagnosed DFU attributes.
#DFQE	DFU create enter/update DFU specifications	Prompt operator for information required to build DFU specifications needed to process an enter/update request.
#DFQI	DFU create inquiry DFU specifications	Prompt operator for information required to build DFU specifications needed to process an inquiry request.
#DFQL	DFU create list DFU specifications	Prompt operator for information required to build DFU specifications needed to process a list request.
#DFSB	DFU build source input for sort portion of DFU list	Create a source member of sort sequence specifications for the data file to be listed.
#DFSS	DFU save DFU specifications source	Prompt for options and perform source save.
#DFUD	DFU enter/update mainline–update mode	Prompt for the review or edit of the previous record processed in an indexed file.
#DFUP	DFU update DFU specifications	Display, print, and/or allow for the editing of DFU specifications.
#DFWS	DFU create work station source specifications	Create work station source specifications for enter/update and inquiry jobs, and write the DFU format as a SUBR library member.

Screen Design Aid (SDA) Utility Directory

Module Name	Descriptive Name	Function
#SAEF	SDA format termination processing	Process end-of-member-options display; copy format member in primary work file to source library.
#SAEM	SDA menu termination processing	Process end-of-menu-options display; copy menu members in primary work file to source library.
#SAEP	SDA print termination processing	Process end-of-print-options display; print formats from format member in primary work file.
#SAER	SDA RPG termination processing	Process end-of-RPG-generation-options display; copy RPG WORKSTN specifications in secondary work file to source library.
#SAEW	SDA WSU termination processing	Process end-of-WSU-generation-options display; copy the WSU program in secondary work file to source library.
#SAFA	SDA screen format add processing	Perform ADD and READD format operations using secondary work file; perform recovery processing on secondary work file; process end-of-format-options display.
#SAFU	SDA screen format update/copy processing	Perform UPDATE and COPY format operations using secondary work file; perform recovery processing on secondary work file; process end-of-format-options display.
#SAIN	SDA initialization	If SDA primary work file exists, show the recovery options display; initialize the LDA.
#SALB	SDA librarian	Service requests from selection routines for a member name list.
#SAMM	SDA menu modification	Process menu definition work displays and update text(s) in secondary work file for user modifications to command and display text or help text; process end-of-text-options screen.
#SAPF	SDA screen format processing	Perform mainline processing for the building of a secondary work file for ADD, RGADD, COPY, and UPDATE operations on a particular format.
#SAPM	SDA menu processing	Build secondary work file for text editing, translating System/34 type display text to System/36 menu text if necessary.
#SAPR	SDA RPG generation processing	Build secondary work file for RPG program generation. Prompt for and process RPG program definition displays.
#SAPW	SDA WSU generation processing	Build secondary work file for WSU program generation. Prompt for and process WSU program definition displays.
#SASC	SDA compiler selection	Process recovery on primary work file and user selection of source member to compile.
#SASE	SDA edit selection	Process recovery on primary work file and user selection of source member for SEU editing.
#SASF	SDA screen format selection	Provides the following functions: • Process recovery on primary work file and user selection of format source member • Prompt for user selection of screen formats and user selection of ADD, UPDATE, DELETE, RESTORE, or PATTERN AFTER screen format operations

Module Name	Descriptive Name	Function
#SASM		SDA menu selection provides the following functions: • Process recovery on primary work file and user selection of menu source members • Prompt for menu texts within a menu source member and user selection of ADD, UPDATE, DELETE, or PATTERN AFTER text operations
#SASO	SDA main option menu processor	Service user's menu selection requests for the following: • Create or update a menu and its help text • Create, add, update, delete, and copy formats for the screen format generator • Create, add, update, delete, copy formats for WSU • Build RPG II WORKSTN file specifications • Build a WSU program • Edit source members via SEU • View screen formats in a $SFGR load member • Print images of screen formats in a source member • Evoke the *screen format generator* 8 to compile screen formats
#SASP	SDA screen format print	Process recovery on primary work file and user selection of a format source member. Initialize primary work file. Process user selection of screen formats within a format source member to print.
#SASR	SDA selection for RPG program selection	Process recovery on primary work file and user selection of source members. Initialize primary work file.
#SASV	SDA screen format view selection	Process recovery on primary work file and user selection of format load member. Initialize primary work file. Process user selection of screen formats within a format load member to view.
#SASW	SDA selection of WSU program generation	Process recovery on primary work file and user selection of source member. Initialize primary work file.
#SAS1	SDA S specification syntax checker	Perform syntax checking on S specifications.
#SAS2	SDA D specification syntax checker	Perform syntax checking on D specifications.
#SAS3	SDA D specification servicing	Move data between work screen image and D specification constant area.
#SAS4	SDA D specification syntax checker	Perform syntax checking on WSU D specifications.

Source Entry Utility (SEU) Directory

Module Name	Descriptive Name	Function
#SEAN	SEU scan	Scan a member for a statement that contains a specified character string and optionally replace the string with another specified character string.
#SEDL	SEU delete	Delete statements from the current member.
#SEDX	SEU data management operations for delete, count, and add index sector function	Response check for all statement numbers, delete a statement by removing index entry, count number of statements between the from statement and the ending statement for move operation, get new index sector when required and chain to previous and next sectors.
#SEEJ	SEU end of job	Copy statements from the SEU work file to the library member and perform statement serialization, program name duplication, and member listing as requested. Delete work file.
#SEEO	SEU end-of-job option and command key display	Allocate a printer to SEU if printing is requested, change number of display screen lines per statement, display the command function key display, and prompt for end-of-job options.
#SEET	SEU enter/update	Enter statements into the member or update statements that already exist in the member. Retrieve selected display screen formats. If print option is on, use *printer data management* 4 to print statements.
#SEIC	SEU include	Include statements from another member into the current member. Perform roll up and search-end-of-source functions for include members. If print option is on, use *printer data management* 4 to print statements.
#SEID	SEU include II	Check responses to Include-From-Library-Name, and Include-From-Member-Name prompts. Change roll factor.
#SEIN	SEU initialization	Initialize variables and diagnose the SEU control statements. If it already exists, copy the member being worked on into the work file from the library.
#SELL	SEU roll	Perform roll up, roll down, and search-end-of-source functions on all members except include member.
#SEMV	SEU move	Copy or move specified statements from one location to another. If move was specified, delete statements at original location.
#SEMX	SEU message translate	Allow the user to enter translated versions of messages that are contained in a message source member.
#SERA	SEU RPG II optional calculation specification syntax checker–phase 3	Continue RPG II calculation specifications syntax checking.
#SERB	SEU RPG II optional calculation specification syntax checker–phase 2	Continue RPG II calculation specifications syntax checking.

