LY20-2339-2

Time Sharing Option 3270 Display Support and Structured Programming Facility Version 2.2: Program Logic Manual

Program Product

Program Number 5740-XT8

The Structured Programming Facility/Time Sharing Option is a program development tool designed to take advantage of the characteristics of IBM 3270 display terminals and to increase productivity in the Time Sharing Option environment for users of both structured and conventional programming techniques.

This document describes the internal logic, program structure, and data areas. It is intended for those who change and maintain this program product.

IBM

Third Edition (October 1979)

This edition is a major revision obsoleting LY20-2339-1.

This edition applies to Version 2, Release 2, Modification Level 1, of the program product TSO-3270 Display Support and Structured Programming Facility (5740-XT8) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein. Therefore, before using this publication, consult your System/370 Bibliography (GC20-0370) for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available outside the United States.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments has been provided at the back of this publication. If this form has been removed, address comments to : IBM Corporation, Technical Publications, Dept. 824, 1133 Westchester Avenue, White Plains, New York 10604. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

PREFACE

THIS MANUAL DESCRIBES THE FUNCTIONS AND LOGIC OF INTERNAL OPERATIONS, PROGRAM STRUCTURE AND CONTROL FLOW, DATA FLOW, AND DATA AREAS OF SPF/TSO VERSION 2, RELEASE 2, MODIFICATION LEVEL 1 (SPF 2.2.1). IT IS INTENDED FOR SYSTEMS PROGRAMMING PERSONNEL WHO NEED TO MODIFY THE DISTRIBUTED VERSION OF THE PROGRAM.

THE MANUAL IS ORGANIZED INTO FIVE MAJOR SECTIONS, AS FOLLOWS:

SECTION 1. INTRODUCTION -
INCLUDES GENERAL INFORMATION ABOUT THE SPF OPERATING ENVIRONMENT, PHYSICAL CHARACTERISTICS, TASK STRUCTURE, AND SPF DATA SETS.

SECTION 2. METHOD OF OPERATION -
DESCRIBES THE LOGIC AND DATA FLOW IN SUFFICIENT DETAIL FOR THE READER TO BE ABLE TO IDENTIFY THE SPF MODULE WHICH PERFORMS A PARTICULAR OPERATION.

SECTION 3. PROGRAM ORGANIZATION -
DESCRIBES THE OVERALL PROGRAM STRUCTURE AND LISTS PERTINENT INFORMATION ABOUT EACH MODULE, INCLUDING INTERFACE REQUIREMENTS.

SECTION 4. DIRECTORY -
CONTAINS CROSS-REFERENCE INFORMATION BY OBJECT MODULE, SUMMARIZING USAGE OF SPF COMMON SUBROUTINES, MENUS, MESSAGES, AND SVC ROUTINES.

SECTION 5. DATA AREAS -
SHOWS THE FORMAT AND CONTENTS OF MAJOR SPF TABLES AND CONTROL BLOCKS.

RELATED PUBLICATIONS

THE READER SHOULD BE FAMILIAR WITH THE FOLLOWING PUBLICATIONS:

SPF/TSO PROGRAM REFERENCE MANUAL, SH20-1975-2

THIS MANUAL PROVIDES A DETAILED DESCRIPTION ON HOW TO USE SPF. THE MANUAL INCLUDES SEVERAL EXAMPLES OF SPF USAGE AND A SAMPLE PROBLEM.

SPF/TSO INSTALLATION AND CUSTOMIZATION GUIDE, SH20-2402-0

THIS MANUAL PROVIDES INFORMATION ON HOW TO INSTALL AND CUSTOM TAILOR SPF.

FOR ADDITIONAL DETAIL ABOUT SPF PROGRAMS AND DATA AREAS, THE READER SHOULD CONSULT THE PL/S COMPILER AND ASSEMBLER OUTPUT LISTINGS WHICH ARE AVAILABLE ON MICROFICHE (LYB0-2481-0).

CONTENTS

SECTION 1

INTRODUCTION

THIS SECTION CONTAINS GENERAL INFORMATION ABOUT THE SPF OPERATING ENVIRONMENT, PHYSICAL CHARACTERISTICS, TASK STRUCTURE, AND DATA SETS. THE SECTION IS ORGANIZED AS FOLLOWS:

PROGRAM OVERVIEW
PROGRAMMING SYSTEMS
MACHINE CONFIGURATIONS
TERMINALS
PHYSICAL CHARACTERISTICS
TASK STRUCTURE
DATA SET USAGE
DATA SET NAMES
DATA SET ATTRIBUTES

PROGRAM OVERVIEW

SPF IS A PROGRAMMING AID THAT OPERATES IN THE TIME SHARING OPTION (TSO) ENVIRONMENT AND IS DESIGNED TO INCREASE PRODUCTIVITY IN DEVELOPING AND MODIFYING PROGRAMS.

SPF SUPPORTS BOTH STRUCTURED AND CONVENTIONAL PROGRAMMING TECHNIQUES. IT CAN BE USED EITHER BY AN INDIVIDUAL PROGRAMMER, OR BY MANY PROGRAMMERS WORKING TOGETHER ON A PROJECT. THE PRIMARY FUNCTIONS THAT IT PROVIDES INCLUDE:

-- FULL SCREEN, CONTEXT EDITING WHICH ALLOWS MULTIPLE LINES TO BE MODIFIED IN A SINGLE INTERACTION.

-- FORWARD, BACKWARD, AND SIDEWAYS SCROLLING OF SOURCE CODE OR OUTPUT LISTINGS.

-- SPLIT SCREEN, ALLOWING TWO SPF FUNCTIONS TO BE PERFORMED INDEPENDENTLY ON THE SAME DISPLAY TERMINAL.

-- ALLOCATION AND MAINTENANCE OF PROGRAMMING LIBRARIES, AUTOMATIC COLLECTION OF LIBRARY ACTIVITY STATISTICS, AND PRINTING OF LIBRARY CONTENTS.

-- INTERFACE WITH STANDARD LANGUAGE PROCESSORS FOR EXECUTION IN THE FOREGROUND OR BACKGROUND.

-- ONLINE TUTORIAL FOR INSTRUCTION AND REFERENCE.

PROGRAMMING SYSTEMS

SPF OPERATES AS A TSO COMMAND PROCESSOR UNDER THE TIME SHARING OPTION OF VS2 RELEASE 1.7 (SVS), OR VS2 RELEASE 3.7 OR 3.8 (MVS). SPF IS WRITTEN IN PL/S AND TRANSLATED INTO OS/VS ASSEMBLER LANGUAGE. THE BPAM AND BSAM ACCESS METHODS ARE EMPLOYED BY SPF FOR READING AND WRITING DATA SETS, AND THE FACILITIES OF TSO/TCAM OR TSO/VTAM ARE USED FOR READING AND WRITING THE DISPLAY.

SPF PROVIDES INTERFACES TO THE FOLLOWING IBM PRODUCTS:

SYSTEM ASSEMBLER (SUPPLIED WITH OS/VS2)	
OS/VS COBOL COMPILER	5740-CB1
FORTRAN IV G1 COMPILER	5734-F02
PL/I CHECKOUT COMPILER	5734-PL2
PL/I OPTIMIZING COMPILER	5734-PL1
LINKAGE EDITOR (SUPPLIED WITH OS/VS2)	
COBOL INTERACTIVE DEBUG (FOREGROUND ONLY)	5734-CB4
FORTRAN INTERACTIVE DEBUG (FOREGROUND ONLY	5734-F05
TSO ASSEMBLER PROMPTER (FOREGROUND ONLY)	5734-CP2
TSO COBOL PROMPTER (FOREGROUND ONLY)	5734-CP1
TSO FORTRAN PROMPTER (FOREGROUND ONLY)	5734-CP3
DOCUMENT COMPOSITION FACILITY (SCRIPT/VS) WITH THE FOREGROUND ENVIRONMENT FEATURE	5748-XX9
OS/VS2 MVS 3270 EXTENDED DISPLAY SUPPORT - SESSION MANAGER, RELEASE 2	5740-XE2
TSO/TCAM COMMAND PROCESSOR "DSPRINT"	5798-AYF
TSO/VTAM DATA SET PRINT (DSPRINT)	5798-CPF
TSO/VS2 PROGRAMMING CONTROL FACILITY (PCF)	5798-BBJ
TSO PROGRAMMING CONTROL FACILITY - II (PCF2)	5798-CLW

ALL THE PROGRAM-NUMBERED PRODUCTS LISTED ABOVE CAN BE ORDERED SEPARATELY UNDER IBM LICENSING AGREEMENTS. NONE OF THE ABOVE PRODUCTS ARE DISTRIBUTED AS PART OF SPF.

MACHINE CONFIGURATION

THE COMPUTER SYSTEM REQUIREMENTS ARE THE SAME AS NEEDED FOR OS/VS2 WITH THE TIME SHARING OPTION (TSO).

THE STORAGE REQUIREMENTS FOR THE USER REGIONS WILL VARY DEPENDING UPON THE SIZE OF THE DATA SETS BEING EDITED AND THE EXTENT THAT "SPLIT SCREEN" WILL BE USED. THE SPF PROGRAMS ARE REENTERABLE AND SHOULD BE PLACED IN THE SYSTEM LINK PACK AREA. THIS WILL REDUCE THE SIZE REQUIREMENT FOR THE USER REGIONS AND SHOULD ALSO IMPROVE PERFORMANCE.

THE FOLLOWING MINIMUM REGION SIZES ARE SUGGESTED FOR SVS. THESE SIZES MAY HAVE TO BE EXPANDED IF LARGE CODE SEGMENTS ARE TO BE EDITED.

256K - IF SPF RESIDES IN THE LINK-PACK AREA
512K - IF SPF DOES NOT RESIDE IN THE LINK-PACK AREA

TERMINALS

SPF SUPPORTS THE FOLLOWING IBM 3270 DISPLAY STATIONS:

3275 MODELS 2 AND 12
3276 MODELS 2, 3, 4, 12, 13, AND 14
3277 MODEL 2 (LOCAL OR REMOTE ATTACHMENT)
3278 MODELS 2, 3, AND 4 (LOCAL OR REMOTE ATTACHMENT)

THE FOLLOWING KEYBOARDS ARE SUPPORTED:

FOR 3275 OR 3277 DISPLAY STATIONS:

78 KEY OPERATOR CONSOLE (FEATURE 4632)
78 KEY EBCDIC TYPEWRITER (FEATURE 4633)
78 KEY ASCII TYPEWRITER (FEATURE 4635)
78 KEY EBCDIC TYPEWRITER/APL (FEATURE 4638), APL SWITCH OFF

FOR 3276 OR 3278 DISPLAY STATIONS:

75 KEY EBCDIC TYPEWRITER (FEATURE 4621)
75 KEY ASCII TYPEWRITER (FEATURE 4624)
87 KEY EBCDIC TYPEWRITER (FEATURE 4627)
87 KEY ASCII TYPEWRITER (FEATURE 4628)
87 KEY EBCDIC TYPEWRITER/APL (FEATURE 4626), APL SWITCH OFF
87 KEY EBCDIC TYPEWRITER/TEXT (FEATURE 4629), TEXT SWITCH OFF

THE STANDARD CHARACTER SET (94 GRAPHICS PLUS BLANK AND NULL) IS SUPPORTED ON 3276 AND 3278 DISPLAY STATIONS.

THE FOLLOWING ARE SUPPORTED, BUT NOT REQUIRED:

AUDIBLE ALARM (FEATURE #1090)
IBM 3284, 3286, 3287, 3288, AND 3289 PRINTERS
PRINT DUAL-CASE CHARACTER SET (RPQ #8K0366)

THE IBM 3284, 3286, 3287, 3288, AND 3289 PRINTERS, IF USED, ARE SUPPORTED VIA THE "DSPRINT" TSO COMMAND PROCESSOR, WHICH MUST BE INSTALLED IF SPF OUTPUT IS DIRECTED TO ONE OF THESE PRINTERS.

PHYSICAL CHARACTERISTICS

SPF IS COMPRISED OF 30 REENTERABLE LOAD MODULES AND 1 NON-REENTERABLE LOAD MODULE. THESE LOAD MODULES ARE BUILT FROM 156 OBJECT MODULES. THE OBJECT MODULES ARE ARE DISTRIBUTED ON TAPE IN SMP INSTALLABLE FORMAT AND MUST BE LINK EDITED TO CREATE THEIR RESPECTIVE LOAD MODULES. IT IS RECOMMENDED THAT MOST OF THE LOAD MODULES BE COPIED TO THE SYSTEM LINK PACK AREA (DATA SET 'SYS1.LPALIB'). SEE THE INSTALLATION AND CUSTOMIZATION GUIDE FOR MORE INFORMATION.

TASK STRUCTURE

THE SPF TASK STRUCTURE IS SHOWN IN FIGURE 1.1. THE SPF MAIN CONTROLLER IS ATTACHED BY TSO WHENEVER THE USER ENTERS THE "SPF" COMMAND. THE MAIN CONTROLLER PERFORMS INITIALIZATION/TERMINATION FUNCTIONS AND HANDLES DISPLAY I/O (VIA TCAM OR VTAM) ON BEHALF OF THE OTHER SPF PROGRAMS. IT ATTACHES THE PROCESSOR MAIN DRIVER, WHICH DISPLAYS THE PRIMARY OPTION MENU AND LINKS TO ONE OF SEVERAL PROCESSING PROGRAMS, DEPENDING ON THE OPTION SELECTED.

IF THE USER ENTERS SPLIT SCREEN MODE, THE MAIN CONTROLLER AGAIN ATTACHES THE PROCESSOR MAIN DRIVER TO HANDLE OPERATIONS ON THE SECOND LOGICAL SCREEN. THUS, IN SPLIT SCREEN MODE THREE TASKS ARE ACTIVE:

SPF CONTROL TASK
SPF PROCESSING TASK FOR LOGICAL SCREEN 1
SPF PROCESSING TASK FOR LOGICAL SCREEN 2

THE WAIT/POST LOGIC BETWEEN THE CONTROL TASK AND EITHER OF THE PROCESSING TASKS IS SHOWN IN FIGURE 1.2. AFTER THE CONTROL TASK ATTACHES A PROCESSING TASK, IT WAITS FOR EITHER OF TWO EVENT CONTROL BLOCKS (ECB'S) TO BE POSTED:

1. DISPLAY REQUEST ECB
2. TASK COMPLETION ECB

WHEN A PROCESSING TASK WANTS THE CONTROL TASK TO PERFORM DISPLAY I/O OR SOME OTHER FUNCTION, IT POSTS THE DISPLAY REQUEST ECB (ECB #1) WITH ONE OF THREE SPECIAL CODES AND ISSUES A WAIT ON:

3. PROCESS REQUEST ECB

THE THREE TYPES OF DISPLAY REQUESTS ARE:

1. NORMAL FULL SCREEN I/O - THIS REQUEST IS USED WHEN A PROCESSING TASK WANTS TO DO NORMAL SPF DISPLAY I/O.

   IN RESPONSE TO A DISPLAY REQUEST, THE CONTROL TASK:
   - WRITES INFORMATION TO THE TERMINAL (TPUT),
   - WAITS FOR A RESPONSE FROM THE USER (TGET),
   - POSTS THE PROCESS REQUEST ECB (ECB #3), AND
   - WAITS FOR DISPLAY REQUEST OR COMPLETION OF THE PROCESSING TASK.

   PROCESSING PROGRAMS GENERALLY CALL THE COMMON DISPLAY SUBROUTINE, CDISPL, TO PERFORM THE POST-WAIT SEQUENCE.

2. LINE I/O - THIS REQUEST IS USED WHEN A PROCESSING TASK WANTS TO ENTER STANDARD TSO LINE I/O MODE.

   IN RESPONSE TO A LINE I/O REQUEST, THE CONTROL TASK:
   - CLEARS PART OF THE SCREEN AND SETS THE LINE COUNT,
   - POSTS THE PROCESS REQUEST ECB (ECB #3), AND
   - WAITS FOR DISPLAY REQUEST OR COMPLETION OF THE PROCESSING TASK.

3. COMMON CONTROL INTERFACE - THIS REQUEST IS USED WHEN A PROCESSING TASK WANTS TO EXECUTE A COMMON SUBROUTINE UNDER THE CONTROL TASK'S TCB.

   IN RESPONSE TO A COMMON CONTROL INTERFACE REQUEST, THE CONTROL TASK:
   - CALLS THE COMMON SUBROUTINE,
   - POSTS THE PROCESS REQUEST ECB (ECB #3), AND
   - WAITS FOR DISPLAY REQUEST OR COMPLETION OF THE PROCESSING TASK.

WHEN A PROCESSOR TASK COMPLETES (EITHER NORMALLY VIA END FUNCTION REQUEST, OR ABNORMALLY DUE TO AN ABEND), THE TASK COMPLETION ECB IS POSTED (ECB #2). FOR ABNORMAL COMPLETION, THE CONTROLLER RE-ATTACHES THE PROCESSOR TASK. FOR NORMAL COMPLETION, THE CONTROLLER TERMINATES SPLIT SCREEN MODE (IF IT WAS IN EFFECT), OR TERMINATES SPF.

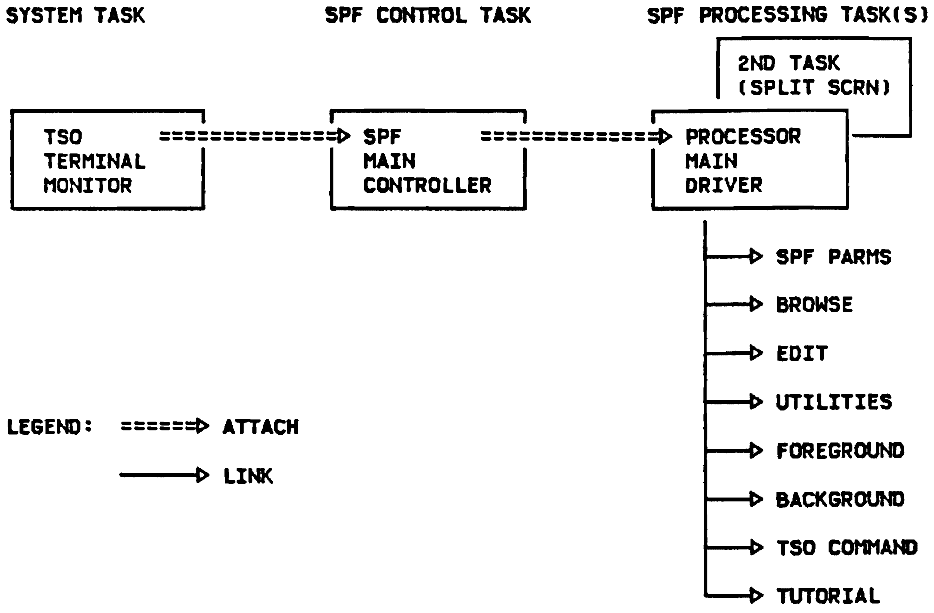


FIGURE 1.1 SPF TASK STRUCTURE

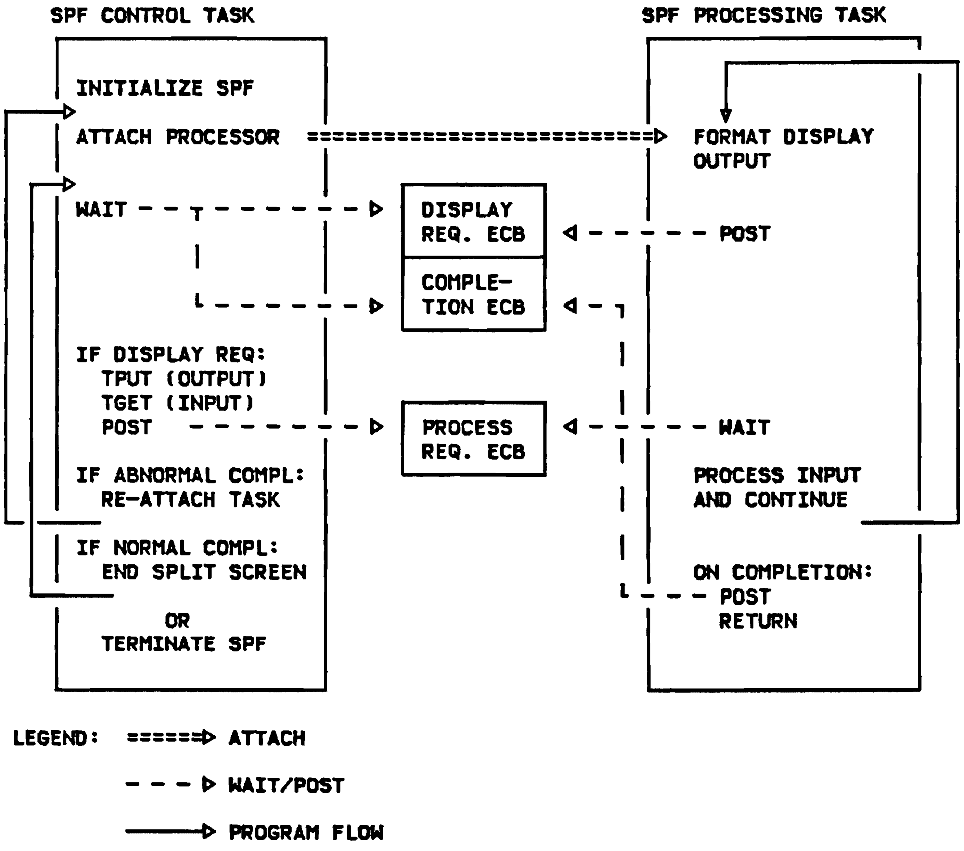


FIGURE 1.2 WAIT/POST LOGIC

DATA SET USAGE

THE DATA SETS USED BY SPF ARE SHOWN IN FIGURE 1.3. SOME OF THE DATA SETS ARE PERMANENT DATA SETS THAT ARE REQUIRED TO EXECUTE SPF. OTHERS ARE TEMPORARY DATA SETS THAT ARE ALLOCATED BY SPF IF THEY ARE REQUIRED.

THE NAMES SHOWN ARE THE NAMES USED TO REFER TO THESE DATA SETS FOR DOCUMENTATION PURPOSES. "TEMPCNTL" IS SOMETIMES USED GENERICALLY TO REFER TO EITHER TEMPCNTL1 OR TEMPCNTL2. LIKEWISE, "TEMPLIST" REFERS TO EITHER TEMPLIST1 OR TEMPLIST2. ACTUAL DATA SET NAMES AND THE FILE NAMES TO WHICH THEY ARE ASSIGNED ARE DISCUSSED LATER IN THIS SECTION.

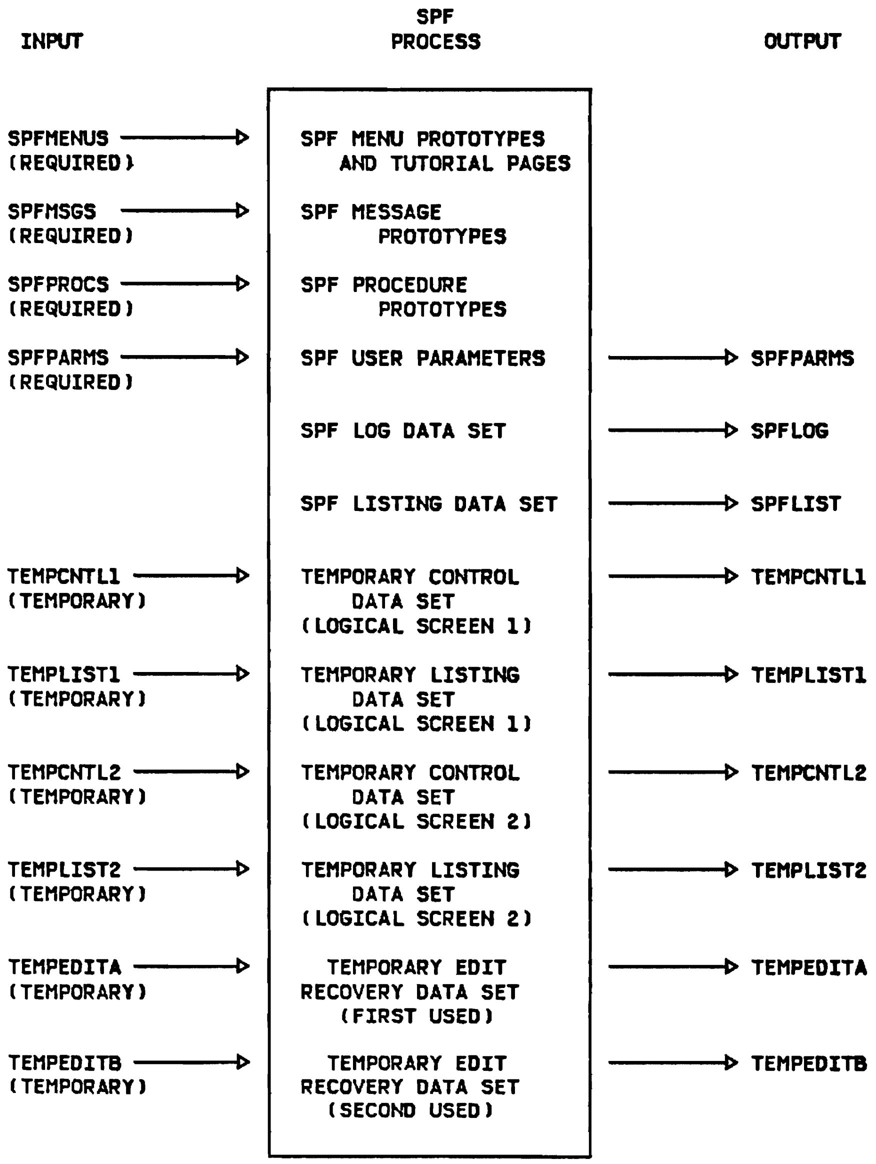


FIGURE 1.3 SPF DATA SET USAGE

SPFMENUS: THIS DATA SET CONTAINS PROTOTYPE MENUS THAT ARE FORMATTED FOR THE DISPLAY SCREEN. MENUS CAN BE EASILY EDITED, MAKING IT POSSIBLE TO CHANGE THE FORMAT OF A DISPLAY WITHOUT PROGRAM CHANGES. INSTALLATIONS MAY ALSO WANT TO CHANGE EXISTING MENUS OR ADD NEW MENUS TO BE USED BY FOREGROUND (OPTION 4) OR BACKGROUND (OPTION 5).

SPFMSGS: THIS DATA SET CONTAINS PROTOTYPE MESSAGES THAT ARE FORMATTED AND USED PRIMARILY AS ERROR, INFORMATION, OR LOG MESSAGES. EACH MESSAGE CONSISTS OF A SHORT PORTION (24 CHARACTERS) THAT CAN BE DISPLAYED IN THE UPPER RIGHT HAND CORNER OF THE SCREEN, AND A LONG PORTION (72 CHARACTERS) THAT CAN BE DISPLAYED ON LINE 3 OF THE DISPLAY.

SPFPROCS: THIS DATA SET CONTAINS PROTOTYPE COMMANDS AND JCL. THE PROTOTYPE COMMANDS ARE USED BY THE FOREGROUND OPTION (OPTION 4) IN BUILDING TSO COMMANDS OR CLISTS. PROTOTYPE JCL IS USED BY THE BACKGROUND OPTION (OPTION 5) IN BUILDING JOB STREAMS FOR BACKGROUND EXECUTION.

SPFPARMS: THIS DATA SET CONTAINS SPF USER INFORMATION TO BE RETAINED FROM ONE SESSION TO ANOTHER. IT CONSISTS OF ONE MEMBER FOR EACH SPF USER. WHEN A TSO USER FIRST USES SPF, A NEW MEMBER IS AUTOMATICALLY CREATED. THEREAFTER, IT IS READ DURING SPF INITIALIZATION AND UPDATED DURING SPF TERMINATION.

SPFLOG: THIS DATA SET CONTAINS LOG MESSAGES WHICH SHOW SIGNIFICANT ACTIONS DURING THE SESSION. THE USER CAN CONTROL WHETHER OR NOT A LOG DATA SET IS TO BE CREATED (OPTION 0.2). SETTING THE PRIMARY ALLOCATION TO 0 WILL ELIMINATE THE OVERHEAD OF ALLOCATING, WRITING, AND FREEING THE LOG DATA SET.

SPFLIST: THIS DATA SET CONTAINS FORMATTED PRINTER OUTPUT REQUESTED BY THE USER. ALLOCATION OF A LIST DATA SET CAN BE PREVENTED BY AVOIDING THE FOLLOWING OPTIONS THAT WRITE TO SPFLIST:

- PRINT SCREEN IMAGE PF KEY.
- EDIT AUTOMATIC PRINT (EDIT COMMAND)
- PRINT INDEX LISTING (OPTION 3.1 X)
- PRINT ENTIRE DATA SET (OPTION 3.1 L)
- PRINT MEMBER (OPTION 3.1 P)
- PRINT CATALOG ENTRIES (OPTION 3.4 P)
- PRINT VTOC ENTRIES (OPTION 3.7 P)

TEMPCNTL: THIS TEMPORARY DATA SET CONTAINS CONTROL CARD IMAGES, OR UTILITY OUTPUT WHICH HAS BEEN FORMATTED FOR DISPLAY. IT IS ALLOCATED THE FIRST TIME IT IS REQUIRED, AND DELETED WHEN A LOGICAL SCREEN TERMINATES. OPTIONS THAT USE TEMPCNTL ARE:

- EDIT (SUBMIT COMMAND) FOR JCL TO BE SUBMITTED
- BACKGROUND (OPTION 4) FOR JCL TO BE SUBMITTED
- HARDCOPY UTILITY (OPTION 3.6 J) FOR JCL TO BE SUBMITTED
- CATALOG UTILITY (OPTION 3.4) FOR IEHLIST OR IDCAMS CONTROL CARDS
- VTOC UTILITY (OPTION 3.7) FOR DATA TO BE DISPLAYED
- TERMINATION (FINAL MENU) FOR JCL TO BE SUBMITTED

TEMPLIST: THIS TEMPORARY DATA SET CONTAINS LISTINGS THAT HAVE BEEN GENERATED BY OS UTILITIES, INVOKED BY SPF. THE LISTINGS ARE DISPLAYED ON THE SCREEN. IT IS ALLOCATED THE FIRST TIME IT IS REQUIRED, AND DELETED WHEN A LOGICAL SCREEN TERMINATES. THE FUNCTION THAT USES TEMPLIST IS:

- CATALOG UTILITY (OPTION 3.4) FOR IEHLIST OR IDCAMS OUTPUT LISTINGS TO BE DISPLAYED

TEMPEDIT: THIS TEMPORARY DATA SET CONTAINS DATA THAT IS SAVED DURING AN EDIT SESSION WHEN YOU HAVE "RECOVERY ON" SPECIFIED. IT IS ALLOCATED THE FIRST TIME YOU CHANGE DATA IN AN EDIT SESSION. IT REMAINS ALLOCATED UNTIL YOU TERMINATE SPF. THERE ARE TWO DATA SETS ("A" AND "B") WHICH ALLOWS YOU TO DO RECOVERY PROCESSING IN SPLIT SCREEN MODE.

DATA SET NAMES

THE DEFAULT NAMES THAT ARE USED BY SPF FOR THE VARIOUS DATA SETS ARE:

```
SPFMENUS  -  'SPF22.MOD1.MENUS'    --
SPFMSGS   -  'SPF22.MOD1.MSGS'       |- THE NAMES SHOWN ARE THE
SPFPROCS  -  'SPF22.MOD1.PROCS'      |  FULLY QUALIFIED DATA SET NAMES.
SPFPARMS  -  'SPF22.MOD1.PARMS'    --

SPFLOG    -   SPFLOG1.LIST    (1)  --
SPFLIST   -   SPF1.LIST       (1)    |- THE NAMES SHOWN ARE QUALIFIED
TEMPCNTL  -   SPFTEMP1.CNTL   (2)    |  WITH THE TSO USERID OR
TEMPLIST  -   SPFTEMP1.LIST   (2)    |  PREFIX.USERID
TEMPEDIT  -   SPFEDITA.BACKUP (3)  --
```

(1) THESE DATA SETS MAY HAVE NUMBERS OTHER THAN "1". FOR EXAMPLE: SPFLOG - SPFLOG4.LIST, SPFLIST - SPF8.LIST

(2) THESE DATA SETS WILL HAVE NUMBERS "1" FOR LOGICAL SCREEN 1 AND "2" FOR LOGICAL SCREEN 2. FOR EXAMPLE: TEMPCNTL - SPFTEMP2.CNTL

(3) THESE DATA SETS WILL HAVE CHARACTERS "A" FOR FIRST USE AND "B" FOR SECOND USE. FOR EXAMPLE: TEMPEDIT - SPFEDITA.BACKUP

DATA SET ATTRIBUTES

THE DEFAULT ATTRIBUTES THAT ARE DISTRIBUTED ARE SHOWN BELOW. SOME OF THE VALUES ARE REQUIRED AS SHOWN. OTHER VALUES CAN BE CHANGED AT YOUR INSTALLATION BY SUPERZAPING THE TSV. STILL OTHERS CAN BE CHANGED BY INDIVIDUAL USERS BY USING OPTION 0.2 (LOG/LIST DEFAULT MENU).

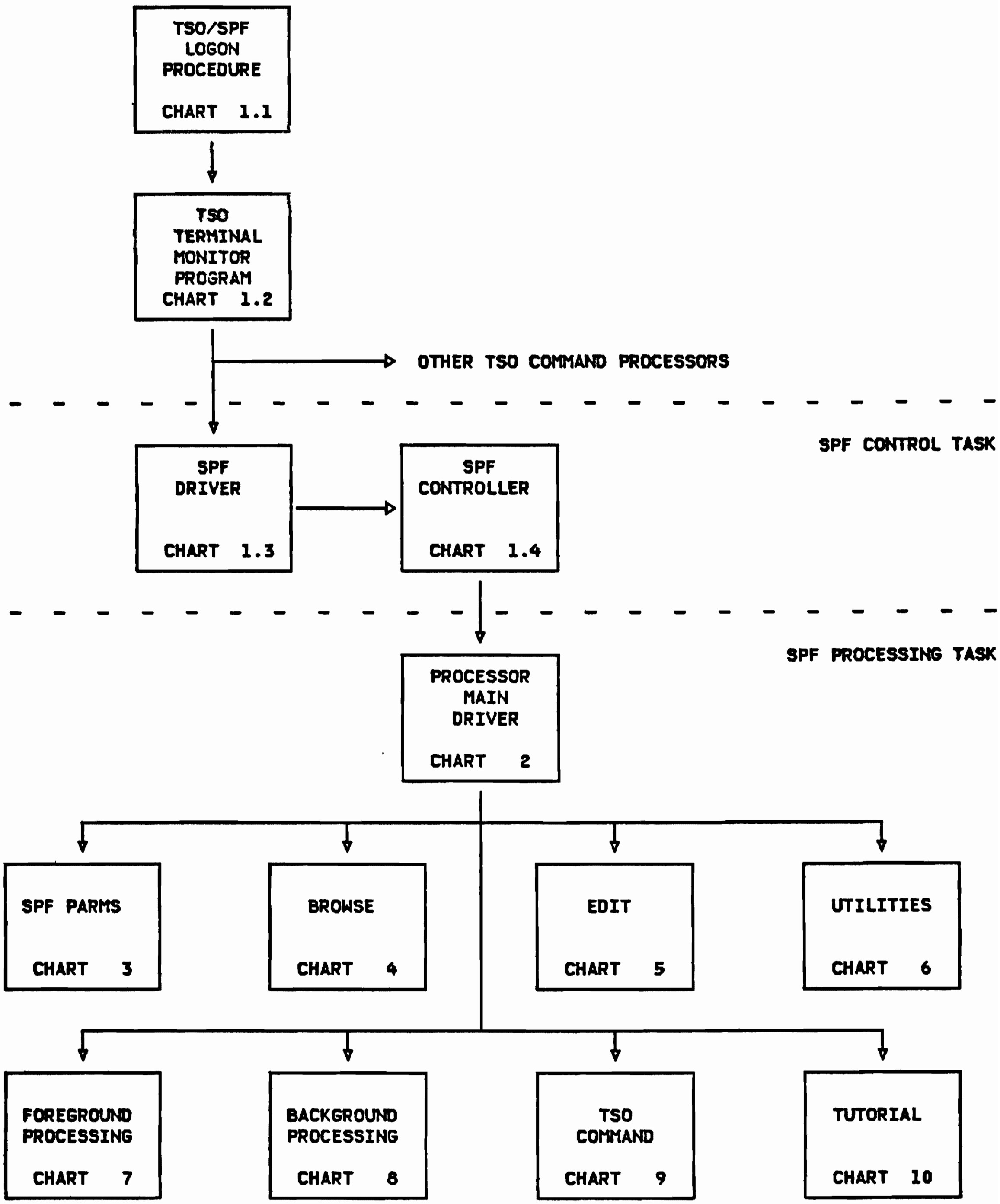
FOR SPF					ALLOCATION PARAMETERS		
DATA SET	FILE NAME	LRECL	BLKSIZE	RECFM	UNITS	PRIMARY	SECONDARY
SPFMENUS	SPFMENUS	84	3120	VB	BLKS	300	40
SPFMSGS	SPFMSGS	76	3120	VB	BLKS	36	8
SPFPROCS	SPFPROCS	80	3120	FB	BLKS	32	6
SPFPARMS	SPFPARMS	6000(1)	6000(1)	F	BLKS	**(2)	**(2)
SPFLOG	SPFNNNNN(3)	125	129	VA	PAGES	10	10
SPFLIST	SPFNNNNN(3)	121	3146	FBA	PAGES	100	200
TEMPCNTL	SPFNNNNN(3)	80	800	FB	BLKS	10	100
TEMPLIST	SPFNNNNN(3)	121	3146	FBA	BLKS	10	10
TEMPEDIT	SPFNNNNN(3)	3120	3120	U	BLKS	40	200

(1) SPFPARMS BLOCKSIZE CAN BE INCREASED LARGER THAN 6000 AT INSTALLATION TIME.

(2) SPFPARMS SPACE ALLOCATION SPECIFIED AT INSTALLATION TIME. SEE INSTALLATION AND CUSTOMIZATION GUIDE.

(3) THE VALUE FOR "NNNNN" IS A UNIQUE NUMBER DETERMINED BY SPF AT THE TIME OF ALLOCATION.

VISUAL TABLE OF CONTENTS



EXTENDED DESCRIPTION	LOAD MODULE	OBJECT MODULE	LABEL
1. INITIALIZE TABLES, VARIABLES, ETC. SUBROUTINE CBS MUST BE CALLED BEFORE CALLING CBR TO SET UP THE COMMON BROWSE TABLE (CBT). THE CBT IS GETMAINED FROM SUBPOOL 4 VIA SUBROUTINE CSM.	SPFSUBS	CBS CBR	CBS CBR10IH
2. THE SCREEN IMAGE IS FILLED FROM BROWSE LINE BUFFERS. DATA IS READ INTO THE LINE BUFFERS ON A DEMAND BASIS. LOGICAL RECORDS ARE READ A CALL TO SUBROUTINE CDG IN SUBROUTINE CBG. WHEN THE DISPLAY IS COMPLETE, SUBROUTINE CDISPL IS CALLED TO DISPLAY THE SCREEN IMAGE.	SPFSUBS	CBR CBG	CBR20FS CBG
3. IF THE END KEY WAS PRESSED, STEP 6 IS NEXT. OTHERWISE PROCESSING PROCEEDS WITH STEP 4.	SPFSUBS	CBR	CBR
4. INPUT ON LINE 2 IS ANALYZED. IF A "FIND" COMMAND WAS ENTERED (OR IF THE REPEAT FIND PF KEY WAS PRESSED) THE LINE BUFFERS ARE SEARCHED FOR THE DESIGNATED CHARACTER STRING. CBG IS CALLED, WHEN NECESSARY, TO READ MORE DATA INTO THE LINE BUFFERS. IF A "CAPS" OR "ASIS" COMMAND WAS ENTERED, THE INTERNAL MODE INDICATORS ARE UPDATED. IF A "LOCATE" COMMAND WAS ENTERED, THE DISPLAYED DATA IS SCROLLED TO THE PROPER LINE. PROCESSING PROCEEDS WITH STEP 2.	SPFSUBS	CBR CBF CBR	CBR50AI CBF CBR54LN
5. IF A SCROLL KEY WAS PRESSED, THE NEW STARTING LINE OR COLUMN IS DETERMINED. PROCESSING PROCEEDS WITH STEP 2.	SPFSUBS	CBR	CBR80PS
6. WHEN THE END KEY IS PRESSED, CONTROL IS RETURNED TO THE CALLER. SUBROUTINE CBC MUST BE CALLED AFTER CALLING CBR TO CLEAN UP THE CBT.	SPFSUBS	CBR CBC	CBR CBC

CHART 5
EDIT

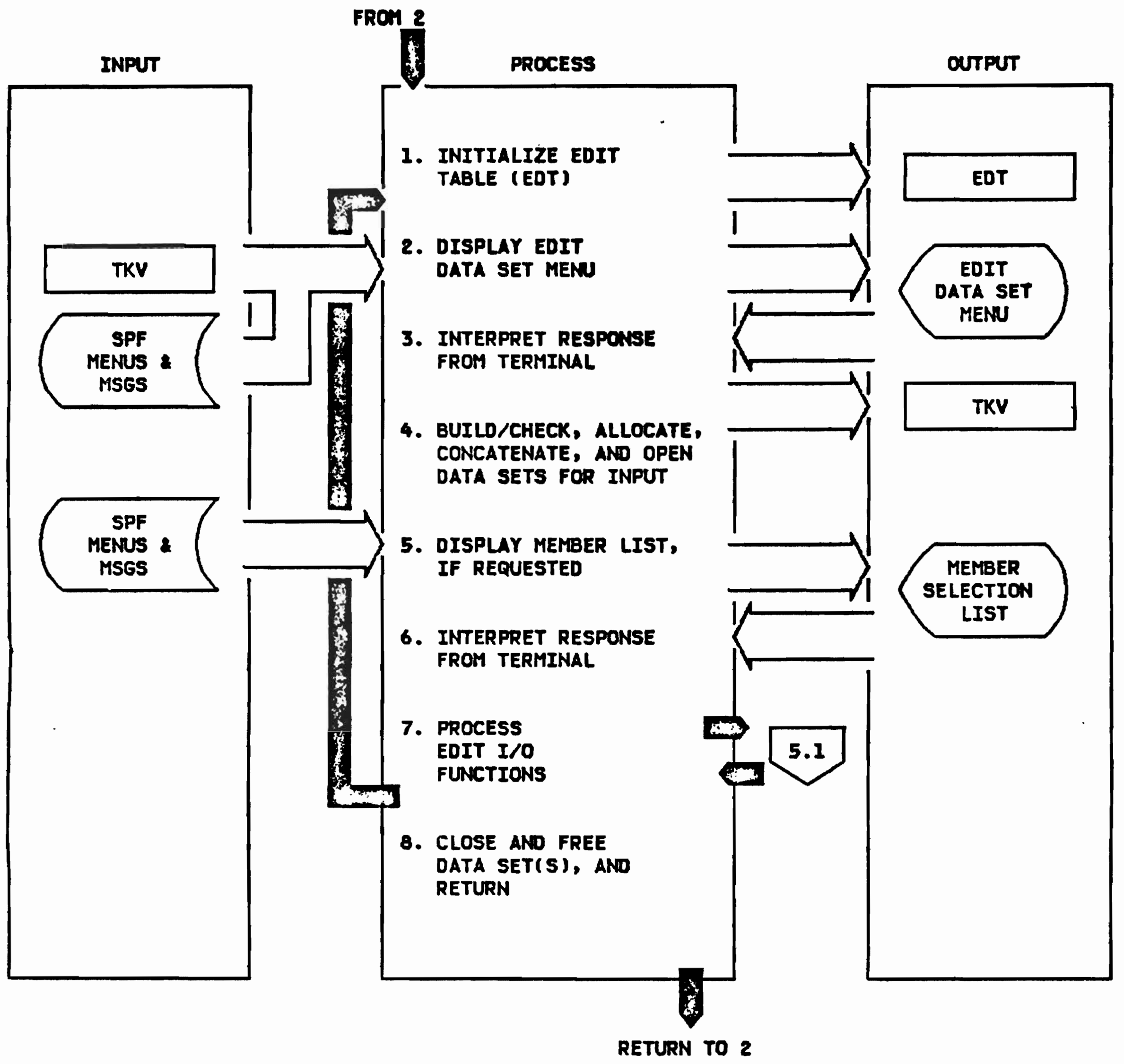
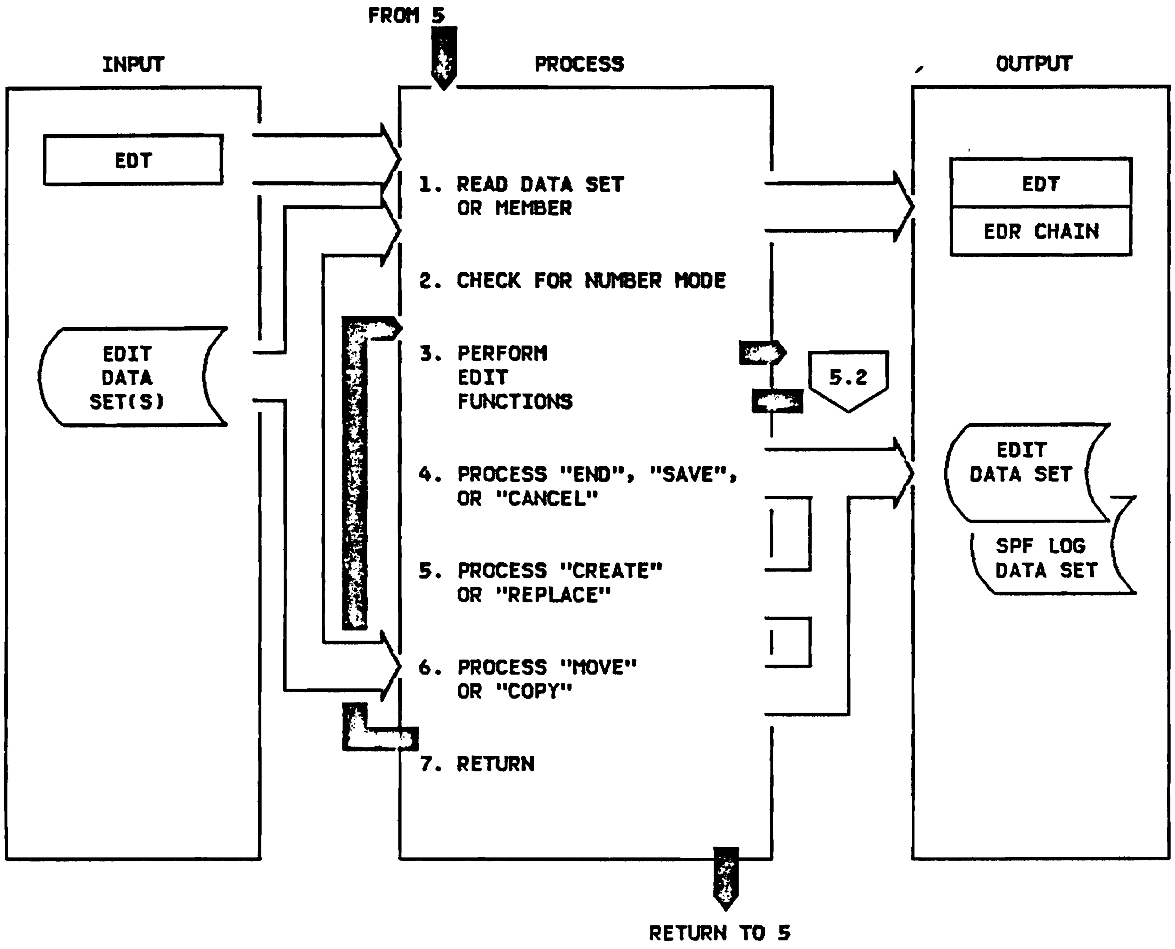

FROM 2
INPUT
PROCESS
OUTPUT
1. INITIALIZE EDIT TABLE (EDT)
2. DISPLAY EDIT DATA SET MENU
3. INTERPRET RESPONSE FROM TERMINAL
4. BUILD/CHECK, ALLOCATE, CONCATENATE, AND OPEN DATA SETS FOR INPUT
5. DISPLAY MEMBER LIST, IF REQUESTED
6. INTERPRET RESPONSE FROM TERMINAL
7. PROCESS EDIT I/O FUNCTIONS
8. CLOSE AND FREE DATA SET(S), AND RETURN
TKV
SPF MENUS & MSGS
SPF MENUS & MSGS
EDT
EDIT DATA SET MENU
TKV
MEMBER SELECTION LIST
5.1
RETURN TO 2

CHART 5
EDIT

EXTENDED DESCRIPTION	LOAD MODULE	OBJECT MODULE	LABEL
1. THE EDIT TABLE (EDT) IS INITIALIZED. THIS TABLE IS USED FOR COMMUNICATION BETWEEN THE EDIT PROGRAMS.	SPFSUBS	ETS	ETS
2. SUBROUTINE MHA IS CALLED TO DISPLAY THE EDIT MENU, AND SUBROUTINE MERR IS CALLED TO DISPLAY AN ERROR OR CONFIRMATION MESSAGE FROM THE PREVIOUS PASS THROUGH THE LOOP (IF ANY). INITIAL VALUES FOR PROJECT, LIBRARY, TYPE, AND PASSWORD ARE FROM THE TKV.	SPFSUBS	EMP	EMP
3. IF THE END KEY WAS PRESSED, STEP 10 IS EXECUTED NEXT. OTHERWISE, PROCESSING CONTINUES WITH STEP 3. ENTERED VALUES ARE PLACED BACK IN THE TKV.	SPFSUBS	EMP	EMP
4. SUBROUTINE CDA IS CALLED TO CONSTRUCT THE DATA SET NAMES (FROM "PROJECT", UP TO 4 "LIBRARY" NAMES, AND "TYPE") OR VALIDITY CHECK THE NAME (IF "OTHER" DATA SET NAME WAS ENTERED). THE PREVIOUSLY EDITED DATA SET(S), IF ANY, ARE CLOSED AND FREED. THE NEW DATA SET(S) ARE ALLOCATED (DISP=SHR) AND CONCATENATED IF PARTITIONED, OR ALLOCATED (DISP=OLD) IF SEQUENTIAL. SUBROUTINE CDO IS CALLED TO OPENED THEM FOR INPUT, AND THE DATA SET CHARACTERISTICS ARE CHECKED FOR VALIDITY.	SPFSUBS	EMP	EMP
5. STEPS 5 AND 6 ARE EXECUTED ONLY IF THE DATA SET(S) ARE PARTITIONED AND THE MEMBER NAME WAS NOT SPECIFIED. SUBROUTINE CML IS CALLED TO DISPLAY THE MEMBER LIST.	SPFSUBS	EPO	EPO
6. IF THE END KEY WAS PRESSED, STEP 2 IS EXECUTED NEXT. OTHERWISE, PROCESSING CONTINUES WITH STEP 7.	SPFSUBS	EPO	EPO
7. IF THE DATA SET(S) ARE PARTITIONED, THE BLDL AND FIND SYSTEM SERVICES ARE ISSUED FOR THE SELECTED MEMBER. THE EDIT HEADER LINES ARE THEN SET UP AND THE EDIT PROCESS ROUTINE EPR IS CALLED TO CONTINUE PROCESSING.	SPFSUBS	EPO EPS	EPO EPS
8. WHEN THE END KEY IS PRESSED, THE DATA SET(S) ARE CLOSED AND FREED, AND CONTROL IS RETURNED TO THE CALLER (PMD).	SPFSUBS	EMP	EMP

CHART 5.1
EDIT I/O ROUTINE



FROM 5
INPUT
PROCESS
OUTPUT
EDT
EDIT DATA SET(S)
1. READ DATA SET OR MEMBER
2. CHECK FOR NUMBER MODE
3. PERFORM EDIT FUNCTIONS
4. PROCESS "END", "SAVE", OR "CANCEL"
5. PROCESS "CREATE" OR "REPLACE"
6. PROCESS "MOVE" OR "COPY"
7. RETURN
5.2
EDT
EDR CHAIN
EDIT DATA SET
SPF LOG DATA SET
RETURN TO 5

CHART 5.1
EDIT I/O ROUTINE

EXTENDED DESCRIPTION	LOAD MODULE	OBJECT MODULE	LABEL
1. EDIT DATA INPUT (EDI) IS CALLED TO READ IN THE DATA. EACH LOGICAL RECORD IS STORED IN AN EDR (EDIT RECORD). THE EDR'S ARE CHAINED TO THE EDIT TABLE.	SPFSUBS	EDI	EDI
2. IF RECORDS HAVE ASCENDING SEQUENCE NUMBERS, EDIT IS INITIALIZED TO "NUMBER" MODE. OTHERWISE, IT IS "NONUM" MODE. AN UPPER/LOWER CASE CHECK IS ALSO DONE SO "CAPS" MODE CAN BE SET ON OR OFF.	SPFSUBS	EDI	EDI
3. THE EDIT PROCESSING ROUTINE (EPR) PERFORMS BASIC EDIT FUNCTIONS. (I.E. SCROLLING, LINE COMMANDS, AND PRIMARY COMMANDS). EPR RETURNS CONTROL WHEN THE PRIMARY COMMANDS SAVE OR CANCEL ARE EXECUTED, OR WHEN THE END KEY IS PRESSED.	SPFSUBS	EPR	EPR
4. STEP 4 IS EXECUTED IN THE EVENT OF "END", "SAVE", OR "CANCEL". FOR "SAVE", THE DATA IS WRITTEN TO THE EDIT DATA SET, BUT NOT THE LIST DATA SET. FOR "END", THE DATA IS WRITTEN TO THE EDIT DATA SET IF ANY CHANGES HAVE OCCURRED, AND IT IS RECORDED IN THE LIST DATA SET IF "PRINT" MODE IS ON VIA SUBROUTINE CPRINT. THE PROCEDURES FOR WRITING THE EDIT DATA SET ARE AS FOLLOWS: - THE SUBROUTINE CDO IS CALLED TO OPEN THE DATA SET FOR OUTPUT AND PERFORM A RESERVE. - THE SUBROUTINE CDP IS CALLED TO WRITE EACH RECORD. - THE STOW-REPLACE SYSTEM SERVICE IS ISSUED,IF IT IS A PARTITIONED DATA SET. - THE SUBROUTINE CDO IS CALLED TO CLOSE AND RELEASE (AND DEQ) THE DATA SET. WHENEVER THE DATA IS SAVED, A RECORD IS WRITTEN TO THE LOG DATA SET BY CALLING THE SUBROUTINE CLOG. FOLLOWING "END" OR "CANCEL", CONTROL IS RETURNED TO THE CALLING PROCEDURE (STEP 7). FOLLOWING "SAVE", STEP 3 IS EXECUTED.	SPFSUBS	EPR	EPR
5. STEP 5 IS EXECUTED FOR A "CREATE" OR "REPLACE" PRIMARY COMMAND. THE EDIT, LIST, AND LOG DATA SETS ARE WRITTEN USING THE SAME PROCEDURES DESCRIBED FOR STEP 4. THEN STEP 3 IS EXECUTED.	SPFSUBS	ECR	ECR
6. STEP 6 IS EXECUTED FOR A "MOVE" OR "COPY" PRIMARY COMMAND. SUBROUTINE CGET IS CALLED TO READ EACH RECORD OF THE EDIT DATA SET MEMBER. FOR "MOVE", A STOW-DELETE SYSTEM SERVICE IS ISSUED TO DELETE THE MEMBER FOLLOWING A SUCCESSFUL COPY. THE SUBROUTINE CLOG IS CALLED TO WRITE A RECORD TO THE LOG DATA SET. THEN STEP 3 IS EXECUTED.	SPFSUBS	EMC	EMC
7. WHEN END OR CANCEL HAS BEEN PROCESSED, CONTROL IS RETURNED TO THE CALLER (EPO OR EPS).	SPFSUBS	EPR	EPR

CHART 5.2
EDIT PROCESSOR

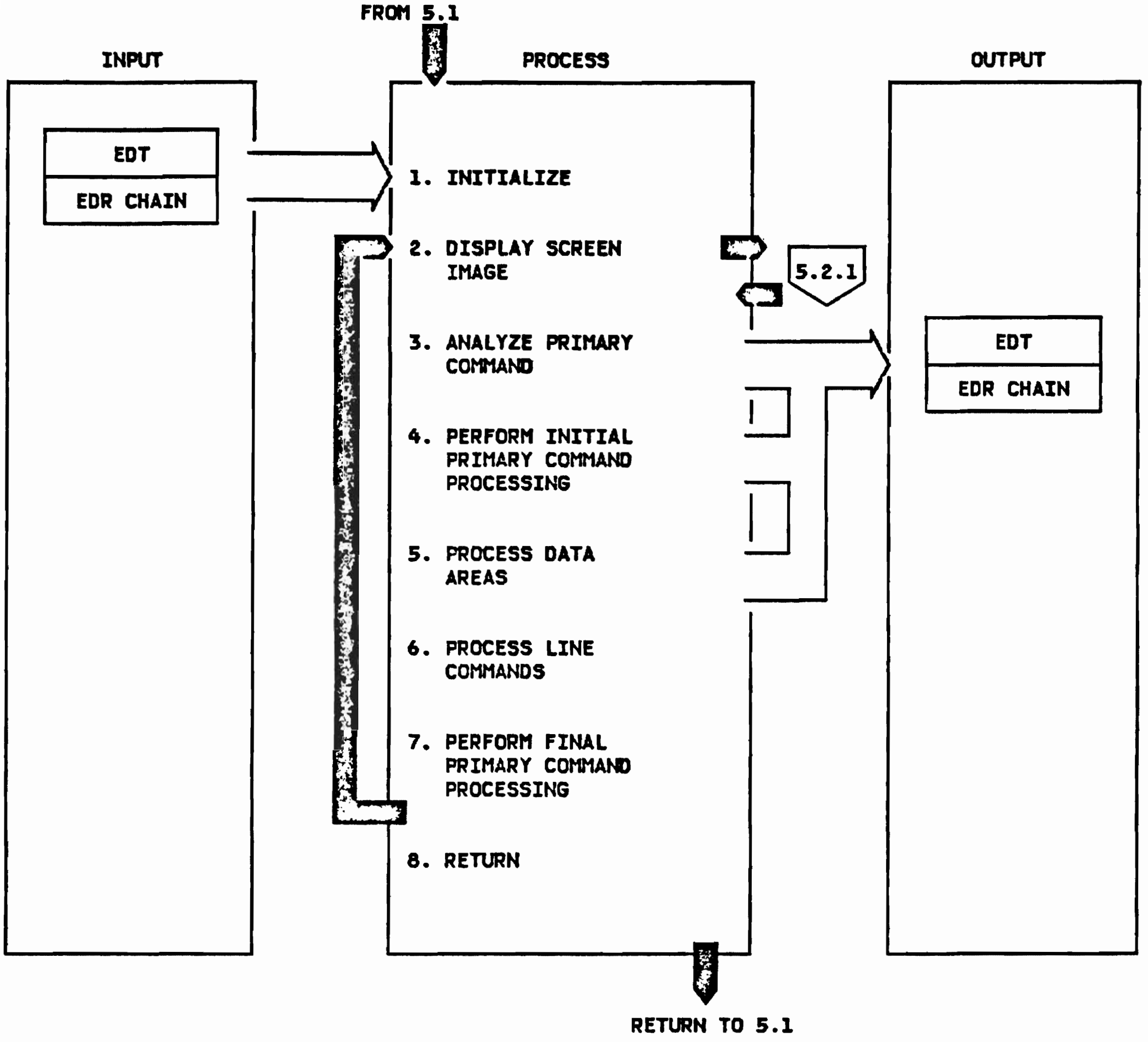
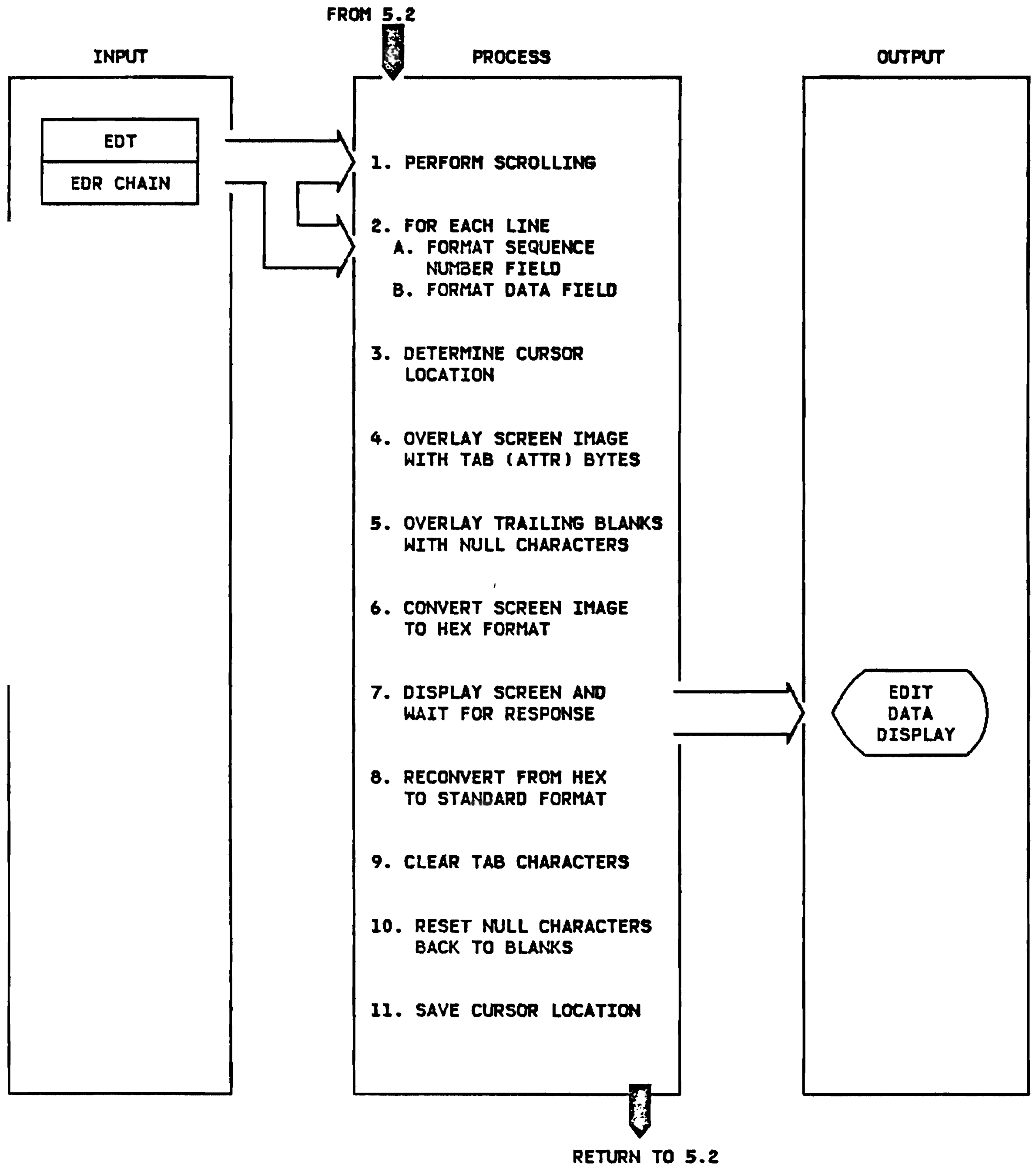


CHART 5.2
EDIT PROCESSOR

EXTENDED DESCRIPTION	LOAD MODULE	OBJECT MODULE	LABEL
1. SELECTED FIELDS IN THE EDIT TABLE (EDT) ARE INITIALIZED. - IF THE EDIT RECORD CHAIN IS EMPTY, ENOUGH "INSERT" LINES ARE INSERTED BETWEEN THE TOP AND BOTTOM EDIT RECORDS TO FILL THE CURRENT LOGICAL SCREEN.	SPFSUBS	EPR	EPR10IH EPR20IR
2. THE LOGICAL SCREEN IS FORMATTED. EITHER SUBROUTINE CERR OR CDISPL IS CALLED TO DISPLAY IT. THEN THE SCREEN IMAGE IS RESTORED BY REMOVING TABS AND NULLS.	SPFSUBS	EFR	EFR
3. IF THE COMMAND INPUT FIELD IS NOT BLANK, IT IS ANALYZED. THE COMMAND VERB IS IDENTIFIED AND PARAMETERS ARE SCANNED. THE EDPCWDS ARRAY IN THE EDIT TABLE (EDT) IS BUILT.	SPFSUBS	EPI	EPI
4. INITIAL COMMAND PROCESSING IS PERFORMED FOR THE FOLLOWING PRIMARY COMMANDS:			
AUTONUM	SPFSUBS	EPI	EPIAUTON
CANCEL	SPFSUBS	EPI	EPICANCL
CAPS	SPFSUBS	EPI	EPICAPS
CREATE	SPFSUBS	EPI	EPIMEMB
COPY	SPFSUBS	EPI	EPIMEMB
HEX	SPFSUBS	EPI	EPIHEX
MOVE	SPFSUBS	EPI	EPIMEMB
NULLS	SPFSUBS	EPI	EPINULLS
NUMBER	SPFSUBS	EPI	EPINUMB
PRINT	SPFSUBS	EPI	EPIPRINT
PROFILE	SPFSUBS	EPI	EPIPROF
RECOVERY	SPFSUBS	EPI	EPIRECVR
RENUM	SPFSUBS	EPI	EPINUMB
REPLACE	SPFSUBS	EPI	EPIMEMB
RESET	SPFSUBS	EPI	EPIRESET
STATS	SPFSUBS	EPI	EPISTATS
TABS	SPFSUBS	EPI	EPITABS
UNNUM	SPFSUBS	EPI	EPINUMB
5. EACH DATA AREA THAT HAS BEEN MODIFIED IS PROCESSED AS FOLLOWS: - MASK DATA IS MOVED TO THE MASK FIELD, - TABS DATA IS VALIDATED AND MOVED TO THE TABS FIELD. - BOUNDS DATA IS USED TO COMPUTE NEW BOUNDS. - ALL OTHER DATA IS MOVED INTO A CORRESPONDING EDIT RECORD (EDR).	SPFSUBS	EPD	EPD

EXTENDED DESCRIPTION	LOAD MODULE	OBJECT MODULE	LABEL
6. LINE COMMANDS THAT HAVE BEEN ENTERED IN THE SEQUENCE FIELDS ARE MOVED TO THE CORRESPONDING EDIT RECORDS. THE EDIT RECORD CHAIN IS THEN SCANNED AND LINE COMMANDS ARE VALIDITY CHECKED. FINALLY THE EDIT RECORD CHAIN IS SCANNED AND THE LINE COMMANDS ARE EXECUTED. THE FUNCTION IS PERFORMED BY USING THE SUBROUTINE CODE FROM THE EDIT LINE COMMAND DEFINITION (ELC ENTRY) AND INDEXING INTO A LIST OF ADDRESSES OF INTERNAL PROCEDURES. THE COMMANDS THAT ARE EXECUTED ARE:	SPFSUBS	EPC	EPC
A (AFTER)	SPFSUBS	EPC	ECLAFTER
B (BEFORE)	SPFSUBS	EPC	ECLBEFOR
BOUNDS	SPFSUBS	EPC	ECLBOUND
C (COPY)	SPFSUBS	EPC	ECLCOPY
COLS	SPFSUBS	EPC	ECLCOLS
D (DELETE)	SPFSUBS	EPC	ECLDEL
I (INSERT)	SPFSUBS	EPC	ECLINSRT
M (MOVE)	SPFSUBS	EPC	ECLMOVE
MASK	SPFSUBS	EPC	ECLMASK
O (OVERLAY)	SPFSUBS	EPC	ECLOVER
R (REPEAT)	SPFSUBS	EPC	ECLREP
L (LAST)	SPFSUBS	EPC	ECLSBOT
S (SHOW)	SPFSUBS	EPC	ECLSHOW
F (FIRST)	SPFSUBS	EPC	ECLSTOP
TABS	SPFSUBS	EPC	ECLTABS
TE	SPFSUBS	EPC	ECLTENTR
TF	SPFSUBS	EPC	ECLTFLOW
TS	SPFSUBS	EPC	ECLTSPLT
X (EXCLUDE)	SPFSUBS	EPC	ECLXCLUD
> (SHIFT)	SPFSUBS	EPC	ECLSR
< (SHIFT)	SPFSUBS	EPC	ECLSL
) (SHIFT COLS)	SPFSUBS	EPC	ECLSCR
((SHIFT COLS)	SPFSUBS	EPC	ECLSCL
7. THE FOLLOWING PRIMARY COMMANDS ARE PROCESSED:	SPFSUBS	EPF	EPF
CANCEL	SPFSUBS	EPF	EPFCANCL
CHANGE	SPFSUBS	EFC	EFC
COPY	SPFSUBS	EMC	EMC
CREATE	SPFSUBS	ECR	ECR
FIND	SPFSUBS	EFC	EFC
LOCATE	SPFSUBS	EPF	EPFLOC
MOVE	SPFSUBS	EMC	EMC
NUMBER	SPFSUBS	EPF	EPFNUMB
PROFILE	SPFSUBS	EPF	EPFPROF
RENUM	SPFSUBS	EPF	EPFNUMB
REPLACE	SPFSUBS	ECR	ECR
RESET	SPFSUBS	EPF	EPFRESET
SAVE	SPFSUBS	EPF	EPFSAVE
SUBMIT	SPFSUBS	EPF	EPFSUBMT
UNNUM	SPFSUBS	EPF	EPFNUMB
THE FOLLOWING PROGRAM FUNCTION KEYS (PFK'S) ARE ALSO PROCESSED: END	SPFSUBS	EPR	EPR
REPEAT FIND	SPFSUBS	EFC	EFC
REPEAT CHANGE	SPFSUBS	EFC	EFC
8. IF NO ERRORS HAVE BEEN DETECTED AND NO INFORMATION MESSAGES ARE TO BE DISPLAYED, AND IF THE END KEY HAS BEEN PRESSED, OR A CANCEL COMMAND HAS BEEN ENTERED, THEN EDIT PROCESSING IS TERMINATED.	SPFSUBS	EPR	EPR

CHART 5.2.1
EDIT DISPLAY SCREEN



1. PERFORM SCROLLING

2. FOR EACH LINE
   A. FORMAT SEQUENCE NUMBER FIELD
   B. FORMAT DATA FIELD

3. DETERMINE CURSOR LOCATION

4. OVERLAY SCREEN IMAGE WITH TAB (ATTR) BYTES

5. OVERLAY TRAILING BLANKS WITH NULL CHARACTERS

6. CONVERT SCREEN IMAGE TO HEX FORMAT

7. DISPLAY SCREEN AND WAIT FOR RESPONSE

8. RECONVERT FROM HEX TO STANDARD FORMAT

9. CLEAR TAB CHARACTERS

10. RESET NULL CHARACTERS BACK TO BLANKS

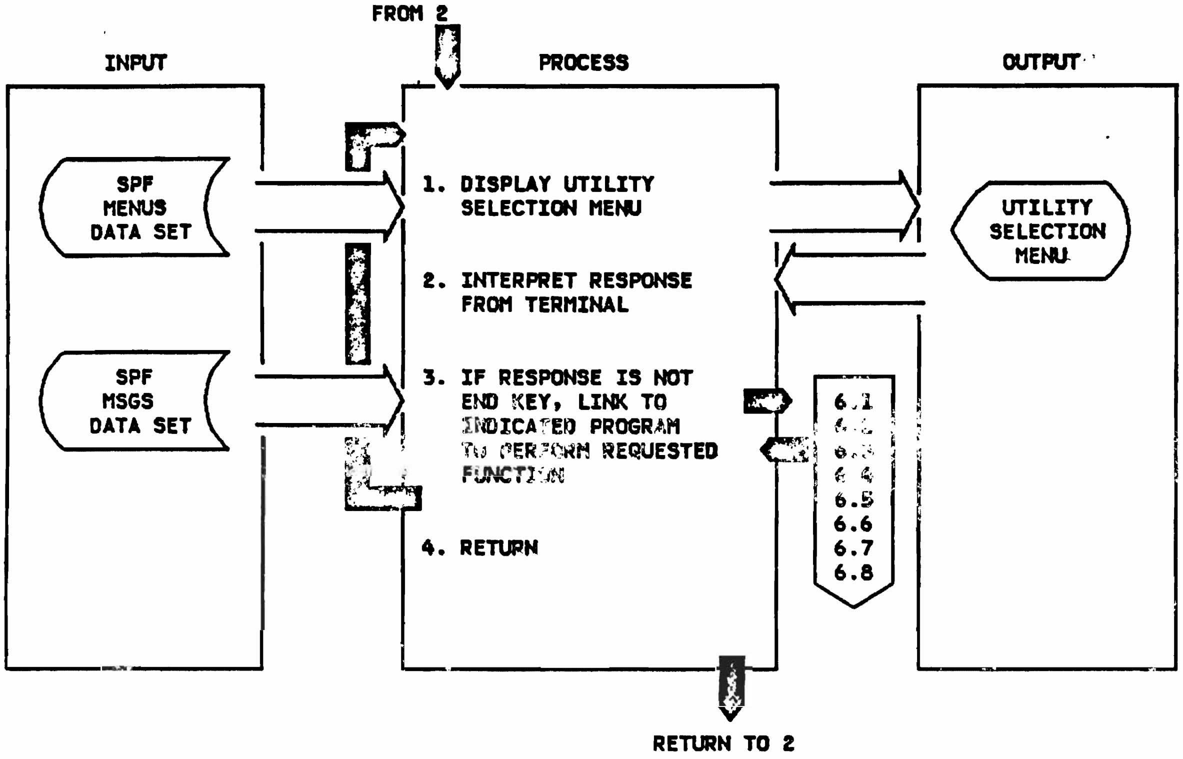
11. SAVE CURSOR LOCATION

CHART 5.2.1
EDIT DISPLAY SCREEN

EXTENDED DESCRIPTION	LOAD MODULE	OBJECT MODULE	LABEL
1. IF A SCROLL REQUEST WAS MADE (TLDSCROL = ON), A NEW FIRST DISPLAY LINE (EDRP(EDFDISPL)) OR A NEW FIRST DISPLAY COLUMN (EDCOL(EDRECDSP,EDLEFT)) IS SET UP, AND SUBROUTINE CSCROLL IS CALLED.	SPFSUBS	EFR	EFR15PS
2. FORMAT THE DISPLAY, PROCESSING EDIT RECORDS, STARTING AT THE FIRST DISPLAY LINE (EDRP(EDFDISPL))	SPFSUBS	EFR	EFR20FS
A. IN THE SEQUENCE FIELD, SET UP EITHER: 1. PENDING LINE COMMAND FROM PREVIOUS IMAGE 2. A MASK (******,.....,BOUNDS,- - -,COLS MASK , TABS , =ERR=>, OR =CHG=>) 3. A SEQUENCE NUMBER EITHER FROM THE EDIT RECORD (IF RENUM MODE) OR THE RELATIVE NUMBER OF THE EDIT RECORD IN THE EDIT RECORD CHAIN (IF NONUM MODE). SET THE INTENSITY LOW FOR SEQUENCE NUMBERS AND HIGH FOR ALL OTHERS.	SPFSUBS	EFR	EFR21CA
B. IN THE DATA FIELDS: 1. MOVE IN DATA FROM THE EDIT RECORD, OR 2. MOVE IN DATA FROM THE MASK FIELD, OR 3. MOVE IN DATA FROM THE TABS FIELD, OR 4. MOVE IN THE EXCLUDED LINES MESSAGE AND ENTER THE NUMBER OF EXCLUDED LINES, OR 5. GENERATE THE BOUNDS LINE, OR 6. GENERATE THE COLS LINE.	SPFSUBS	EFR	EFR22DA
3. DETERMINE THE CURSOR LOCATION. IF EDCSRSET = ON THE CURSOR POSITION HAS ALREADY BEEN DETERMINED AND STORED IN THE TLDCSRP FIELD. IF NOT, THE THE CURSOR MAY BE ASSOCIATED WITH AN EDIT RECORD AND AN OFFSET IN THE RECORD. IF SO DETERMINE THE CORRESPONDING POSITION ON THE SCREEN. IF THE POSITION ON THE SCREEN CANNOT BE DETERMINED, PUT THE CURSOR ONTO LINE 2. IF THE CURSOR IS WITHIN THE DATA PORTION OF THE SCREEN, USE THE TABS LINE TO DETERMINE THE NEXT POSITION FOR THE CURSOR AND SET IT TO THAT LOCATION.	SPFSUBS	EFR	EFR30SC
4. IF TABS MODE IS IN EFFECT PUT ATTRIBUTE BYTES IN EACH COLUMN POSITION WHERE AN ASTERISK IS FOUND IN THE TABS LINE (OVERLAY NON-BLANK CHARACTERS ONLY IF IN TABS ANY MODE).	SPFSUBS	EFR	EFR40ST
5. BACKWARD SCAN EACH FIELD OF THE DATA PORTION OF THE SCREEN FOR TRAILING BLANKS TO REPLACE WITH NULL CHARACTERS. IF ENTIRE FIELD IS BLANK, DO NOT REPLACE FIELD WITH NULLS UNLESS TABS ALL IS IN EFFECT (EDTABBO = OFF). IF CURSOR IS WITHIN A FIELD, DO NOT REPLACE BLANKS AT OR BEFORE CURSOR WITH NULLS.	SPFSUBS	EFR	EFR45SN
6. IF HEX MODE IS ON, CONVERT SCREEN IMAGE FROM STANDARD FORMAT TO HEX FORMAT. THE FIRST FEW LINES ON THE SCREEN WILL BE EXPANDED TO FILL THE SCREEN. THE CURSOR IS REPOSITIONED FOR HEX MODE.	SPFSUBS	EFR	EFR50FH

EXTENDED DESCRIPTION	LOAD MODULE	OBJECT MODULE	LABEL
7. IF AN ERROR OR INFORMATION MESSAGE IS TO BE DISPLAYED SUBROUTINE CERR IS CALLED, OTHERWISE SUBROUTINE CDISPL IS CALLED.	SPFSUBS	EFR	EFR60DS
8. IF HEX MODE WAS ON, RECONVERT THE SCREEN IMAGE FROM HEX FORMAT TO STANDARD FORMAT. REPOSITION CURSOR.	SPFSUBS	EFR	EFR70RH
9. REPLACE ATTRIBUTE BYTES THAT WERE CREATED IN STEP 4 WITH THE DATA CHARACTERS THAT THEY OVERLAYED.	SPFSUBS	EFR	EFR80CT
10. IF TABS MODE IS NOT ON, ONLY MODIFIED LINES WILL BE EXAMINED, AND THE DISPLAY INTERFACE WILL HAVE TRANSLATED NULLS TO BLANKS. IF TABS ARE ON ONLY PART OF A LINE MIGHT HAVE BEEN MODIFIED, SO TRANSLATE ANY NULLS TO BLANKS.	SPFSUBS	EFR	EFR80CT
11. IF THE CURSOR WAS ON A LINE, ASSOCIATE IT WITH THE CORRESPONDING EDIT RECORD (EDR).	SPFSUBS	EFR	EFR90RC

CHART 6
UTILITIES


FROM 2
INPUT
PROCESS
OUTPUT
SPF MENUS DATA SET
SPF MSGS DATA SET
1. DISPLAY UTILITY SELECTION MENU
2. INTERPRET RESPONSE FROM TERMINAL
3. IF RESPONSE IS NOT END KEY, LINK TO INDICATED PROGRAM TO PERFORM REQUESTED FUNCTION
4. RETURN
6.1
6.2
6.3
6.4
6.5
6.6
6.7
6.8
UTILITY SELECTION MENU
RETURN TO 2

SECTION 3

PROGRAM ORGANIZATION

THIS SECTION ILLUSTRATES THE PHYSICAL STRUCTURE AND ORGANIZATION OF SPF PROGRAMS AND DESCRIBES THE PROGRAM INTERFACE REQUIREMENTS. THE SECTION IS ORGANIZED AS FOLLOWS:

PROGRAM COMPONENTS - LISTS THE NAMES OF ALL SPF LOAD MODULES AND OBJECT MODULES.