Module Name	Descriptive Name	Function
#SERC	SEU RPG II optional calculation specification syntax checker–phase 1	Syntax check RPG II calculation specifications.
#SERD	SEU RPG II optional calculation specification syntax checker–phase 4	Continue RPG II calculation specifications syntax checking.
#SERE	SEU RPG II extension, line counter, copy statement, auto report, and control card specifications syntax checker–phase 1	Syntax check RPG II control, extension and line counter specifications; and auto report option and /COPY specifications.
#SERF	SEU RPG II file description specification syntax checker–phase 1	Syntax check RPG II file description specifications.
#SERG	SEU RPG II file description specification syntax checker–phase 2	Continue syntax checking RPG II file description specifications.
#SERH	SEU RPG II file description specification syntax checker–phase 3	Continue syntax checking of RPG II file description specifications.
#SERI	SEU RPG II input specification syntax checker–phase 1	Syntax check RPG II and auto report input specifications.
#SERJ	SEU RPG II input specification syntax checker–phase 2	Continue syntax checking RPG II and auto report input specifications.
#SERL	SEU RPG II extension specification syntax checker–phase 2	Continue syntax checking RPG II and auto report control, extension and line counter specifications; and auto report option and /COPY specifications.
#SERO	SEU RPG II output specification syntax checker–phase 1	Syntax check RPG II and auto report output specifications.
#SERP	SEU RPG II output specification syntax checker–phase 2	Syntax check RPG II and auto report output specifications.
#SERT	SEU RPG II telecommunications specification syntax checker–phase 1	Syntax check RPG II and auto report telecommunications specifications.
#SERU	SEU RPG II telecommunications specification syntax checker–phase 2	Continue syntax checking RPG II and auto report telecommunications specifications.
#SEU	SEU common	Load the SEU local data area and the four main storage resident routines (SEUDM, SEUKEY, SEUMSG, and SEUPNT).

Work Station Utility (WSU) Directory

WSU COMPILER MODULES

Compile-Time Module	Descriptive Name	Function
#WSGAF	WSU assign data base field values	Assign data base field values; place the JDB size, WDB size, and counts of data base fields, system fields, program labels, and subroutines in GCOMMON.
#WSGAI	WSU indicator assignment phase	Use local copy of system indicator table to update indicator-referencing T, M, S, D, and C records with the corresponding mask/displacements. Print indicator usage lists.
#WSGAL	WSU assign program label values	Update field name table with relative command data block numbers and data block displacements for program labels.
#WSGAM	WSU message assignment phase	Update C records that have message-defining literals with assigned relative message numbers. Build MIC table and update C records that have message-referencing MIC numbers with assigned relative message numbers. Print list of referenced MICs.
#WSGAS	WSU build array specification area	Generate array specification area (ASA) for the WSU object program.
#WSGBF	WSU field name table building phase	Build field name table from data field and program label definition/references on T, M, F, S, D, and C records.
#WSGCD	WSU command data area building phase	Format the command data area from C records, GCSBDFSB, GCSBDRSD, and GCSBDPSB and put the area in work file object at the preassigned location.
#WSGCH	WSU control header building phase	Format the control header from sections of GCOMMON and put it in the first work file object sector.
#WSGCS	WSU command specification area building	Format the command specification area for C records and put the area in the work file object at the preassigned location.
#WSGDS	WSU data specification area building phase	Format the data specification area from T, M, F, S, D, and C records and the system fields table and put the area in the work file object at the preassigned location.
#WSGEP	WSU execution procedure member generating phase	Calculate the minimum execution region size. Format the execution procedure statements and put them in the new procedure member. Print an execution region map and the OCL for the new procedure.
#WSGFO	WSU object program formatting phase	Build file and record type specification block displacement tables from T, M, I, and F records. Build a process segment specification block displacement table and table of display attributes and field counts from S and D records. Determine total object size and assign relative locations to individual object areas.
#WSGFS	WSU file specification area building phase	Format the file specification area from T, M, I, and F records and put the area in the object work file at the preassigned location.

Compile-Time Module	Descriptive Name	Function
#WSGLS	WSU local specification area building phase	Format the local specification area from F records and put the area in the object work file at the preassigned location.
#WSGMD	WSU message data area building phase	Format the local specification area from C records and put the area in the object work file at the preassigned location.
#WSGML	WSU message length assignment phase	Build tables from D records that describe display screen format field distribution. Determine the number of bytes allowed for message text on each display screen format. Update D records with assigned message lengths. Print extended diagnostics.
#WSGMS	WSU message specification area building phase	Format the literal section of the message specification area from C records and put the section in the work file object at the preassigned location. Format the MIC section of the message specification area from C records and put the section in the work file object after the literal section.
#WSGMT	WSU diagnostic message text listing and cleanup phase	Print list of diagnostic message texts cross-referenced to error numbers and severity codes. If an abort or terminal error has been signaled, abnormally terminate generation.
#WSGOB	WSU object program member generating phase	Load *library sector PUT/GET* 6 and copy the preformatted work file object to the new object member.
#WSGOM	WSU generator post-assign initialization phase	Initialize the post-assign GCOMMON extension, overlaying assign-only code and data.
#WSGPS	WSU process specification area building phase	Format the process specification area from S records and GCSCRNTB and put the area in the work file object at the preassigned location.
#WSGRF	WSU field data block resolution phase	Update embedded field data blocks in T, M, F, S, D and C records and system fields table with name definitions and relative number and displacement assignments from corresponding field name table entries.
#WSGSS	WSU screen specification area building phase	Format the screen specification area from S and D records and GCSCRNTB and put this area in the work file object at the preassigned location.
#WSGWF	WSU generator post-syntax initialization phase	Initialize the post-syntax GCOMMON extension, overlaying syntax-only code and data, and move a copy of the system fields table into GCOMMON.
#WSG0	WSU generator common initialization phase	Initialize the generator's common data areas (GCOMMON, GCCHGEND, and GCSYNTAX). Print source records up to first noncomment or end of file.
#WSG0I	WSU generation initialization	Initialize generator environment. Prime common data area with control information, open required files, and start WSU compilation listing.
#WSG1	WSU J specification syntax checking phase	Print J specification and read and print all source records to the first T, M, S, D, or C specification EOF.

Compile-Time Module	Descriptive Name	Function
#WSG2	WSU T and M specification, F and I specification syntax checking phase	Print records in source buffer and read and print T, M, F, and I specifications and all source records to the first S, D, or C specification or end of file. Put T, M, I, and F records in the work file intermediate text.
#WSG2A	WSU E specification syntax checking phase	Syntax check all E specification fields, build intermediate text entries and common data areas, write MICs for all errors detected.
#WSG3A	WSU S specification syntax checking phase	Print record in source buffer and read and print S specifications and all other source records to first D or C specification or end of file. Put S record in work file intermediate text.
#WSG3B	WSU D specification syntax checking phase	Print record in source buffer and read and print D specifications and all other source records to next C or S specification or end of file. Put D record in work file intermediate text.
#WSG4A	WSU calculation specification syntax checker	Begin checking the syntax of calculation specifications. Put special C and/or S records in the work file intermediate text.
#WSG4B	WSU calculation specification operation, table, and continuation syntax phase	Complete syntax checking for table operations. Print C specification and any continuation line and all other source records to next C, S, or D specification or end of file. Put C record in work file intermediate text.
#WSG4C	WSU calculation specification operation, MIC/message text, and continuation syntax checking phase	Complete syntax checking for message operations. Print C specification and read and print any continuation line and all other source records to next C, S, or D specification or end of file. Put C record in work file intermediate text.
#WSG4D	WSU calculation specification operation, factor 2, result field, and half-adjust syntax checking phase	Complete syntax checking for non-message/table operations. Print C specification and read and print all source records to next C, S, or D specification or end of file. Put C record in work file intermediate text.
#WSG5	WSU file, record type, and processing level assignment phase	Build relational tables for file, and record type definitions. Update C and S records with relative file and record type numbers and processing level codes. Print extended diagnostics.
#WSG6	WSU format assignment phase	Build relational tables for format definitions. Update C records with relative screen format numbers. Print extended diagnostics.
#WSG8	WSU operation code assignment phase	Print unreferenced multiply-defined and undefined field names. Print extended diagnostics. Adjust field-name-definition-dependent operation codes in C records.
#WSG9	WSU SFGR source member generating phase	Format SFGR S and D records and put them in a new source member.

WSU EXECUTION-TIME MODULES

Execution-Time Modules	Descriptive Name	Function
#TXCR	WSU TXCR file build (CREATE) mainline	Initialize for TXCR utility.
#TXC1	WSU TXCR end statement processor	Process TXCR END control statement.
#TXC2	WSU TXCR recover statement processor	Process TXCR RECOVER control statement.
#TXC3	WSU TXCR refresh end statement processor	Process TXCR REFRESH control statement.
#TXC4	WSU TXCR recreate statement processor	Process TXCR RECREATE control statement.
#TXC5	WSU TXCR txcrfile statement processor	Process TXCR TXCRFILE control statement.
#TXC6	WSU TXCR run processor	Process TXCR RUN control statement.
#WSUCR	WSU transaction file create routine	Build transaction file record-by-record.
#WSUEX	WSU transaction file extraction routine	On transaction file record retrieval, selectively exclude retrieved records.
#WSURV	WSU recovery extraction routine	On transaction-file-trailer-containing-file record retrieval, selectively exclude retrieved records.
#WSXI1	WSU execution initialization phase 1	Initialize for WSU object program execution by finding required message, object program, and format members, and by intialization all required data areas.
#WSXI2	WSU execution initialization phase 2	Allocate and open disk files, set up WSU execution work file according to information in object member, format common data areas.
#WSXP	WSU execution process mainline	Perform the following execution-time subroutine processing: • Initial load low storage resident data/code. • Initialize low storage resident data areas. • Perform enter linkage for called subroutine and, if required, fetch subroutine-containing WSU transient. • Perform return linkage for called subroutine and, if required, fetch calling-routine-containing transient. • Dispatch work session driver to complete, restart, or begin interaction cycle for work station, or work session abort to end cycle. • Abort job or session. • Process storage requests.

Execution-Time Modules	Descriptive Name	Function
#WSX00	WSU execution transient 0	Log driver trace entries, work station trace entries, and file trace entries.
#WSX01	WSU execution transient 1	Read records from specified data file, and suspend session if necessary to avoid record contention.
#WSX02	WSU execution transient 2	Write record to specified data file.
#WSX03	WSU execution transient 3	Consists of routines for the following: • Trace pool element functions, if pool trace active. • Restore registers and return to caller. • Display specified WSU/user format and select any required messages.
#WSX04	WSU execution transient 4	Process the RANGE operation for alphameric factors.
#WSX05	WSU execution transient 5	Process all calculation specifications for the current processing segment.
#WSX06	WSU execution transient 6	Consists of routines for the following: • Perform sign-on initialization of transaction file, session activation, and initiate the IJ, IW, and first sequence screen processing segments. • Validate work session data record chain in transaction file.
#WSX07	WSU execution transient 7	Determine next segment to be processed in enter mode.
#WSX08	WSU execution transient 8	Consists of routines for the following: • Handle automatic puts and program-defined processing within current segment. • Get specified CDB from work file and place it in the pool.
#WSX09	WSU execution transient 9	Perform session deactivation, sign-off transaction file and work session termination, and initiate the EW and EJ processing segments.
#WSX10	WSU execution transient 10	Consists of routines for the following: • Drive interaction cycles, restore get-suspended session, accept user, set command/function key indicators, and select appropriate routine for keyed response or request. Save or restore normal WDB for cycle turnover. • Get normal or alternate WDB from work file into pool element and transfer corresponding sections to work station level JDB fields, common indicators, and common data pointers. • Put work station level JDB fields, common indicators, and common data pointers into corresponding sections of normal or alternate WDB in pool element.

Execution-Time Modules	Descriptive Name	Function
#WSX11	WSU execution transient 11	Process WSU calculation specification multiply operation.
#WSX12	WSU execution transient 12	Process WSU calculation specification divide operations and move remainder operations.
#WSX13	WSU execution transient 13	Process WSU calculation specification table compare operations for table of alphameric literals or table of alphameric fields.
#WSX14	WSU execution transient 14	Process WSU calculation specification table compare operations for table of numeric literals or table of numeric fields.
#WSX15	WSU execution transient 15	Convert specified binary value to decimal character equivalent in JDB or decimal character value in JDB to binary equivalent in key area.
#WSX16	WSU execution transient 16	Route control to appropriate processor for response to WSU message or MENU display. Handle abnormal completion code from *work station data management* 4.
#WSX17	WSU execution transient 17	Get and update trailer information from transaction file record.
#WSX18	WSU execution transient 18	Perform the following: • I/O for calculation specification or review of GETPH, GETPR, GETNH, or GETNR operation. • I/O for GET calculation specification operation. • Format trailer information transaction file adds or inserts. • Allocate pool element.
#WSX19	WSU execution transient 19	Handle review mode processing. Zero or blank all normal or alternate work station level JDB fields.
#WSX20	WSU execution transient 20	Perform LOKUP operation for WSU array manipulation.