LOAD MODULE HIERARCHY - SHOWS THE RELATIONSHIP BETWEEN LOAD MODULES.

OBJECT MODULE DESCRIPTIONS - BRIEFLY DESCRIBES THE PURPOSE OF EACH OBJECT MODULE. FOR OBJECT MODULES THAT ARE PROGRAMS, CALLING SEQUENCE INFORMATION IS ALSO GIVEN. THIS SECTION IS ORDERED ALPHABETICALLY BY MODULE NAME.

PROGRAM COMPONENTS

THE FOLLOWING IS A LIST IN ALPHABETICAL ORDER OF THE LOAD MODULES WHICH COMPRISE SPF, SHOWING THE OBJECT MODULE(S) INCLUDED IN EACH LOAD MODULE. THE NAME OF THE ENTRY POINT (EP) MODULE FOLLOWS THE LOAD MODULE TITLE.

LOAD MODULE	OBJECT MODULE	DESCRIPTION (EP: ENTRY POINT)

```
SPF ------------- SPF DRIVER LOAD MODULE  (EP: SPF)
         SPF ------ SPF DRIVER

SPFBRO ---------- BROWSE LOAD MODULE  (EP: BRO)
         BRO ------ BROWSE MAIN

SPFCALCP -------- CALL COMMAND PROCESSOR (EP: SPFCALCP)

SPFEDIT --------- EDIT LOAD MODULE  (EP: EDD)
         EDD ------ EDIT MAIN DRIVER

SPFFOR ---------- FOREGROUND PROCESSOR LOAD MODULE  (EP: FOR)
         FOR ------ FOREGROUND PROCESSOR

SPFJOB ---------- BACKGROUND PROCESSOR LOAD MODULE  (EP: JOB)
         JOB ------ BACKGROUND PROCESSOR

SPFMAIN --------- SPF CONTROLLER LOAD MODULE  (EP: SMD)
         CIPARMS -- COMMON INITIALIZE USER PARMS
         SIP ------ SPF INPUT PARMS EXIT ROUTINE
         SMA ------ SPF MAIN ATTACH
         SMC ------ SPF MAIN CONTROLLER
         SMD ------ SPF MAIN DRIVER
         SMI ------ SPF MAIN INITIALIZATION
         SML ------ SPF MAIN LINE I/O INTERFACE
         TKV ------ SPF KEYWORD/VALUE PROTOTYPE TABLE
         TKW ------ SPF KEYWORD TABLE
         TRT ------ SPF TABLE OF REENTRANT TABLES

SPFOPT ---------- SPF PARAMETERS AND DEFAULTS LOAD MODULE  (EP: OPT)
         OPT ------ SPF PARAMETERS AND DEFAULTS OPTION

SPFPMD ---------- PROCESSOR MAIN DRIVER LOAD MODULE  (EP: PMD)
         PFT ------ PROCESSOR FINAL TERMINATION
         PMD ------ PROCESSOR MAIN DRIVER
         PRS ------ PROCESSOR RESTART

SPFSCAN --------- BACKGROUND SCAN LOAD MODULE  (EP: SCN)
         SCN ------ BACKGROUND SCAN

SPFSPC ---------- SPF PARMS CONVERSION LOAD MODULE  (EP: SPC)
         SPC ------ SPF PARMS CONVERSION (VERSION 2.1 TO VERSION 2.2)

SPFSUBS --------- COMMON SUBROUTINE LOAD MODULE  (EP: TSC)
         BCD ------ BROWSE COMMAND DEFINITIONS
         CAT ------ COMMON ATTACH COMMAND
         CBC ------ COMMON BROWSE CLEANUP
         CBDSN ---- COMMON BUILD DATASET NAME
         CBF ------ COMMON BROWSE FIND
         CBG ------ COMMON BROWSE GET
         CBR ------ COMMON BROWSE
         CBS ------ COMMON BROWSE SETUP
         CCB ------ COMMON COMMAND BUILD
         CCD ------ COMMON CONVERT DIRECTORY ENTRY
         CCP ------ COMMON COMMAND PARSE
         CCS ------ COMMON COMMAND SCAN
         CDA ------ COMMON DATASET ALLOCATE
```

(CONTINUED ON NEXT PAGE)

PROGRAM COMPONENTS (CONTINUED)

CDAIR	COMMON DAIR INTERFACE
CDATE	COMMON CONVERT DATE
CDC	COMMON DATASET CLOSE
CDERR	COMMON DAIR ERROR
CDF	COMMON DATASET FREE
CDG	COMMON DATASET GET
CDISPL	COMMON DISPLAY
CDO	COMMON DATASET OPEN
CDP	COMMON DATASET PUT
CDT	COMMON GET DEVICE TYPE
CERR	COMMON ERROR MESSAGE
CFI	COMMON FIND
CHC	COMMON HARDCOPY
CHELP	COMMON HELP
CHPJ	COMMON HARDCOPY JOB
CHPL	COMMON HARDCOPY LOCAL
CIR	COMMON GET DIRECTORY ENTRY
CIV	COMMON GET DSCB INFORMATION
CJC	COMMON JOB CARD
CJF	COMMON JOB NAME FIND
CJN	COMMON JOB NAME SETUP
CKVGET	COMMON KEYWORD/VALUE GET
CKVPUT	COMMON KEYWORD/VALUE PUT
CLM	COMMON LOAD MODULE LOADER
CLOG	COMMON LOG
CMB	COMMON MENU BUILD
CML	COMMON MEMBER LIST
CMSG	COMMON MESSAGE
CPRINT	COMMON PRINT DATASET
CRELS	COMMON RELEASE DASD
CRESV	COMMON RESERVE DASD
CSB	COMMON SUBMIT
CSCROLL	COMMON SCROLL
CSM	COMMON STORAGE MANAGEMENT
CTA	COMMON ALLOCATE TEMPORARY DATASET
CTF	COMMON FREE TEMPORARY DATASET
CTGET	COMMON TGET
CTPUT	COMMON TPUT
CT1	COMMON ALLOCCATE TEMPORARY DATASET
CT2	COMMON FREE TEMPORARY DATASET
CUPARMS	COMMON UPDATE USER PARMS
CVM	COMMON VERIFY MEMBER NAME
CVSDE	COMMON VERIFY SPF DIRECTORY ENTRY
EBA	EDIT RECOVERY INITIALIZATION
EBE	EDIT BACKUP END
EBI	EDIT BACKUP INITIALIZATION
EBR	EDIT RECOVERY READ
EBS	EDIT BACKUP STORE
EBX	EDIT BACKUP RESET
ECD	EDIT COMMAND DEFINITIONS
ECR	EDIT CREATE/REPLACE COMMAND FINAL
EDI	EDIT DATA INPUT
EDO	EDIT DATA OUTPUT
EFC	EDIT FIND/CHANGE
EFR	EDIT FORMAT DISPLAY
EFT	EDIT FLOW TEXT
EGN	EDIT GENERAL NUMBER
EGR	EDIT GENERAL RESET
EMC	EDIT MOVE/COPY COMMAND FINAL
EML	EDIT MESSAGE LINE
EMP	EDIT MENU PROCESSOR
EPC	EDIT PROCESS LINE COMMAND
EPD	EDIT PROCESS DATA
EPF	EDIT PROCESS FINAL
EPI	EDIT PROCESS INITIAL
EPO	EDIT PDS PROCESSOR
EPP	EDIT PROFILE PROCESSOR

(CONTINUED ON NEXT PAGE)

PROGRAM COMPONENTS (CONTINUED)

```
          EPR ------ EDIT MAIN PROCESSOR
          EPS ------ EDIT SEQUENTIAL DATASET PROCESSOR
          EPX ------ EDIT OTHER DATASET PROCESSOR
          ERA ------ EDIT RECORD ALLOCATE
          ERC ------ EDIT RECORD CHANGE
          ERD ------ EDIT RECORD DELETE
          ERF ------ EDIT RECORD FREE
          ERI ------ EDIT RECORD INSERT
          ERN ------ EDIT RECORD NUMBER
          ERO ------ EDIT RECORD DELETE ORIGINAL
          ERR ------ EDIT RECORD RESET
          ERS ------ EDIT RECORD SHOW
          ERX ------ EDIT RECORD EXCLUDE
          EST ------ EDIT SPLIT TEXT
          ETC ------ EDIT TABLE CLEANUP
          ETS ------ EDIT TABLE SETUP
          ETL ------ EDIT LINE COMMAND TABLE
          MERR ----- COMMON MENU ERROR
          MHA ------ COMMON MENU HANDLER
          SOP ------ SPF OUTPUT PARMS EXIT
          TSC ------ TABLE OF COMMON SUBS

  SPFTBLS --------- COMMON TABLES LOAD MODULE  (EP: TSI)
          TSI ------ COMMON TABLES

  SPFTCM ---------- COMMAND TABLE LOAD MODULE  (EP: TCM)
          TCM ------ COMMAND TABLE

  SPFTMENU -------- MENU TESTER LOAD MODULE  (EP: MNT)
          MNT ------ TEST MENU (DEBUGGING AID)

  SPFTSO ---------- TSO COMMAND PROCESSOR LOAD MODULE  (EP: PTC)
          PTC ------ TSO COMMAND PROCESSOR

  SPFTUTOR -------- TUTORIAL PROCESSOR LOAD MODULE  (EP: TUT)
          TUT ------ TUTORIAL PROCESSOR

  SPFUCA ---------- CATALOG MANAGEMENT DRIVER LOAD MODULE  (EP: UCA)
          UCA ------ CATALOG MANAGEMENT DRIVER

  SPFUC1 ---------- SVS CATALOG MANAGEMENT LOAD MODULE  (EP: UC1)
          UC1 ------ SVS CATALOG MANAGEMENT

  SPFUC2 ---------- MVS CATALOG MANAGEMENT LOAD MODULE  (EP: UC2)
          UC2 ------ MVS CATALOG MANAGEMENT

  SPFUDA ---------- LIBRARY/DATA SET UTILITY LOAD MODULE  (EP: UDA)
          UAA ------ ALLOCATE NEW DATA SET
          UAC ------ CATALOG/UNCATALOG DATA SET
          UAD ------ DELETE DATA SET
          UAI ------ DISPLAY DATA SET INFORMATION
          UAR ------ RENAME DATA SET
          UDA ------ LIBRARY/DATA SET UTILITY DRIVER
          UDM ------ LIBRARY UTILITY MEMBER LIST
          UDMS ----- LIBRARY UTILITY MEMBER SELECT
          UDP ------ PRINT DATA SET
          UDX ------ PRINT INDEX LISTING
          UDZ ------ COMPRESS DATA SET

  SPFUHC ---------- HARDCOPY UTILITY LOAD MODULE  (EP: UHC)
          UHC ------ HARDCOPY UTILITY
  SPFUMC ---------- MOVE/COPY UTILITY LOAD MODULE  (EP: UMC)
          UMC ------ MOVE/COPY UTILITY
          UMCS ----- MOVE/COPY UTILITY MEMBER SELECT

  SPFUOL ---------- OUTLIST UTILITY LOAD MODULE  (EP:UOL)
          UOL ------ OUTLIST UTILITY
```

(CONTINUED ON NEXT PAGE)

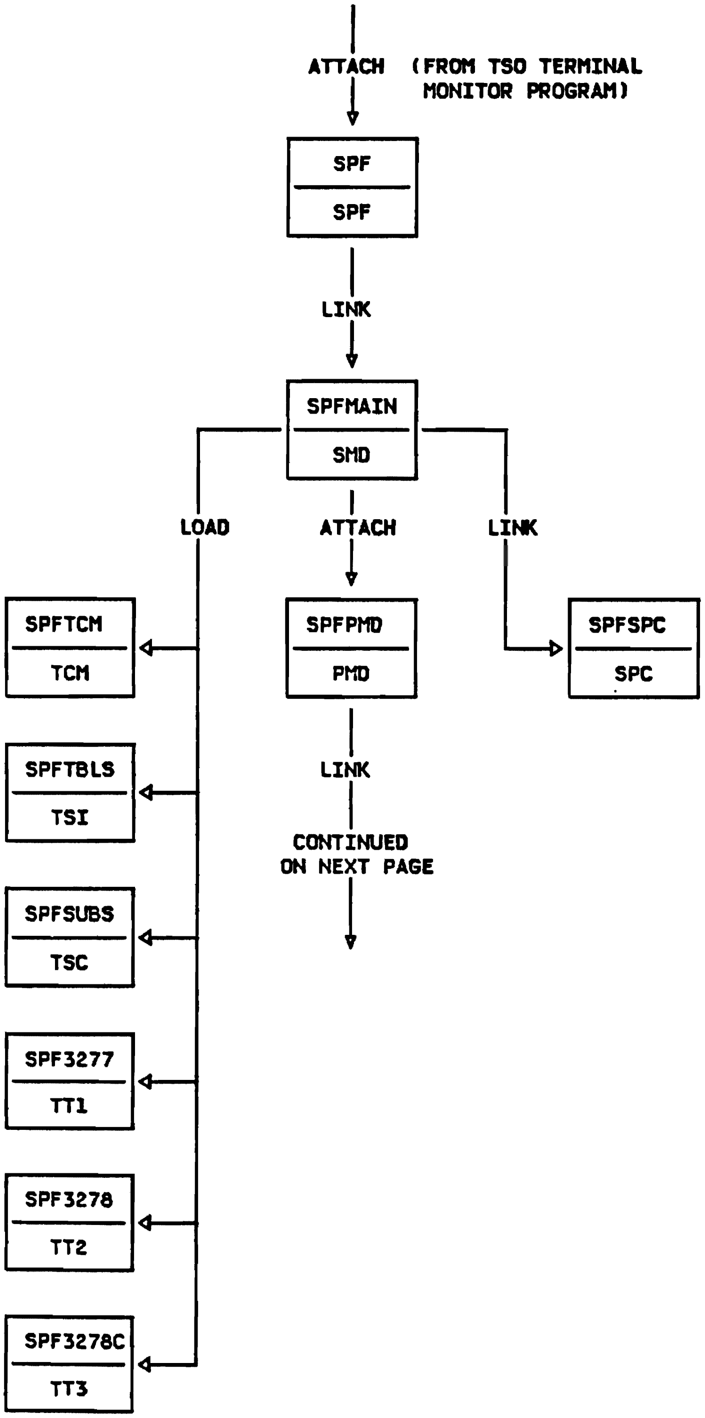
PROGRAM COMPONENTS (CONTINUED)

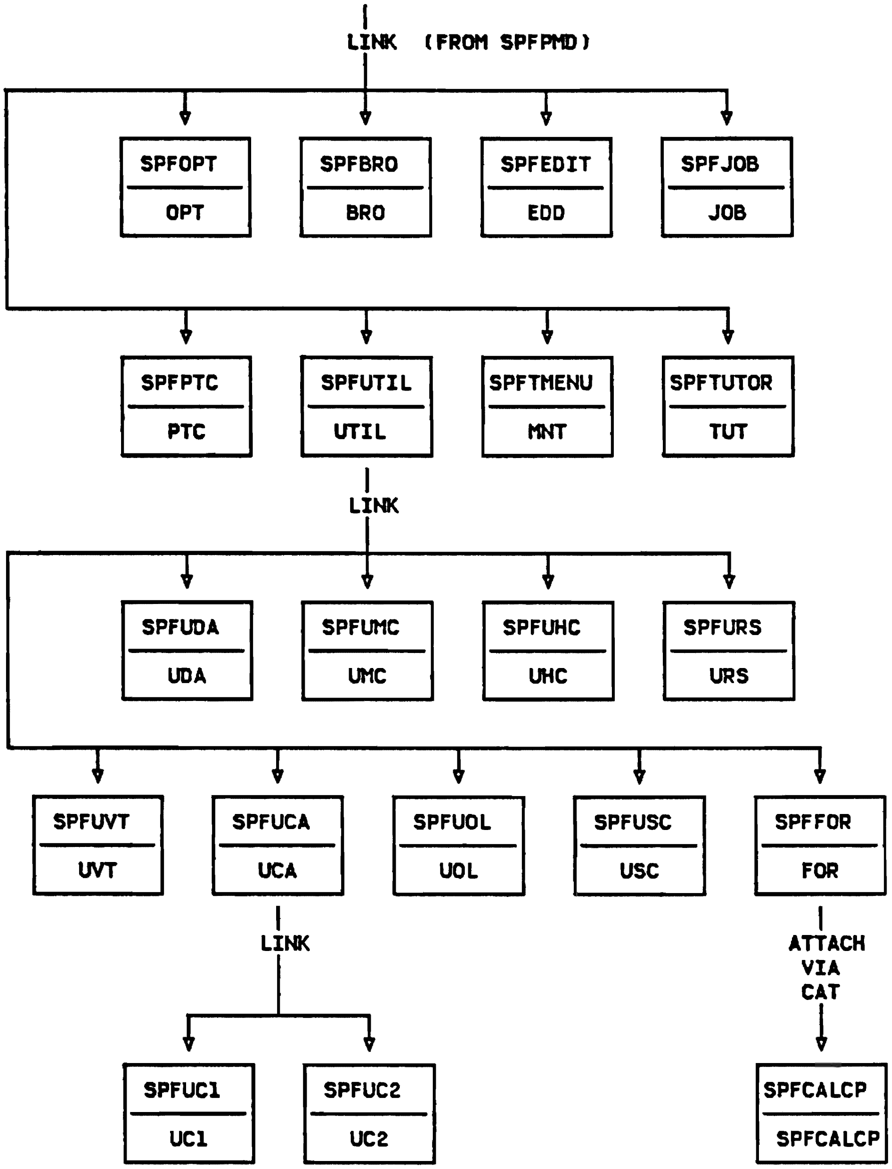
```
SPFURS ---------- RESET SPF STATISTICS LOAD MODULE  (EP: URS)
         URS ------ RESET STATISTICS UTILITY
         URSS ----- RESET STATISTICS UTILITY MEMBER SELECT

SPFUSC ---------- SCRIPT/VS UTILITY LOAD MODULE  (EP: USC)
         USC ------ SCRIPT/VS UTILITY

SPFUTIL --------- UTILITY DRIVER LOAD MODULE  (EP: UTIL)
         UTIL ----- UTILITY DRIVER

SPFUVT ---------- VTOC UTILITY LOAD MODULE  (EP: UVT)
         UVT ------ VTOC UTILITY

SPF3277 --------- 3277 TABLES LOAD MODULE  (EP: TT1)
         TT1 ------ 3277 TABLES

SPF3278 --------- 3278 TABLES LOAD MODULE  (EP: TT2)
         TT2 ------ 3278 TABLES

SPF3278C -------- 3278 CANADIAN(FRENCH) TABLES LOAD MODULE  (EP: TT3)
         TT3 ------ 3278 CANADIAN(FRENCH) TABLES
```

LOAD MODULE HIERARCHY

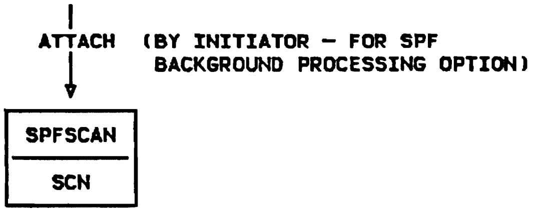
THE FOLLOWING FIGURE SHOWS THE SPF LOAD MODULE HIERARCHY. THE UPPER NAME IN EACH BOX IS THE LOAD MODULE NAME. THE LOWER NAME IN EACH BOX CONTAINS THE ENTRY POINT NAME.

THE FLOW BETWEEN THE LOAD MODULES IS ACCOMPLISHED VIA OS ATTACH, LOAD, OR LINK MACROS, AS SHOWN IN THE FIGURE.




LINK (FROM SPFPMD)
SPFOPT
OPT
SPFBRO
BRO
SPFEDIT
EDD
SPFJOB
JOB
SPFPTC
PTC
SPFUTIL
UTIL
SPFTMENU
MNT
SPFTUTOR
TUT
LINK
SPFUDA
UDA
SPFUMC
UMC
SPFUHC
UHC
SPFURS
URS
SPFUVT
UVT
SPFUCA
UCA
SPFUOL
UOL
SPFUSC
USC
SPFFOR
FOR
LINK
ATTACH
VIA
CAT
SPFUC1
UC1
SPFUC2
UC2
SPFCALCP
SPFCALCP

. . . .


ATTACH (BY INITIATOR – FOR SPF
BACKGROUND PROCESSING OPTION)
SPFSCAN
SCN

OBJECT MODULE DESCRIPTIONS

THIS SECTION CONTAINS DESCRIPTIONS OF ALL SPF OBJECT MODULES. THE FOLLOWING IS A DESCRIPTION OF THE FORMAT USED IN THIS SECTION:

XXXX - TITLE OF OBJECT MODULE.

PURPOSE:

THIS SECTION IS SPECIFIED ONLY FOR OBJECT MODULES THAT ARE PROGRAMS AND BRIEFLY DESCRIBES THE PURPOSE OF THE PROGRAM (OBJECT MODULE XXXX).

INVOKED WITH:

THIS SECTION IS SPECIFIED ONLY FOR OBJECT MODULES THAT ARE PROGRAMS. THE SECTION IDENTIFIES HOW OBJECT MODULE XXXX IS INVOKED. INVOCATION IS VIA EITHER CALL, LINK OR ATTACH. FOR MODULES OTHER THAN COMMON SUBROUTINES, THE INVOKING OBJECT MODULE IS GIVEN IN PARENTHESIS. FOR EXAMPLE:

LINK TO SPFZZZZ (FROM PMD).

REFERENCED VIA:

THIS SECTION IS SPECIFIED ONLY FOR OBJECT MODULES THAT ARE TABLES (NOT PROGRAMS) AND DESCRIBES HOW PROGRAMS GET ADDRESSABILITY TO THE TABLE.

CALLING SEQUENCE PARAMETERS:

THIS SECTION IS SPECIFIED ONLY FOR OBJECT MODULES THAT ARE PROGRAMS. THE SECTION LISTS ALL PARAMETERS PASSED TO THE MODULE BY THE INVOKING MODULE. THE COLUMNS ARE FROM LEFT TO RIGHT: PARAMETER NUMBER, PARAMETER NAME, FORMAT, USAGE, TITLE. THE FORMAT COLUMN EITHER CONTAINS THE FORMAT OF THE PARAMETER (E.G. "CHAR(8)") OR THE DATA AREAS SECTION DESCRIBING THE TABLE (E.G. "<TLD>") OR AN ASTERISK (*) INDICATING THAT THE PARAMETER IS DESCRIBED BELOW IN THE "WHERE" SECTION. FOR EXAMPLE:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MEMBER	CHAR(8)	INPUT	NAME OF PDS MEMBER
2.	DSNS	*	INPUT	DSNAME STRUCTURE

RETURN CODE:

THIS SECTION IS SPECIFIED ONLY FOR OBJECT MODULES THAT ARE PROGRAMS. THIS SECTION LISTS REGISTER 15 CONTENTS WHEN THE MODULE RETURNS TO THE INVOKING MODULE. FOR ROUTINES USING THE SPFPROC/SPFRETRN INTERFACE, THE RETURN CODE IS ALSO PLACED IN THE TLDRC FIELD.

NOTES:

THIS SECTION CONTAINS OTHER INFORMATION USEFUL FOR UNDERSTANDING THE OBJECT MODULE.

PURPOSE:

BCD IS THE BROWSE COMMAND DEFINITION TABLE. IT CONTAINS ONE ENTRY FOR EACH BROWSE PRIMARY COMMAND. THE COMMAND DEFINITION TABLE IS INPUT TO THE COMMON COMMAND PARSE (CCP) ROUTINE, AND IS USED FOR ERROR CHECKING AND FOR ORDERING PARAMETERS.

REFERENCED VIA:

THE ADDRESS OF THE BCD IS IN THE TSC. IT IS SYMBOLICALLY REFERENCED BY THE NAME "BCD" DEFINED IN SEGMENT "TSCDCLS".

NOTES:

- THE BCD TABLE IS TERMINATED WITH AN X'FF' CHARACTER.
- EACH COMMAND DEFINITION WITHIN THE TABLE IS TERMINATED WITH AN X'FE' CHARACTER.
- EACH PARAMETER DEFINITION WITHIN A COMMAND DEFINITION IS TERMINATED WITH AN X'FD' CHARACTER.
- IN ADDITION, THE LENGTH OF EACH COMMAND DEFINITION ENTRY IS PART OF THE ENTRY, AND THE NUMBER OF PARAMETERS DEFINED IS ALSO PART OF THE ENTRY.

- BROWSE DRIVER ROUTINE

PURPOSE:

BRO IS INVOKED BY PMD WHEN OPTION 1 IS SELECTED FROM THE PRIMARY OPTION MENU. IT ALLOCATES APPROPRIATE DATA SETS, DISPLAYS MEMBER LISTS, AND PERFORMS BROWSING.

INVOKED WITH:

LINK TO SPFBRO

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

CAT - COMMON ATTACH ROUTINE CAT

PURPOSE:

CAT IS USED TO ATTACH OTHER COMMANDS AND CLISTS UNDER SPF.

INVOKED WITH:

CALL TO CAT

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CBUF	*	INPUT	COMMAND BUFFER
3.	CCODE	FIXED(31)	OUT	COMPLETION CODE

WHERE

CBUF - IS THE TSO COMMAND BUFFER CONTROL BLOCK.

CCODE - IS SET TO THE COMPLETION CODE OF THE COMMAND OR CLIST. IF THE COMMAND COMPLETED DUE TO ATTENTION, THIS FIELD WILL BE SET TO ZERO.

RETURN CODES:

0 - NORMAL COMPLETION.

4 - ATTENTION TERMINATION.

8 - ABEND TERMINATION.

NOTES:

CAT WILL HANDLE COMMANDS AND CLISTS SUBJECT TO THE FOLLOWING RESTRICTIONS:

1. THE FOLLOWING COMMANDS ARE NOT SUPPORTED: LOGON, LOGOFF, SPF, TEST, AUTHORIZED COMMANDS, COMMANDS INVOKING AUTHORIZED PROGRAMS.
2. CLISTS MAY NOT INVOKE ANY OF THE RESTRICTED COMMANDS LISTED ABOVE.
3. CLIST ATTENTION EXITS ARE NOT SUPPORTED.
4. COMMAND PROCEDURE STATEMENT TERMIN IS NOT SUPPORTED.

CBC - COMMON BROWSE CLEANUP ROUTINE

PURPOSE:

CBC CLEANS UP (FREEMAINS) THE COMMON BROWSE TABLE (CBT). IT MUST BE CALLED, IF CBS IS CALLED TO SET UP THE CBT.

INVOKED WITH:

CALL TO CBC

CALLING SEQUENCE PARAMETERS:

```
1.  TLD       <TLD>       INPUT     LOGICAL DISPLAY TABLE

2.  CBTP      PTR(31)     INPUT     PTR TO BROWSE TABLE (CBT)
```

RETURN CODE:

0 - ALWAYS.

NOTES:

```
TO USE THE COMMON BROWSE ROUTINE:
    FIRST CALL CBS (COMMON BROWSE SETUP ROUTINE)
    THEN  CALL CBR (COMMON BROWSE ROUTINE) ONE OR MORE TIMES
    THEN  CALL CBC (COMMON BROWSE CLEANUP ROUTINE)
FIND STRINGS, AND BROWSE MODES (ASIS, HEX ETC) WILL BE REMEMBERED
FROM ONE CBR CALL TO THE NEXT (UNTIL CBC IS CALLED).
```

NOTE THAT CBR CAN BE CALLED WITHOUT EXPLICITLY CALLING CBS/CBC AND WITHOUT PASSING A VALID COMMON BROWSE TABLE (CBT). IN THIS CASE, CBR CALLS CBS/CBC AND SETS UP THE CBT INTERNALLY.

CBDSN - COMMON BUILD DSNAME ROUTINE • CBDSN

PURPOSE:

CBDSN IS PASSED PARAMETERS FROM A DATA SET MENU. IT BUILDS THE APPROPRIATE FULLY-QUALIFIED DATA SET NAME AND VERIFIES THAT IT IS NOT AN SPF DATA SET THAT IS CURRENTLY OPEN. IT STORES THE DATA SET NAME AND LENGTH IN A STRUCTURE SUITABLE FOR USE WITH DAIR.

INVOKED WITH:

CALL TO CBDSN

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MPROJ	CHAR(8)	INPUT	PROJECT NAME FROM MENU
3.	MLIBR	CHAR(8)	INPUT	LIBRARY NAME FROM MENU
4.	MTYPE	CHAR(8)	INPUT	TYPE NAME FROM MENU
5.	MMEMB	CHAR(8)	INPUT	MEMBER NAME FROM MENU
6.	MDSN	CHAR(56)	INPUT	"OTHER" DATA SET NAME FROM MENU
7.	DSNS	CHAR(46)	OUTPUT	SELECTED DATA SET NAME STRUCTURE
8.	MEMBER	CHAR(8)	OUTPUT	SELECTED MEMBER
9.	MSGID	CHAR(4)	OUTPUT	ERROR/PROMPTING ERROR MESSAGE ID
10.	PARM#	FIXED(31)	OUTPUT	PARAMETER NUMBER FOR CURSOR

WHERE

DSNS - IS THE DATA SET NAME STRUCTURE (STANDARD DAIR FORMAT).

```
E.G.,   DCL 1 DSNS,
              2 DSNL FIXED(15),  /* DSNAME LENGTH */
              2 DSN  CHAR(44);   /* DSNAME        */
```

MEMBER - IS THE USER-SPECIFIED MEMBER NAME (IF ANY) WHICH IS EXTRACTED FROM "OTHER" DSNAME OR COPIED FROM THE "MMEMB" INPUT PARAMETER.

MSGID - IS THE ID OF AN ERROR OR PROMPTING MESSAGE (IF ANY). IF NO ERROR IS DETECTED, THE MSGID IS NOT CHANGED.

PARM# - IS THE PARAMETER NUMBER ASSOCIATED WITH THE ERROR OR PROMPTING MESSAGE, FOR CALLING MERR. NOTE: A PARM# IS ALWAYS RETURNED, EVEN IF THERE WAS NO ERROR.

RETURN CODE:

0 - DSNAME IS NOT A GENERATION DATASET FORMAT NAME, EG. A.B(-1)

4 - DSNAME IS A GENERATION DATASET FORMAT NAME.

(CONTINUED ON NEXT PAGE)

NOTES:

THE MSGID AND PARM# VALUES THAT MAY BE RETURNED BY CBDSN ARE:

MSGID		MEANING	PARM#
'G002'	-	ENTER PROJECT NAME	1
'G003'	-	ENTER LIBRARY NAME	2
'G004'	-	ENTER TYPE QUALIFIER	3
'G054'	-	DATA SET IS OPEN	1 OR 5*
'G090'	-	MISSING QUOTE	5
'G091'	-	DSN LENGTH ERROR	5
'G092'	-	INVALID MEMBER NAME	5
'G093'	-	IMBEDDED BLANKS IN DSN	5
'G094'	-	GDS NOT CATALOGUED	5
(UNCHANGED)	-	NO ERROR	1 OR 5*

* PARM# = 1 IF PROJECT, LIBRARY, TYPE WAS SPECIFIED
* PARM# = 5 IF "OTHER" DSNAME WAS SPECIFIED

MSGID 'G054' IS RETURNED ONLY IF AN SPF DATA SET (LIST, LOG, TEMPLIST, TEMPCNTL, EDIT RECOVERY) WAS SPECIFIED AND IT IS OPEN. IN THIS CASE, A VALID DATA SET NAME STRUCTURE HAS BEEN RETURNED, AND THE USING PROGRAM MAY CHOOSE TO IGNORE THE 'G054' CONDITION.

CBDSN DOES NOT PRODUCE ERRORS FOR MANY TYPES OF UNACCEPTABLE DATA SET NAMES, I.E. INVALID CHARACTERS OR QUALIFIERS MORE THAN EIGHT CHARACTERS LONG. THESE KINDS OF ERRORS WILL BE CAUGHT BY THE DYNAMIC ALLOCATION ROUTINE (DAIR) WHICH IS CALLED FROM CDAIR.

CBF - COMMON BROWSE FIND ROUTINE CBF

PURPOSE:

CBF IS CALLED BY COMMON BROWSE (CBR) IF A FIND COMMAND IS ENTERED AS A PRIMARY COMMAND, OR IF THE FIND PF KEY IS PRESSED. IT DECODES THE FIND COMMAND (IF REQUIRED), AND PERFORMS THE SEARCH TO FIND THE REQUIRED STRING.

INVOKED WITH:

CALL TO CBF

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CBT	*	IN/OUT	COMMON BROWSE TABLE

WHERE

CBT - IS THE COMMON BROWSE TABLE, WHICH IS INITIALIZED BY CBS AND USED BY CBF, CBR, CBG AND CBC.

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

CBG - COMMON BROWSE GET ROUTINE CBG

PURPOSE:

CBG IS CALLED BY COMMON BROWSE (CBR) TO GET A SPECIFIED RELATIVE RECORD. IT ATTEMPTS TO FIND THE RECORD ALREADY IN MAIN MEMORY (IN AN I/O BUFFER) AND IF IT CANNOT, CALLS COMMON DATASET GET (CDG) TO READ A LOGICAL RECORDS.

INVOKED WITH:

CALL TO CBG

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CBT	*	IN/OUT	COMMON BROWSE TABLE

WHERE

CBT - IS THE COMMON BROWSE TABLE, WHICH IS INITIALIZED BY CBS AND USED BY CBF, CBR, CBG AND CBC.

RETURN CODE:

0 - ALWAYS.

NOTES:

CBR BUILDS A TRACK/RECORD TABLE AS RECORDS ARE READ, TO ENABLE IT TO COMPUTE A TTRN FOR ANY LOGICAL RECORD ALREADY READ. IT ALSO MAINTAINS IN THE COMMON BROWSE TABLE AND ARRAY OF POINTERS AND LENGTHS TO LOGICAL RECORDS THAT ARE ALREADY IN MAIN MEMORY IN AN I/O BUFFER. THESE FUNCTIONS IMPROVE PERFORMANCE AND MINIMIZE THE AMOUNT OF ACTUAL I/O REQUIRED TO GO TO SPECIFIC LOCATIONS IN THE DATA SET OR MEMBER.

SPECIAL INTERFACES TO COMMON DATASET GET (CDG) ARE USED BY CBG TO ENABLE CBG TO DETERMINE THE RELATIVE TTRN OF EACH LOGICAL RECORD, AND TO POINT DIRECTLY TO THE CORRECT TTR WHEN I/O IS NOT SEQUENTIAL.

PURPOSE:

CBR IS USED TO DISPLAY (AND ALLOW SCROLLING) OF A SEQUENTIAL DATA SET OR MEMBER OF A PARTITIONED DATA SET.

INVOKED WITH:

CALL TO CBR

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	INPUT	FILE DEFINITION TABLE
3.	CBTP	PTR(31)	INPUT	PTR TO COMMON BROWSE TABLE

WHERE

```
    TFD     - IS A FILE CONTROL BLOCK FOR THE DATA SET TO BE BROWSED.
              THE ASSOCIATED DCB MUST BE OPEN AND, FOR A PDS, A FIND
              MUST HAVE BEEN ISSUED FOR THE MEMBER.

    CBTP    - EITHER ZERO (0), OR A POINTER TO THE COMMON BROWSE
              TABLE (CBT).
                ZERO - IS PASSED IF CBR IS TO CALL CBS/CBT TO GET
                       AND RELEASE THE CBT.
                PTR  - IS PASSED IF CBS HAS BEEN CALLED TO SET UP
                       THE COMMON BROWSE TABLE, AND CBT WILL BE
                       CALLED TO CLEAN UP THE CBT.
```

RETURN CODE:

0 - BROWSE COMPLETED SUCCESSFULLY.