Ideographic Sort Utility Directory

Module Name	Function
0.#KA$A	Print sort's COMMON area with labels after phases 0A, 0B, 0D, 0E, and all passes.
0.#KA$D	Contain phase 0D COMMON constants. Serve an input to 0.#KA$A when dumping COMMON.
0.#KA$E	Determine if there is enough available storage to load 0.#KA$A (sort COMON dump routine). If there is enough available storage, load 0.#KA$A.
0.#KA$X	Contain phase 1, 2, and 3 COMMON constants. Serve as input to 0.#KA$A when dumping COMMON.
0.#KA@HELP	Display screen formats for ideographic sort prompts (for nonideographic session).
0.#KA$HELP	Display screen formats for ideographic sort prompts (for ideographic session).
0.#KABA (phase 0B)	Finish diagnosing the header statement and issue messages if any errors are detected. Pseudo open and allocate the input file.
	Allocate the work file, if work file (FILE) statement present.
	Read the statement following the header statement.
	If ALTSEQ specified, load and call 0.#KABC.
0.#KABC	Read and diagnose ALTSEQ statements and issue messages if any errors are detected.
	Modify the 256-byte alternate collating sequence table as specified on the ALTSEQ statements that are read in.
0.#KACA (phase 0C)	Read and diagnose sort sequence specifications for one sort run.
0.#KACB	Issue error messages based on the error table generated in phase OC. Determine if any terminal errors were diagnosed by phase OC; if they were, pass control to 0.#KAGA.
0.#KACC	Move the select/build routine to its final position in storage. Move the summary table behind select/build routine if summary sort is used.
0.#KACE	Diagnose the sequence specification statements to determine if they are valid include or omit statements. Generate code segments from the information on the include/omit statements.
0.#KACF	Diagnose the sequence specification statements to determine if they are valid field statements. Generate code segments from the information on the field statements.
0.#KACK	Diagnose include/omit record type specifications with a factor 2 keyword. Generate a valid include/omit sequence specification statement.
0.#KACL	Calculate module length.
0.#KACZ	Complete select/build code. Move the summary table to phase 1 location behind COMMON.
0.#KADA	Determine the active program lengths of modules 0.#KA1X, 0.#KA2A, 0.#KA3A, and 0.#KA3S.
0.#KADB (phase 0D)	Design the execution phases of the sort program (phase 1, phase 2, and phase 3).
	Check for terminal error in design of sort. If found, pass to 0.#KAGA.
0.#KAEA (phase 0E)	Allocate output file, if not the same file as the input file. If work file (FILE) statement omitted, call automatic work file allocation (@KAEG).

Module Name	Function
0.#KAGA (phase 0G)	Issue messages indicating errors found and information determined in the generation phases.
0.#KASE	Convert 1-byte EBCDIC KANA reading field to Seion KANA control field.
0.#KASRT (phase 0A)	Do initial sort.
0.#KATB	Determine if #KAMAST and #KACTIVE are on disk. Generate the active collating sequence tables.
0.#KATL	Build ideographic control field.
0.#KAZA	Move a given number of bytes from one area in storage to another.
0.#KA1D	Calculate and place in COMMON the sector address (SSS) of the current work block. Calculate and place in COMMON the SSS of the next work block.
0.#KA1L	Perform initialization for 0.#KA1X. Load into storage the required phase 1 modules for 0.#KA1X.
0.#KA1X (phase 1)	Perform a tournament sort, produce variable-length strings of sequence work records, and place a string of a variable number of work record blocks onto the work file. At end of pass for a summary sort, move the summary table from the next byte after the select/build code to the next byte after COMMON.
0.#KA1Z	Issue messages indicating errors found and information determined in the preceding execution phase.
0.#KA2A (phase 2)	Merge strings created in phase 1 until the total number of strings is less than or equal to the sort's order of merge.
0.#KA2L	Load into storage the required phase 2 modules for 0.#KA2A.
0.#KA3A (phase 3)	Allocate the output file, if it is the same file as the input file, perform last pass of the merge, and place the sorted records consecutively in the output file.
0.#KA3L	Perform initialization for the phase 3 modules (0.#KA3A/0.KA3S), and load into storage the modules required by 0.#KA3A/0.KA3S to perform phase 3. If the number of remaining strings is less than the order of merge, redesign the sort for phase 3.
0.#KA3S (phase 3)	Allocate the output file, if it is the same file as the input file, and perform last pass of the merge. If no summary table exists, delete duplicate records; if a summary table exists, summarize duplicate records as specified, and place the sorted records consecutively in the output file.
0.#KA4A (phase 4)	Issue displayed message and end the job after unsuccessful completion via the SYSLOG transient routine. Terminate the sort program after successful completion.
0.#KA9G	Provide the logical get I/O function for the work file.
0.#KA9I	Read sort sequence specification statements, which are in a procedure member or are entered from the display station keyboard.
0.#KA9M	Read sort sequence specification statements from the loadable sort parameter list.
0.#KA9P	Provide the logical put I/O function for the work file.
0.#KA9S	Read sort sequence specification statements from a source member.

Module Name	Function
@KADC	Compute the initial area in phase 1 to be assigned to the internal sort area.
@KAEG	Determine the space needed for the work file and use special allocate to automatically allocate it.
@KAZB	Convert an unsigned zoned decimal number (up to 7 bytes long) to a 3-byte hex number.
@KAZC	Convert a 3-byte hex number to a 7-byte zoned decimal number and concatenate a sign byte to the result.
@KAZD	Divide a 3-byte hex number into another 3-byte hex number.
@KAZE	Convert 4-byte groups into their equivalent 9-bit EBCDIC values.
@KAZF	Convert 1 byte of storage to printable EBCDIC values, 1 bit at a time.
@KAZM	Multiple two 3-byte hex numbers.

The following module is the loadable sort transient that is packaged as part of the SSP.

0.#MAGS	Load and give control to #KASRT. Set the completion indicator and return control to the program that called sort.

Ideographic Character Generator Utility

Module Name	Descriptive Name	Function
#CGIN	CGU initialization	Open extended ideographic character file (#EXTN) and master sort information file (#KAMAST), and allocate the active collating sequence file (#KACTIVE).
		Allocate and open the display station.
		Locate all CGU modules and build a parameter table.
		Load #CGCM (common data areas) and #CGSB (common subroutines) into lower portion of #CGIN.
		Retrieve keywords from #CG#MG and save them in common area.
		Initialize data areas in common area.
#CGCM	CGU common data areas	Contain the initialized common data structure (CGCOM) used by all modules.
		Contain all file DTFs, physical buffers, IOBs, display station DTFs, printer DTFs, physical print buffers, and format index I/O areas.
#CGSB	CGU common subroutines	Load requested module, if required, and specify which modules are already loaded.
		Issue SYSLOG messages.
		Perform display station I/O.
		Analyze function and command keys (called by WSRTN).
		Right-adjust a numeric field.
		Convert extended character numbers to the RRN in #EXTN file.
		Convert RRN to sector address in #EXTN file.
		Convert sector address to RRN in #EXTN file.
		Convert RRN to extended character number.
		Convert IBM code to RRN in #KAMAST.
		Retrieve a record from #EXTN file.
		Retrieve a record from #KAMAST file.
#CGML	CGU mode control	Determine what function is requested via command key (entry, update, sort, delete, list, end of job).
		Load and pass control to the appropriate execution module when function is determined.

Module Name	Descriptive Name	Function
#CGEN	CGU entry mode	Determine next available location for new character in the #EXTN file. Prompt the operator for the number of the next character to be created. Check that the #EXTN character is not already defined. Call #CGEU to process the description of the new character. Call #CGAL to process the sort information, and write the new character and sort information to the respective files.
#CGUP	CGU update mode	Prompt the operator for the number of the character to be updated. Check that the requested character is a user-defined character. Check that the #EXTN character is defined.
#CGSO	CGU sort mode	Prompt the operator for the character number of the sort master record to be updated. Display and allow the operator to change the sort information in the sort master file; for IMB-supplied characters only, allow update to single pronunciation. Call #CGAL to write the updated sort information in the master sort file and to print the updated sort information if the print status is on.
#CGDE	CGU delete mode	Prompt the operator for the number of the #EXTN character to be deleted and display the character before deleting it. Call #CGAL to delete the character from the #EXTN file and the master sort file. Call #CGAL to print deleted records.
#CGEJ	CGU end of job	Prompt the operator for ending or continuing the job. Display number of records processed. If YES response to end-of-job prompt, print number of records processed if any printing has occurred. If NO response to end-of-job prompt, return to the exited mode.
#CGLI	CGU list mode	Prompt the operator for the upper and lower limits of the characters to be printed. Check that the character print limits correspond to the characters in the IBM-supplied extended ideographic character set (excluding special symbols), or to the characters in the user defined set. Check that the upper limit is greater than or equal to the lower limit. Print the characters and master sort information within the limits specified.

Module Name	Descriptive Name	Function
#CGEU	CGU character definition	Prompt the operator for a description of new and updated characters.
		Process the description of new and updated characters. (#CGEU displays the character as it is being defined or changed, and allows the user to restart character definition.)
		Process the Copy command function key to allow the user to copy and optionally combine characters during character definition.
		Call #CGAL to write new and updated character definition in the #EXTN file.
#CGAL	CGU common subroutines	Flag #KACTIVE file when active collating sequence changes.
		Write record to #EXTN file.
		Write record to #KAMAST file.
		Move record from the I/O buffer to the image buffer for processing.
		Prompt for sort information.
		Initialize sort hold fields.
		Convert information from #KAMAST to displayable format.
		Process user response to sort information prompt.
		Process Enter key response to sort information prompt.
		Check for blank fields with special symbol character type.
		Convert image to matrix form.
		Perform printing.
		Allocate and open printer.
		Print title and column heading lines.
		Print second line of matrix and updated sort information.
		Print remaining matrix lines.
		Interface with printer data management.
		Convert binary value to decimal.
		Convert decimal value to binary.
		Convert hex value to EBCDIC.

Query/36 Directory

Module Name	Descriptive Name
#QDDE	Entry and insert modes mainline (for sequential or direct files)
#QDDI	File specifications diagnosis processor
#QDDU	Update mode mainline (for sequential or direct files)
#QDEI	Entry and insert modes mainline (for indexed files)
#QDEU	Mainline subroutines
#QDEX	Initialization for execution processor
#QDFB	Data entry control statement generator
#QDGN	File, data definition, and data entry specifications generator
#QDLN	Data dictionary link processor
#QDMP	Data entry attribute specifications generator
#QDSS	Data entry source specification save processor
#QDUD	Update mode mainline (indexed files)
#QDWS	Work station source specification generator
#QUDA	Query/36 initialization
#QUDB	Work with queries display
#QUDC	Define the query display
#QUDD	File/format selection
#QUDE	Join files display and diagnostics
#QUDF	Record selection display
#QUDG	Field selection
#QUDH	Sort field selection
#QUDI	Select collating sequence and user defined collating sequence
#QUDJ	Column formatting and summarization display
#QUDK	Result field definition display
#QUDL	Break fields display
#QUDM	Select output device and output type display
#QUDN	Show file selections and join tests display
#QUDO	Copy queries display
#QUDP	Library list display
#QUDQ	Review/view query, reads query from library into task work space
#QUDR	Delete queries display
#QUDS	End this query display

Module Name	Descriptive Name
#QUDT	File/format definition extract and field table build
#QUDX	Show long comments for files, format, and fields
#QUDY	Translated language collating sequence tables
#QUDZ	Query subroutine build (compile query subroutine)
#QUD2	File list build
#QUF2	Record selection diagnostics
#QUK2	Result field definition diagnostics
#QUK3	Save/restore for result field definition
#QUPD	Print query definition
#QUPF	Print format of query output to disk file
#QURA	Query run initialization
#QURB	Run query to display or show report layout
#QURC	Return data to DW/36 print for multicopy
#QURD	Run query to printer
#QURE	Determine report format
#QURF	Compile code to format detail lines
#QURG	Initialize for run-time record selection
#QURH	Compile code to format summary lines and process breaks
#QURI	Run query to disk when summary output only requested
#QURJ	Call query data management to open the query
#QURK	Run query to display format data routine
#QUTA	Return data field instructions to DW/36 editor
#QUTB	Document output driver and return data to DW/36 print for column list
#QUTC	Return data to DW/36 editor

PS/36 Directory

Module Name	Function
#ODBA	DIA–Update status log.
#ODBC	DIA–Time-stamp the queue for connection failure.
#ODBD	DIA–Document-unit decoder.
#ODBE	DIA–Negotiate sign-on to partner.
#ODBG	DIA–Get reply from session printer.
#ODBH	DIA–Search command encoder.
#ODBK	DIA–Parser.
#ODBM	DIA–Get inbound data area.
#ODBN	DIA–Send command to session partner.
#ODBO	DIA–SRL decoder.
#ODBP	DIA–File server, terminate write.
#ODBQ	DIA–File server, write.
#ODBR	DIA–Get command from queue.
#ODBS	DIA–Search transaction processor.
#ODBZ	DIA session manager.
#ODFF	File document to DISOSS.
#ODFI	DLS temporary document cleanup.
#ODFS	Select document for file to DISOSS.
#ODL1	Maintain access codes.
#ODL2	Maintain keywords.
#ODL3	Maintain document classes.
#ODL5	View list of access codes.
#ODL6	View list of keywords.
#ODL7	View list of document classes.
#ODL8	View list of access codes.
#ODL9	View list of document classes.
#ODPH	Copy search results to another user.
#ODPL	View list of PS/36 users.
#ODQA	DLS request handler.
#ODQE	Send manager.
#ODQJ	NEP cancel.