4 - MEMBER OR DATA SET DOES NOT CONTAIN ANY RECORDS.

8 - I/O ERROR ATTEMPTING TO READ FIRST RECORD.

12 - INSUFFICIENT MAIN STORAGE FOR BUFFERS ETC.

NOTES:

THIS MODULE HAS REPLACED THE SPF VERSION 2.1 CBRO SUBROUTINE.

CBR IS CALLED BY BROWSE (SPF OPTION 1), AND VARIOUS UTILITIES.

CBR ASSUMES THAT THE FIRST TWO LINES OF THE TLS HAVE ALREADY BEEN SET UP. IT DOES NOT CHANGE THESE LINES EXCEPT TO DISPLAY COLUMN NUMBERS. CBR ALSO ASSUMES THAT THE INITIAL SCROLL AMOUNT HAS ALREADY BEEN STORED IN THE TLD (TLDSCAMT).

CBR NEVER CLOSES OR FREES THE DATA SET WHICH IT IS PASSED.

CBS - COMMON BROWSE SET UP ROUTINE CBS

PURPOSE:

CBS SET UPS (ACQUIRE AND INITIALIZE) THE COMMON BROWSE TABLE (CBT). THIS FUNCTION CAN BE PERFORMED BEFORE CALLING CBR (ONE OR MORE TIMES). IF CBS IS CALLED, CBC MUST BE CALLED TO CLEANUP (RELEASE) THE CBT. CBS RETURNS A POINTER TO THE CBT TO THE CALLER.

INVOKED WITH:

CALL TO CBS

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CBTP	PTR(31)	OUTPUT	PTR TO BROWSE TABLE (CBT)

RETURN CODE:

0 -- ALWAYS.

NOTES:

```
TO USE THE COMMON BROWSE ROUTINE:
    FIRST CALL CBS (COMMON BROWSE SETUP ROUTINE)
    THEN  CALL CBR (COMMON BROWSE ROUTINE) ONE OR MORE TIMES
    THEN  CALL CBC (COMMON BROWSE CLEANUP ROUTINE)
FIND STRINGS, AND BROWSE MODES (ASIS, HEX ETC) WILL BE REMEMBERED
FROM ONE CBR CALL TO THE NEXT (UNTIL CBC IS CALLED).
```

NOTE THAT CBR CAN BE CALLED WITHOUT EXPLICITLY CALLING CBS/CBC AND PASSING A VALID COMMON BROWSE TABLE (CBT). IN THIS CASE, CBR CALLS CBS/CBC AND SETS UP THE CBT INTERNALLY.

CCB - COMMON COMMAND BUILD ROUTINE CCB

PURPOSE:

CCB IS USED TO PROCESS A MENU / PROC PAIR. IF THE PROC CONTAINS A COMMAND STATEMENT, THE CORRESPONDING COMMAND WILL BE BUILT AND RETURNED TO THE CALLER.

INVOKED WITH:

CALL TO CCB

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MHAPM	*	INPUT	MHA PARAMETER LIST
3.	CMDPM	*	OUTPUT	COMMAND BUILD AREA
4.	MSGPM	CHAR(104)	OUTPUT	MESSAGE BUILD AREA
5.	DDNPM	CHAR(548)	OUTPUT	DDNAME / DSNAME BUILD AREA
6.	KNTPM	CHAR(*)	INPUT	KEYBLOCK AREA
7.	VNTPM	CHAR(*)	INPUT	KEYWORD VALUE AREA

WHERE

MHAPM - IS AN AREA FORMATTED AND INITIALIZED EXACTLY THE SAME AS THE INPUT PARAMETER LIST TO MHA. INITIALIZATION OF THIS AREA WILL OCCUR AUTOMATICALLY BY A PREVIOUS CALL TO CMB.

CMDPM - IS AN AREA WHERE THE COMMAND WILL BE BUILT.
IT IS FORMATTED BY CCB TO CONTAIN:
2 BYTE - LENGTH VALUE,
2 BYTE - OFFSET VALUE,
VARIABLE LENGTH - COMMAND STRING, (MAX LENGTH OF 250 BYTES).
IN THIS FORMAT, IT MAY BE PASSED DIRECTLY TO CAT FOR COMMAND EXECUTION.

MSGPM - IS AN AREA WHERE ERROR MESSAGE DATA IS RETURNED.
IT IS MEANINGFUL ONLY IF THE CCB RETURN CODE IS NON-ZERO. IT'S FORMAT IS:
2 BYTE - MESSAGE ID,
2 BYTE - CURSOR POSITION,
24 BYTE - SHORT MESSAGE,
72 BYTE - LONG MESSAGE.

(CONTINUED ON NEXT PAGE)

DDNPM - IS AN OPTIONAL PARAMETER AND DEFINES AN AREA WHERE DDNAME AND DATA SET NAME DATA IS RETURNED. IF THE PROC INVOKED BY CCB CONTAINS ALLOC AND/OR FREEDSN CONTROL CARDS, THE ASSOCIATED DDNAME (FOR ALLOC) AND THE DATA SET NAME (FOR FREEDSN) INFORMATION IS RETURNED. THE FORMAT OF THIS AREA IS:

```
    4   BYTE - DDNAME INDEX (NUMBER OF DDNAME'S BELOW)
    80  BYTE - DDNAME LIST (UP TO TEN 8 BYTE DDNAME'S)
    4   BYTE - DATA SET NAME INDEX (NUMBER OF DATA SET
               NAMES BELOW)
    460 BYTE - DATA SET NAME LIST (UP TO TEN 46 BYTE
               DATA SET NAMES FORMATTED AS FOLLOWS:
               2  BYTE - DATA SET NAME LENGTH
               44 BYTE - DATA SET NAME NAME (EBCDIC)
```

KNTPM - IS AN OPTIONAL PARAMETER AND DEFINES AN AREA RESEMBLING A KVBLOCK. THAT IS, IT CONTAINS THE NAMES OF KEYWORDS IN THE SAME FORMAT AS A KVBLOCK AREA. IF A KEYWORD VALUE CANNOT BE OBTAINED FROM THE NAME BEING PROCESSED, THIS AREA IS SEARCHED PRIOR TO CALLING CKVGET.

VNTPM - IS AN OPTIONAL PARAMETER (REQUIRED IF KNTPM IS PRESENT) AND DEFINDS AN AREA OF KEYWORD VALUES. IF A KEYWORD EXISTS IN THE KNTPM AREA, ITS CORRESPONDING VALUE IS OBTAINED FROM THIS AREA.

RETURN CODE:

0 - SUCCESSFUL.

4 - ERROR ENCOUNTERED. MSGPM AREA HAS DATA ABOUT THE NATURE OF THE ERROR.

NOTES:

CCB IS INTENDED TO BE USED AFTER A PREVIOUS CALL TO CMB. FOR THE PURPOSE OF SUBSTITUTING KEYWORD VALUES, CCB ASSUMES THAT THE MENU HANDLER BUFFER (MHAF) AND THE MENU ACTION ENTRIES (MHAFACTN) ARE BOTH PROPERLY INITIALIZED (WHICH CMB WILL DO).

IF A PROC KEYWORD VALUE SUBSTITUTION IS REQUIRED FOR AN ITEM WHICH DOES NOT EXIST ON THE CORRESPONDING MENU, CCB WILL CALL CKVGET IN AN EFFORT TO OBTAIN THE MISSING KEYWORD VALUE. IT WILL FIRST DETERMINE IF OPTIONAL PARAMETERS 6 AND 7 ARE PRESENT. IF THEY ARE, THE KNTPM AREA WILL BE SCANNED FOR THE KEYWORD. IF FOUND, THE CORRESPONDING KEYWORD VALUE WILL BE OBTAINED FROM THE VNTPM AREA. IF PARAMETERS 6 AND 7 ARE NOT PRESENT, OR IF THE CORRECT KEYWORD COULD NOT BE FOUND IN THE KNTPM AREA, CCB WILL CALL CKVGET TO DETERMINE THE VALUE OF THE KEYWORD IF IT EXISTS IN THE TKV.

CCD - COMMON CONVERT DIRECTORY ROUTINE CCD

PURPOSE:

CCD CONVERTS A DIRECTORY ENTRY TO EBCDIC FOR PRINTING OR DISPLAYING.

INVOKED WITH:

CALL TO CCD

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	BLDLENT	*	INPUT	BLDL LIST ENTRY
3.	SPFFLAG	CHAR(1)	OUTPUT	SPF DIRECTORY ENTRY FLAG
4.	NAME	CHAR(8)	OUTPUT	MEMBER NAME
5.	LIB	CHAR(1)	OUTPUT	LIBRARY NUMBER
6.	VERSMOD	CHAR(7)	OUTPUT	VERSION/MODIFICATION LEVEL
7.	CDATE	CHAR(8)	OUTPUT	CREATION DATE
8.	MDATE	CHAR(8)	OUTPUT	LAST MODIFIED DATE
9.	MTIME	CHAR(5)	OUTPUT	LAST MODIFIED TIME
10.	CURLIN	CHAR(5)	OUTPUT	CURRENT NUMBER LINES
11.	INITLIN	CHAR(5)	OUTPUT	INITIAL NUMBER LINES
12.	MODLIN	CHAR(5)	OUTPUT	MODIFIED NUMBER LINES
13.	USERID	CHAR(7)	OUTPUT	USER ID

WHERE

BLDLENT - IS AN ENTRY FROM THE OS BLDL CONTROL BLOCK.

SPFFLAG - INDICATES IF THE ENTRY IS IN SPF LIBARAY ENTRY FORMAT:
'00' HEX - NOT SPF FORMAT
'01' HEX - SPF FORMAT

LIB - CONCATINATION LEVEL NUMBER OR '-' IF MEMBER NOT FOUND.

VERSMOD - VERSION/MODIFICATION LEVEL - ' VV.MM ' EBCDIC.

CDATE - CREATION DATE - 'YY/MM/DD' EBCDIC.

MDATE - LAST MODIFIED DATE - 'YY/MM/DD' EBCDIC.

MTIME - LAST MODIFIED TIME - 'HH:MM' EBCDIC.

USERID - ID OF USER WHO LAST MODIFIED THE MEMBER.

RETURN CODES:

0 - ALWAYS.

NOTES:

NONE.

CCP - COMMON COMMAND PARSE CCP

PURPOSE:

CCP PARSES A COMMAND THAT HAS BEEN PRESCANNED BY CCS (COMMON COMMAND SCAN). CCP DETECTED CERTAIN ERRORS, AND REORDERS THE PARAMETERS TO MAKE THEIR USE BY PROCESSING PROGRAMS EASIER.

INVOKED WITH:

CALL TO CCP

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TCS	<TCS>	IN/OUT	COMMAND SCAN TABLE
3.	TCD	*	INPUT	COMMAND DEFINITION TABLE
4.	ERRCODE	CHAR(4)	OUTPUT	ERROR CODE
5.	ERRPTR1	PTR(31)	OUTPUT	ERROR MESSAGE PTR #1.
6.	ERRPTR2	PTR(31)	OUTPUT	ERROR MESSAGE PTR #2.
7.	ERRPTR3	PTR(31)	OUTPUT	ERROR MESSAGE PTR #3.

WHERE

TCS - IS A COMMAND SCAN TABLE THAT HAS BEEN SET UP BY CCS.

TCD - IS THE COMMAND DEFINITION TABLE THAT IS USED IN PROCESSING THE TCS.

ERRCODE - IS AN ERROR MESSAGE CODE THAT CAN BE PASSED TO MERR OR CERR. NOT USED IF NO ERRORS ARE DETECTED.

ERRPTR1 - IS A POINTER THAT MAY BE SET UP IF AN ERROR CODE IS ALSO SET UP.

RETURN CODE:

0 - NORMAL RETURN.

'N' - A NON-ZERO CODE OF 'N' INDICATES THAT AN ERROR WAS DETECTED IN PROCESSING THE N'TH PARAMETER IN THE COMMAND.

NOTES

NONE.

CCS - COMMON COMMAND SCAN CCS

PURPOSE:

CCS IS USED TO SCAN A PRIMARY COMMAND. IT DETERMINES THE NUMBER OF WORDS IN THE COMMAND AND BUILDS AN ARRAY WITH ONE ENTRY FOR EACH WORD. THE ENTRY CONTAINS A PTR TO THE WORD, THE LENGTH OF THE WORD, THE KEY/WORD CODE, IF THE WORD WAS FOUND IN THE PRIMARY COMMAND WORD TABLE, FLAGS TO INDICATE THE TYPE OF WORD (OR STRING) AND FLAGS TO INDICATE ANY ERROR CONDITIONS THAT WERE FOUND.

INVOKED WITH:

CALL TO CCS

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TCS	*	IN/OUT	COMMAND SCAN TABLE

WHERE

TCS - COMMAND SCAN TABLE. FIELDS TCSCIP, TCSCISZ, AND TCSWDCT (MAX POSSIBLE VALUE) MUST BE FILLED IN BY THE CALLER. CCS WILL RETURN TCSWDCT (ACTUAL) AND AN TSCWDS ENTRY FOR EACH WORD (OR STRING) FOUND.

RETURN CODES:

0 - NORMAL RETURN.

'N' - A NON-ZERO CODE OF 'N' INDICATES THAT AN ERROR WAS DETECTED FOR WORD 'N' IN THE COMMAND. AN ERROR FLAG WILL BE SET IN THE 'N'TH ENTRY TO INDICATE THE TYPE OF ERROR. POSSIBLE ERRORS ARE:
- TCSIVSIZ = ON - HEX STRING CONTAINS ODD NUMBER OF DIGITS.
- TCSIVHEX = ON - HEX STRING CONTAINS NON-HEX CHARACTERS.

NOTES:

CCS MOVES THE LINE TO A LOCAL AREA AND TRANSLATES IT TO UPPER CASE BEFORE SCANNING IT.

POINTERS FOR QUOTED STRINGS POINT TO THE FIRST CHARACTER OF THE STRING, AND NOT TO THE QUOTE OR PREFIX CHARACTER. THE LENGTH OF A QUOTED STRING IS THE NUMBER OF CHARACTERS CONTAINED WITHIN THE QUOTES.

CDA - COMMON DATASET ALLOCATE ROUTINE CDA

PURPOSE:

CDA IS USED TO ALLOCATE A SINGLE DATA SET OR TO ALLOCATE AND CONCATENATE FROM TWO TO FOUR DATA SETS. CDA WILL HANDLE ALLOCATION OF EXISTING DATA SETS ONLY. OPTIONALLY, CDA WILL CLOSE AND FREE THE DATA SET(S) ASSOCIATED WITH A TFD, AND THEN ALLOCATE THE NEW FILE.

INVOKED WITH:

CALL TO CDA

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	SMSG	CHAR(24)	OUTPUT	SHORT ERROR MESSAGE
4.	LMSG	CHAR(72)	OUTPUT	LONG ERROR MESSAGE

WHERE

TFD - FILE DEFINITION TABLE FOR THE FILE BEING ALLOCATED.

SMSG - RETURN AREA FOR SHORT ERROR MESSAGE IF ERROR OCCURRED.

LMSG - RETURN AREA FOR LONG ERROR MESSAGE IF ERROR OCCURRED.

RETURN CODES:

0 - NORMAL RETURN.

1 - UNACCEPTABLE OR MISSING "PROJECT" (FAILED CBDSN).
- PROBLEM FREEING OLD PROJECT.LIB1.TYPE.
- PROBLEM ALLOCATING PROJECT.LIB1.TYPE.
- PROBLEM CONCATENATING DATASETS.

2 - UNACCEPTABLE OR MISSING "LIB1" (FAILED CBDSN).

3 - UNACCEPTABLE OR MISSING "LIB2" (FAILED CBDSN).
- PROBLEM FREEING OLD PROJECT.LIB2.TYPE.
- PROBLEM ALLOCATING PROJECT.LIB2.TYPE.
- CONCATENATION DATASETS HAVE UNLIKE DSORG.

4 - UNACCEPTABLE OR MISSING "LIB3" (FAILED CBDSN).
- PROBLEM FREEING OLD PROJECT.LIB3.TYPE.
- PROBLEM ALLOCATING PROJECT.LIB3.TYPE.
- CONCATENATION DATASETS HAVE UNLIKE DSORG.

5 - UNACCEPTABLE "LIB4" (FAILED CBDSN).
- PROBLEM FREEING OLD PROJECT.LIB4.TYPE.
- PROBLEM ALLOCATING PROJECT.LIB4.TYPE.
- CONCATENATION DATASETS HAVE UNLIKE DSORG.

6 - UNACCEPTABLE OR MISSING "TYPE" (FAILED CBDSN).

7 - UNACCEPTABLE "OTHER" DSN (FAILED CBDSN).
- PROBLEM FREEING "OTHER" DATASET.
- PROBLEM ALLOCATING "OTHER" DATASET.

8 - UNACCEPTABLE VOLUME SERIAL.

(CONTINUED ON NEXT PAGE)

NOTES:

IF RETURN CODE > 0 THEN MESSAGES ARE RETURNED IN SMSG AND LMSG AND NO DATASET(S) ASSOCIATED WITH THE TFD WILL BE ALLOCATED.

IF REQUESTED (TFDMENUP ¬= 0), CDA WILL BUILD THE DSNAME(S) BY CALLING CBDSN.

THE DA08 AND DA0C BLOCKS WILL BE CONSTRUCTED FOR THE DURATION OF CDA ONLY AND CDAIR WILL BE CALLED TO PERFORM THE ALLOCATE AND CONCATENATE FUNCTIONS.

ERROR MESSAGES WILL BE OBTAINED FROM CDERR FOR ALLOCATION ERRORS, AND CMSG FOR CBDSN ERRORS.

IF CDA IS CALLED WITH A TFD IN WHICH A FILE IS CURRENTLY DESCRIBED, (I.E., TFDDDN(1) IS NOT ZERO BITS), THEN CDA WILL COMPARE THE IDENTIFICATION (DSN'S, VOLUME, DISPOSITION, AND PASSWORD) OF THE CURRENT AND REQUESTED FILES AND, IF DIFFERENT, WILL CLOSE AND FREE THE CURRENT ALLOCATION, AND ALLOCATE THE REQUESTED FILE.

IF CDA IS CALLED WITH MENU DATA (TFDMENUP ¬= 0), THEN ALLOCATION WILL BE FOR THE DATASET(S) DESCRIBED IN THE MENU DATA. OTHERWISE ALLOCATION WILL BE FOR THE DATASET DESCRIBED IN TFDDSNP(1) -> TFDDSNS.

PURPOSE:

CDAIR SERVES AS THE COMMON INTERFACE WITH THE DAIR MODULE "IKJEFD00". IT HANDLES THE SPECIAL CASE OF A DATA SET ALLOCATED SHARED, BUT TO BE TREATED BY PCF (PROGRAM CONTROL FACILITY) AS OLD FOR PURPOSES OF VOLUME VERIFICATION.

INVOKED WITH:

CALL TO CDAIR

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	BLOCK	*	IN/OUT	DAIR BLOCK

WHERE

BLOCK - IS A TSO DYNAMIC ALLOCATION INTERFACE ROUTINE (DAIR) CONTROL BLOCK (DA08,DA18, ETC.).

RETURN CODE:

RETURN CODE THAT IS RETURNED IN REG 15 FROM "IKJEFD00".

NOTES:

IF A DA08 (ALLOCATE) BLOCK IS ENCOUNTERED, AND IF BOTH THE SHR (DA08SSHR) AND OLD (DA08SOLD) FLAG BITS ARE ON, A SPECIAL PCF INTERFACE IS RECOGNIZED, SINCE THIS COMBINATION IS INVALID AS INPUT TO DAIR. IN THIS CASE, THE DA08SOLD BIT IS TURNED OFF BY CDAIR, AND A '01'X IS STORED IN THE FIRST BYTE OF THE ECTSCMD FIELD AS A SIGNAL TO PCF THAT THIS SHR REQUEST SHOULD BE TREATED FOR VOLUME VERIFICATION PURPOSES AS AN OLD REQUEST.

IF A DA08 (ALLOCATE) BLOCK IS ENCOUNTERED, AND IF THE VOLUME SERIAL FIELD IS BLANK (REQUEST FOR A CATALOGED DATA SET), A LOCATE MACRO IS ISSUED BY CDAIR TO FIND THE VOLUME SERIAL, AND CDT IS CALLED TO DETERMINE THE UNIT TYPE. THE VOLUME AND UNIT VALUES ARE PLACED IN THE DA08 BLOCK BEFORE DAIR IS CALLED.

IF A DA18 (FREE) BLOCK IS ENCOUNTERED, AND IF BOTH THE DELETE (DA18NDL) AND UNCATALOG (DA18NUC) BIT FLAGS ARE SET, THE DATA SET WILL BE UNCATALOGED BY A CATALOG MACRO REQUEST AFTER DAIR HAS BEEN CALLED TO SCRATCH AND DEALLOCATE THE DATA SET. IN THIS CASE, THE THE UNCATALOG FLAG BIT IS TURNED OFF BEFORE THE BLOCK IS PROCESSED BY DAIR.

CDAIR CAN EITHER LINK TO "IKJEFD00" OR BRANCH DIRECTLY TO THE ADDRESS STORED IN TSIDAIRP. THE LATER CASE IS THE NORMAL CASE AND RESULTS IN IMPROVED PERFORMANCE. TSIDAIRP IS SET UP BY SMI WHICH LOADS "IKJEFD00" AND THEN STORES ITS ADDRESS. IF TSIDAIRP < 4 (BECAUSE SMI DID NOT LOAD "IKJEFD00") THEN LINKING TAKES PLACE. THIS OPTION COULD SAVE STORAGE AT THE EXPENSE OF CPU OVERHEAD IF THE DAIR MODULES WERE NOT ALREADY IN THE LINK PACK AREA.

CDATE - COMMON CONVERT DATE ROUTINE CDATE

PURPOSE:

CDATE CONVERTS A FIXED POINT OR PACKED DECIMAL DATE TO EBCDIC.

INVOKED WITH:

CALL TO CDATE

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	DATEIN	CHAR(4)	INPUT	INPUT DATE
3.	DATEOUT	CHAR(8)	OUTPUT	OUTPUT DATE

WHERE

DATEIN - THE INPUT DATE IN ONE OF THE FOLLOWING FORMATS:

PACKED DECIMAL - 'CCYYDDDS' HEX:
CC = '00' HEX
YY = DECIMAL DIGITS FOR YEAR
DDD = DECIMAL DIGITS FOR DAY OF YEAR
S = PACKED DECIMAL SIGN (IGNORED)

FIXED POINT - 'CCYYDDDD' HEX:
CC = 'FF' HEX
YY = BINARY VALUE FOR YEAR
DDDD = BINARY VALUE FOR DAY OF YEAR

DATEOUT - THE OUTPUT DATE - 'YY/MM/DD' EBCDIC:
YY = EBCDIC YEAR
MM = EBCDIC MONTH
DD = EBCDIC DAY

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

CDC - COMMON DATASET CLOSE CDC

PURPOSE:

CDC IS USED TO CLOSE A FILE. THE FILE IS EXPECTED TO HAVE PREVIOUSLY OPENED BY CALLING CDO. IN ADDITION TO CLOSING THE DCB, THE CDC WILL OPTIONALLY CALL CRELS TO DEQ THE DATA SET RESOURCE AND OPTIONALLY CALL CSM TO FREEMAIN THE I/O BUFFER.

INVOKED WITH:

CALL TO CDC

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	MSGID	CHAR(4)	OUTPUT	ERROR MESSAGE ID (NOT USED)

WHERE

TFD - FILE DEFINITION TABLE FOR THE FILE BEING ALLOCATED.

MSGID - ERROR MESSAGE ID IS PROVIDED FOR FUTURE POSSIBLE USE.

RETURN CODES:

0 - ALWAYS.

NOTES:

CDC WILL CLEAR THE DCB I/O ERROR SWITCH BEFORE CLOSING THE DCB.

CDERR - COMMON DAIR ERROR ROUTINE CDERR

PURPOSE:

CDERR IS CALLED AFTER AN ERROR IS ENCOUNTERED FROM THE TSO "DAIR" ROUTINE. IT DETERMINES THE SPF MESSAGE ID WHICH DESCRIBES THE ERROR, THEN OPTIONALLY INVOKES THE SPECIFIED ERROR HANDLING ROUTINE.

INVOKED WITH:

CALL TO CDERR

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	RETCODE	FIXED(15)	INPUT	DAIR RETURN CODE
3.	DA08	*	INPUT	DA08 CONTROL BLOCK
4.	SHORT	CHAR(24)	OUTPUT	SHORT (LEVEL 1) ERROR MESSAGE
5.	LONG	CHAR(72)	OUTPUT	LONG (LEVEL 2) ERROR MESSAGE

WHERE

RETCODE - THE SAVED CONTENTS OF REGISTER 15 ON RETURN FROM DAIR.

DA08 - IS THE TSO DAIR DA08 CONTROL BLOCK, WHICH IS USED TO ALLOCATE A FILE.

SHORT - THIS PARAMETER IS USED TO RETURN SHORT ERROR MESSAGES.

LONG - THIS PARAMETER IS USED TO RETURN LONG ERROR MESSAGES.

RETURN CODE:

0 - ALWAYS.

NOTES:

THE SPF MESSAGES IDS USED BY CDERR TO GENERATE THE SHORT AND LONG ERROR MESSAGES ARE D001 THROUGH D022.

CDF - COMMON DATASET FREE ROUTINE CDF

PURPOSE:

CDF IS USED TO FREE ALLOCATION(S) OF A FILE PREVIOUSLY ALLOCATED USING CDA. THE FILE MAY CONSIST OF A SINGLE DATA SET OR A CONCATENATION OF TWO TO FOUR DATASETS. CDF WILL CALL CDC TO CLOSE THE DCB, IF NECESSARY, BEFORE CALLING CDAIR TO FREE THE ALLOCATION.

INVOKED WITH:

CALL TO CDF

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	SMSG	CHAR(24)	OUTPUT	SHORT ERROR MESSAGE
4.	LMSG	CHAR(72)	OUTPUT	LONG ERROR MESSAGE

WHERE

TFD - FILE DEFINITION TABLE FOR THE FILE BEING FREED.

SMSG - SHORT ERROR MESSAGE RETURNED WHEN RETURN CODE IS NOT 0.

LMSG - LONG ERROR MESSAGE RETURNED WHEN RETURN CODE IS NOT 0.

RETURN CODES:

0 - NORMAL RETURN.

1 - PROBLEM FREEING PROJECT.LIB1.TYPE.

2 - PROBLEM FREEING PROJECT.LIB2.TYPE.

3 - PROBLEM FREEING PROJECT.LIB3.TYPE.

4 - PROBLEM FREEING PROJECT.LIB4.TYPE.

5 - PROBLEM FREEING "OTHER" DATASET.

NOTES:

CDF WILL FREE THE DDNAMES SAVED IN THE TFD BY CDA, BY CONSTRUCTING A DA18 BLOCK AND CALLING CDAIR FOR EACH DDNAME TO BE FREED.

IF A DAIR ERROR OCCURS, AN ERROR MESSAGE WILL BE PLACED IN THE SMSG AND LMSG FIELDS. AN ATTEMPT WILL BE MADE TO FREE ALL DDNAMES IN THE EVENT THAT ANY ERROR IS ENCOUNTERED.

CDG - COMMON GET RECORD ROUTINE CDG

PURPOSE:

CDG IS USED TO READ A LOGICAL RECORD FROM A SEQUENTIAL DATA SET OR MEMBER OF A PDS AND OPTIONALLY NOTE THE TTR OF EACH BLOCK READ. CDG MAY ALSO BE USED FOR UPDATE IN PLACE OF A PDS.

INVOKED WITH:

CALL TO CDG

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE

WHERE

TFD - IS A STRUCTURE WHICH INCLUDES THE FOLLOWING PARAMETERS:

TFDRECP	PTR(31)	IN/OUT	RECORD POINTER
TFDRECL	FIXED(15)	OUTPUT	RECORD LENGTH
TFDECODE	FIXED(15)	IN/OUT	ENTRY CODE
TFDTTRN	CHAR(4)	IN/OUT	FIRST OR CURRENT TTRN

PARAMETER USAGE

TFDRECP - A ZERO VALUE ON INPUT TO CDG INDICATES 'LOCATE' MODE, IN WHICH CASE CDG WILL RETURN A POINTER TO THE LOGICAL RECORD IN TFDRECP. (NOTE: LOCATE MODE MUST BE USED FOR UPDATE IN PLACE OF A PDS.)

A NON-ZERO VALUE ON INPUT TO CDG INDICATES 'MOVE' MODE, IN WHICH CASE CDG WILL MOVE THE LOGICAL RECORD TO THE ADDRESS SPECIFIED BY TFDRECP.

TFDRECL - OUTPUT PARAMETER ONLY. CONTAINS THE LENGTH OF THE LOGICAL RECORD RETURNED BY CDG.

TFDECODE - INDICATES THE ENTRY CONDITION, AS FOLLOWS:

0 - 1ST ENTRY TO CDG (FOR "STANDARD" MODE OF OPERATIONS). 1ST RECORD WILL BE RETURNED. THE TTRN OF THE FIRST BLOCK WILL BE STORED IN TFDTTRN. CDG WILL SET TFDECODE=1.

1 - ITH ENTRY TO CDG. ITH RECORD WILL BE RETURNED. NO CHANGE TO TFDECODE.

2 - RESET TO THE TTRN CONTAINED IN TFDTTRN (INVALID FOR UPDATE IN PLACE). 1ST RECORD STARTING AT THE NEW TTRN WILL BE RETURNED. CDG WILL SET TFDECODE=1.

3 - CLOSE OUT. CDG WILL SET TFDECODE=0. NO RECORD IS RETURNED.
NOTE: CLOSE OUT IS AUTOMATIC WHENEVER RETURN CODE > 0, EXCEPT WHEN ORIGINAL VALUE OF TFDECODE WAS 5 (SEE NOTES).

(CONTINUED ON NEXT PAGE)

4 - 1ST ENTRY TO CDG FOR UPDATE IN PLACE.
1ST RECORD WILL BE RETURNED.
CDG WILL SET TFDECODE=1.

5 - 1ST ENTRY TO CDG IF CURRENT TTRN FOR EACH
RECORD IS TO BE RETURNED IN TFDTTRN.
1ST RECORD WILL BE RETURNED.
CDG WILL SET TFDECODE=1.
TFDECODE=5 IS USED BY BROWSE.

RETURN CODE:

0 - NORMAL COMPLETION.

1 - END OF FILE CONDITION. NO RECORD WAS RETURNED. CDG HAS CLOSED ITSELF OUT UNLESS TFDECODE=5 WAS USED, IN WHICH CASE CLOSE OUT IS NOT AUTOMATIC ON END OF FILE.

2 - UNRECOVERABLE I/O ERROR. NO RECORD WAS RETURNED. CDG HAS CLOSED ITSELF OUT UNLESS TFDECODE=5 WAS USED, IN WHICH CASE CLOSE OUT IS NOT AUTOMATIC ON I/O ERROR. (NOTE: RETURN CODE ALSO SET TO 2 IF CERTAIN USER ERRORS ARE DETECTED.)

NOTES:

THIS MODULE HAS REPLACED THE SPF VERSION 2.1 CGET SUBROUTINE.

IT IS ASSUMED THAT CALLERS OF CDG HAVE PREVIOUSLY ALLOCATED AND OPENED THE DATASET BY CALLING EITHER CTA OR CDA AND CDO.

CDG ALWAYS RETURNS THE TTRN OF THE FIRST BLOCK, TO ALLOW THE DATA SET OR MEMBER TO BE RE-READ FROM THE TOP (SEE TFDECODE=2). OPTIONALLY, CDG MAY BE INITIALIZED TO RETURN THE CURRENT TTRN FOR EACH RECORD AS IT READS THROUGH THE DATA (SEE TFDECODE=5).

FOR VARIABLE LENGTH RECORDS, THE TFDRECP AND TFDRECL PARAMETERS REFER TO THE DATA PORTION ONLY, I.E. THEY EXCLUDE THE FOUR BYTE RECORD PREFIX AREA.

FOR UNDEFINED RECORD FORMAT (RECFM=U), THE RECORD LENGTH RETURNED IN TFDRECL IS THE SIZE OF THE PHYSICAL BLOCK.

ON CLOSE OUT, CDG DOES NOT CLOSE THE DATA SET; TO REUSE CDG FOR A DIFFERENT MEMBER OF THE SAME PDS, THE FOLLOWING STEPS ARE REQUIRED:

1. CLOSE OUT CDG (TFDECODE=3) IF IT HAS NOT ALREADY CLOSED ITSELF OUT.
2. ISSUE FIND MACRO FOR NEW MEMBER.
3. CALL CDG (TFDECODE=0) TO START READING NEW MEMBER.

THE ONLY USER ERROR WHICH WILL CAUSE RETURN CODE = 2 IS FOR CDG TO BE CALLED WITHOUT FIRST CALLING CDO. TFDOPN MUST BE ON.

IF THE ORIGINAL TFDECODE WAS 0 OR 4, TFDECODE WILL BE RESET TO ZERO FOR ANY CONDITION WHICH RESULTS IN A NON-ZERO RETURN CODE.

IF THE ORIGINAL TFDECODE WAS 5, AND AN END OF FILE OR I/O ERROR OCCURS, TFDECODE WILL BE SET TO 1.

CDG USES BSAM I/O.

CDISPL - COMMON DISPLAY ROUTINE

PURPOSE:

CDISPL IS USED TO DISPLAY THE CURRENT CONTENTS OF THE TLD/TLS AND WAIT FOR USER RESPONSE (POST DISPLAY REQUEST ECB AND WAIT ON PROCESS REQUEST ECB).