Module Name	Function
#ODQS	View library request status.
#ODQT	DLS time-stamp.
#ODQY	DLS copy task.
#ODQZ	DLS copy task status update.
#ODRL	Select from list of remote libraries.
#ODSC	Copy, change, create a search.
#ODSD	Delete searches.
#ODSL	Document function router.
#ODSO	Choose print options.
#ODSR	Work with documents found.
#ODSW	Work with searches.
#OUAA	Analyze mail folders.
#OUAC	Control communications queues.
#OUAD	Install OSU files and libraries.
#OUAE	Display queue entries.
#OUAG	Get number of logical records.
#OUAL	Maintain remote destinations.
#OUAM	Maintain mail folder.
#OUAP	Process remote destinations.
#OUAQ	Maintain communication queue definitions.
#OUAR	Maintain communications routes.
#OUAS	Generate save/reorganize OCL.
#OUAT	Start transmission on communication queues.
#OUAW	Delete documents confirmation screen.
#OUAX	Delete routes confirmation screen.
#OUAY	Delete queue confirmation.
#OUAZ	Delete queue entry confirmation.
#OUA1	Install OFFICE/36 files on mail library.
#OUA2	Install non-OFFICE/36 files.
#OUA3	Install OFFICE/36 files on mail library.
#OUA4	Profile file conversion.
#OUBB	Build the asynchronous feedback records.
#OUBE	Log a router destination error.

Module Name	Function
#OUBF	Route for FINDSLOT.
#OUBJ	Cancel PS/36 NEPs.
#OUBK	Confirm PS/36 NEPs stopped.
#OUBR	Route and direct services.
#OUBS	Perform routing service.
#OUBX	Flush newest slot in fanout list.
#OUCA	Perform calendar processing.
#OUCB	Update/delete/add calendar items.
#OUCC	Print and/or delete calendar entries.
#OUCD	MODULE TO USE OUCLCF.
#OUCE	Overlay module to call common abort program.
#OUCF	Overlay module for OUCLF.
#OUCG	Set up week-, day-at-a-glance views.
#OUCI	Calendar select.
#OUCJ	Call #OUCL to select one calendar.
#OUCK	Define calendar session test version.
#OUCL	Prompt user for calendar selection.
#OUCM	Perform simple call to OUCLOP.
#OUCN	OUCLX overlay version.
#OUCO	Dummy driver for OUCLOP.
#OUCP	Overlay interface to OUFCMN.
#OUCQ	Build combined view of invitee list calendars.
#OUCR	Determine range for print/delete.
#OUCS	Calendar session.
#OUCT	Print or delete calendar entries.
#OUCU	Create, change, or delete calendars, managers, or descriptions.
#OUCV	Build list view for normal processing.
#OUCW	OVERLAY FOR OUCLVU.
#OUCX	Build list view for EXP VERF AND CONF.
#OUCY	Prepare for batch print delete of calendar.
#OUCZ	Batch print/delete of calendar items.
#OUC1	Add/change group meetings.
#OUC2	Delete group meetings/update meeting items.

Module Name	Function
#OUC3	View/update the invitee list of a group meeting.
#OUC4	Schedule/delete group meeting on 2-30 calendars.
#OUC5	Update meeting item on an invitee calendar.
#OUC6	Six-month calendar view.
#OUC7	Build combined view of invitee list calendars.
#OUDA	DIA session state controller.
#OUDB	Get command work area (CWA).
#OUDC	Put reply work area (RWA).
#OUDD	Process data distributions of DIA catcher.
#OUDF	Process request-distribution command.
#OUDG	Process SRR-cancel-distribution command.
#OUDH	Process SRR-set-control-value command.
#OUDI	DIA catcher–print for indirect users.
#OUDJ	Process SRR-sign-on-request command.
#OUDK	Process list command.
#OUDL	Process NRR-ACK-4 command.
#OUDM	Process obtain command.
#OUDN	Get next distribution.
#OUDO	Obtain documents for a general user.
#OUDP	Process acknowledge-3 command.
#OUDQ	Status correlation.
#OUDR	DIA catcher program.
#OUDS	Distribute a document and/or message for a general user.
#OUDT	Handle DIA catcher–status distribution.
#OUDW	Provide transfer overlay to do correlation.
#OUDX	Build page heading from list document.
#OUDY	Distribute data distribution to group.
#OUFE	Disk I/O abort.
#OUGD	Process group delete confirmation screen.
#OUGL	Control display of screen OUGR06.
#OUGP	Process present print options screen.
#OUGQ	Present print option screen for multiple selections.
#OUGR	Resolve group.
#OUGU	Update/create a group.

Module Name	Function
#OUGV	View merged or unmerged group.
#OUGW	Control display of screen OUGR02.
#OUGX	Enrollment and authority check.
#OUHC	Log error message from DIU parser.
#OUHL	Log messages to system console and history file.
#OUHM	SNADS send messages to console/history file.
#OUHR	SNADS receive manager.
#OUHS	Decoder.
#OUHT	SNADS receive manager.
#OUHU	SNADS receive manager.
#OUJD	SNADS distribute data/status common.
#OUJK	SNADS distribute data/status common.
#OUJS	SNADS distribute status.
#OUJU	SNADS distribute data/status common.
#OUKD	Decoder.
#OUKS	Decoder.
#OUK1	Write DOD, SPST, or DEST structure to queue.
#OUK2	Read SNADS directory table.
#OUK3	Read SPST, DOD, and destination structure.
#OUK4	SNADS addressing token translate.
#OUK5	SNADS addressing token translate.
#OUK6	SNADS parser, get next route.
#OUMA	Send/forward messages.
#OUMB	Print a document from mail.
#OUMC	Process create/delete mail log screen.
#OUMD	Process send/forward mail screen.
#OUME	Interface between DW/36 and #OUMD.
#OUMF	File a document out of the mail library.
#OUMG	Allocate/deallocate SNADS file at beginning/end of send mail.
#OUMH	Log hard-copy mail.
#OUMI	Sort/eliminate duplicate recipients.
#OUMJ	Message type substitution.
#OUMK	Print mail log listing.