INVOKED WITH:

CALL TO CDISPL

CALLING SEQUENCE PARAMETERS:

```
1.  TLD       <TLD>       IN/OUT    LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

CDISPL SCANS THE TLS INPUT LINE (LINE 2) FOR AN ATTRIBUTE BYTE INDICATING THE BEGINNING OF THE TLSINPUT FIELD (IF ANY) AND INITIALIZES THE TLD POINTER (TLDIDABP) ACCORDINGLY.

AFTER THE USER RESPONSE, CDISPL PERFORMS THE FOLLOWING:

IF THE HELP PFK WAS PRESSED AND THE TLDHELP FIELD IS NOT BLANK:
CHELP IS CALLED.

IF A PRIMARY COMMAND PFK WAS PRESSED:
THE ASSOCIATED COMMAND IS RETRIEVED FROM THE TKV AND PLACED IN THE TLSINPUT FIELD.

IF A LINE COMMAND PFK WAS PRESSED:
THE ASSOCIATED COMMAND IS RETRIEVED FROM THE TKV AND PLACED IN THE TLS EDIT LINE COMMAND FIELD.

IF THE RETURN PFK WAS PRESSED:
CDISPL INTERFACES WITH THE CALLING PROGRAMS AS THOUGH THE END KEY HAD BEEN PRESSED, BUT WITHOUT INTERFACING WITH THE SPF MAIN TASK, AND THUS WITHOUT CAUSING ANY ACTUAL DISPLAY OUTPUT. PMD STOPS THIS FUNCTION WHEN IT GETS CONTROL.

IF ANY DISPLAYABLE INPUT FIELD HAS AN EXTENDED RETURN VALUE (A PRIMARY OPTION VALUE) PRECEEDED BY AN EQUAL SIGN (=) THEN IT WILL BE PLACED IN THE TLSINPUT FIELD WHEN THE PRIMARY OPTION MENU IS DISPLAYED. CDISPL WILL THEN RETURN TO PMD AS THOUGH THE ENTER KEY HAD BEEN PRESSED.

IF THE SCROLL FIELD WAS MODIFIED:
THE MDT BIT FOR THE SCROLL FIELD IS SET TO OFF.

IF A SCROLL PFK WAS PRESSED AND THE TLSINPUT FIELD CONTAINS A SCROLL AMOUNT:
THE NEW AMOUNT IS MOVED TO THE SCROLL FIELD AND THE MDT BIT FOR THE INPUT FIELD IS SET TO OFF. THIS HAS THE EFFECT OF A TEMPORARY CHANGE TO THE SCROLL FIELD JUST FOR THIS INTERACTION.

NOTE: WHENEVER CDISPL MODIFIES AN MDT (MODIFIED DATA TAG) IN THE TLS, IT ADJUSTS THE MDT COUNT IN THE TLD ACCORDINGLY.

PURPOSE:

CDO IS USED TO OPEN A FILE. THE FILE IS EXPECTED TO HAVE PREVIOUSLY BEEN ALLOCATED BY CALLING CDA OR CTA. IF THE FILE IS AN EMPTY SEQUENTIAL DATASET BEING OPENED FOR INPUT (AN INVALID THING TO DO), THE DSCB WILL BE READ TO OBTAIN DATASET ATTRIBUTES, BUT OPEN WILL NOT BE DONE. OPTIONALLY CDO WILL VALIDATE DATASET ATTRIBUTES AS REQUIRED BY THE CALLER. VALID COMBINATIONS OF DSORG, LRECL, BLKSIZE AND RECORD FORMAT WILL BE CHECKED. CDO WILL OPTIONALLY CALL CRESV TO ISSUE A SHARED DASD RESERVE AGAINST THE DATA SET. CDO WILL ALSO OPTIONALLY CALL CSM TO GETMAIN SPACE FOR AN I/O BUFFER.

INVOKED WITH:

CALL TO CDO

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	MSGID	CHAR(4)	OUTPUT	ERROR MESSAGE ID

WHERE

TFD - THE FILE DEFINITION TABLE FOR THE FILE BEING OPENED.

MSGID - CONTAINS ID OF ERROR MESSAGE ON ABNORMAL RETURN. OTHERWISE, THE MSGID WILL NOT BE MODIFIED.

RETURN CODES:

0 - NORMAL RETURN.

4 - ERROR RETURN (MSGID CONTAINS THE ERROR MSG ID).

NOTES:

ON AN ABNORMAL RETURN, THE DCB WILL BE CLOSED AND CRELS WILL BE CALLED, IF NECESSARY, TO RELEASE THE DATA SET RESOURCE.

CDP - COMMON DATA PUT ROUTINE CDP

PURPOSE:

CDP IS USED TO OUTPUT A LOGICAL RECORD TO A SEQUENTIAL DATA SET OR MEMBER OF A PDS. CDP WILL OPTIONALLY NOTE THE TTR OF A PHYSICAL BLOCK WRITTEN. CDP USES BSAM I/O.

INVOKED WITH:

CALL TO CDP

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE

WHERE

TFD - IS A STRUCTURE WHICH INCLUDES THE FOLLOWING PARAMETERS:

TFDRECP	PTR(31)	INPUT	RECORD POINTER
TFDRECL	FIXED(15)	INPUT	RECORD LENGTH
TFDECODE	FIXED(15)	IN/OUT	ENTRY CODE

PARAMETER USAGE

TFDRECP - CONTAINS A POINTER TO THE LOGICAL RECORD. CDP SUPPORTS 'MOVE' MODE ONLY FOR RECFM=F OR RECFM=V AND 'LOCATE' MODE ONLY FOR RECFM=U DATA.

TFDRECL - IS THE LENGTH OF THE LOGICAL RECORD IN BYTES.

TFDECODE - IS AN ENTRY CODE, AS FOLLOWS:

0 - 1ST ENTRY TO CDP. 1ST RECORD WILL BE OUTPUT. CDP WILL SET TFDECODE=1.

1 - 'I'TH ENTRY TO CDP. ITH RECORD WILL BE OUTPUT. TFDECODE WILL NOT BE CHANGED.

2 - CLOSE OUT. CDP WILL WRITE THE FINAL BLOCK, IF NECESSARY AND SET TFDECODE=0. NOTE: CLOSE OUT IS AUTOMATIC WHENEVER RETURN CODE > 0.

RETURN CODE:

0 - NORMAL COMPLETION.

2 - UNRECOVERABLE I/O ERROR. CDP HAS CLOSED ITSELF OUT.

(CONTINUED ON NEXT PAGE)

NOTES:

THIS MODULE HAS REPLACED THE SPF VERSION 2.1 CPUT SUBROUTINE.

IT IS ASSUMED THAT CALLERS OF CDP HAVE EITHER PREVIOUSLY ALLOCATED AND OPENED THE DATA SET BY CALLING CDA AND CDO OR BY CALLING CTA. THIS ASSURES THAT THE TFD IS INITIALIZED PROPERLY.

FOR VARIABLE LENGTH RECORDS, THE TFDRECP AND TFDRECL PARAMETERS REFER TO THE DATA PORTION ONLY, I.E. THEY EXCLUDE THE FOUR BYTE RECORD PREFIX AREA. CDP WILL AUTOMATICALLY CONSTRUCT THE RECORD AND BLOCK PREFIXES IN THE OUTPUT BUFFER.

THE RECORD LENGTH INDICATED BY TFDRECL NEED NOT AGREE WITH THE DCB LRECL. FOR FIXED RECORD FORMATS, CDP WILL EITHER TRUNCATE OR PAD WITH BLANKS TO MAKE THE RECORD LENGTH EQUAL THE DCB LRECL. FOR VARIABLE RECORD FORMATS, CDP WILL TRUNCATE IF THE LOGICAL RECORD LENGTH EXCEEDS DCB LRECL-4. IN ADDITION, CDP WILL AUTOMATICALLY REMOVE ANY TRAILING BLANKS FOR VARIABLE LENGTH RECORDS.

FOR UNDEFINED RECORD FORMAT (RECFM=U), TFDRECL MUST SPECIFY THE SIZE OF THE PHYSICAL BLOCK. FOR RECFM=U, CDP WRITES THE BLOCK DIRECTLY FROM THE AREA SPECIFIED VIA TFDRECP, AND ISSUES A CHECK IMMEDIATELY FOLLOWING THE WRITE.

ON CLOSE OUT, CDP DOES NOT WRITE END-OF-FILE NOR DOES IT CLOSE THE DATA SET, IT SIMPLY FLUSHES THE I/O BUFFER. TO REUSE CDP FOR A DIFFERENT MEMBER OF THE SAME PDS:

1. CALL CDP FOR CLOSE OUT (TFDECODE=2) TO FLUSH THE BUFFER.
2. ISSUE STOW MACRO FOR THIS MEMBER.
3. CALL CDP (TFDECODE=0) TO START WRITING NEW MEMBER.

CDT - COMMON DEVICE TYPE ROUTINE

PURPOSE:

CDT IS USED TO OBTAIN THE DEVICE TYPE OF A VOLUME, WHEN THE VOLUME SERIAL OR DEVICE CODE IS KNOWN. THE DEVICE TYPE IS RETURNED IN CHARACTER FORMAT.

INVOKED WITH:

CALL TO CDT

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	VOLUME	CHAR(6)	INPUT	VOLUME SERIAL OR DEVICE CODE
3.	DEVICE	CHAR(8)	OUTPUT	DEVICE TYPE
4.	OPTION	FIXED(31)	INPUT	OPTION CODE

WHERE

VOLUME - IF THE FIRST TWO BYTES ARE X'FFFF' THEN THE NEXT FOUR BYTES ARE ASSUMED TO BE THE CATALOG DEVICE CODE FOR THE VOLUME. IF THE FIRST TWO BYTES ARE X'FFFE' THEN THE NEXT FOUR BYTES ARE ASSUMED TO BE THE UCB DEVICE CODE FOR THE VOLUME. OTHERWISE, THE FIELD IS ASSUMED TO BE THE ACTUAL VOLUME SERIAL.

DEVICE - VALUE FROM THE SYSTEM DEVICE NAME TABLE 'DEVNAMET' (E.G. '3330' OR '2314'), '3330V', OR BLANK.

OPTION - 0 OR 1.

RETURN CODE:

0 - ALWAYS.

NOTES:

IF THE OPTION VALUE IS 0, THEN BLANK IS RETURNED AS THE DEVICE TYPE. IF THE OPTION IS 1, THEN THE FOLLOWING NOTES APPLY. ALL THE CALLS TO CDT BY THE DISTRIBUTED SYSTEM USE OPTION 1.

IF THE VOLUME SERIAL IS SUPPLIED THEN THE UCBS ARE SEARCHED TO FIND THE DEVICE CODE. IF THE DEVICE CODE IS NOT FOUND AND THERE IS A MASS STORAGE SYSTEM (MSS) AVAILABLE, THE DEVICE TYPE RETURNED IS '3330V'. IF THE CODE IS NOT FOUND AND THERE IS NO MSS, BLANK IS RETURNED AS THE DEVICE TYPE.

THE DEVICE CODE FOR A CATALOGED DATA SET IS AVAILABLE THROUGH THE CATALOG VIA THE LOCATE MACRO AND THEN CAN BE USED AS INPUT TO CDT.

ONCE CDT HAS A DEVICE CODE, THEN THE SYSTEM DEVICE NAME TABLE (MODULE NAME 'DEVNAMET') IS SEARCHED FOR A DEVICE TYPE (E.G. '3330' OR '2314'). IF CDT FINDS NO CORRESPONDING DEVICE TYPE FOR A DEVICE CODE, A BLANK IS RETURNED.

CERR - COMMON ERROR DISPLAY ROUTINE

PURPOSE:

CERR IS USED TO FORMAT AND DISPLAY SPF MESSAGES. THE MESSAGE IS READ AND FORMATTED BY CMSG. A SHORT MESSAGE OF UP TO 24 CHARACTERS IS FIRST DISPLAYED IN THE UPPER RIGHT HAND CORNER OF THE LOGICAL SCREEN. IF THE SPF USER RESPONDS BY PRESSING THE HELP KEY, A 2ND LEVEL MESSAGE OF UP TO 77 CHARACTERS IS DISPLAYED ON THE THIRD LINE OF THE DISPLAY. IF THE SPF USER AGAIN RESPONDS BY PRESSING THE HELP KEY, CHELP IS CALLED BY CDISPL TO DISPLAY TUTORIAL INFORMATION.

INVOKED WITH:

CALL TO CERR

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MSGID	CHAR(4)	INPUT	MESSAGE ID
3.	PARM1	*	INPUT	MESSAGE PARAMETER 1
..	...			
..	PARMN	*	INPUT	MESSAGE PARAMETER 'N'

WHERE

PARM(S) - PARM1 THROUGH PARMN CAN BE IN ANY FORMAT THAT CAN BE HANDLED BY CMSG IN FORMATTING MESSAGES. THE PARAMETERS NEED NOT BE ACTUALLY REFERENCED BY A MESSAGE. A MAXIMUM OF 50 PARAMETERS IS SUPPORTED.

RETURN CODE:

0 - ALWAYS.

NOTES:

THE CERR PARAMETER LIST MUST BE TERMINATED WITH A VLIST FLAG.

ON EXIT, CERR RESTORES THE SCREEN TO ITS CONDITION AT ENTRY TO CERR.

CERR CALLS CMSG TO OBTAIN THE REQUESTED MESSAGE.

IF THE FIRST CHARACTER OF THE MESSAGE ID IS '-', THEN CERR DOES NOT CALL CMSG, BUT FILLS THE SHORT MESSAGE AREA WITH DASHES AND AND DISPLAYS THE SCREEN. IN THIS CASE, NO 2ND LEVEL MESSAGE IS PROCESSED.

PURPOSE:

CFI IS USED TO FIND MEMBERS OF THE SPFMENUS, SPFMSGS, AND SPFPROCS DATA SETS. CFI MAINTAINS BLDL LISTS LOCATED IN THE FIND TABLE (TFI) FOR THE DATA SETS, THUS ELIMINATING MOST PDS DIRECTORY SEARCHES FOR THESE DATA SETS. IF THE REQUESTED MEMBER IS NOT IN THE TFI, CFI ISSUES A BLDL AND THEN A FIND MACRO.

INVOKED WITH:

CALL TO CFI

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	INPUT	FILE DEFINITION TABLE
3.	MEMBER	CHAR(8)	INPUT	SEARCH MEMBER NAME

WHERE

TFD - THE TFD FOR THE DATA SET TO BE SEARCHED.

MEMBER - IS THE MEMBER FOR WHICH THE FIND IS TO BE DONE.

RETURN CODE:

0 - NORMAL RETURN, MEMBER FOUND.

4 - MEMBER NOT FOUND.

8 - I/O ERROR RETURN FROM BLDL.

NOTES:

THE CALLING PROGRAM MUST SET UP THE TFD AND OPEN THE DATA SET BEFORE CALLING CFI.

SEE THE DATA AREAS SECTION FOR A DESCRIPTION OF THE FIND TABLE (TFI).

PURPOSE:

CHC PERFORMS THE HARDCOPY FUNCTION FOR UOL AND USC.

INVOKED WITH:

CALL TO CHC

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	MENU	CHAR(8)	INPUT	MENU NAME

WHERE

MENU - IS THE NAME OF THE MENU TO BE DISPLAYED BY CHC.

RETURN CODE:

0 - ERROR RETURN

4 - DELETE REQUESTED

8 - KEEP REQUESTED

NOTES:

NONE.

PURPOSE:

CHELP IS USED TO INVOKE THE SPF TUTORIAL STARTING AT A SPECIFIC PAGE. THE TLS AND APPROPRIATE TLD FIELDS ARE SAVED BEFORE INVOKING THE TUTORIAL, AND RESTORED BEFORE RETURNING. WHEN CHELP IS INVOKED, THE USER MAY PAGE THROUGH THE TUTORIAL AS DESIRED. WHEN HE ENDS THE TUTORIAL, THE SCREEN IS RESTORED.

INVOKED WITH:

CALL TO CHELP

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

WHERE

TLD - TLDHELP (CHAR(8)) CONTAINS THE MEMBER NAME OF THE FIRST TUTORIAL PAGE TO BE DISPLAYED.

RETURN CODE:

0 - SPFTUTOR WAS INVOKED.

4 - SPFTUTOR WAS NOT INVOKED BECAUSE THE TLDHELP FIELD CONTAINED AN INVALID MEMBER NAME.

NOTES:

NONE.

PURPOSE:

CHPJ IS USED TO PRINT A SPECIFIED DATA SET VIA A BACKGROUND JOB. CHPJ GENERATES JCL FOR A JOB STEP AND WRITES THE JCL TO THE TEMPCNTL DATA SET ('USERID.SPFTEMP*.CNTL'). THE PROTOTYPE FOR THE GENERATED JCL IS TAKED FROM THE SPFPROCS DATA SET (MEMBER CHPJ).

INVOKED WITH:

CALL TO CHPJ

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICIAL DISPLAY TABLE
2.	DSNS	CHAR(58)	INPUT	DATA SET NAME STRUCTURE
3.	VOLUME	CHAR(6)	INPUT	VOLUME SERIAL NUMBER
4.	DEVICE	CHAR(8)	INPUT	DEVICE TYPE
5.	DISP	CHAR(1)	INPUT	DATA SET DISPOSITION
6.	SCLAS	CHAR(15)	INPUT	SYSOUT CLASS
7.	RECFM	CHAR(6)	INPUT	OUTPUT RECORD FORMAT
8.	LRECL	FIXED(15)	INPUT	OUTPUT LOGICAL RECORD LENGTH
9.	BLKSIZE	FIXED(15)	INPUT	OUTPUT BLOCK SIZE
10.	OPTJ	CHAR(1)	INPUT	OUTPUT OPTCD

WHERE

DSNS - THE DATA SET NAME STRUCTURE (STANDARD DAIR FORMAT).

```
E.G.,   DCL 1 DSNS,
                2 DSNL FIXED(15),  /* DSNAME LENGTH */
                2 DSN  CHAR(56);   /* DSNAME        */
```

NOTE: THE DSN MAY INCLUDE A MEMBER NAME IN PARENS.

VOLUME - THE SERIAL NUMBER OF VOLUME THAT THE DATA SET IS ON, IF IT IS NOT CATALOGED.

DEVICE - THE TYPE OF THE DEVICE THAT THE DATA SET IS ON, IF IT IS NOT CATALOGED.

DISP - A CODE THAT INDICATES THE REQUESTED DISPOSITION OF THE DATA SET AFTER PRINTING:
'D' - DELETE
'K' - KEEP

SCLAS - THE SYSOUT CLASS USED FOR THE OUTPUT OF THE PRINT STEP.

RECFM - THE DATA SET RECORD FORMAT, WHICH IS USED TO CONTROL THE OUTPUT OF THE PRINT STEP.

LRECL - THE DATA SET LOGICIAL RECORD LENGTH, WHICH IS USED TO CONTROL THE OUTPUT OF THE PRINT STEP.

BLKSIZE - THE DATA SET BLOCK SIZE, WHICH IS USED TO CONTROL THE OUTPUT OF THE PRINT STEP.

OPTJ - THE OPTCD VALUE FOR THE 3800, 'J' OR BLANK.

(CONTINUED ON NEXT PAGE)

RETURN CODE:

0 - NO ERRORS DETECTED.

1-39 - CDP ERROR RETURN CODE WRITING TO TEMPCNTL DATA SET.

40 - CHPJ MEMBER NOT FOUND IN PROCS DATA SET.

44 - I/O ERROR READING SPFPROCS MEMBER CHPJ.

48 - CMSG ERROR CONSTRUCTING A CARD FROM THE MODEL.

NOTES:

THE CALLING SEQUENCE PARAMETERS REFER TO THE DATA SET TO BE PRINTED, NOT THE TEMPCNTL DATA SET.

PURPOSE:

CHPL IS USED TO ROUTE A SPECIFIED DATA SET TO A LOCAL PRINTER VIA DSPRINT.

INVOKED WITH:

CALL TO CHPL

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICIAL DISPLAY TABLE
2.	ENTCODE	FIXED(31)	INPUT	CHPL ENTRY CODE
3.	DSNS	CHAR(58)	INPUT	DATA SET NAME STRUCTURE
4.	PSWD	CHAR(8)	INPUT	DATA SET PASSWORD
5.	LPID	CHAR(8)	INPUT	LOCAL PRINTER ID

WHERE

ENTCODE - THE ENTRY CODE TO CHPL:

0 - NO CARRIAGE CONTROL CHARACTERS IN THE INPUT DATA SET, USE DSPRINT'S DEFAULTS.
1 - CARRIAGE CONTROL CHARACTERS PRESENT IN THE INPUT DATA SET, INDICATE SUCH TO DSPRINT.

DSNS - THE DATA SET NAME STRUCTURE (STANDARD DAIR FORMAT).

```
E.G.,   DCL 1 DSNS,
              2 DSNL FIXED(15),  /* DSNAME LENGTH */
              2 DSN  CHAR(56);   /* DSNAME        */
```

NOTE: THE DSN MAY INCLUDE A MEMBER NAME IN PARENS.

PSWD - THE PASSWORD, IF THE DATA SET TO BE PRINTED IS PASSWORD PROTECTED.

LPID - LOCAL PRINTER ID KNOWN BY DSPRINT.

RETURN CODE:

0 - NO ERROR DETECTED.

4 - THE USER ATTENTIONED OUT OF DSPRINT.

8 - DSPRINT ABENDED.

12 - DSPRINT HAD A RETURN CODE GREATER THAN ZERO.

NOTES:

NONE.

CIPARMS - COMMON INITIALIZE USER PARAMETERS ROUTINE CIPARMS

PURPOSE:

CIPARMS GETMAINS THE AREA FOR THE SPF TABLE OF KEYWORD VALUES (TKV) AND INITIALIZES IT WITH A MEMBER OF THE SPF PARMS DATA SET. THE MEMBER HAS THE SAME NAME AS THE USERS TSO LOGON ID.

INVOKED WITH:

CALL TO CIPARMS (FROM SMI)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - PARMS MEMBER READ AND PROCESSED. NO ERRORS DETECTED.

1 - PARMS MEMBER READ AND CONVERTED FROM SPF VERSION 2.1 FORMAT. NO ERRORS DETECTED.

2 - NO PARMS MEMBER READ, NEW MEMBER CREATED. NO ERRORS DETECTED.

3 - NO PARMS MEMBER READ, NO NEW MEMBER CREATED. NO ERRORS DETECTED.

5 - OPEN ERROR OPENING PARMS DATA SET TO OUTPUT NEW MEMBER.

6 - DIRECTORY FULL ERROR ATTEMPTING TO STOW NEW MEMBER.

7 - I/O FIND ERROR FINDING PARMS MEMBER.

8 - I/O READ ERROR READING PARMS MEMBER.

9 - I/O STOW ERROR STOWING DIRECTORY ENTRY FOR NEW MEMBER.

10- I/O WRITE ERROR WRITING NEW MEMBER TO PARMS DATA SET.

12- DATA SET FULL ERROR ATTEMPTING TO WRITE NEW MEMBER.

NOTES:

IF THE PARMS MEMBER WAS CREATED BY SPF VERSION 2.1 IT IS CONVERTED TO THIS VERSION'S FORMAT BY SPC (VIA LINK TO SPFSPC).

IF THE PARMS MEMBER IS NOT FOUND, DEFAULT VALUES FROM A COMPILED VERSION OF THE TKV ARE USED TO INITIALIZE THE TKV. IF THERE ARE NO OTHER ERRORS, THIS DEFAULT TKV IS WRITTEN TO THE PARMS DATA SET.

AFTER THE TKV IS INITIALIZED THE SUBROUTINE SIP (SPF INPUT PARMS) IS CALLED. SIP WAS DESIGNED AS A USER EXIT ROUTINE AND CAN BE MODIFIED IF AN INSTALLATION REQUIRES THAT DATA BE VERFIED OR MODIFIED BEFORE BEING USED BY SPF. SIP ALSO COPIES SOME DATA FROM THE TKV TO THE TSV. FOR MORE INFORMATION SEE THE DESCRIPTION OF SIP.

RETURN CODES 5 TO 12 ARE ERROR CODES. THEY WILL CAUSE AN INITIALIZATION ERROR MENU TO BE DISPLAYED WHEN PMD IS FIRST EXECUTED, BUT FOLLOWING THAT SPF WILL CONTINUE PROCESSING.

FOR RETURN CODES 3 TO 12 DEFAULT TKV VALUES ARE USED DURING THE SESSION BUT WILL NOT BE REMEMBERED FOR THE NEXT SESSION.

CIR - COMMON READ PDS DIRECTORY ROUTINE CIR

PURPOSE:

CIR READS PDS DIRECTORY ENTRIES AND RETURNS THE INFORMATION TO THE CALLER. IT IS USED BY THE LIBRARY AND DATA SET UTILITIES.

INVOKED WITH:

CALL TO CIR

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	WORK	PTR(31)	IN/OUT	WORK AREA

WHERE

TFD - TFDBLDLP POINTS TO A BLDL LIST IN THE SAME FORMAT AS THAT REQUIRED BY THE OS/VS BLDL MACRO.

WORK - A FULL WORD WHICH CIR USES TO POINT TO A WORK AREA BETWEEN CALLS. THE CALLER MUST SET THIS WORD TO ZERO BEFORE THE FIRST CALL TO CIR. OTHERWISE IT IS OF NO CONCERN TO THE CALLER.

RETURN CODE:

0 - NO ERRORS DETECTED.

4 - I/O ERROR READING DIRECTORY.

8 - UNABLE TO OPEN DCB FOR DIRECTORY READ.

NOTES:

CIR READS THE PDS DIRECTORY SEQUENTIALLY. IT RETURNS ONE DIRECTORY ENTRY TO THE CALLER ON EACH CALL.

THE CALLER SETS UP A BLDL FORMAT LIST (POINTED TO FROM TFDBLDLP) WITH THE NUMBER OF ENTRIES SET TO ONE AND THE ENTRY LENGTH FILLED IN, TO DESCRIBE TO CIR HOW MUCH DATA TO RETURN ON EACH CALL. CIR FILLS IN THE REMAINDER OF THE BLDL LIST FOR THE ENTRY. CIR MAY BE CALLED REPEATEDLY UNTIL ALL DIRECTORY ENTRIES HAVE BEEN READ.

AFTER THE ENTIRE DIRECTORY HAS BEEN READ, CIR RETURNS A TRAILER ENTRY, INDICATED BY THE FIRST 4 BYTES OF THE MEMBER NAME BEING X'FFFFFFFF'. THE NEXT TWO BYTES THEN CONTAIN THE TOTAL NUMBER OF DIRECTORY BLOCKS, AND THE LAST 2 BYTES CONTAIN THE NUMBER OF USED DIRECTORY BLOCKS.

CIV - COMMON READ VTOC ROUTINE CIV

PURPOSE:

CIV OBTAINS VTOC AND DIRECTORY INFORMATION ABOUT THE DATA SET SPECIFIED IN THE TFD AND PLACES IT IN THE CIV COMMON AREA.

INVOKED WITH:

CALL TO CIV

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<CIV>	OUTPUT	CIV COMMON AREA
4.	OPTION	FIXED(31)	INPUT	OPTION

WHERE

```
     OPTION  - CONTROL CIV PROGRAM AS FOLLOWS:
                 0 - NORMAL PROCESSING.
                 1 - NO PARTITIONED DATA SET DIRECTORY INFORMATION
```

RETURN CODE:

0 - NO ERRORS DETECTED. CIV COMMON AREA IS COMPLETE.

>0 - OBTAIN OF VTOC INFORMATION FAILED. THE OBTAIN MACRO RETURN CODE IS PASSED BACK AS THE CIV RETURN CODE. THE CIV COMMON IS SET TO ZEROS.

NOTES:

NONE.

PURPOSE:

CJC IS USED TO WRITE JOBCARD IMAGES TO THE TEMPORARY CONTROL CARD DATA SET ('USERID.SPFTEMP*.CNTL'). UP TO FOUR RECORDS ARE WRITTEN.

INVOKED WITH:

CALL TO CJC

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	JOBCARDS	(4)CHAR(72)	INPUT	JOBCARD IMAGES (FROM MENU)

RETURN CODE:

0 - NORMAL COMPLETION.

NON-ZERO - ABNORMAL COMPLETION (RETURN CODE PASSED THROUGH FROM CDP).

NOTES:

BEFORE CJC IS CALLED, THE CONTROL CARD TEMPORARY DATA SET MUST BE ALLOCATED AND OPENED BY CALLING CTA. CJC WILL INITIALIZE THE FOLLOWING FIELDS OF THE TFD BEFORE CALLING CDP:

```
TFDRECP = ADDR(CJC OUTPUT BUFFER)
TFDRECL = 80
TFDECODE = 0
```

THE CALLING PROGRAM MUST USE CDP TO WRITE OUT ADDITIONAL JCL TO THE TEMPORARY DATSET, LATER CLOSEOUT CDP, AND CLOSE THE TEMPORARY DATA SET BEFORE SUBMITTING THE JCL TO OS BY CALLING THE CSB ROUTINE.

CJC DOES NOT SYNTAX CHECK THE JOBCARD INFORMATION. IT IS ASSUMED THE USER HAS VERIFIED THE JOBCARD INFORMATION BEFORE CJC IS CALLED.

CJF - COMMON JOBNAME FIND ROUTINE CJF

PURPOSE:

CJF IS USED TO SEARCH JOBCARD IMAGES TO FIND THE JOBNAME AND TO UPDATE THE TSVJCHAR FIELD APPROPRIATELY. CJF IS CALLED AFTER DISPLAY OF THE JOBCARDS. CJF IS USED IN CONJUNCTION WITH CJN AND CJC IN PROCESSING USER JOBCARDS.

INVOKED WITH:

CALL TO CJF

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CURCHAR	CHAR(1)	IN/OUT	CURRENT JOBNAME CHARACTER
3.	NEXTCHAR	CHAR(1)	IN/OUT	NEXT JOBNAME CHARACTER
4.	JOBCARDS	(4)CHAR(72)	INPUT	ARRAY OF 4 JOBCARDS
5.	JOBNAME	CHAR(8)	OUTPUT	JOBNAME

WHERE

CURCHAR - IS THE OUTPUT FIELD FROM CJN. IT IS UPDATED BY CJF TO REFLECT POSSIBLE USER MODIFICATION OF THE JOBNAME IF JOBCARDS WERE DISPLAYED.

NEXTCHAR - IS THE OUTPUT FIELD FROM CJN. IT IS UPDATED BY CJF TO REFLECT POSSIBLE USER MODIFICATION OF THE JOBNAME IF JOBCARDS WERE DISPLAYED.

JOBNAME - IS SET TO THE JOBNAME IF ANY IS FOUND, OTHERWISE, IT IS SET TO BLANKS.

RETURN CODES:

0 - ALWAYS.

NOTES:

SEE THE NOTES FOR OBJECT MODULE CJN.

- COMMON JOBNAME INIT ROUTINE

PURPOSE:

CJN IS USED TO INITIALIZE THE JOBNAME ON A SET OF JOBCARDS BEFORE DISPLAYING THEM. CJN ALSO UPDATES THE TSVJCHAR FIELD IF AN SPF FORMAT JOBNAME IS BEING USED. SPF JOBNAMES CONSIST OF THE TSO USERID FOLLOWED BY A SINGLE CHARACTER FROM "A" TO "Z" OR "0" TO "9". CJN IS USED IN CONJUCTION WITH CJF AND CJC IN HANDLING USER JOBCARDS.

INVOKED WITH:

CALL TO CJN

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CURCHAR	CHAR(1)	OUT	CURRENT JOBNAME CHARACTER
3.	NEXTCHAR	CHAR(1)	OUT	NEXT JOBNAME CHARACTER
4.	JOBCARDS	(4)CHAR(72)	IN/OUT	ARRAY OF 4 JOBCARDS

WHERE

CURCHAR - IS THE JOBNAME CHARACTER TAKEN FROM TSVJCHAR AND USED TO INITIALIZE THE JOBCARDS.

NEXTCHAR - IS SET TO THE VALUE OF TSVJCHAR AFTER IT IS INCREMENTED BY CJN.

JOBCARDS - IS THE JOBCARD ARRAY INTO WHICH THE JOBNAME WILL BE PLACED BY CJN.

RETURN CODES:

0 - ALWAYS.

NOTES:

IF THE CALLING PROGRAM DOES NOT SUBMIT A JOB AFTER CALLING CJN, IT MUST COMPARE NEXTCHAR WITH TSVJCHAR AND IF EQUAL IT MUST SET TSVJCHAR TO CURCHAR. THIS PROCEDURE INSURES THAT THE JOBNAME IS INCREMENTED ONLY WHEN SPF FORMAT JOBNAMES ARE SUBMITTED.

CKVGET - COMMON KEYWORD-VALUE GET ROUTINE CKVGET

PURPOSE:

CKVGET IS USED TO RETRIEVE THE "REMEMBERED" VALUE(S) OF A LIST OF NAMED KEYWORDS FROM THE KEYWORD/VALUE TABLE (TKV). CKVPUT IS USED TO STORE KEYWORD/VALUES INTO THE TKV.

INVOKED WITH:

CALL TO CKVGET

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	KVBLOCK	<KVBLOCK>	INPUT	KEYWORD/VALUE CONTROL BLOCK
3.	AREA	CHAR(*)	OUTPUT	VALUE AREA

WHERE

KVBLOCK - CONSISTS OF ONE OR MORE KEYWORD/VALUE ENTRIES DELIMITED BY A BYTE SET TO '00'X. SEE DATA AREAS SECTION FOR A DESCRIPTION OF THE KEYWORD/VALUE BLOCK.

AREA - AN AREA EQUAL IN LENGTH TO THE SUM OF THE VALUE LENGTH FIELD(S) IN THE KVBLOCK.

RETURN CODES:

0 - ALL KEYWORDS WERE FOUND IN TKV.

4 - ONE OR MORE KEYWORDS WERE NOT FOUND IN TKV.

NOTES:

EACH VALUE RETRIEVED IS PLACED IN THE NEXT N BYTES OF AREA. N IS THE VALUE LENGTH FROM A GIVEN KVBLOCK ENTRY. IF A GIVEN KEYWORD IS NOT FOUND IN THE TKV, ITS AREA SPACE IS SET TO BLANKS. IF THE LENGTH OF THE SPACE IS GREATER THAN THE VALUE RETRIEVED, THE SPACE IS PADDED WITH BLANKS.

IF THE FIRST CHARACTER OF A KEYWORD NAME IS '*' (ASTERISK) THEN THE KEYWORD/VALUE ENTRY IS ASSUMED NOT TO BE IN THE TKV. THE SEARCH OF THE TKV IS SKIPPED, THE OUTPUT AREA IS SET TO BLANKS AND THE RETURN CODE IS SET TO 4.

CKVPUT - COMMON KEYWORD-VALUE PUT ROUTINE

PURPOSE:

CKVPUT IS USED TO STORE KEYWORDS AND THEIR "REMEMBERED" VALUES INTO THE KEYWORD/VALUE TABLE (TKV). CKVGET IS USED TO RETRIEVE VALUES FROM THE TABLE.

INVOKED WITH:

CALL TO CKVPUT

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	KVBLOCK	<KVBLOCK>	INPUT	KEYWORD/VALUE CONTROL BLOCK
3.	AREA	CHAR(*)	INPUT	VALUE AREA

WHERE

KVBLOCK - CONSISTS OF ONE OR MORE KEYWORD/VALUE ENTRIES DELIMITED BY A BYTE SET TO '00'X. SEE DATA AREAS SECTION FOR A DESCRIPTION OF THE KEYWORD/VALUE BLOCK.

AREA - AN AREA EQUAL IN LENGTH TO THE SUM OF THE VALUE LENGTH FIELD(S) IN THE KVBLOCK. AREA CONTAINS THE VALUES TO BE STORED INTO THE TKV.

RETURN CODES:

0 - ALWAYS.

NOTES:

EACH KEYWORD/VALUE PAIR REPLACES AN EXISTING TKV ENTRY OF THE SAME KEYWORD NAME, OR IS ADDED TO THE TKV IF NOT CURRENTLY STORED THERE.

EACH VALUE IS BACKSCANNED FOR THE LAST NON-BLANK CHARACTER BEFORE STORING, AND ONLY THE NON-BLANK CHARACTERS ARE STORED, UNLESS THE KEYWORD IS FOUND IN THE "FIXED" PART OF THE TKV. BLANK VALUES ARE NOT STORED IN THE "VARIABLE" PART OF THE TKV.

THESE TECHNIQUES REDUCE THE TKV LENGTH AS WELL AS THE SPFPARMS DATA SET BLKSIZE REQUIRED. IF THERE IS INSUFFICIENT SPACE TO STORE A KEYWORD/VALUE IN THE TKV, THE USER IS NOTIFIED AND THE TKVFULL FLAG IN THE TKV HEADER IS SET.

IF THE FIRST CHARACTER OF A KEYWORD IS '*' (ASTERISK) THEN THE KEYWORD/VALUE ENTRY IS NOT STORED IN THE TKV.

PURPOSE:

CLM IS A SUBROUTINE USED TO LOAD COMMON SUBROUTINE MODULES. THESE MODULES CONTAIN SUBROUTINES AND THEIR ASSOCIATED ADDRESSES. THE ADDRESSES FROM THE PSEUDO TSC (AT THE ENTRY POINT OF THE MODULE) ARE STORED IN THE CORRESPONDING ENTRIES IN THE REAL TSC FOR THE CURRENT PROCESSOR TASK.

INVOKED WITH:

CALL TO CLM

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	NAME	CHAR(8)	INPUT	LOAD MODULE NAME

WHERE

NAME - THE NAME OF THE MODULE TO BE LOADED.

RETURN CODES:

0 - NORMAL COMPLETION.

4 - ERROR IN THE LOADED MODULE. AN ENTRY FROM THE LOADED MODULE CANNOT BE STORED IN THE REAL TSC BECAUSE THERE IS NO CORRESPONDING ENTRY.

NOTES:

CLM IS USED BY PMD TO LOAD A MODULE THAT IS IDENTIFIED WITH INPUT FROM APRIOPT. THIS ALLOWS EXIT ROUTINES OR ALTERNATE ROUTINES TO TAKE EFFECT DURING THE LIFE OF A PROCESSOR. THE MODULE IS DELETED BEFORE THE NEXT PRIMARY OPTION IS INVOKED BY PMD.

CLOG - COMMON LOG ROUTINE CLOG

PURPOSE:

CLOG IS USED TO WRITE AN ENTRY TO THE SPF LOG DATA SET. CLOG ADDS A TIME STAMP TO THE MESSAGE BEFORE WRITING IT. IT ALSO WRITES PAGE HEADINGS.

INVOKED WITH:

CALL TO CLOG

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MSGID	CHAR(4)	INPUT	MESSAGE ID
3.	PARM1	*	INPUT	LOG MESSAGE PARAMETER 1
..	...			
..	PARMN	*	INPUT	LOG MESSAGE PARAMETER 'N'

WHERE

MSGID - IDENTIFIES THE MESSAGE FROM THE SPF MESSAGE DATA SET THAT IS TO BE USED IN CREATING THE LOG RECORD.

PARM(S) - PARM1 THROUGH PARMN ARE OPTIONAL PARAMETERS. THEY CAN BE IN ANY FORMAT THAT CAN BE HANDLED BY CMSG AND ARE USED AS SUBSTITUTIONAL PARAMETERS IN FORMATTING THE LOG MESSAGE. A MAXIMUM OF 10 OPTIONAL PARAMETERS IS SUPPORTED BY CLOG.

RETURN CODE:

0 - ALWAYS.

NOTES:

THE CLOG PARAMETER LIST MUST BE TERMINATED BY A VLIST FLAG.

SPF PROGRAMS SHOULD USE CLOG TO WRITE LOG MESSAGES ANY TIME A PERMANENT CHANGE IS MADE TO A DATA SET, OR A SIGNIFICANT EVENT OCCURS SUCH AS SPF INITIALIZATION/TERMINATION.

TO FORM THE LOG RECORD, THE LEVEL 1 AND LEVEL 2 MESSAGES FROM THE SPF MESSAGE DATA SET ARE COMBINED TO PRODUCE A LOG RECORD OF 96 CHARACTERS.

IF THE LOG DATA SET PRIMARY ALLOCATION (AS SPECIFIED USING OPTION 0.2) IS NOT ZERO, CLOG WILL CALL CTA TO ALLOCATE THE LOG DATA SET THE FIRST TIME THAT A LOGGING REQUEST IS MADE. THEN IT WILL LOG THE START OF SESSION MESSAGE (P001) BEFORE LOGGING THE REQUEST. THE ONE EXCEPTION IS WHEN THE FIRST REQUEST IS FOR THE SESSION TERMINATION MESSAGE (P002). IN THIS CASE, THE LOG IS ALLOCATED AND START AND END OF SESSION MESSAGES ARE LOGGED ONLY IF TSV FIELD "$LOGFLAG" IS "A" (ALWAYS).

CMB - COMMON MENU BUILD ROUTINE CMB

PURPOSE:

CMB IS USED TO BUILD A PARAMETER LIST WHICH CAN BE USED TO CALL MHA. ALL PARAMETER ENTRIES IN THE LIST WILL BE INITIALIZED TO POINT TO THE PROPER KEYWORD VALUES (ALSO OBTAINED BY CMB).

INVOKED WITH:

CALL TO CMB

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MHAPM	*	IN/OUT	MHA PARAMETER LIST
3.	KVBPM	*	OUTPUT	KVBLOCK AREA
4.	KVGPM	*	OUTPUT	KEYWORD VALUE AREA

WHERE

MHAPM - IS AN AREA FORMATTED EXACTLY THE SAME AS THE INPUT PARAMETER LIST TO MHA. THE CALLER MUST INITIALIZE THE TLD POINTER FIELD, THE MENU ID FIELD, AND THE PARM1, ..., PARMN FIELDS. THE PARM1, ..., PARMN FIELDS SHOULD BE EACH INITIALIZED TO ZERO WITH THE HIGH ORDER BIT ON IN THE LAST (NTH) FIELD. IT WILL BE RETURNED BY CMB WITH THE PARM FIELDS SET TO THE ADDRESSES OF THE CORRESPONDING KEYWORD VALUES, (IN THE KVGPM AREA).

KVBPM - IS AN AREA WHERE A KVBLOCK IS BUILT. IT IS FORMATTED BY CMB BASED ON DATA RETURNED BY A NON-DISPLAY CALL TO MHA FOR THE REQUESTED MENU ID. IT IS THE CALLER'S RESPONSIBILITY TO SUPPLY A LARGE ENOUGH AREA.

KVGPM - IS AN AREA INTO WHICH THE KEYWORD VALUES, ASSOCIATED WITH THE REQUESTED MENU, ARE READ. IT IS THE CALLER'S RESPONSIBILITY TO SUPPLY A LARGE ENOUGH AREA.

RETURN CODE:

0 OR 4 - RETURN CODE IS PASSED BACK FROM CALL TO CKVGET.

NOTES:

THIS ROUTINE IS CALLED BY UOL AND USC.

PURPOSE:

CML IS USED TO PROCESS MEMBER LISTS. SEVERAL SERVICES ARE PROVIDED:

1. READ PDS DIRECTORY AND BUILD IN-MEMORY MEMBER NAME LIST.
2. FREEMAIN PREVIOUSLY CONSTRUCTED IN-MEMORY MEMBER LIST.
3. ADD AN ENTRY TO A PREVIOUSLY CONSTRUCTED IN-MEMORY MEMBER LIST.
4. DISPLAY A MEMBER LIST AND PROCESS SCROLLING AND USER SELECTIONS. SPF STATISTICS ARE DISPLAYED IF PRESENT. CML INTERFACES WITH A SELECTION ROUTINE SUPPLIED BY THE CALLING PROGRAM OR A BUILT-IN SELECTION ROUTINE. SELECTION ROUTINE INTERFACES ARE DESCRIBED IN FOLLOWING PAGES OF THIS SECTION.

INVOKED WITH:

CALL TO CML

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	CODE	BIT(32)	INPUT	CONTROL BIT CODES
4.	SUBR	PTR(31)	INPUT	SELECT SUBROUTINE
5.	NAME	CHAR(8)	IN/OUT	MEMBER NAME
6.	PARM	*	INPUT	PARM FOR SELECT SUBROUTINE
7.	MSGID	CHAR(4)	OUTPUT	ERROR MESSAGE ID

WHERE

TFD - THE DATA SET MUST BE A PDS WITH AN OPEN DCB WITH DSORG=PO. TFDDDNAM MUST BE SET UP.

CODE - 32 BIT SWITCHES, AS FOLLOWS:

ECODE(1) TO ECODE(24) - RESERVED

ECODE(25) 0 - NOT AN ADD ENTRY REQUEST
1 - ADD AN ENTRY TO MEMBER LIST

ECODE(26) 0 - NO TTR IN MEMBER LIST
1 - PLACE TTR IN MEMBER LIST

ECODE(27) 0 - ALLOW SELECTION OF MEMBERS NOT IN LIST
1 - ALLOW SELECTION OF MEMBERS IN LIST ONLY

ECODE(28) 0 - DISPLAY MEMBER LIST
1 - SUPPRESS DISPLAY

ECODE(29) 0 - NO RENAME FIELD
1 - RENAME FIELD

ECODE(30) 0 - SINGLE INPUT DATA SET
1 - CONCATENATED INPUT DATA SETS

(CONTINUED ON NEXT PAGE)

ECODE(31) 0 - RETAIN MEMBER LIST FOR FUTURE CALLS
1 - FREE MEMBER LIST ON EXIT

ECODE(32) 0 - DO NOT BUILD A MEMBER LIST
1 - READ DIRECTORY AND BUILD MEMBER LIST

SUBR - ADDRESS OF A SELECT ROUTINE WHICH CML WILL INVOKE WHEN A SELECT CODE IS ENTERED. IF SUBR IS ZERO, CML WILL CALL THE BUILT-IN SELECT ROUTINE DESCRIBED BELOW.

NAME - ON INPUT, NAME OF THE FIRST MEMBER TO BE DISPLAYED ON THE SCREEN. IF THE MEMBER REQUESTED DOES NOT EXIST, THE LIST IS DISPLAYED STARTING WITH THE MEMBER PRECEDING THE MEMBER REQUESTED IN COLLATING SEQUENCE.

ON OUTPUT, NAME OF THE MEMBER SELECTED IF THE BUILT-IN SELECT ROUTINE IS USED.

PARM - PARAMETER THAT IS PASSED TO THE SELECT ROUTINE WHEN ONE IS SPECIFIED.

MSGID - CONTAINS AN ERROR MESSAGE ID IF RETURN CODE IS NOT 0.

RETURN CODES:

0 - NORMAL RETURN.

4 - ERROR RETURN. ONE OF THE FOLLOWING OCCURRED.
- NO MEMBERS IN DATA SET.
- BLDL ERROR OCCURRED.
- I/O ERROR READING THE PDS DIRECTORY.
- OPEN OF DCB FOR DIRECTORY READ FAILED.

NOTES:

ON EXIT FROM CML, TFDCML POINTS TO THE MEMBER LIST IN SUBPOOL 3 IF THE MEMBER LIST WAS RETAINED.

IF CML IS PASSED A ZERO 'SUBR' ADDRESS, IT CALLS A BUILT-IN SELECT SUBROUTINE (USED BY BROWSE AND EDIT). THE BUILT-IN ROUTINE PERFORMS THE FOLLOWING FUNCTIONS:

VALIDITY CHECKS THE SELECT CODE -- 'S' IS THE ONLY VALID CODE.
CHANGES A VALID SELECT CODE TO BLANK.
RETURNS THE MEMBER SELECTED IN THE NAME PARAMETER.
IGNORES THE RENAME FIELD.
SETS THE ENDFLAG ON.

SEE THE NEXT PAGE FOR INFORMATION ON CML SELECT SUBROUTINES INVOKED WHEN THE SUBR PARAMETER IS NON-ZERO.

(CONTINUED ON NEXT PAGE)

CML INVOKED SELECT SUBROUTINE:

CML INVOKES A "SELECT SUBROUTINE" TO PROCESS EACH MEMBER SELECTED FROM THE LIST. THE 'SUBR' PARAMETER PASSED TO CML IS THE ADDRESS OF THE SELECT SUBROUTINE.

THE CALLING SEQUENCE PARAMETERS FOR THE SELECT SUBROUTINE ARE:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	CODE	BIT(32)	INPUT	CONTROL BIT CODES
4.	NAME	CHAR(8)	IN/OUT	MEMBER NAME TO CML
5.	PARM	*	IN/OUT	PARAMETER FOR SELECT SUBROUTINE
6.	SCODE	CHAR(1)	IN/OUT	SELECT CODE
7.	MEMBER	CHAR(8)	INPUT	SELECTED MEMBER NAME
8.	RENAME	CHAR(8)	IN/OUT	RENAME FIELD
9.	FLAGS	BIT(8)	OUTPUT	FLAGS

WHERE

CODE - 32 BIT SWITCHES, AS PASSED TO CML.

NAME - MEMBER NAME, AS PASSED TO CML.

PARM - PARAMETER THAT IS PASSED TO THE SELECT ROUTINE FROM THE PROGRAM THAT CALLED CML.

SCODE - IS THE SELECTION CODE THAT WAS ENTERED BY THE TERMINAL USER. THIS CODE SHOULD BE SET TO BLANK OR '*' BEFORE EXITING UNLESS ENDFLAG IS SET.

MEMBER - IS THE NAME OF THE MEMBER THAT WAS SELECTED BY THE TERMINAL USER.

RENAME - IS THE CONTENTS OF THE RENAME FIELD ASSOCIATED WITH THE MEMBER. THE SELECT ROUTINE CAN CHANGE THIS FIELD.

FLAGS - IS A BYTE CONTAINING TWO FLAGS IN THE FOLLOWING FORMAT.