Module Name	Function
#OUML	Process review-mail-log screen.
#OUMM	Process send-a-message screen.
#OUMO	Specify print options.
#OUMP	Define a sublist of the mail log.
#OUMQ	Access Q-file.
#OUMR	Print the mail audit report.
#OUMS	Process mail status screen.
#OUMU	Delete files or documents.
#OUMW	View/change mail log entry.
#OUMX	View a memo slip.
#OUMY	Check the personal document PSWD.
#OUM1	Send/forward/reply note screen.
#OUM2	Other mailing information for a note.
#OUM3	Recover a note.
#OUM4	Build include instruction for forwarded note.
#OUPC	Create calendar and mail log during enrollment.
#OUPD	Enroll Displaywriter users.
#OUPF	Install office default.
#OUPG	Driver for updating of general user profile.
#OUPO	Delete enrollment.
#OUPR	Enroll PS/36 users.
#OUPS	Enroll self.
#OUPU	View general user information.
#OUPW	View Displaywriter user information.
#OUQP	Process start/stop-activity queue.
#OUQS	Process activity queue.
#OURG	View names within a group.
#OURI	List directory by user ID and name.
#OURL	Select directory entries.
#OURM	Select receivers for mail.
#OURN	Alphabetically list names.
#OURP	View update for #OURG.
#OURR	Process list for #OURG.

Module Name	Function
#OURS	Select receivers for messages.
#OURT	List telephone directory.
#OURU	Process update, delete, view for RI, RT, and RN.
#OURZ	Get system values, authority, enrollment.
#OUSA	Server–initiate read.
#OUSB	Server–read.
#OUSC	Server–lock and release data object.
#OUSD	Server–terminate read.
#OUSE	Server–initiate write.
#OUSF	Server–write.
#OUSG	Server–terminate write.
#OUSH	Scottish bid main line.
#OUSM	Server–terminate write profile.
#OUSN	Server–backout write.
#OUSP	SNADS send–start session.
#OUSU	SNADS send–no more DIUs.
#OUSV	SNADS send.
#OUSW	SNADS send–control connection.
#OUSY	SNADS send–purge queue.
#OUSZ	SNADS send–suffix T2.
#OUS1	SNADS send–RCV NACK.
#OUS2	SNADS send–end of NACK.
#OUS3	SNADS send–decode NACK.
#OUS4	SNADS send–abort NACK.
#OUS5	SNADS send–feedback.
#OUS6	SNADS send–data object.
#OUS7	Server–decode DIA document unit.
#OUXA	Build queue definition in SQS.
#OUXH	Dequeue the task block for the present task.
#OUXI	Asynchronous lock task.
#OUXL	SNADS address token validation.
#OUXQ	Build queue definition in SQS.
#OUXR	PS/36 IPL procedure for scheduler.

Module Name	Function
#OUXT	Clean up control block chain.
#UMCA	Process migrate calendar A.
#UMCB	Process migrate calendar B.
#UMDL	Process migrate distribution list.
#UMPR	Process migrate profile.
#UMRA	Produce migrate profile report.
#UMRB	Produce migrate calendar report.
#UMRC	Produce migrate distribution list report.

DW/36 Directory

Module Name	Function
#PFAB	Perform final processing subroutines.
#PFAD	Create/delete #TEXTWRK.
#PFAI	Format index pages.
#PFAL	Allocate DTFs for printers/files.
#PFAT	Copy page/line formats.
#PFCP	Create cover page.
#PFDL	Deallocate DTFs for printers/files.
#PFED	Perform editing for data merge/summary math.
#PFER	Open resolved document.
#PFEX	Display the EXIT A DOCUMENT display.
#PFFP	Perform final processing/line numbers/change flag.
#PFHF	Create the headers and footers.
#PFHH	Create the outline headings/help tag tables.
#PFIB	Create the build table for auto-index.
#PFIC	Perform the interactive L/A copy function.
#PFIF	Perform format changes in an include.
#PFIG	Perform interactive German hyphenation.
#PFII	Perform interactive L/A instruction parser.
#PFIL	Perform interactive line adjustment during edit.
#PFIN	Perform include processing.
#PFIP	Perform interactive L/A attribute propagate.
#PFIX	Perform include transformations.
#PFLA	Perform output line adjust.
#PFLO	Display first two of the three LABEL PRINT OPTIONS displays.
#PFLP	Merge labels into a composite document.
#PFL3	Display third of the three LABEL PRINT OPTIONS displays.
#PFMD	Perform merge from a document.
#PFMG	Perform the merge function.
#PFMO	Display merge options.
#PFMP	Perform the merge initialization function.
#PFNT	Perform PS/36 note processing.
#PFOC	Perform the output L/A copy function.

Module Name	Function
#PFOE	Perform output L/A error logging.
#PFOG	Perform output L/A German hyphenation.
#PFOH	Perform resolution of outline heading and outline heading text instructions.
#PFOP	Perform output L/A attribute propagation.
#PFPD	Display the 6580 DISPLAYWRITER OPTIONS display.
#PFPG	Display the SAVE OPTIONS display.
#PFPH	TEXTREL procedure handler.
#PFPI	Display the 6670 PRINT OPTIONS display.
#PFPL	Perform pagination and line adjustment of printed document and control the flow of print resolution/pagination/line adjustment.
#PFPM	Perform print function mainline routing.
#PFPO	Display first two of the three PRINT OPTIONS displays.
#PFPP	Perform output pagination.
#PFPQ	Interface to query to resolve data base markup during print.
#PFPR	Resolve markup that is already resolved to text.
#PFPS	Perform print resolution subroutines.
#PFPW	Perform widow line processing for output.
#PFPX	Perform #PFPM extra subroutines.
#PFP3	Display the third of the three PRINT OPTIONS displays.
#PFQM	Spool/JOBQ manager.
#PFRE	Remove spelling errors from document.
#PFRO	Display the two PRINT OPTIONS displays for printing resolved documents.
#PFSL	Perform pagination and line adjustment of a source document.
#PFSM	Resolve summary math instructions.
#PFSP	Perform source pagination.
#PFTB	Resolve tabs for data/text and math.
#PFTC	Build the table of contents page.
#PFTU	Perform parse instructions.
#PFXE	Log errors to the error page.
#PFXF	Interface to print transformations and evoke procedures.

Module Name	Function
#TUAA	Add to addenda dictionary.
#TUAD	Create/delete work library.
#TUAV	Process average instruction.
#TUBK	Process begin keep.
#TUBO	Process begin overstrike.
#TUBR	Process spell box input.
#TUBX	Build box on AID request.
#TUCE	Perform character field editing.
#TUCF	Process change format.
#TUCK	Process command keys for #TUEC.
#TUCL	Process column copy, move, or delete.
#TUCN	Process comment.
#TUCO	Process color instruction.
#TUCP	Clear pending commands.
#TUCR	Process CR/RCR aids.
#TUCV	Process count statement.
#TUDA	Process date statement.
#TUDD	Create or revise document description.
#TUDE	Execute document requests.
#TUDF	Process data file options.
#TUDI	Process document ID statement.
#TUDL	Perform document selection.
#TUDN	Rename document.
#TUDO	Process document options.
#TUDP	Print document list.
#TUDR	Delete document.
#TUDT	Process data field instructions menus.
#TUDV	Process document variable.
#TUDW	Edit document list.
#TUDX	Execute document requests-extended.
#TUDY	Copy document.