```
1 FLAGS,
  2 ENDFLAG  BIT(1),
  2 PAGEFLAG BIT(1),
  2 *        BIT(6),
```

```
ENDFLAG - INDICATES ACTION TO BE TAKEN BY CML UPON
          RETURN FROM SELECT SUBROUTINE, AS FOLLOWS:
          0 - RESCAN SCREEN IMAGE FOR SELECT CODES.
          1 - RETURN TO CALLING PROGRAM.

PAGEFLAG - SET BY CML AS FOLLOWS:
          0 - IF THE CURRENT SELECTION IS NOT THE
              LAST SELECTION ON THE PAGE.
          1 - IF THE CURRENT SELECTION IS THE LAST
              ON THE PAGE.  THE SELECT ROUTINE SHOULD
              CLOSE AND RELEASE THE DATA SET BEING
              PROCESSED IN THIS CASE.
```

THE RETURN CODE (IN REG 15) FROM THE SELECT SUBROUTINE IS NOT USED BY CML.

NOTES:

THE NAMES OF THE OBJECT MODULES WHICH ARE CML SELECT SUBROUTINES ARE AS FOLLOWS:

UDMS - USED BY UDM (MEMBER LIST OPTION OF LIBRARY UTILITY)
UMCS - USED BY UMC (MOVE/COPY UTILITY)
URSS - USED BY URS (RESET STATISTICS UTILITY)

CMSG - COMMON MESSAGE ROUTINE CMSG

PURPOSE:

CMSG IS USED TO FORMAT ERROR MESSAGES. CMSG OPTIONALLY READS A MESSAGE FROM THE SPFMSGS DATA SET, FORMATS THE MESSAGE AND RETURNS THE MESSAGE TO THE CALLER. PARAMETERS MAY BE SUBSTITUTED INTO THE MESSAGE.

INVOKED WITH:

CALL TO CMSG

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MSGID	CHAR(4)	INPUT	MESSAGE ID
3.	LEVEL	FIXED(31)	INPUT	MESSAGE LEVEL
4.	AREA	CHAR(*)	OUTPUT	AREA FOR MESSAGE
5.	SIZE	FIXED(31)	INPUT	SIZE OF AREA (BYTES)
6.	PARM1	*	SUBSTITUTION PARAMETER 1	
..	...			
..	PARMN	*	SUBSTITUTION PARAMETER N	

WHERE

MSGID - IS THE MESSAGE ID. THE FIRST CHARACTER MUST BE AN UPPER CASE ALPHA CHARACTER. THE REMAINING CHARACTERS MUST BE NUMERIC.

LEVEL - 0 - TO REQUEST THE HELP (TUTOR) NAME (CHARS(8)) BE RETURNED TO TLDHELP.
1 - TO REQUEST THE LEVEL 1 (SHORT) MESSAGE
2 - TO REQUEST THE LEVEL 2 (LONG) MESSAGE
3 - TO REQUEST SHORT AND LONG MESSAGES, AND FOR TLDHELP AND TLDALARM TO BE SET FROM THE MESSAGE.

AREA - IS THE AREA WHERE THE MESSAGE(S) IS TO RETURNED. IF LEVEL 3 IS REQUESTED, THE SHORT MESSAGE (24 BYTES) IS FOLLOWED BY THE LONG MESSAGE (LENGTH SPECIFIED BY THE SIZE PARAMETER, I.E. AREA MUST BE 24 PLUS SIZE).

SIZE - IS THE NUMBER OF AREA BYTES PROVIDED. IF THE MESSAGE EXCEEDS THE SIZE, IT WILL BE TRUNCATED. NORMALLY, THE VALUE OF THIS PARAMETER IS:

LEVEL	SIZE
0	NOT USED
1	24
2	72
3	72

PARM(S) - PARM1 THROUGH PARMN ARE OPTIONAL PARAMETERS FOR SUBSTITUTION INTO THE CORRESPONDING PARAMETER FIELDS OF THE MESSAGE (SEE THE INSTALLATION AND CUSTOMIZATION GUIDE FOR A DESCRIPTION OF SPF MESSAGE FORMATS). A MAXIMUM OF 50 PARAMETERS IS SUPPORTED.

(CONTINUED ON NEXT PAGE)

RETURN CODE:

0 - ALWAYS.

NOTES:

THE INPUT PARAMETER LIST TO CMSG MUST TERMINATE WITH A VLIST FLAG.

CMSG NORMALLY ACCESSES THE SPF MESSAGES DATA SET (SEE THE INSTALLATION AND CUSTOMIZATION GUIDE FOR A DESCRIPTION OF SPF MESSAGE FORMATS). THE REQUESTED MESSAGE IS PLACED IN THE SPECIFIED ANSWER AREA. IF THE AREA IS TOO LONG, IT IS PADDED WITH TRAILING BLANKS. IF TOO SHORT, THE MESSAGE IS TRUNCATED.

IF THE FIRST CHARACTER OF THE MESSAGE ID IS BLANK OR BINARY ZERO, THE MESSAGE IS CONTAINED IN THIS CALLING SEQUENCE, AND THE MESSAGE DATA SET IS NOT ACCESSED. IN THIS CASE, PARM1 IS THE ENTIRE LEVEL 1 MESSAGE (24 CHARACTERS), AND PARM2 IS THE ENTIRE LEVEL 2 MESSAGE (72 CHARACTERS). THE REMAINING PARMS, IF ANY, CORRESPOND TO THE SUBSTITUTABLE PARAMETERS IN THE MESSAGE. THIS SPECIAL CASE ALLOWS PREVIOUSLY RETRIEVED (OR INTERNALLY GENERATED) MESSAGES TO BE PROCESSED BY CMSG FOR THE PURPOSE OF FILLING IN SUBSTITUTABLE PARAMETERS.

CPRINT - COMMON PRINT ROUTINE CPRINT

PURPOSE:

CPRINT WRITES A MEMBER OF A PDS OR A SEQUENTIAL DATA SET TO THE SPF LIST DATA SET. HEADERS AND OTHER INFORMATION ARE PRINTED BASED ON A FORMAT CODE. THE DATA SET OR MEMBER TO BE PRINTED IS NOT MODIFIED.

INVOKED WITH:

CALL TO CPRINT

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	INPUT	FILE DEFINITION TABLE
3.	PROJECT	CHAR(8)	INPUT	PROJECT NAME
4.	LIBRARY	CHAR(8)	INPUT	LIBRARY NAME
5.	TYPE	CHAR(8)	INPUT	TYPE QUALIFIER
6.	MEMBER	CHAR(8)	INPUT	MEMBER NAME
7.	FORMAT	CHAR(1)	INPUT	OUTPUT FORMAT TYPE CODE

WHERE

FORMAT - ONE OF THE FOLLOWING OUTPUT FORMAT TYPE CODES:

'S' - SPF LIBRARY PDS HEADER, START COLS, MOD FLAGS, AND DATA LEN (IF VARIABLE LRECL). PARAMETERS 3, 4, AND 5 ARE USED ONLY FOR THIS FORMAT.

'N' - NONSPF LIBRARY PDS OR SEQUENTIAL HEADER, START COLS, AND DATA LEN (IF VARIABLE LRECL). THE DATASET NAME IN THE TFD IS USED.

'X' - NO HEADERS OR ADDITIONAL INFORMATION.

RETURN CODE:

0 - THE DATA HAS BEEN SUCESSFULLY WRITTEN.

4 - THE DATA WAS NOT WRITTEN OR WAS PARTIALLY WRITTEN DUE TO AN ALLOCATION, OPEN, OR I/O ERROR.

8 - THE DATA WAS NOT WRITTEN DUE TO DATA SET CHARACTERISTICS. A MESSAGE WAS WRITTEN TO THE SPF LISTING DATA SET.

12 - THE DATA WAS NOT WRITTEN BECAUSE THE MEMBER SPECIFIED WAS NOT FOUND OR THERE WAS A FIND OR BLDL ERROR. A MESSAGE WAS WRITTEN TO THE SPF LISTING DATA SET.

16 - THE DATA WAS PARTIALLY WRITTEN DUE TO AN I/O ORROR READING THE DATA. A MESSAGE WAS WRITTEN TO THE SPF LISTING DATA SET.

20 - THE MEMBER OR DATA SET IS EMPTY. A MESSAGE WAS WRITTEN TO THE SPF LISTING DATA SET.

NOTES:

THE DCB MUST BE OPENED FOR INPUT BEFORE CALLING CPRINT.

PURPOSE:

CRELS IS USED TO RELEASE A SHARED DASD PACK AFTER UPDATING A DATA SET, AND TO DEQ THE RESOURCE. CRELS IS USED IN CONJUNCTION WITH THE COMMON RESERVE SUBROUTINE (CRESV).

INVOKED WITH:

CALL TO CRELS

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	INPUT	FILE DESCRIPTION TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

THE TFD BLOCK MUST BE THE SAME AS THAT USED FOR CRESV.

IF THE TFDLENQ FLAG IS SET BY THE CALLER, AN ADDITIONAL DEQ IS ISSUED OF THE FORM USED BY THE LINK EDITOR.

CRELS CLEARS THE TFDRESV FLAG.

PURPOSE:

CRESV IS USED TO RESERVE A SHARED DASD VOLUME PRIOR TO UPDATING A DATA SET. CRESV SHOULD BE CALLED BEFORE WRITING, DELETING, OR RENAMING ANY USER DATA SET OR MEMBER, EXCEPT FOR TEMPORARY DATA SETS WHICH ARE ALLOCATED BY SPF ON BEHALF OF THE USER (SUCH AS SPF*.LIST, SPFLOG*.LIST, SPFTEMP*.LIST, AND SPFTEMP*.CNTL).

INVOKED WITH:

CALL TO CRESV

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	INPUT	FILE DEFINITION TABLE

RETURN CODE:

0 - NORMAL RETURN.

4 - DDNAME IN TFD WAS NOT FOUND IN THE TIOT. RESERVE NOT ISSUED.

NOTES:

THE RESERVE MACRO ISSUED BY CRESV INCLUDES AN "ENQ" FUNCTION. THE ENQ PARAMETERS ARE DERIVED FROM THE TFD PARAMETER.

IF THE DATA SET IS A POTENTIAL LOAD MODULE DATA SET, THE CALLER SHOULD SET THE TFDLENQ SWITCH SO THAT CRESV WILL ISSUE A SECOND RESERVE OR ENQ. THE SECOND ENQ IS THE SAME AS THAT USED BY THE LINK EDITOR.

TO DEQ THE RESOURCE FOLLOWING UPDATE, THE COMMON RELEASE SUBROUTINE (CRELS) IS USED.

REFER TO APPENDIX A FOR MORE INFORMATION ON THE ENQ/DEQ LOGIC.

CSB - COMMON SUBMIT ROUTINE CSB

PURPOSE:

CSB IS A COMMON SUBROUTINE USED FOR SUBMITTING BACKGROUND JOBS TO OS. THE JCL TO BE SUBMITTED IS PASSED TO CSB IN A TEMPORARY CONTROL CARD DATA SET. THE TEMPORARY DATA SET SHOULD CONTAIN A JOB STATEMENT AS THE FIRST RECORD(S), UNLESS THE SUBMIT COMMAND IS SET UP TO PROVIDE VALID JOBCARDS AT YOUR INSTALLATION. THE TEMPORARY DATA SET MUST BE CLOSED AT ENTRY TO CSB.

INVOKED WITH:

CALL TO CSB

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	DSNS	CHAR(46)	INPUT	DATA SET NAME STRUCTURE

WHERE

DSNS - IS THE DATA SET NAME STRUCTURE (STANDARD DAIR FORMAT).

```
E.G.,   DCL 1 DSNS,
              2 DSNL FIXED(15),  /* DSNAME LENGTH */
              2 DSN  CHAR(44);   /* DSNAME        */
```

THE DATA SET NAME IS NORMALLY 'USERID.SPFTEMP1.CNTL' OR 'USERID.SPFTEMP2.CNTL', AS SET UP BY CTA.

RETURN CODE:

0 - NORMAL COMPLETION.

4 - ATTENTION TERMINATION.

8 - ABEND TERMINATION.

OTHER - RETURN CODE FROM SUBMIT COMMAND.

NOTES:

CSB CALLS CAT TO ATTACH THE TSO SUBMIT COMMAND WHICH ACTUALLY SUBMITS THE JOB TO THE JOB QUEUE.

CSCROLL - COMMON SCROLL ROUTINE

PURPOSE:

CSCROLL IS USED TO INTERPRET THE SCROLL PFK'S AND THE SCROLL AMOUNT FIELD, AND RETURN AN UPDATED BINARY VALUE FOR THE FIRST LINE OR COLUMN BEING VIEWED.

INVOKED WITH:

CALL TO CSCROLL

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	PARM	*	IN/OUT	PARAMETER STRUCTURE

WHERE

PARM - IS A STRUCTURE CONSISTING OF:

CURLINE	FIXED(31)	IN/OUT	CURRENT RELATIVE LINE NUMBER
MAXLINE	FIXED(31)	INPUT	MAXIMUM RELATIVE LINE NUMBER
PGLEN	FIXED(31)	INPUT	PAGE LENGTH (NUMBER OF LINES)
CURCOL	FIXED(31)	IN/OUT	CURRENT COLUMN NUMBER
MAXCOL	FIXED(31)	INPUT	MAXIMUM COLUMN NUMBER
PGWIDTH	FIXED(31)	INPUT	PAGE WIDTH (NUMBER OF COLUMNS)

PARAMETER USAGE

CURLINE - ON INPUT, INDICATES THE FIRST LINE BEING DISPLAYED (VALID RANGE: 1 <= CURLINE <= MAXLINE). ON OUTPUT, CURLINE WILL BE SET AS FOLLOWS:

FOR SCROLL DOWN: CURLINE = CURLINE + AMOUNT
(BUT NOT EXCEEDING MAXLINE)
FOR SCROLL UP: CURLINE = CURLINE - AMOUNT
(BUT NOT LESS THAN ONE)

WHERE "AMOUNT" IS EITHER THE NUMERIC VALUE ENTERED BY THE USER OR:

PGLEN IF 'PAGE' WAS SPECIFIED
PGLEN/2 IF 'HALF' WAS SPECIFIED

IF SCROLL UP "MAX" IS REQUESTED, CURLINE IS SET TO 1. IF SCROLL DOWN "MAX" IS REQUESTED, CURLINE IS SET TO A NEGATIVE NUMBER = -PGLEN.

MAXLINE - (INPUT ONLY - NOT CHANGED BY CSCROLL). INDICATES THE MAXIMUM VALID LINE NUMBER TO BE VIEWED. NOTE: IF THE ACTUAL NUMBER OF LINES IN THE DATA SET IS UNKNOWN, MAXLINE SHOULD BE SET TO A VERY LARGE NUMBER, SUCH AS '7FFFFFFF'X.

PGLEN - (INPUT ONLY - NOT CHANGED BY CSCROLL). INDICATES THE CURRENT NUMBER OF LINES PER LOGICAL PAGE (FOR SCROLLING DOWN OR UP BY A PAGE).

(CONTINUED ON NEXT PAGE)

CURCOL - ON INPUT, INDICATES THE 1ST COLUMN BEING DISPLAYED (VALID RANGE: 1 <= CURCOL <= MAXCOL-PGWIDTH+1). ON OUTPUT, CURCOL WILL BE SET AS FOLLOWS:

```
FOR SCROLL RIGHT: CURCOL = CURCOL + AMOUNT
                  (NOT EXCEEDING MAXCOL-PGWIDTH+1)
FOR SCROLL LEFT:  CURCOL = CURCOL - AMOUNT
                  (BUT NOT LESS THAN ONE)
```

WHERE "AMOUNT" IS EITHER THE NUMERIC VALUE ENTERED BY THE USER OR:

PGWIDTH IF 'PAGE' WAS SPECIFIED
PGWIDTH/2 IF 'HALF' WAS SPECIFIED

IF SCROLL RIGHT "MAX" IS REQUESTED, CURCOL IS SET TO MAXCOL - PGWIDTH + 1.
IF SCROLL LEFT "MAX" IS REQUESTED, CURCOL IS SET TO 1.

MAXCOL - THE MAXIMUM COLUMN TO BE VIEWED (SHOULD NOT BE LESS THAN PGWIDTH).

PGWIDTH - THE CURRENT NUMBER OF COLUMNS PER LOGICAL PAGE (FOR SCROLLING RIGHT OR LEFT BY A PAGE).

RETURN CODES:

0 - NORMAL COMPLETION.

1 - INVALID SCROLL AMOUNT.

2 - NONE OF THE SCROLL PFK'S WAS PRESSED. THIS IS A PROGRAMMING ERROR -- CSCROLL SHOULD BE CALLED ONLY WHEN BIT TLDSCROL IS ON IN THE TLD.

NOTES:

IF RETURN CODE IS NON-ZERO, NEITHER CURLINE NOR CURCOL IS CHANGED.

CSCROLL "CLEANS UP" THE VALUE IN THE SCROLL AMOUNT FIELD AS FOLLOWS:

```
'P---' REPLACED WITH 'PAGE'
'H---' REPLACED WITH 'HALF'
'C---' REPLACED WITH 'CSR'
'M---' REPLACED WITH PREVIOUS VALUE ('MAX' DOES
                           NOT REMAIN IN EFFECT).
FOR NUMERICS FOLLOWED BY ALPHABETICS (E.G., '15XX')
THE LOW ORDER ALPHABETIC CHARACTERS ARE BLANKED OUT.
```

CSCROLL ALWAYS LEAVES THE CURSOR POSITIONED UNDER THE BEGINNING OF THE SCROLL AMOUNT FIELD.

CSM - COMMON STORAGE MANAGEMENT ROUTINE CSM

PURPOSE:

CSM PROVIDES AN INTERFACE WITH THE GETMAIN AND FREEMAIN SYSTEM SERVICES.

INVOKED WITH:

CALL TO CSM

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CODE	FIXED(31)	INPUT	GETMAIN/FREEMAIN CODE
3.	LENGTH	FIXED(31)	OUTPUT	LENGTH OF AREA

4.	ADDRESS	PTR(31)
		FIXED(31)
			IN/OUT
			IN/OUT
				ADDRESS OF PTR, OR
				REGISTER NUMBER (1-12)

5.	SUBPOOL	FIXED(31)	INPUT	SUBPOOL TO BE USED

WHERE

CODE - SYMBOLICALLY - GETMAINC, GETMAINU OR FREEMAIN.
NUMERIC CODE - (1) , (2) OR (3)

LENGTH - NUMBER OF BYTES TO GETMAIN OR FREEMAIN. USE ZERO IF SUBPOOL (PARM 5) IS TO BE FREEMAINED.

ADDRESS - IF A NUMBER FROM 1-12 IS PRESENT, THE CORRESPONDING REGISTER IS USED TO PASS AN ADDRESS FROM OR TO CSM.
- IF AN ADDRESS IS PRESENT IT POINTS TO A WORD TO BE USED TO PASS AN ADDRESS FROM OR TO CSM.
- IF A GETMAIN IS TO BE DONE, THE ADDRESS OF THE AREA WHICH IS GETMAINED IS PASSED AS OUTPUT FROM CSM BACK TO THE CALLER.
- IF A FREEMAIN IS TO BE DONE, THE ADDRESS OF THE AREA TO BE FREEMAINED IS PASSED FROM THE CALLER TO CSM.

RETURN CODE:

0 - GETMAIN OR FREEMAIN SUCCESSFUL.

4 - CONDITIONAL GETMAIN WAS UNSUCCESSFUL.

8 - INVALID REQUEST CODE WAS ENTERED OR ERROR RETURNED FROM GETMAIN OR FREEMAIN.

NOTES:

CSM DOES NOT USE SPFPROC/SPFRETRN BECAUSE NO DYNAMIC AREA IS USED.

CTA - COMMON ALLOC-OPEN TEMP DATA SET ROUTINE CTA

PURPOSE:

CTA PASSES CONTROL TO CT1 TO ALLOCATE AND OPEN THE FOLLOWING DATA SETS THAT MAY BE USED DURING AN SPF SESSION.

USERID.SPFX.LIST	- LIST DATA SET
USERID.SPFLOGX.LIST	- LOG DATA SET
USERID.SPFTEMP1.CNTL	- CONTROL CARD DATA SET (LOGICAL DISPLAY 1)
USERID.SPFTEMP1.LIST	- LISTING DATA SET (LOGICAL DISPLAY 1)
USERID.SPFTEMP2.CNTL	- CONTROL CARD DATA SET (LOGICAL DISPLAY 2)
USERID.SPFTEMP2.LIST	- LISTING DATA SET (LOGICAL DISPLAY 2)
USERID.SPFEDIT1.BACKUP	- FIRST EDIT RECOVERY DATA SET
USERID.SPFEDIT2.BACKUP	- SECOND EDIT RECOVERY DATA SET

INVOKED WITH:

CALL TO CTA

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	IN/OUT	LOGICAL DISPLAY TABLE
2.	TYPE	CHAR(8)	INPUT	TYPE OF DATA SET
3.	OPTION	FIXED(31)	INPUT	TYPE OF OPEN REQUIRED

WHERE

TYPE - 8 CHARACTERS CODED EXACTLY AS SHOWN BELOW (INCLUDING THE CHARACTER 'X' AND TRAILING BLANKS). ANY OTHER TYPE CODE CAUSES CTA TO ABEND WITH USER CODE 970.

'SPFLIST '	- FOR THE LIST DATA SET.
'SPFLOG '	- FOR THE LOG DATA SET.
'SPFCNTLX'	- FOR A CONTROL CARD DATA SET.
'SPFLISTX'	- FOR A LISTING DATA SET.
'SPFEDITX'	- FOR AN EDIT RECOVERY DATA SET.

OPTION - THE OPEN TYPE REQUIRED.

0 - OUTPUT
1 - INPUT

RETURN CODE:

0 - ALWAYS.

(CONTINUED ON NEXT PAGE)

NOTES:

THIS MODULE HAS REPLACED THE SPF VERSION 2.1 CALLOC SUBROUTINE.

SPFLOG, SPFLIST, AND THE EDIT RECOVERY DATA SETS HAVE TFD'S COMPILED INTO THE TSI (TABLE TDS). TFD'S FOR THE SPFCNTLX AND SPFLISTX DATA SETS ARE CREATED BY CTA THE FIRST TIME THAT THE DATA SET IS USED. THE TFD FOR EACH TYPE OF DATA SET IS LOCATED AS FOLLOWS:

```
'SPFLOG  ' - TDSLOGP  -> TFD FOR LOG DATA SET.
'SPFLIST ' - TDSLISTP -> TFD FOR LIST DATA SET.
'SPFCNTLX' - TLDTFDCP -> TFD FOR CONTROL CARD DATA SET.
'SPFLISTX' - TLDTFDLP -> TFD FOR LISTING DATA SET.
'SPFEDITX' - TLDTFDEP -> TFD FOR EDIT RECOVERY DATA SET.
```

ON RETURN FROM CTA, THE APPROPRIATE TFD, DATA SET NAME, AND DCB HAVE BEEN SET UP. IF CTA WAS SUCCESSFUL, THE DCB WILL BE OPENED. IF ALLOCATION OR OPEN FAILED, THE DCB WILL BE CLOSED. THE CALLING PROGRAM IS RESPONSIBLE FOR TESTING "TFDOPN" BEFORE ATTEMPTING TO USE THE DCB.

THE ADDRESSES OF THE DATA SET NAME STRUCTURE (TFDDSNSP) AND DCB (TFDDCBP) ARE CONTAINED IN THE TFD.

IF THE DATA SET HAS ALREADY BEEN ALLOCATED WHEN CTA IS CALLED, THE DCB WILL BE OPENED. IF THE DCB IS OPEN WHEN CTA IS CALLED, THE DCB WILL BE CLOSED AND REOPENED.

IF ALLOCATION FAILS, CTA WILL TPUT LINE MESSAGES THAT INDICATE THE DATA SET NAME AND REASON FOR FAILURE. THE USER CAN THEN ATTEMPT TO CONTINUE WITHOUT THE DATA SET (FOR EXAMPLE WITHOUT THE LOG DATA SET), OR HE CAN CHOOSE TO EXIT FROM SPF.

WHEN AN ATTEMPT (EITHER SUCCESSFUL OR UNSUCCESSFUL) HAS BEEN MADE TO ALLOCATE A DATA SET THE "TFDALLOC" FLAG BIT IS SET ON. CALLING PROGRAMS CAN CHECK THIS BIT BEFORE CALLING CTA TO AVOID REPEATED ATTEMPTS TO ALLOCATE A DATA SET THAT, FOR SOME REASON OR OTHER, CANNOT BE ALLOCATED, AND THUS AVOID REPEATED NOTIFICATIONS TO THE TERMINAL USER THAT THE DATA SET CANNOT BE ALLOCATED.

A TYPICAL SEQUENCE FOR USING CTA FOR SPFLIST OR SPFLOG MIGHT BE:

```
IF TFDALLOC = OFF THEN
  CALL CTA (TLD,'SPFLIST ');      (NOTE 8 CHARACTER NAME)
ELSE
  ;
IF TFDOPN = ON THEN
  WRITE OUTPUT TO LIST DATA SET;
ELSE
  DISPLAY ERROR MESSAGE;
```

A TYPICAL SEQUENCE FOR USING CTA FOR SPFLISTX, SPFCNTLX, OR SPFEDITX MIGHT BE:

```
CALL CTA (TLD,'SPFCNTLX');
IF TFDOPN = ON THEN
  WRITE OUTPUT TO TEMPCNTL DATA SET;
ELSE
  DISPLAY ERROR MESSAGE;
```

PURPOSE:

CTF PASSES CONTROL TO CT2 TO CLOSE AND FREE THE DATA SETS THAT WERE ALLOCATED AND OPENED BY CTA.

INVOKED WITH:

CALL TO CTF

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	IN/OUT	LOGICAL DISPLAY TABLE
2.	TYPE	CHAR(8)	INPUT	TYPE OF DATA SET
3.	OPTION	CHAR(1)	INPUT	FREE OPTION

WHERE

TYPE - 8 CHARACTERS CODED EXACTLY AS SHOWN BELOW (INCLUDING THE CHARACTER 'X' AND TRAILING BLANKS). ANY OTHER TYPE CODE CAUSES CTF TO ABEND WITH USER CODE 971.

```
'SPFLIST ' - FOR THE LIST DATA SET.
'SPFLOG  ' - FOR THE LOG DATA SET.
'SPFCNTLX' - FOR A CONTROL CARD DATA SET.
'SPFLISTX' - FOR A LISTING DATA SET.
'SPFEDITX' - FOR AN EDIT RECOVERY DATA SET.
```

OPTION - SINGLE CHARACTER AS SHOWN BELOW. ANY OTHER CODE IS TREATED LIKE 'K'.

```
'D' - IF THE DATA SET IS TO BE DELETED.
'K' - IF THE DATA SET IS TO BE KEPT.
```

RETURN CODE:

0 - ALWAYS.

NOTES:

THIS MODULE HAS REPLACED THE SPF VERSION 2.1 CFREE SUBROUTINE.

ON RETURN FROM CTF, THE APPROPRIATE DCB WILL HAVE BEEN CLOSED, AND THE ALLOCATION FREED.

FOR FURTHER INFORMATION ON THE DATA SETS HANDLED BY CTA/CTF SEE THE DESCRIPTION OF CTA.

FOR FURTHER INFORMATION ON THE CTF/CT2 INTERFACE, SEE THE DESCRIPTIONS OF CT1 AND CT2.

CTGET - COMMON TGET (TERMINAL GET) ROUTINE CTGET

PURPOSE:

CTGET IS USED TO ISSUE A TGET SVC.

INVOKED WITH:

CALL TO CTGET

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPALY TABLE
2.	AREA	CHAR(*)	OUTPUT	INPUT/OUTPUT AREA
3.	AREASIZE	FIXED(31)	INPUT	INPUT AREA SIZE
4.	OPTIONS	BIT(32)	INPUT	TGET OPTIONS
5.	INSIZE	FIXED(31)	OUTPUT	INPUT DATA SIZE

WHERE

AREA - THE AREA WHERE SCREEN DATA WILL BE RECEIVED FROM THE TERMINAL ACCESS METHOD.

AREASIZE - THE MAXIMUM SCREEN DATA SIZE ACCEPTABLE.

OPTIONS - CODES REPRESENTING THE VARIOUS TGET MACRO OPTIONS. THE ONLY CODE USED BY SPF IS '129' = 'ASIS,WAIT'

INSIZE - THE ACTUAL SIZE OF SCREEN DATA PLACED IN AREA.

RETURN CODE:

RETURN CODE FROM TGET SVC.

NOTES:

NONE.

CTPUT - COMMON TPUT (TERMINAL PUT) ROUTINE CTPUT

PURPOSE:

CTPUT IS USED TO ISSUE A TPUT SVC.

INVOKED WITH:

CALL TO CTPUT

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	AREA	CHAR(*)	OUTPUT	OUTPUT DATA AREA
3.	AREASIZE	FIXED(31)	INPUT	OUTPUT DATA SIZE
4.	OPTIONS	BIT(32)	INPUT	TPUT OPTIONS

WHERE

AREA - THE OUTPUT DATA FOR THE TERMINAL ACCESS METHOD TO SEND TO THE SCREEN.

AREASIZE - THE SIZE OF THE OUTPUT DATA.

OPTIONS - CODES REPRESENTING THE VARIOUS TPUT MACRO OPTIONS. THE FOLLOWING ARE USED BY SPF:

0 - 'NO OPTIONS'
3 - 'FULLSCR'
8 - 'HOLD'
11 - 'FULLSCR,HOLD'

RETURN CODE:

RETURN CODE FROM TPUT SVC.

NOTES:

NONE.

PURPOSE:

CT1 IS USED TO ALLOCATE AND OPEN TEMPORARY SPF DATA SETS. IT HANDLES ALLOCATION ERROR RECOVERY IF REQUIRED.

INVOKED WITH:

CALL FROM CTA (EITHER DIRECLY OR VIA "SYSCCI" - COMMON CONTROLLER INTERFACE, AN SPF INTERNAL PLS PROCEDURE)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	DA08	*	IN/OUT	DA08 BLOCK
4.	DA34	*	INPUT	DA34 BLOCK

WHERE

TFD - A TFD WHICH CONTAINS A DEFAULT DDNAME AND POINTERS TO A DATA SET NAME STRUCTURE AND DCB.

DA08 - IS THE TSO DAIR DA08 CONTROL BLOCK USED TO ALLOCATE A DATASET.

DA34 - IS THE TSO DAIR DA34 BLOCK USED TO SPECIFY DATASET ATTRIBUTES.

RETURN CODE:

0 - SUCCESSFUL ALLOCATION AND OPENING OF THE DATA SET.

1 - UNABLE TO OPEN DATA SET.

>1 - DAIR RETURN CODE FROM UNSUCCESSFUL ALLOCATION.

(CONTINUED ON NEXT PAGE)

NOTES:

RECOVERY FROM THE FOLLOWING ALLOCATION ERRORS IS ATTEMPTED.

- IF THE DATA SET TO BE ALLOCATED IS CATALOGED BUT DOES NOT EXIST ON THE CATALOGED DASD VOLUME, OR IF THE VOLUME IS NOT MOUNTED, CT1 WILL UNCATALOG THE DATA SET AND ATTEMPT ALLOCATION AGAIN.
- IF THE DATA SET TO BE ALLOCATED AND OPENED IS NOT CATALOGED, BUT EXISTS ON A DASD VOLUME, THE EXISTING DATA SET IS SCRATCHED, AND CT1 WILL ATTEMPT ALLOCATION AGAIN.

UP TO FOUR ALLOCATION ATTEMPTS ARE MADE. THIS HANDLES MOST CASES OF TEMPORARY DATA SETS NOT BEING CLEANED UP AFTER A SYSTEM CRASH.

WHEN CT1 IS INVOKED TO ALLOCATE AND OPEN 'SPFLOG ', 'SPFLIST ', OR 'SPFEDITX', IT MUST BE EXECUTING UNDER THE SPF MAIN TASK. WHEN IT IS INVOKED TO ALLOCATE AND OPEN 'SPFCNTLX' OR 'SPFLISTX' IT MUST BE EXECUTING UNDER THE PROCESSOR TASK. THIS IS BECAUSE THE LOG, LIST, AND EDIT BACKUP DATA SETS BELONG TO THE SPF MAIN TASK AND MUST BE AROUND EVEN WHILE PROCESSOR TASKS ARE ATTACHED AND DETACHED TO SUPPORT SPLIT SCREEN. THE TEMPCNTL AND TEMPLIST DATA SETS BELONG TO THE PROCESSOR AND WILL BE CLEANED UP BY TASK TERMINATION IF THE PROCESSOR TASK TERMINATES ABNORMALLY.

SINCE CTA (WHICH INVOKES CT1) MAY BE CALLED UNDER EITHER TASK, A DUAL SCHEME FOR INVOKING CT1 IS PROVIDED. IF CTA AND CT1 ARE UNDER THE SAME TASK, A DIRECT CALL IS PERFORMED. IF CTA IS EXECUTING UNDER THE PROCESSOR TASK, AND CT1 MUST BE EXECUTED UNDER THE SPF MAIN TASK, AN INTERNAL SPF PROCEDURE NAMED 'SYSCCI' (COMMON CONTROLLER INTERFACE) IS INVOKED. IT IN TURN EXECUTES POST/WAIT LOGIC SO THAT THE SPF MAIN TASK WILL ACTUALLY CALL THE SUBROUTINE. WHEN THE SUBROUTINE RETURNS, THE SPF MAIN TASK USES POST/WAIT LOGIC TO RETURN TO SYSCCI WITH THE REG 15 RETURN CODE.

THE SAME LOGIC THAT APPLIES TO CTA INVOKING CT1 ALSO APPLIES TO CTF (CLOSE/FREE DATA SETS) INVOKING CT2, SINCE THE DCBS MUST BE OPENED AND CLOSED UNDER THE SAME TASK.

THE FOLLOWING FILE NAMES CAN BE PREALLOCATED. IF CT1 RECOGNIZES THAT A FILE HAS BEEN PREALLOCATED BY USING READJFCB, IT SKIPS ALLOCATION AND GOES IMMEDIATELY TO OPEN THE DATA SET. NO ERROR CHECKING OF PREALLOCATED FILES IS DONE.

SPFLOG	- LOG DATA SET
SPFLIST	- LIST DATA SET
SPFCNTL1	- CONTROL CARD DATA SET, LOGICAL SCREEN 1
SPFCNTL2	- CONTROL CARD DATA SET, LOGICAL SCREEN 2
SPFLIST1	- LISTING DATA SET, LOGICAL SCREEN 1
SPFLIST2	- LISTING DATA SET, LOGICAL SCREEN 2
SPFEDITA	- FIRST EDIT RECOVERY DATA SET
SPFEDITB	- SECOND EDIT RECOVERY DATA SET

PURPOSE:

CT2 IS USED TO CLOSE AND FREE THE DATA SETS THAT WERE ALLOCATED AND OPENED BY CTA/CT1.

INVOKED WITH:

CALL FROM CTF (EITHER DIRECLY OR VIA "SYSCCI" - COMMON CONTROLLER INTERFACE, AN SPF INTERNAL PLS PROCEDURE)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	DA18	*	INPUT	DA18 BLOCK

WHERE

DA18 - IS THE TSO DAIR DA18 BLOCK USED TO FREE A FILE.

RETURN CODE:

0 - SUCCESSFUL CLOSE AND FREE OF THE DATA SET.

>0 - DAIR RETURN CODE FROM UNSUCCESSFUL FREE.

NOTES:

IF CT2 IS PASSED THE SPFLOG OR SPFLIST TFD, CDP IS CALLED TO COMPLETE THE OUTPUT TO THE DATA SET.

IF THE TFDPREAL FLAG BIT IS ON INDICATING THAT THE CORRESPONDING FILE WAS PREALLOCATED, CT2 DOES NOT FREE THE FILE.

CT2 MAY BE CALLED DIRECTLY BY CTF, OR INDIRECTLY VIA "SYSCCI". THIS DUAL INTERFACE IS REQUIRED FOR THE SAME REASONS THAT CTA CALLS CT1 WITH A DUAL INTERFACE. SEE THE DESCRIPTION OF CT1 FOR AN EXPLANATION.

PURPOSE:

CUPARMS DOES AN UPDATE IN PLACE OF A MEMBER OF THE SPF PARMS DATA SET. THE MEMBER BEING UPDATED IS THE SAME MEMBER THAT WAS READ IN OR CREATED DURING SPF INITIALIZATION BY CIPARMS. THE NAME OF THE MEMBER IS THE SAME AS THE USER'S TSO LOGON ID. THE DATA WHICH IS WRITTEN COMES FROM THE SPF TABLE OF KEYWORD VALUES (TKV).

INVOKED WITH:

CALL TO CUPARMS

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - PARMS MEMBER WAS UPDATED AND NO ERRORS DETECTED.

4 - FIND ERROR ATTEMPTING TO FIND THE PARMS MEMBER.

8 - I/O ERROR ATTEMPTING TO READ THE PARMS MEMBER.

12 - I/O ERROR ATTEMPTING TO WRITE THE PARMS MEMBER.

NOTES:

BEFORE WRITING OUT THE TKV TO THE PARMS DATA SET, A COPY IS MADE AND SET UP TO BE ADDRESSED BY TSITKVP. THEN THE SUBROUTINE SOP (SPF OUTPUT PARMS) IS CALLED. SOP WAS DESIGNED AS A USER EXIT ROUTINE AND CAN BE MODIFIED IF A INSTALLATION REQUIRES THAT DATA BE VERIFIED OR MODIFIED BEFORE BEING WRITTEN TO THE PARMS DATA SET. SOP ALSO COPIES SOME DATA FROM THE TSV TO THE TKV. FOR MORE INFORMATION SEE THE DESCRIPTION OF SOP.

CVM - COMMON VERIFY MEMBER NAME ROUTINE

PURPOSE:

CVM VERIFIES THAT A MEMBER NAME THAT IS TO BE GENERATED BY SPF IS A VALID MEMBER NAME. IT MUST BEGIN WITH AN ALPHA OR @,#,$ CHARACTER AND THE REMAINING 7 CHARACTERS MUST BE ALPHAMERIC.

INVOKED WITH:

CALL TO CVM

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MEMBER	CHAR(8)	INPUT	NAME OF MEMBER TO BE VERIFIED

RETURN CODES:

0 - MEMBER NAME IS VALID.

4 - MEMBER NAME IS INVALID.

NOTES:

SUPERZAP CAN BE USED TO MODIFY THIS SUBROUTINE SO THAT SPF WILL ALLOW ANY INVALID MEMBER NAME TO BE USED IN CREATING NEW MEMBERS. THE CODE CAN BE MODIFIED AND REASSEMBLED TO ALLOW SOME, BUT NOT ALL TYPES OF INVALID MEMBER NAMES TO BE GENERATED BY SPF. FOR EXAMPLE, THE SUBROUTINE COULD BE MODIFIED TO ALLOW MEMBER NAMES STARTING WITH A NUMBER, AND REJECT NAMES THAT INCLUDE SPECIAL CHARACTERS.

CVSDE - COMMON VERIFY SPF DIRECTORY ENTRY ROUTINE CVSDE

PURPOSE:

CVSDE DETERMINES IF A PDS DIRECTORY ENTRY PASSED TO IT IS IN SPF DIRECTORY ENTRY FORMAT OR NOT.

INVOKED WITH:

CALL TO CVSDE

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	BLDLENT	*	INPUT	BLDL ENTRY

WHERE

BLDLENT - IS AN ENTRY FROM THE OS BLDL CONTROL BLOCK.

RETURN CODES:

0 - SPF ENTRY.

4 - NOT SPF ENTRY, NO USER DATA.

8 - NOT SPF ENTRY, USER DATA NOT IN SPF FORMAT.

NOTES:

SPF DIRECTORY ENTRIES ARE 15 HALFWORDS IN LENGTH. THE LAST THREE BYTES MUST BE BLANKS, AND CREATION DATE AND LAST MODIFIED DATE MUST BE PACKED DECIMAL FIELDS. SEE DATA AREAS SECTION ON SPF DIRECTORY ENTRY FORMAT <SDE>.

PURPOSE:

EBA DETERMINES IF RECOVERY IS PENDING AND PERFORMS THE RECOVERY PROCESS INITIALIZATION. IT ALLOCATES AND OPENS THE RECOVERY DATA SET AND ESTABLISHES THE BACKUP/RECOVERY CONTROL TABLE (EBT). IT ALSO VALIDATES THE OUTPUT DATA SET CHARACTERISTICS IF RECOVERY IS INDICATED.

INVOKED WITH:

CALL TO EBA

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - NO RECOVERY AVAILABLE IF RECOVERY NOT IN PROGRESS (1ST CALL FROM EMP), OR OUTPUT DATA SET ACCEPTABLE IF RECOVERY ALREADY IN PROGRESS (2ND CALL FROM EMP).

4 - RECOVERY AVAILABLE (1ST CALL FROM EMP).

8 - OUTPUT DATA SET NOT ACCEPTABLE (2ND CALL FROM EMP).

NOTES:

EBA IS ALWAYS CALLED AT THE BEGINNING OF EDIT PROCESSING BY EMP. THE RECOVERY PROCESS IS COMPLETED BY A CALL TO EBR.

EBE - EDIT BACKUP END ROUTINE EBE

PURPOSE:

EBE FREES CONTROL OF THE RECOVERY DATA SET AND ELIMINATES THE BACKUP/RECOVERY CONTROL TABLE (EBT).

INVOKED WITH:

CALL TO EBE

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
    REG 3 ->  <EDT>       IN/OUT     EDIT TABLE

    REG 9 ->  <TLD>       INPUT      LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

EBE IS ALWAYS CALLED AT THE TERMINATION OF EDIT PROCESSING. IT IS ALSO CALLED WHEN CERTAIN ERRORS ARE ENCOUNTERED DURING THE BACKUP OR RECOVERY PROCESS.

PURPOSE:

EBI PERFORMS THE BACKUP PROCESS INITIALIZATION. IT ALLOCATES AND OPENS A RECOVERY DATA SET AND ESTABLISHES THE BACKUP/RECOVERY CONTROL TABLE (EBT).

INVOKED WITH:

CALL TO EBI

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

EBI IS CALLED FROM EBS ONLY WHEN "RECOVERY ON" IS INITIALLY REQUESTED. THE ACTUAL BACKUP PROCESS IS STARTED AND CONTINUED BY CALLS TO EBS.

EBR - EDIT RECOVERY READ ROUTINE

PURPOSE:

EBR READS THE RECOVERY DATA SET.

INVOKED WITH:

CALL TO EBR

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKABE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	IN/OUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

EBR IS CALLED TO READ THE RECOVERY DATA SET. FOLLOWING THE RECOVERY PROCESS THE BACKUP PROCESS IS IN EFFECT AND CAN BE CONTINUED BY CALLS TO EBS.

PURPOSE:

EBS STORES DATA IN THE RECOVERY DATA SET.

INVOKED WITH:

CALL TO EBS

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	IN/OUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

EBS IS CALLED EVERY TIME A RECORD IS ADDED, DELETED, OR MODIFIED IN THE EDIT RECORD CHAIN. THE FIRST TIME EBS IS CALLED FOR A DATA SET OR MEMBER THE BACKUP PROCESS IS STARTED. EBI IS CALLED BY EBS IF BACKUP/RECOVERY HAD NOT BEEN PREVIOUSLY INITIALIZED THE BACKUP PROCESS IS STOPPED BY A CALL TO EBX WHEN THE DATA SET OR MEMBER IS SAVED OR CANCELLED OR WHEN THE USER REQUESTS "RECOVERY OFF".

PURPOSE:

EBX RESETS THE BACKUP/RECOVERY CONTROL.

INVOKED WITH:

CALL TO EBX

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

EBX STOPS THE BACKUP PROCESS WHEN A DATA SET OR MEMBER IS SAVED OR CANCELLED OR WHEN THE USER REQUESTS "RECOVERY OFF". THE BACKUP PROCESS CAN BE RESTARTED BY A CALL TO EBS.

ECD - EDIT COMMAND DEFINITIONS TABLE

PURPOSE:

ECD IS THE EDIT COMMAND DEFINITION TABLE. IT CONTAINS ONE ENTRY FOR EACH EDIT PRIMARY COMMAND. THE COMMAND DEFINITION TABLE IS INPUT TO THE COMMON COMMAND PARSE (CCP) ROUTINE, AND IS USED FOR ERROR CHECKING AND FOR ORDERING PARAMETERS.

REFERENCED VIA:

THE ADDRESS OF THE ECD IS IN THE TSC. IT IS SYMBOLICALLY REFERENCED BY THE NAME "ECD" DEFINED IN SEGMENT "ECSDCLS" (EDIT COMMON SUBS).

NOTES:

- THE ECD TABLE IS TERMINATED WITH AN X'FF' CHARACTER.
- EACH COMMAND DEFINITION WITHIN THE TABLE IS TERMINATED WITH AN X'FE' CHARACTER.
- EACH PARAMETER DEFINITION WITHIN THE COMMAND DEFINITIONS IS TERMINATED WITH AN X'FD' CHARACTER.
- IN ADDITION, THE LENGTH OF EACH COMMAND DEFINITION ENTRY IS PART OF THE ENTRY, AND THE NUMBER OF PARAMETERS DEFINED IS ALSO PART OF THE ENTRY.

PURPOSE:

ECR PERFORM PROCESSING FOR THE "CREATE" AND "REPLACE" COMMANDS.

INVOKED WITH:

CALL TO ECR

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
    REG 3 ->  <EDT>       IN/OUT    EDIT TABLE
    REG 9 ->  <TLD>       INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE

EDD - EDIT DRIVER ROUTINE EDD

PURPOSE:

EDD IS INVOKED BY PMD WHEN OPTION 2 IS SELECTED FROM THE PRIMARY OPTION MENU. IT CALLS SUBROUTINES TO DISPLAY THE EDIT DATA SET, ALLOCATE APPROPRIATE DATA SETS, DISPLAY MEMBER LISTS, AND PERFORM EDITING.

INVOKED WITH:

LINK TO SPFEDIT

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MENU	CHAR(8)	INPUT	MENU NAME

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EDI - EDIT DATA INPUT ROUTINE EDI

PURPOSE:

EDI PERFORMS DATA INPUT FOR EDIT. IT IS CALLED AT INITIALIZATION TO READ THE INITIAL DATA, AND IS ALSO CALLED IF A "MOVE" OR "COPY" PRIMARY COMMAND IS PERFORMED. IN ADDITION TO READING DATA, EDI LIMITS THE RECORDS ACTUALLY PUT ON THE EDR CHAIN (FOR A "COPY" WITH START/END RECORDS SPECIFIED). IT ALSO CHECK TO SEE IF SEQUENCE NUMBERS ARE PRESENT IN THE DATA AND IF LOWER CASE AND/OR INVALID CHARACTERS ARE PRESENT.

INVOKED WITH:

CALL TO EDI

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE
3.	TFD	<TFD>	INPUT	INPUT FILE DEFINITION TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EDO - EDIT DATA OUTPUT ROUTINE

PURPOSE:

EDO PERFORMS THE WRITING OF OUTPUT FOR EDIT. IT IS CALLED FOR A STANDARD "END" (IF A CHANGE IN THE DATA HAS OCCURRED), IF A "SAVE" IS DONE, OR IF A "CREATE" OR "REPLACE" PRIMARY COMMAND IS ISSUED. EDO PERFORMS AUTO-RENUMBERING IF BOTH NUMBER MODE AND AUTONUM MODE ARE ON.

INVOKED WITH:

CALL TO EDO

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE
3.	TFD	<TFD>	INPUT	INPUT FILE DEFINITION TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EFC - EDIT FIND-CHANGE ROUTINE

PURPOSE:

EFC IS CALLED TO ANALYZE AND/OR PERFORM THE FIND/CHANGE COMMANDS.

INVOKED WITH:

CALL TO EFC

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
    REG 3 ->  <EDT>        IN/OUT    EDIT TABLE
    REG 9 ->  <TLD>        INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EFR - EDIT FORMAT DISPLAY ROUTINE EFR

PURPOSE:

EFR IS CALLED TO FORMAT THE DISPLAY SCREEN.

INVOKED WITH:

CALL TO EFR

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EFT - EDIT FLOW TEXT ROUTINE

PURPOSE:

EFT FLOWS TEXT TO SUPPORT BOTH THE "TF" AND "TE" COMMANDS.

INVOKED WITH:

CALL TO EFT

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
    REG 3 ->  <EDT>       IN/OUT    EDIT TABLE
    REG 9 ->  <TLD>       INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

EGN IS CALLED TO NUMBER OR RENUMBER ALL OF THE STANDARD RECORDS ON THE EDR CHAIN.

INVOKED WITH:

CALL TO EGN

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

EGN USES THE EDTNUMBR FIELD TO DETERMINE WHETHER TO NUMBER, RENUMBER OR UNNUMBER THE DATA.

```
  EDTNUMBR = 'R' MEANS RENUMBER STARTING AT THE EDTDELTA NUMBER AND
             INCREMENTING BY THE SAME DELTA NUMBER.
  EDTNUMBR = 'U' MEANS UNNUMBER, BLANKING OUT EXISTING SEQUENCE
             NUMBERS.
  EDTNUMBR = 'I' MEANS INITIAL NUMBER OPERATION.  MOD FLAGS ARE
             RESET IF APPROPRIATE (I.E. IF MOD FLAGS ARE BEING USED
             AND THE CURRENT MOD FLAG IS TOO LARGE) AND A STANDARD
             NUMBER OPERATION IS PERFORMED.
  EDTNUMBR = 'N' (OR ANY OTHER CODE) MEANS STANDARD NUMBER OPERATION
             CHANGING THE SEQUENCE NUMBER ONLY IF REQUIRED TO FORCE
             A VALID SEQUENCE NUMBER (I.E. IF THE RECORD ALREADY
             CONTAINS A VALID SEQUENCE NUMBER IT IS NOT CHANGED).
```

EITHER NUMBERING OR RENUMBERING RESULTS IN EVERY RECORD CONTAINING A VALID ASCENDING SEQUENCE NUMBER.

PURPOSE:

EGR IS CALLED TO RESET THE EDR CHAIN. THIS CONSISTS OF DELETING ANY SPECIAL RECORDS THAT ARE ON THE CHAIN (BOUNDS,COLS,MASK, AND TABS), MARKING ALL RECORDS NOT-EXCLUDED, AND RESETTING ALL SPECIAL FLAGS (=ERR=>, =CHG=>, ETC).

INVOKED WITH:

CALL TO EGR

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EMC - EDIT MOVE-COPY ROUTINE EMC

PURPOSE:

EMC PERFORMS PROCESSING FOR THE "MOVE" AND "COPY" COMMANDS.

INVOKED WITH:

CALL TO EMC

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

EML HANDLES -CAUTION- AND -WARNING- MESSAGES FOR EDIT. IT READS TWO MESSAGES FROM THE MESSAGE DATA SET AND PUTS THEM ON THE EDR CHAIN, AFTER THE FIRST LINE ON THE DISPLAY.

INVOKED WITH:

CALL TO EML

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MSG	CHAR(4)	INPUT	LONG MESSAGE ID
3.	CODE	FIXED(31)	INPUT	MESSAGE TYPE CODE

WHERE

MSG - IS THE IDENTIFIER OF THE FIRST MESSAGE THAT IS TO BE PUT ON THE EDR CHAIN. THE SECOND MESSAGE IS ONE DIGIT HIGHER IN SEQUENCE (I.E. INPUT OF E702 RESULTS IN MESSAGE E702 AND E703 BEING PUT ON THE EDR CHAIN.)

CODE - IS A CODE THAT IS STORED IN THE EDRTYPE FIELD OF THE EDIT RECORDS THAT CONTAIN THE MESSAGES. THE CODES ARE TAKEN FROM EDRDCLS AND ARE USED TO DELETE THE MESSAGE LINE IF THE CONDITION INDICATED HAS CHANGED.

RETURN CODE:

0 - ALWAYS.

NOTES:

AN EXAMPLE OF A -CAUTION- MESSAGE THAT IS HANDLED BY EML IS THE MESSAGE INIDICATING THAT "STANDARD NUMBER MODE HAS BEEN TURNED OFF". THE CODE WILL INDICATE THAT NUMBER MODE IS OFF. IF A NUMBER COMMAND IS ISSUED, NUMBER MODE WILL BE TURNED ON, AND THE MESSAGE WILL BE ERRONEOUS. THE CODE WHICH IS STORED IN THE EDR IS USED AS A TRIGGER TO CAUSE THE MESSAGE EDR TO BE DELETED.

EMP - EDIT MENU PROCESSOR ROUTINE

PURPOSE:

EMP DISPLAYS THE EDIT DATA SET MENU, AND DISPLAYS ERROR MESSAGES. IT CALLS COMMON SUBROUTINES TO ALLOCATE AND OPEN THE APPROPRIATE DATA SETS.

INVOKED WITH:

CALL TO EMP

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE
3.	MENU	CHAR(8)	INPUT	MENU NAME

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EPC - EDIT PROCESS (LINE) COMMANDS ROUTINE EPC

PURPOSE:

EPC PROCESSES EDIT LINE COMMANDS.

INVOKED WITH:

CALL TO EPC (FROM EPR)

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
    REG 3 ->  <EDT>        IN/OUT    EDIT TABLE

    REG 9 ->  <TLD>        INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

EPC IS INVOKED BY EPR WHEN ANY EDIT LINE COMMANDS ARE TO BE PROCESSED. IT SCANS THE EDIT RECORD CHAIN, PERFORMS VALIDITY CHECKING AND, IF NO ERRORS ARE DETECTED, PERFORMS LINE COMMAND PROCESSING.

PURPOSE:

EPD IS CALLED TO PROCESS DATA INPUT FROM THE DISPLAY SCREEN.

INVOKED WITH:

CALL TO EPD

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

- EDIT PROCESS (PRIMARY) FINAL ROUTINE

PURPOSE:

EPF PERFORMS ANY FINAL PROCESSING THAT IS REQUIRED TO COMPLETE EDIT PRIMARY COMMANDS. THE PROCESSING IS DONE AFTER EDIT LINE COMMANDS HAVE BEEN PROCESSED.

INVOKED WITH:

CALL TO EPF

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

EPF IS INVOKED BY EPR WHEN FINAL PROCESSING OF A PRIMARY EDIT COMMAND IS REQUIRED (AFTER LINE COMMAND PROCESSING HAS BEEN PERFORMED). IT PERFORMS VALIDITY CHECKING AND, IF NO ERRORS ARE DETECTED, COMPLETES THE PRIMARY COMMAND PROCESSING.

EPI - EDIT PROCESS (PRIMARY) INITIAL ROUTINE EPI

PURPOSE:

EPI PERFORMS INITIAL PROCESSING OF EDIT PRIMARY COMMANDS.

INVOKED WITH:

CALL TO EPI (FROM EPR)

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

EPI IS INVOKED BY EPR WHEN INITIAL PROCESSING OF A PRIMARY EDIT COMMANDS IS REQUIRED (BEFORE LINE COMMAND PROCESSING HAS BEEN PERFORMED). IT PERFORMS VALIDITY CHECKING AND, IF NO ERRORS ARE DETECTED, PERFORMS INITIAL (IN SOME CASES COMPLETE) PRIMARY COMMAND PROCESSING.

PURPOSE:

EPO PROCESSES PARTITIONED DATA SET. IT CALLS CML TO DO MEMBER LIST PROCESSING IF APPROPRIATE.

INVOKED WITH:

CALL TO EPO

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

EPP IS INVOKED TO RETRIEVE AND/OR STORE EDIT PROFILE OPTIONS.

INVOKED WITH:

CALL TO EPP

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE
3.	GETPROF	CHAR(8)	INPUT	GET PROFILE NAME
4.	PUTPROF	CHAR(8)	INPUT	PUT PROFILE NAME

WHERE

GETPROF - IDENTIFIES THE PROFILE TO BE RETRIEVED. IF THE NAME IS BLANK, NO RETRIEVAL IS PERFORMED.

PUTPROF - IDENTIFIES THE PROFILE TO BE STORED. IF THE NAME IS BLANK, NO STORING IS PERFORMED.

RETURN CODE:

0 - ALWAYS.

NOTES:

EPP RETRIEVES PROFILE INFORMATION INCLUDING MASK AND TABS LINES FROM THE TKV BY CALLING CKVGET. IT MOVES THE INFORMATION INTO THE EDIT TABLE. STORING PROFILE INFORMATION IS THE REVERSE OPERATION.

PURPOSE:

EPR GETS CONTROL WITH THE INPUT DATA SET ALLOCATED AND OPEN. IT READS THE DATA IN (EDI), FORMATS THE SCREEN (EFR) AND PERFORMS THE LOGICAL EDITING OF THE DATA (EPI,EPD,EPC,EPF).

INVOKED WITH:

CALL TO EPR

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

EPS - EDIT PHYSICAL SEQUENTIAL ROUTINE

PURPOSE:

EPS PROCESSES PHYSICAL SEQUENTIAL DATA SETS.

INVOKED WITH:

CALL TO EPS

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

- EDIT "OTHER" DATA SET ORGANIZATION ROUTINE

PURPOSE:

EPX PROCESSES DATA SETS THAT ARE NEITHER PARTITIONED NOR SEQUENTIAL. IT CAUSES AN ERROR MESSAGE TO BE DISPLAYED, BUT COULD REPLACED TO PROCESS OTHER TYPES OF DATA SETS.

INVOKED WITH:

CALL TO EPX

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

ERA - EDIT RECORD ALLOCATE ROUTINE

PURPOSE:

ERA ALLOCATES ONE EDIT RECORD AND STORES ITS ADDRESS IN THE EDIT TABLE (EDRP(EDFREE)). ALLOCATION CONSISTS OF UNCHAINING ONE RECORD FROM THE FREE CHAIN, OR IF NO RECORDS EXIST ON THE FREE CHAIN, OF GETMAINING A DATA BLOCK AND BREAKING IT UP INTO FREE EDIT RECORDS WHICH ARE PUT ON THE FREE CHAIN. THE HEADER OF THE NEW EDIT RECORD IS ZEROED AND ITS DATA AREA IS BLANKED.

INVOKED WITH:

CALL TO ERA

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

WHEN STORAGE IS REQUIRED, AN UNCONDITIONAL GETMAIN IS PERFORMED. THIS WILL RESULT IN AN 80A ABEND IF INSUFFICIENT STORAGE EXISTS TO SATISFY THE REQUEST.

THE SIZE OF THE GETMAIN IS 4K. IT IS TAKEN FROM A FIELD IN THE EDIT TABLE (EDGMSIZE) WHICH IS INITIALIZED BY ED.

ERC - EDIT RECORD CHANGE ROUTINE ERC

PURPOSE:

ERC MARKS A RECORD AS CHANGED BY SETTING THE CHANGED BIT IN THE EDR AND IT ALSO SETS THE OVERALL CHANGED BIT IN THE EDT. THE MOD FLAG PART OF THE SEQUENCE NUMBER IS SET (IF APPROPRIATE) AND THE BACKUP ROUTINE EBS IS CALLED IF RECOVERY IS ON.

INVOKED WITH:

CALL TO ERC

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

ALL ROUTINES THAT CHANGE THE DATA WITHIN AN EDR MUST CALL ERC TO INDICATE THAT THE CHANGE WAS MADE.

THE TYPE OF CHANGE IS INDICATED BY SETTING THE APPROPRIATE BIT IN EDTEDRBS WHICH IS OR'ED TO THE EDR THAT IS BEING CHANGED BY ERC.

ERD - EDIT RECORD DELETE ROUTINE ERD

PURPOSE:

ERD DELETES ONE EDIT RECORD FROM THE EDR CHAIN.
ON INPUT:
  EDRP(EDCURR) - POINTS TO THE EDR TO BE DELETED FROM THE EDR CHAIN.
ON OUTPUT:
  EDRP(EDCURR) - POINTS TO THE EDR PRECEDING THE DELETED EDR.
  EDRP(EDFREE) - POINTS TO THE DELETED EDR.

INVOKED WITH:

CALL TO ERD

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

ERD DECREMENTS THE TOTAL EDR COUNT.

POINTERS IN THE EDIT TABLE THAT MUST POINT TO EDR'S THAT ARE ON THE EDR CHAIN ARE EITHER ZEROED OUT, OR ARE RESET TO POINT TO THE PRECEDING OR FOLLOWING EDR ON THE CHAIN.

ERD INSURES THAT BLOCKS OF EXCLUDED (X'ED) RECORDS ARE CONSISTENT AFTER THE DELETION HAS OCCURRED.

ERD USES EDTSA2 (EDIT TABLE SAVE AREA 2) AS A SAVE AREA SO THAT IT CAN CALL A LOWER LEVEL PROGRAM. THIS AREA IS USED INSTEAD OF A DYNAMICALLY GETMAINED AREA TO IMPROVE PERFORMANCE SINCE ERD IS INVOKED FREQUENTLY..

PURPOSE:

ERF TAKE ONE RECORD (EDR) AND PUTS IT ON THE FREE CHAIN SO THAT IT WILL BE AVAILABLE FOR REUSE.
ON INPUT:
  EDRP(EDFREE) - POINTS TO THE EDR TO BE FREED.

INVOKED WITH:

CALL TO ERF

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

```
   ERI INSERTS ONE EDIT RECORD ONTO THE EDR CHAIN.  IN ADDITION TO
   PERFORMING FORWARD AND BACKWARD CHAINING, ERI COMPUTES AN INTERNAL
   EDR SEQUENCE NUMBER FOR THE EDR HEADER.  IT ALSO CALLS ERN (EDIT
   RECORD NUMBER) IF SEQUENCE NUMBERS ARE TO BE STORED IN THE DATA.
   ERI CALLS EBS (EDIT BACKUP) IF RECOVERY MODE IS ON, AND INCREMENTS
   THE TOTAL EDR COUNT AND THE TOTAL STANDARD EDR COUNT.
   ON INPUT:
     EDRP(EDAFTER) - POINTS TO THE EDR AFTER WHICH THE INSERTION
                     IS TO BE MADE.
     EDRP(EDFREE)  - POINTS TO THE EDR TO BE INSERTED.
```

INVOKED WITH:

CALL TO ERI

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
     REG 3 ->  <EDT>       IN/OUT    EDIT TABLE

     REG 9 ->  <TLD>       INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

ERN - EDIT RECORD NUMBER ROUTINE ERN

PURPOSE:

ERN PUTS A SEQUENCE NUMBER INTO A RECORD THAT IS ON THE EDR CHAIN.
ON INPUT:
EDRP(EDFREE) - POINTS TO THE EDR WHICH IS TO BE SEQUENCE NUMBERED.

INVOKED WITH:

CALL TO ERN

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

ERO - EDIT RECORD ORIGINAL ROUTINE ERO

PURPOSE:

ERO MAKES A COPY OF AN ORIGINAL RECORD (ONE THAT WAS INITIALLY READ IN) AND QUEUES IT TO AN ORIGINAL RECORD CHAIN.
ON INPUT:
  EDRP(EDFREE) - POINTS TO THE ORIGINAL EDR TO BE COPIED.

INVOKED WITH:

CALL TO ERO

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

ERO IS CALLED ONLY FROM ERD.

KEEPING A COPY OF ORIGINAL EDR'S ALLOWS A DETERMINALTION AT SAVE TIME OF WHAT WAS DELETED FROM THE ORIGINAL DATA. THIS CAPABILITY IS NOT CURRENTLY UTILIZED.

PURPOSE:

ERR TAKE ONE RECORD (EDR) AND PERFORMS THE CLEANUP NECESSARY TO REMOVE A COMMAND THAT WAS ASSOCIATED WITH IT. THIS INCLUDES CLEARING THE COMMAND BIT, FREEING THE EDR EXTENSION IF APPROPRIATE, AND HANDLING EXCLUDED LINE CONSIDERATIONS. ERR DECREMENTS THE COMMAND COUNT AND INSURES THAT THE CHAIN OF EDR'S WITH ASSOCIATED COMMANDS IS UPDATED, AS ARE THE POINTER TO THE FIRST AND LAST COMMAND EDR.
ON INPUT:
  EDRP(ERRP) - POINTS TO THE EDR TO BE RESET.

INVOKED WITH:

CALL TO ERR

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
    REG 3 ->  <EDT>       IN/OUT    EDIT TABLE

    REG 9 ->  <TLD>       INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

THE COMMAND COUNT IS DECREMENTED, POINTERS TO THE FIRST AND LAST EDR CONTAINING COMMANDS ARE UPDATED IF REQUIRED, AND THE CHAIN OF ERS'S CONTAINING COMMANDS IS ALSO UPDATED IF REQUIRED.

ERS - EDIT RECORD SHOW ROUTINE

PURPOSE:

```
   ERS MARKS A RECORD AS NOT EXCLUDED.
   ON INPUT:
     EDRP(EDXCURR) - POINTS TO THE EDR TO BE 'UN'-EXCLUDED.
```

INVOKED WITH:

CALL TO ERS

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
       REG 3 ->  <EDT>       IN/OUT    EDIT TABLE

       REG 9 ->  <TLD>       INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

ERS INSURES THAT BLOCKS OF EXCLUDED (X'ED) RECORDS ARE CONSISTENT AFTER THE EDR IS RESET SO IT IS NOT EXCLUDED.

ERX - EDIT RECORD EXCLUDE ROUTINE

PURPOSE:

```
   ERX MARKS A RECORD AS EXCLUDED.
   ON INPUT:
     EDRP(EDXCURR) - POINTS TO THE EDR TO BE EXCLUDED.
```

INVOKED WITH:

CALL TO ERX

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

```
        REG 3 ->  <EDT>        IN/OUT    EDIT TABLE

        REG 9 ->  <TLD>        INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

ERX INSURES THAT BLOCKS OF EXCLUDED (X'ED) RECORDS ARE CONSISTENT AFTER THE EDR IS RESET SO IT IS EXCLUDED.

EST - EDIT SPLIT TEXT ROUTINE

PURPOSE:

EST SPLITS TEXT INTO TWO LINES AND INSERTS ONE OR MORE LINES BETWEEN THE PARTS OF THE TEXT. IT IS INVOKED WHEN THE "TS" LINE COMMAND IS PROCESSED.

INVOKED WITH:

CALL TO EST

CALLING SEQUENCE PARAMETERS:

NONE - SPECIAL SPF EDIT LINKAGE CONVENTIONS ARE USED:

REG 3 ->	<EDT>	IN/OUT	EDIT TABLE
REG 9 ->	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE

ETC - EDIT TABLE CLEANUP ROUTINE

PURPOSE:

ETC IS INVOKED TO FREE THE EDIT TABLE.

INVOKED WITH:

CALL TO ETC

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDT	<EDT>	IN/OUT	EDIT TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

ETL - EDIT LINE COMMANDS TABLE ETL

PURPOSE:

ETL IS THE EDIT LINE COMMAND DEFINITION TABLE. IT CONTAINS ONE ENTRY FOR EACH EDIT LINE COMMAND. THE LINE COMMAND DEFINITION TABLE IS REFERENCED BY EPC IN VALIDATING AND PROCESSING LINE COMMANDS.

REFERENCED VIA:

THE ADDRESS OF THE ETL IS IN THE TSC. IT IS SYMBOLICALLY REFERENCED BY THE NAME "ETL" DEFINED IN SEGMENT "ECSDCLS" (EDIT COMMON SUBS).

NOTES:

THE ETL TABLE IS TERMINATED WITH AN X'FF' CHARACTER. EACH COMMAND DEFINITION ENTRY IS FIXED LENGTH. THE ENTRIES INCLUDE THE COMMAND NAME, THE COMMAND TYPE (MOVE, COPY, AFTER, ETC), THE PASS (1-3) ON WHICH THE COMMAND SHOULD BE EXECUTED, THE DEFAULT SUFFIX, A CURSOR POSITION CODE, INVALID LINE MASK AND THE INDEX IN THE TSC OF THE ROUTINES THAT IS TO BE EXECUTED WHEN THE COMMAND IS ENTERED.

ETS - EDIT TABLE SETUP ROUTINE ETS

PURPOSE:

ETS IS INVOKED TO GETMAIN AND INITIALIZE THE EDIT TABLE.

INVOKED WITH:

CALL TO ETS

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	EDTPTR	PTR(31)	OUTPUT	PTR TO THE GETMAINED EDT

RETURN CODE:

0 - ALWAYS.

NOTES:

THE PTR WHICH IS RETURNED BY ETS SHOULD BE LOADED INTO REG 3 BEFORE CALLING EDIT SUBROUTINES.

PURPOSE:

FOR PROCESSES FOREGROUND SUBOPTIONS. A FOREGROUND SUBOPTION MENU IS DISPLAYED AND THE APPROPRIATE SPFPROCS DATASET MEMBER IS READ AND PROCESSED. A TSO COMMAND MAY BE GENERATED FROM THE USER INPUT AND A COMMAND PROTOTYPE IN THE SPFPROCS DATASET. FOR CALLS COMMON SUBROUTINE CAT TO EXECUTE A TSO COMMAND OR CLIST, IF SPECIFIED IN THE SPF PROC.

INVOKED WITH:

LINK TO SPFFOR (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MENUID	CHAR(8)	INPUT	MENU NAME

WHERE

MENUID - INDICATES THE FOREGROUND SUBOPTION MENU TO BE DISPLAYED. THIS PARAMETER ALSO IS THE NAME OF THE SPFPROCS DATASET MEMBER TO BE USED TO GENERATE THE COMMAND.

RETURN CODE:

0 - ALWAYS.

NOTES:

THE FOREGROUND SELECTION MENU IS PROCESSED BY OBJECT MODULE UTIL, NOT THE FOR OBJECT MODULE. SEE THE DESCRIPTION OF OBJECT MODULE UTIL FOR FURTHER INFORMATION.

THE INSTALLATION CAN CREATE NEW SPF OPTIONS SIMILAR TO OPTION 4 OR ADDITIONAL OPTION 4 SUBOPTIONS WITHOUT MAKING ANY PROGRAM MODIFICATIONS. SEE INSTALLATION AND CUSTOMIZATION GUIDE FOR FURTHER INFORMATION.

PURPOSE:

JOB IS INVOKED WHEN OPTION 5 IS SELECTED FROM THE PRIMARY OPTION MENU. IT DISPLAYS THE BACKGROUND SELECTION MENU (JOBA) AND GENERATES JCL BY MERGING USER INPUT THAT IS ENTERED FROM A SECONDARY BACKGROUND MENU AND A JCL PROTOTYPE READ FROM THE SPFPROCS DATA SET. JCL IS WRITTEN TO A TEMPORARY DATA SET ('USERID.SPFTEMPX.CNTL') AND SUBMITTED TO THE JOB STREAM BY CALLING SUBROUTINE CSB.