Module Name	Function
#TUEA	Process for spelling aid.
#TUEB	Build edit text lines.
#TUEC	Execute command requests.
#TUED	Edit display.
#TUEF	Find options.
#TUEG	Get options.
#TUEK	Build command key template.
#TUEL	Build format scale lines.
#TUEM	Format change lines.
#TUEP	Process pending line functions.
#TUET	Set up for text delete with WSC.
#TUEV	Perform spelling verification.
#TUEX	Execute command keys.
#TUEY	Process synonyms.
#TUFH	Process data field heading.
#TUFM	Move, copy, or delete format.
#TUFR	Find or replace.
#TUFW	Find word.
#TUGA	Perform Germanic spelling aid.
#TUGF	Change font.
#TUGG	Get graphic file.
#TUGH	Perform Germanic hyphenation.
#TUGL	Perform grade level analysis.
#TUGO	Go to another list module.
#TUGR	Process graphic instruction.
#TUGV	Perform Germanic verification.
#TUGZ	Perform Germanic hyphenation.
#TUHC	Outline heading definition.
#TUHF	Process header or footer.
#TUHI	Process instructions for header or footer.
#TUHP	Process help tutorial.
#TUHT	Process outline heading text.
#TUHV	Process high instruction.
#TUHX	Process outline heading.

Module Name	Function
#TUHY	Perform hyphenation.
#TUIE	Process index entry instruction.
#TUIF	Process begin conditional text.
#TUIH	Process CMD11 to insert hyphens.
#TUIK	Process instruction key interface.
#TUIN	Process include statement.
#TUIPL	Perform DW/36 time initialization.
#TUIS	Perform spelling workspace initialization.
#TUIT	Build translate tables.
#TUIX	Process index instruction.
#TULC	Process locate statement.
#TULD	Process text library description.
#TULF	Process line format options.
#TULL	Perform text library select.
#TULN	Process line commands.
#TULV	Process low instruction.
#TUMF	Perform master format selection.
#TUMK	Perform spelling verification.
#TUMN	Perform text library maintenance.
#TUMT	Process for margins and tabs.
#TUNE	Perform numeric field editing.
#TUNF	Change format.
#TUNL	Process numbered list.
#TUNM	Process text month names.
#TUNS	Select subset list.
#TUOE	Perform object execution and maintenance.
#TUOL	Perform object selection.
#TUOW	Select object list.
#TUPA	Process page statement.
#TUPC	Process position line commands.
#TUPE	Process page end statement.
#TUPF	Perform page format definition.
#TUPG	Perform pagination.

Module Name	Function
#TUPH	Process procedure.
#TUPK	Process print key options.
#TUPN	Process page number statement.
#TUPQ	Display the WORK WITH DOCUMENTS display.
#TURA	Revise supplemental dictionary.
#TURB	Process required backspace statement.
#TURE	Reset errors.
#TURH	Process running head statement.
#TURP	Perform replace for find/replace.
#TURT	Perform root phase processing.
#TURV	Perform recovery.
#TUSA	Process for special aids.
#TUSK	Process skip command.
#TUSM	Process summary math instructions.
#TUSN	Process system page number command.
#TUSO	Process spelling options.
#TUSP	Display the SPELLING OPTIONS display.
#TUSR	Perform DW/36 common subroutines.
#TUSV	Check words.
#TUTA	Process for text aids.
#TUTB	Process table of contents.
#TUTC	Process table of contents menu.
#TUTD	Process text delete.
#TUTE	Process time command.
#TUTF	Process copy of a format.
#TUTG	Process help text label.
#TUTI	Process text body instructions.
#TUTN	Process text delete.
#TUTO	Process text operation complete.
#TUTS	Process for text shift.
#TUTV	Process total instruction.
#TUTX	Move, copy, or delete text.
#TUT1	Process text copy.

Module Name	Function
#TUUA	Update supplemental dictionary.
#TUUE	Update page end.
#TUUL	Update lines.
#TUUM	Process prompt line.
#TUUP	Update edit display.
#TUUR	Update for remote work station.
#TTUS	Perform interactive spell check.
#TUVN	Process data field instruction.
#TUVW	Process view document command.
#TUWD	Process work station description.
#TUWI	Process work station initial.
#TUWL	Process wide lines.
#TUZI	Process zero index command.

Development Support Utility Directory

Module Name	Function
#EDED	Perform edit functions.
#EDEJ	Perform exit functions.
#EDFC	Perform find/change operations.
#EDFD	Perform find operation.
#EDIN	Perform include operation.
#EDIS	Perform session initialization.
#EDIT	Perform edit initialization.
#EDL1	Check all line commands entered for errors.
#EDL2	Execute all line commands.
#EDML	Member list processor.
#EDRA	Syntax checker for RPG II specifications.
#EDRC	Syntax checker for RPG II specifications.
#EDRC	Syntax checker for RPG II specifications.
#EDRD	Syntax checker for RPG II specifications.
#EDRE	Syntax checker for RPG II specifications.
#EDRF	Syntax checker for RPG II specifications.
#EDRI	Syntax checker for RPG II specifications.
#EDRO	Syntax checker for RPG II specifications.
#EDRT	Syntax checker for RPG II specifications.
#EDVW	Perform all view processing except find operations.

A

B

C

D

E

F

G

H

J

K

L

M

N

O

T

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name _____

Company or Organization _____

Address _____

_____
City State Zip Code

Phone No. _____
Area Code

No postage necessary if mailed in the U.S.A.

Fold and tape. **Please do not staple.**

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 / ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and tape. **Please do not staple.**

IBM

LY21-0590-4

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name _____

Company or Organization _____

Address _____

_____
City State Zip Code

Phone No. _____
Area Code

No postage necessary if mailed in the U.S.A.

Fold and tape. **Please do not staple.**

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 / ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and tape. **Please do not staple.**

IBM

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name _____

Company or Organization _____

Address _____

_____
City State Zip Code

Phone No. _____
Area Code

No postage necessary if mailed in the U.S.A.

Fold and tape. **Please do not staple.**

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 / ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and tape. **Please do not staple.**

IBM

International Business Machines Corporation

THIS DOCUMENT CONTAINS RESTRICTED MATERIALS OF IBM CORPORATION.

Program Service Information

Contents

File Number
S36-36

Order Number
LY21-0590-4

Printed in U.S.A.