INVOKED WITH:

LINK TO SPFJOB (FROM PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MENUID	CHAR(8)	INPUT	MENU NAME

WHERE

MENUID - INDICATES THE MENU TO BE DISPLAYED. IF THE FIRST CHARACTER OF THIS PARAMETER IS '*', THEN MENUID(2:7) CONTAINS A SELECTION MENU ID. IF THE FIRST CHARACTER IS NOT '*', THEN MENUID(1:8) CONTAINS THE MENU ID OF A SECONDARY MENU TO BE DISPLAYED, AND THE SELECTION MENU IS BYPASSED.

IF THE SELECTION MENU IS NOT BYPASSED, UPON RETURN FROM A SECONDARY MENU, A TERMINATION SELECTION MENU IS DISPLAYED. THE TERMINATION SELECTION MENU ID IS OBTAINED BY INCREMENTING THE LAST CHARACTER OF THE SELECTION MENU ID BY 1 DECIMAL. FOR EXAMPLE, IF MENUID CONTAINS '*JOBA', THEN THE SELECTION MENU ID IS 'JOBA' AND THE TERMINATION SELECTION MENU ID IS 'JOBB'.

RETURN CODES:

0 - ALWAYS.

NOTES:

THE INSTALLATION CAN CREATE NEW SPF OPTIONS SIMILAR TO OPTION 5 BY PASSING A MENU ID OTHER THAN '*JOBA' TO THIS MODULE. ADDITIONAL OPTION 5 SUBOPTIONS CAN ALSO BE ADDED WITHOUT MAKING ANY PROGRAM CHANGES. SEE THE INSTALLATION AND CUSTOMIZATION GUIDE FOR FURTHER INFORMATION.

MERR - COMMON MENU ERROR ROUTINE MERR

PURPOSE:

THE MENU ERROR ROUTINE IS CALLED TO DISPLAY ERROR MESSAGES AFTER A MENU HAS BEEN PROCESSED BY COMMON SUBROUTINE MHA. MERR ALSO CAUSES THE MENU TO BE RE-PROCESSED. THE CURSOR IS POSITIONED TO THE FIELD CONTAINING THE ERROR (AS INDICATED BY THE PARMID PARAMETER), AND THE AUDIBLE ALARM IS SOUNDED IF SPECIFIED IN THE MESSAGE.

INVOKED WITH:

CALL TO MERR

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	IN/OUT	LOGICAL DISPLAY TABLE
2.	MSGID	CHAR(4)	INPUT	ERROR MESSAGE ID
3.	PARMID	FIXED(31)	INPUT	PARAMETER CURSOR ID
4.	PARM1	*	INPUT	PARAMETER FOR MESSAGE
..	...			
..	PARMN	*	INPUT	PARAMETER FOR MESSAGE

WHERE

PARMID - IDENTIFIES THE PARAMETER IN THE PREVIOUS CALL TO THE MENU HANDLER WHICH CORRESPONDS TO THE FIELD IN ERROR. THE CURSOR WILL BE POSITIONED TO THIS PARAMETER. IF THIS PARAMETER CONTAINS A FULLWORD ZERO, THE CURSOR WILL BE PLACED ACCORDING TO THE SPECIFICATIONS ON THE MENU ACTION STATEMENTS.

PARM(S) - OPTIONAL PARAMETERS ARE USED AS SUBSTITUTIONAL VALUES IN THE SPECIFIED MESSAGE. ANY VALUES THAT ARE VALID IN CALLING CMSG MAY BE USED. A MAXIMUM OF 50 PARAMETERS MAY BE SPECIFIED.

RETURN CODE:

SAME AS MHA.

NOTES:

MERR IS LIKE AN ALTERNATE ENTRY TO THE MENU HANDLER (OBJECT MODULE MHA). MERR SETS TLDMERRC ON AND CALLS MHA.

A CALL TO MERR ACTUALLY RESULTS IN MERR CALLING MHA, MHA CALLING CERR, CERR CALLING CMSG AND CDISPL.

THE MERR PARAMETER LIST MUST BE TERMINATED BY A VLIST FLAG.

WHEN AN ERROR MESSAGE IS DISPLAYED VIA MERR, THE TERMINAL USER MAY PRESS PF1 TO OBTAIN A SECOND LEVEL MESSAGE ON LINE 3. IF PF1 IS AGAIN PRESSED, TUTORIAL MODE IS ENTERED AT AN APPROPRIATE PAGE. WHEN THE USER EXITS FROM TUTORIAL MODE, THE SCREEN IS RESTORED.

THE CALLING PROGRAM MUST BE PREPARED FOR ALTERATIONS TO ANY INPUT FIELD ON THE MENU. THE NEW INPUT FIELDS ARE RETURNED IN THE SAME MANNER AS IN THE PREVIOUS CALL TO THE MENU HANDLER.

ERROR MESSAGES ARE OBTAINED BY CALLING OBJECT MODULE CMSG. SEE THE INSTALLATION AND CUSTOMIZATION GUIDE FOR A DESCRIPTION OF SPF MESSAGE FORMATS.

PURPOSE:

MHA IS CALLED TO READ, DISPLAY AND PROCESS A MENU FROM THE SPFMENUS DATA SET. MHA INITIALIZES FIELDS ON THE MENU FROM PASSED PARAMETERS, AND FROM THE MENU DEFINITION READ FROM THE SPFMENUS DATA SET. AFTER READING INPUT FROM THE TERMINAL, MHA PASSES INFORMATION BACK TO THE CALLING PROGRAM VIA THE CALLING SEQUENCE PARAMETERS.

INVOKED WITH:

CALL TO MHA

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	IN/OUT	LOGICAL DISPLAY TABLE
2.	MENUID	CHAR(8)	INPUT	MENU NAME
3.	OPTIONS	FIXED(31)	INPUT	CONTROL OPTIONS
4.	PARM1	*	IN/OUT	PARAMETER FOR THE MENU
..	...			
..	PARMN	*	IN/OUT	PARAMETER FOR THE MENU

WHERE

MENUID - IS THE NAME OF A MEMBER OF THE SPFMENUS DATA SET TO BE PROCESSED.

OPTIONS - THIS PARAMETER SPECIFIES WHICH OPTIONS ARE TO BE USED.

0 - NO SPECIAL OPTION REQUESTED.

1 - RETURN IF NOT FOUND OPTION. IF THIS OPTION IS SPECIFIED, MHA WILL RETURN TO THE CALLER IF THE REQUESTED MENU IS NOT FOUND IN THE SPFMENUS DATA SET. SEE RETURN CODE SECTION BELOW.

2 - NON-DISPLAY. THIS OPTION CAUSES THIS ROUTINE TO BUILD THE MENU IN THE TLS, BUT NOT TO DISPLAY OR AWAIT RESPONSE. THE CALLING PROGRAM MAY, FOR EXAMPLE, USER MERR TO DISPLAY THE MENU.

3 - BOTH 1 AND 2 ABOVE.

PARM(S) - OPTIONAL PARAMETERS THAT ARE REFERENCED BY THE MENU ACTION STATEMENTS, WHERE VALUES ARE TO BE RETURNED (AND/OR PASSED TO THE MENU). A MAXIMUM OF 100 OPTIONAL PARAMETERS IS SUPPORTED.

RETURN CODE:

0 - NORMAL RETURN.

4 - MENU NOT FOUND (ONLY IF OPTION 1 OR 3 IS SPECIFIED).

998 ABEND - MHA TERMINATES WITH ABEND 998 IF AN ERROR IS ENCOUNTERED PROCESSING THE MENU AND IT IS NOT POSSIBLE TO PROCEED.

NOTES:

THE MHA PARAMETER LIST MUST BE TERMINATED WITH A VLIST FLAG.

SEE THE INSTALLATION AND CUSTOMIZATION GUIDE FOR MORE INFORMATION ON MENU FORMATS.

MNT - MENU TESTER (DEBUGGING AID) ROUTINE MNT

PURPOSE:

MNT IS A TESTING ROUTINE USED TO DISPLAY A MENU SO THAT ITS LAYOUT CAN BE EXAMINED.

INVOKED WITH:

LINK TO SPFTMENU (FROM PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

MNT IS INVOKED BY KEYING "TESTMENU" ON THE PRIMARY OPTION MENU.

MNT CALLS MHA WITH 50 SUBSTITUTIBLE PARAMETERS. MENUS WITH MORE THAN 50 PARAMETERS CANNOT BE TESTED WITH TESTMENU.

PURPOSE:

OPT IS INVOKED WHEN OPTION 0 IS SELECTED FROM THE PRIMARY OPTION MENU. IT DISPLAYS THE SPF PARAMETER OPTIONS MENU, AND THEN EITHER THE TERMINAL CHARACTERISTICS MENU, THE LOG/LIST DEFAULTS MENU, OR THE PROGRAM FUNCTION KEY DEFINITION MENU. PARAMETERS ARE VALIDATED AND THE INFORMATION IS PLACED IN THE TSV OR TKV.

INVOKED WITH:

LINK TO SPFOPT (FROM PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	PARM	CHAR(8)	INPUT	MENU NAME

RETURN CODE:

0 - ALWAYS.

NOTES:

THE INITIAL MENU NAME IS PASSED TO THIS PROGRAM FROM PMD VIA THE SECOND PARAMETER OF THE PRIMARY OPTION MENU. MENU NAME "OPT00" (SPF PARAMETER OPTIONS MENU) IS USED TO INDICATE OPTION 0 WAS SELECTED. MENU NAMES "OPT01" (TERMINAL CHARACTERISTICS MENU), "OPT02" (LOG/LIST DEFAULTS MENU), OR "OPT03" (PROGRAM FUNCTION KEY DEFINITION MENU) INDICATE OPTION 0.1, 0.2, OR 0.3 WERE SELECTED RESPECTIVELY.

IF SESSION MANAGER RELEASE 2 IS INSTALLED, THE TERMINAL CHARACTERISTICS MENU IS CHANGED TO "OPT01SM". THIS MENU HAS MENU AND ACTION STATEMENTS AFTER THE <END> STATEMENT. WHEN THESE LINES ARE MOVED TO THEIR APPROPIATE PLACES, IT ALLOWS THE USER TO CONTROL GOING INTO SESSION MANAGER MODE FOR TSO MESSAGES OUTSIDE OF PRIMARY OPTIONS 4 AND 6.

THE PROGRAM FUNCTION KEY DEFINITION MENU IS EITHER "OPT03A", "OPT03B", OR "OPT03C". THE PROPER MENU IS SELECTED BASED ON THE TERMINAL TYPE AND THE NUMBER OF PF KEYS ENTERED ON THE TERMINAL CHARACTERISTICS MENU.

PFT - PROCESSOR FINAL TERMINATION ROUTINE PFT

PURPOSE:

PFT IS CALLED BY PMD WHEN SPF TERMINATION HAS BEEN REQUESTED. IT DISPLAYS ONE OF THE FINAL MENUS (LOG, LIST, OR LOG/LIST), AND THEN HANDLES THE DISPOSITION OF THE LOG AND LIST DATA SETS.

INVOKED WITH:

CALLED BY PMD

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - NORMAL RETURN, COMPLETE TASK TERMINATION.

4 - TERMINATION ABORTED, RETURN TO PRIMARY OPTION MENU.

NOTES:

NONE.

PURPOSE:

PMD IS INVOKED FROM THE SPF MAIN TASK. IT DISPLAYS THE PRIMARY OPTION MENU ("APRIOPT") AND THEN LINKS TO A LOAD MODULE WHOSE NAME IS TAKEN FROM THE PRIMARY OPTION MENU, BASED ON THE OPTION SELECTED BY THE USER.

INVOKED WITH:

ATTACH TO SPFPMD (FROM SMI OR SMA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TYPE	FIXED(31)	INPUT	ENTRY TYPE - 1(INIT), 2(RESTART)
3.	CODE	FIXED(31)	INPUT	CODE DEPENDS ON ENTRY TYPE

WHERE

CODE - IF TYPE 1 ENTRY (SPF INITIALIZATION) CODE IS:

0-3 - SUCCESSFUL INITIALIZATION
>3 - INITIALIZATION FAILED (SEE MENU PMDPIER FOR A LIST OF REASONS).

- IF TYPE 2 ENTRY (SPLIT SCREEN OR REATTACH) CODE IS:

0 - SPLIT SCREEN
N - 'N' IS THE ABEND CODE FROM PMD TASK TERMINATION WHICH IS TO BE DISPLAYED ON THE PRSTRT (BOX) MENU.

RETURN CODE:

0 - ALWAYS.

NOTES:

WHEN A VALID OPTION IS SELECTED FROM THE PRIMARY OPTION MENU, TWO NAMES ARE RETURNED. PMD LINKS TO THE FIRST OF THE TWO NAMES AND PASSES THE SECOND NAME AS A PARAMETER (ALONG WITH THE TLD).

IF THE PROCESSOR TASK ABENDS, THE SPF MAIN TASK REATTACHES SPFPMD AND PASSES IT THE TASK COMPLETION CODE. PMD THEN CALLS PRS TO LOG THE ABEND INFORMATION AND DISPLAY THE RESTART MENU.

PURPOSE:

PRS IS CALLED BY PMD WHEN PMD IS BEING REATTACHED BECAUSE OF A PREVIOUS TASK ABEND. IT LOGS OUT A MINI-DUMP OF ABEND INFORMATION TO THE LOG DATA SET. IT THEN FORMATS AND DISPLAYS THE ABEND (BOX) MENU.

INVOKED WITH:

CALLED BY PMD

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	CODE	FIXED(31)	INPUT	PREVIOUS TASK TERMINATION CODE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

PTC IS INVOKED WHEN OPTION 6 IS SELECTED FROM THE PRIMARY OPTION MENU AND ALLOWS EXECUTION OF TSO COMMANDS AND CLISTS UNDER SPF.

INVOKED WITH:

LINK TO SPFTSO (FROM PMD)

CALLING SEQUENCE PARAMETERS:

```
1.  TLD       <TLD>       INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

PTC CALLS OBJECT MODULE CAT TO EXECUTE THE COMMAND OR CLIST.

SEE THE NOTES SECTION OF OBJECT MODULE CAT FOR A LIST OF RESTRICTIONS.

PURPOSE:

SCN IS EXECUTED IN THE BACKGROUND AS THE FIRST JOBSTEP OF BACKGROUND JOBS SUBMITTED VIA THE IBM SUPPLIED BACKGROUND PROCESSING SUB-OPTIONS (OPTION 5). THIS JCL IS GENERATED FROM JCL PROTOTYPES IN THE SPFPROCS DATA SET. THE PROGRAM SEARCHES A CONCATENATION OF PARTITIONED DATA SETS FOR A SPECIFIED MEMBER AND, IF THE MEMBER IS FOUND, COPIES IT TO A SEQUENTIAL TEMPORARY DATA SET WHERE IT SERVES AS INPUT TO A PROCESSING PROGRAM IN A SUBSEQUENT JOBSTEP. THIS FUNCTION IS NEEDED TO SUPPORT HIERARCHICAL LIBRARIES.

INVOKED WITH:

```
//SCAN     EXEC  PGM=SPFSCAN,PARM='MEMNAME',COND=(12,LE)
//STEPLIB DD DSN=SPF22.MOD1.SPFLOAD,DISP=SHR
//IN       DD DSN=DSNAME1,DISP=SHR
//         DD DSN=DSNAME2,DISP=SHR
//         DD DSN=DSNAME3,DISP=SHR
//         DD DSN=DSNAME4,DISP=SHR
//OUT      DD UNIT=SYSDA,DISP=(NEW,PASS),SPACE=(CYL,(2,2)),
//            DSN=&TEMP1
```

WHERE

MEMNAME - THE PDS MEMBER BEING SEARCHED FOR
DSNAME1 - FIRST PDS TO BE SEARCHED
DSNAME2 - SECOND PDS TO BE SEARCHED
DSNAME3 - THIRD PDS TO BE SEARCHED
DSNAME4 - FOURTH PDS TO BE SEARCHED

CALLING SEQUENCE PARAMETERS:

1. MEMBER * INPUT PARTITIONED DATA SET MEMBER NAME

WHERE

MEMBER - IS THE STANDARD OS PARAMETER LIST. THE FIRST TWO BYTES CONTAIN THE LENGTH OF THE PARAMETER IN BINARY. THE LENGTH IS FOLLOWED BY A 1 TO 8 BYTE MEMBER NAME.

RETURN CODES:

0 - NORMAL RETURN, MEMBER FOUND AND COPIED.
12 - MEMBER NOT FOUND.
16 - UNABLE TO OPEN INPUT DCB.
20 - I/O ERROR ON INPUT DATA SET.
24 - UNABLE TO OPEN OUTPUT DCB.
28 - I/O ERROR ON OUTPUT DATA SET.

NOTES:

IT IS RECOMMENDED THAT ALL DATA SETS BEING SCANNED HAVE THE SAME BLOCK SIZE. I/O ERRORS MAY RESULT IF THE BLOCK SIZE OF THE FIRST DATA SET IS SMALLER THAN OTHER DATA SETS IN THE CONCATENATION SEQUENCE.

PURPOSE:

SIP IS AN EXIT ROUTINE THAT CAN EXAMINE AND MODIFY SPF PARAMETERS AFTER THEY HAVE BEEN READ FROM THE PARMS DATA SET, AND BEFORE THEY ARE USED FOR SPF PROCESSING. IT ALSO MOVES SELECTED PARAMETERS FROM THE TKV TO THE TSV.

INVOKED WITH:

CALL TO SIP (FROM CIPARMS)

CALLING SEQUENCE:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

SIP IS DESIGNED TO PROVIDE A SIMPLE INTERFACE FOR INSTALLATIONS THAT REQUIRE MODIFICATION TO SPF USER PARMS ON INPUT. IT PROVIDES AN INTERFACE TO CKVGET AND CKVPUT TO RETRIEVE AND THEN STORE BACK SELECTED USER PARAMETERS. IT ALSO PROVIDES ADDRESSABILITY TO SOME COMMON SPF TABLES. BY MODIFYING, ASSEMBLING, AND LINK EDITING SIP AN INSTALLATION CAN PERFORM PARAMETER VALIDATION/MODIFICATION AND STILL MAINTAIN A CLEAN INTERFACE WITH OTHER SPF MODULES.

SIP IS LINK EDITED INTO SPFMAIN.

THE PARAMETERS THAT ARE MODIFIED BY SIP ARE NOT IMMEDIATELY WRITTEN OUT TO THE PARMS DATA SET. THEY WILL NORMALLY BE WRITTEN OUT AT SPF TERMINATION AND MAY BE WRITTEN OUT AT OTHER TIMES DURING SPF PROCESSING.

A SIMILAR TYPE OF EXIT IS PROVIDED WHEN SPF PARAMETERS ARE WRITTEN TO THE PARMS DATA SET. SEE THE DESCRIPTION OF OBJECT MODULE SOP.

SMA - SPF MAIN ATTACH ROUTINE SMA

PURPOSE:

SMA IS CALLED BY SMI AND SMC TO ATTACH A PROCESSOR TASK (SPFPMD). IT IS ALSO CALLED TO DETACH SPFPMD AND CLEAN UP THE ASSOCIATED TLD. SMA CONTAINS THE STAI EXIT ROUTINE THAT IS ENTERED IF THE PROCESSOR TASK ABENDS.

INVOKED WITH:

CALL SMA (FROM SMI AND SMC)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE (TLD0)
2.	PMDTYPE	FIXED(31)	INPUT	PMD ENTRY TYPE
3.	PMDCODE	FIXED(31)	INPUT	PMD INPUT CODE

WHERE

PMDTYPE - ENTRY TYPE FOR PMD (SEE PMD).

PMDCODE - ENTRY CODE FOR PMD (SEE PMD).

RETURN CODE:

0 - ALWAYS.

NOTES:

SMA IS FIRST CALLED BY SMI TO CREATE THE FIRST PROCESSOR TASK. SMC CALLS SMA TO CREATE ANOTHER PROCESSOR TASK WHEN THE SPLIT PF KEY IS PRESSED. SMC ALSO CALLS SMA TO DETACH THE PROCESSOR TASK AND CLEAN UP THE TLD WHEN A PROCESSOR TASK TERMINATES (NORMAL OR ABNORMAL). ABNORMAL TERMINATION CAUSES A REATTACH OF SPFPMD.

PURPOSE:

SMC INTERFACES BETWEEN A PROCESSOR TASK AND THE TERMINAL. IT GENERATES AN OUTPUT STREAM USING THE LOGICAL SCREEN TABLES (TLS) FROM THE PROCESSORS. IT CALLS THE COMMON SUBROUTINE CTPUT TO OUTPUT TO THE TERMINAL AND THEN READS FROM THE TERMINAL USING THE COMMON SUBROUTINE CTGET. AFTER THE INPUT IS RECEIVED, THE ATTENTION ID (AID) IS ANALYZED AND THE INPUT DATA IS MOVED TO THE TLS.

INVOKED WITH:

CALL TO SMC (FROM SMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE (TLDO)

RETURN CODE:

0 - ALWAYS.

NOTES:

THE REDISPLAY KEY (PA2) AND THE SPLIT, SWAP, CURSOR, AND PRINT PF KEYS ARE COMPLETLY PROCESSED BY SMC. OTHER KEYS CAUSE CONTROL TO BE PASSED TO THE PROCESSOR TASKS VIA POST/WAIT LOGIC.

SMD - SPF MAIN DRIVER ROUTINE SMD

PURPOSE:

SMD IS INVOKED FROM THE SPF DRIVER (MODULE SPF). IT LOADS SPFTBLS, SPFSUBS, AND SPFTCM. THEN SMD CALLS SMI AND IF THERE IS NO INITIALIZATION ERROR IT CALLS SMC. SMD CONTAINS THE STAX EXIT ROUTINE THAT IS ENTERED IF THE PA1 KEY IS PRESSED.

INVOKED WITH:

LINK TO SPFMAIN (FROM SPF)

CALLING SEQUENCE PARAMETERS:

1.	TSOPARM	PTR(31)	INPUT	ADDR OF TSO PARAMETER LIST
2.	DCB	*	INPUT	ADDR OF SPFLIB DCB, OR ZERO

WHERE

TSOPARM - IS THE ADDRESS OF THE FOUR-ADDRESS LIST PASSED BY THE TSO TMP TO ANY COMMAND PROCESSOR. THE FOUR ADDRESSES ARE:

1.	CBUF	-	THE TSO COMMAND BUFFER CONTROL BLOCK
2.	UPT	-	THE TSO USER PROFILE TABLE
3.	PSCB	-	THE TSO PROTECTED STEP CONTROL BLOCK
4.	ECT	-	THE TSO ENVIRONMENT CONTROL TABLE

DCB - IS THE ADDRESS OF AN OPEN DCB, IF SPFLIB WAS ALLOCATED PRIOR TO SPF EXECUTION. THE DCB IS USED AS THE TASKLIB DCB WHEN ATTACHING SPFPMD. ZERO IS PASSED IF SPFLIB WAS NOT ALLOCATED.

RETURN CODE:

0 - ALWAYS.

NOTES:

THE TSOPARM LIST CONTAINS THE STANDARD PARAMETERS THAT ARE PASSED TO ANY TSO COMMAND PROCESSOR. SEE "GUIDE TO WRITING A TERMINAL MONITOR OR A COMMAND PROCESSOR" FOR MORE INFORMATION.

SMI - SPF MAIN INITIALIZATION ROUTINE SMI

PURPOSE:

SMI PERFORMS SPF INITIALIZATION. INCLUDED IS THE ALLOCATION AND OPENING OF SPFMENUS, SPFMSGS, SPFPROCS, AND SPFPARMS. THE PARMS INITIALIZATION SUBROUTINE (CIPARMS) IS CALLED TO READ THE USER PARMS RECORD FROM THE PARMS DATA SET OR CREATED A FIRST-TIME SPF USER PARMS RECORD. SMI CONTAINS THE STAE EXIT ROUTINE THAT IS ENTERED IF THE SPF MAIN TASK ABENDS.

INVOKED WITH:

CALL TO SMI (FROM SMD)

CALLING SEQUENCE PARAMETERS:

1. TLD <TLD> INPUT LOGICAL DISPLAY TABLE (TLD0)

RETURN CODE:

0 - SPF IS SUCCESSFULLY INITIALIZED.

>0 - SMI ENCOUNTERED AN ERROR DURING INITIALIZATION.

NOTES:

NONE.

SML - SPF MAIN LINE I-O INTERFACE ROUTINE SML

PURPOSE:

SML IS CALLED BY SMC WHENEVER A PROCESSOR TASK IS ABOUT TO INVOKE A PROGRAM THAT MIGHT PERFORM STANDARD TSO LINE I/O.

INVOKED WITH:

CALL TO SML (FROM SMC)

CALLING SEQUENCE PARAMETERS:

1. TLD <TLD> INPUT LOGICAL DISPLAY TABLE (TLD0)

RETURN CODE:

0 - ALWAYS.

NOTES:

THE PROCESSOR TASK SIGNALS SMC BY POSTING THE DISPLAY REQUEST ECB (TLDDRECB) WITH A SPECIAL CODE. SML POSITIONS THE CURSOR AND CLEARS THE SCREEN FROM THE CURSOR POSITION TO THE BOTTOM. SML THEN WAITS FOR THE NEXT DISPLAY REQUEST, AFTER WHICH IT PASSES CONTROL BACK TO SMC WHICH PROCESS THE DISPLAY REQUEST ALONG WITH A COMPLETE REDISPLAY OF THE SCREEN.

PURPOSE:

SOP IS AN EXIT ROUTINE THAT CAN EXAMINE AND MODIFY SPF PARAMETERS BEFORE THEY ARE WRITTEN TO THE PARMS DATA SET. IT ALSO MOVES SELECTED PARAMETERS FROM THE TSV TO THE TKV. THE PASSWORD VALUE IS BLANKED AND "RACF" VALUES ON JCL JOB CARDS ARE SET TO QUESTION MARKS.

INVOKED WITH:

CALL TO SOP (FROM CUPARMS)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

SOP IS DESIGNED TO PROVIDE AN EASY INTERFACE FOR INSTALLATIONS THAT REQUIRE MODIFICATION TO SPF USER PARMS ON OUTPUT. IT PROVIDES AN INTERFACE TO CKVGET AND CKVPUT TO RETRIEVE AND THEN STORE BACK SELECTED USER PARAMETERS. IT ALSO PROVIDES ADDRESSABILITY TO SOME COMMON SPF TABLES. BY MODIFYING, ASSEMBLING, AND LINK EDITING SOP AN INSTALLATION CAN PERFORM PARAMETER VALIDATION/MODIFICATION AND STILL MAINTAIN A CLEAN INTERFACE WITH OTHER SPF MODULES.

SOP IS LINK EDITED INTO SPFSUBS.

SOP ACTUALLY MODIFIES A COPY OF THE PARAMETERS. THIS COPY IS WRITTEN TO THE PARMS DATA SET, BUT THE ORIGINAL (UNMODIFIED) DATA CONTINUES TO BE USED DURING THE CURRENT SPF SESSION.

A SIMILAR TYPE OF EXIT IS PROVIDED WHEN SPF PARAMTERS ARE READ FROM THE PARMS DATA SET. SEE THE DESCRIPTION OF OBJECT MODULE SIP.

ONLY THE "RACF" "PASSWORD" VALUE IS SET TO QUESTION MARKS. A SUPER-ZAPPABLE SWITCH BYTE IS PROVIDED TO HAVE "USER" AND "GROUP" VALUES ALSO SET TO QUESTION MARKS. REFER TO THE PROGRAM LISTINGS.

SPC - SPF PARMS CONVERSION ROUTINE

PURPOSE:

SPC CONVERTS A PARMS RECORD FROM THE SPF VERSION 2.1 FORMAT INTO THE SPF VERSION 2.2 FORMAT. CIPARMS LINKS TO SPC WHEN IT RECOGNIZES THAT THE PARMS RECORD WHICH HAS BEEN READ IS A VERSION 2.1 RECORD.

INVOKED WITH:

LINK TO SPFSPC (FROM CIPARMS)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE (TLD0)
2.	TKV	<TKV>	INPUT	DEFAULT TKV FOR VERSION 2.2
3.	TKV2	CHAR(*)	INPUT	SPF PARMS DATA SET RECORD

WHERE

TKV2 - IS THE VERSION 2.1 PARMS RECORD.

RETURN CODE:

0 - ALWAYS.

NOTES:

IN CREATING THE TKV, DATA IS TAKEN FROM THE VERSION 2.1 TSV. HOWEVER, NOT ALL OF THE DATA THAT MIGHT BE USED IS COPIED OVER TO THE NEW TKV, SINCE SOME OF IT MIGHT BE INCONSISTENT WITH SPF VERSION 2.2.

SPF - SPF DRIVER ROUTINE SPF

PURPOSE:

SPF IS THE HIGHEST LEVEL ROUTINE IN THE SPF MAIN TASK. IT RECEIVES CONTROL FROM THE TSO TERMINAL MONITOR PROGRAM AND PASSES CONTROL TO THE SPF MAIN DRIVER ROUTINE (SMD).

INVOKED WITH:

ATTACH TO SPF (FROM TERMINAL MONITOR PROGRAM)

CALLING SEQUENCE PARAMETERS:

1.	CBUF	*	INPUT	THE TSO COMMAND BUFFER
2.	UPT	*	INPUT	THE TSO USER PROFILE TABLE
3.	PSCB	*	INPUT	THE TSO PROTECTED STEP CONTROL BLOCK
4.	ECT	*	INPUT	THE TSO ENVIRONMENT CONTROL BLOCK

WHERE

THE FOUR PARAMETERS ARE THE STANDARD PARAMETERS THAT ARE PASSED TO ANY TSO COMMAND PROCESSOR. SEE "GUIDE TO WRITING A TERMINAL MONITOR OR A COMMAND PROCESSOR" FOR MORE INFORMATION.

RETURN CODE:

0 - ALWAYS.

NOTES:

SPF DETERMINES WHETHER OR NOT AN SPFLIB FILE HAS BEEN ALLOCATED. IF IT HAS, A DCB IS OPENED AND ITS ADDRESS IS USED IN LINKING TO SPFMAIN (WHICH IN TURN USES IT FOR LOADS AND AS A TASKLIB IN ATTACHING SPFPMD). IF THE SPFLIB FILE HAS NOT BEEN ALLOCATED, AN ADDRESS OF ZERO IS PASSED TO SPFMAIN.

SPFCALCP - SPF CALL COMMAND PROCESSOR ROUTINE

PURPOSE:

SPFCALCP IS A COPY OF THE TSO CALL COMMAND PROCESSOR. IT IS INVOKED WHEN A CALL COMMAND IS TO BE EXECUTED UNDER SPF.

INVOKED WITH:

CALL TO SPFCALCP

CALLING SEQUENCE PARAMETERS:

1.	CBUF	*	INPUT	THE TSO COMMAND BUFFER
2.	UPT	*	INPUT	THE TSO USER PROFILE TABLE
3.	PSCB	*	INPUT	THE TSO PROTECTED STEP CONTROL BLOCK
4.	ECT	*	INPUT	THE TSO ENVIRONMENT CONTROL TABLE
5.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

WHERE

THE FOUR PARAMETERS ARE THE STANDARD PARAMETERS THAT ARE PASSED TO ANY TSO COMMAND PROCESSOR. SEE "GUIDE TO WRITING A TERMINAL MONITOR OR A COMMAND PROCESSOR" FOR MORE INFORMATION. THE LAST PROGRAM IS THE LOGICAL DISPLAY TABLE AND ALLOWS THE CALLED TO HAVE ACCESS TO ANY SPF TABLES.

RETURN CODE:

FROM THE CALLED PROGRAM.

NOTES:

SPFCALCP IS NEEDED BECAUSE THE TSO CALL COMMAND PROCESSOR (CALL) NO LONGER EXISTS IN MVS SYSTEMS.

ONLY UNAUTHORIZED PROGRAMS MAY BE INVOKED VIA SPFCALCP BECAUSE SPF RUNS AS AN UNAUTHORIZED PROBLEM PROGRAM.

PURPOSE:

SPFSC93X IS AN SVC 93 EXIT ROUTINE WHICH PROVIDES THE SPF/SESSION MANAGER INTERFACE. THIS INTERFACE ALLOWS SPF TO CONTROL WHEN SESSION MANAGER MODE IS ENTERED.

INVOKED WITH:

CALL TO SPFSC93X (FROM IKTTMPX1)

CALLING SEQUENCE PARAMETERS:

REGISTERS 0 AND 1 ARE THE SAME AS ENTRY TO SVC 93.
REGISTERS 2 AND 10 ARE AVAILABLE FOR PROGRAM USE.
ALL OTHER REGISTERS CONTAIN SESSION MANAGER DATA.

RETURN CODE:

-8 - REQUEST HAS NOT BEEN PROCESSED BY SPFSC93X. THE SESSION MANAGER SHOULD PROCESS IT.
-4 - REQUEST HAS BEEN PROCESSED BY SPFSC93X. THE SESSION MANAGER SHOULD PASS THE -4 RETURN CODE BACK TO SVC 94.
0 - REQUEST HAS BEEN PROCESSED BY SPFSC93X. THE SESSION MANAGER SHOULD PASS THE 0 RETURN CODE BACK TO SVC 93.

NOTES:

SESSION MANAGER RELEASE 2 IS REQUIRED FOR THE SPF/SESSION MANAGER INTERFACE TO BE EFFECTIVE.

PURPOSE:

SPFSC94X IS AN SVC 94 EXIT ROUTINE WHICH PROVIDES THE SPF/SESSION MANAGER INTERFACE. THIS INTERFACE ALLOWS SPF TO CONTROL WHEN SESSION MANAGER MODE IS ENTERED.

INVOKED WITH:

CALL TO SPFSC94X (FROM IKTTMPX2)

CALLING SEQUENCE PARAMETERS:

REGISTERS 0 AND 1 ARE THE SAME AS ENTRY TO SVC 94.
REGISTERS 6 AND 8 ARE AVAILABLE FOR PROGRAM USE.
ALL OTHER REGISTERS CONTAIN SESSION MANAGER DATA.

RETURN CODE:

-8 - REQUEST HAS NOT BEEN PROCESSED BY SPFSC94X. THE SESSION MANAGER SHOULD PROCESS IT.
-4 - REQUEST HAS BEEN PROCESSED BY SPFSC94X. THE SESSION MANAGER SHOULD PASS THE -4 RETURN CODE BACK TO SVC 94.
0 - REQUEST HAS BEEN PROCESSED BY SPFSC94X. THE SESSION MANAGER SHOULD PASS THE 0 RETURN CODE BACK TO SVG 94.

NOTES:

SESSION MANAGER RELEASE 2 IS REQUIRED FOR THE SPF/SESSION MANAGER INTERFACE TO BE EFFECTIVE.

TCM - TABLE - COMMANDS-CLISTS

PURPOSE:

TCM IS THE TABLE OF COMMANDS. IT CONTAINS A LIST OF COMMAND NAMES. THE COMMON ATTACH ROUTINE (CAT) LOOKS UP IN THIS TABLE THE NAME OF ANY PROCESSOR OR CLIST THAT IS TO BE EXECUTED.

REFERENCED VIA:

TLD (TLDDCLS - TLDTCMP).

NOTES:

EACH NAME IN THE TCM IS IDENTIFIED AS A COMMAND PROCEDURE (CLIST), A COMMAND PROCESSOR, AN INVALID COMMAND, OR A BLDL REQUIRED COMMAND. IN THE CASE OF A BLDL REQUIRED COMMAND, A BLDL IS PERFORMED (WITH DCB=0) AND IF THE COMMAND IS FOUND IT IS ASSUMED TO BE A COMMAND PROCESSOR. IF IT IS NOT FOUND WITH A BLDL, IT IS ASSUMED TO BE A CLIST. THE LAST ENTRY IN THE TCM SPECIFIES HOW COMMANDS NOT IN THE TABLE SHOULD BE HANDLED

THE TCM OBJECT MODULE IS IN REENTRANT LOAD MODULE SPFTCM, WHICH IS LOADED BY OBJECT MODULE SMD.

TKV - TABLE - KEYWORD VALUES TKV

PURPOSE:

TKV IS THE TABLE OF KEYWORD/VALUES. THIS OBJECT MODULE IS THE USED TO CREATE FOR A NEW USER THE IN-MEMORY TKV CONTROL BLOCK THAT IS USED BY SPF TO SAVE A USERS "REMEMBERED" PARAMETERS. THE IN-MEMORY TKV IS SAVED FROM SESSION TO SESSION IN THE SPFPARMS DATASET.

REFERENCED VIA:

VCON (CIPTKVCP IN CIPARMS OBJECT MODULE)

NOTES:

FOR FURTHER INFORMATION ABOUT THE TKV SEE THE DATA AREAS SECTION OF THIS MANUAL.

THE TKV OBJECT MODULE IS LINK EDITED TO THE SPFMAIN LOAD MODULE.

TKW - TABLE - KEYWORDS TKW

PURPOSE:

TKW IS THE TABLE OF KEYWORDS TABLES. IT CONTAINS POINTERS TO THE THREE KEYWORD TABLES (FOR SPF SYSTEM WIDE KEYWORDS, FOR EDIT AND BROWSE PRIMARY COMMAND KEYWORDS, AND FOR EDIT LINE COMMAND KEYWORDS).

REFERENCED VIA:

TLD (TLDDCLS - TLDTKWP).

NOTES:

KEYWORDS ARE 1 TO 8 CHARACTER NAMES. ASSOCIATED WITH EACH KEYWORD IS ITS LENGTH, AND AN INTERNAL ONE BYTE CODE.

THE USE OF KEYWORD TABLES ENABLES MORE EFFICIENT CODE (SINCE ONLY A SINGLE BYTE NEEDS TO BE REFERENCED IN CHECKING COMMANDS AND PARAMETERS, AND IT ALLOWS ALIAS NAMES SINCE TWO ENTRIES IN THE TABLE CAN HAVE THE SAME INTERNAL CODE.

THE NUMBER OF KEYWORD ENTRIES IS LIMITED TO 255.

CALLED INTERNAL PROCEDURES ARE USED TO REFERENCE INFROMATION IN THE KEYWORD TABLES.
   "SYSCODE" IS USED TO RETRIEVE A CODE, IF A KEYWORD IS KNOWN.
   "SYSWORD" IS USED TO RETRIEVE A KEYWORD, IF A CODE IS KNOWN.

THE TKW OBJECT MODULE IS LINK EDITED TO THE SPFMAIN LOAD MODULE.

TRT - TABLE - STATIC TRANSLATE TABLES

PURPOSE:

TRT IS A TABLE OF STATIC TRANSLATE TABLES.

REFERENCED VIA:

INDIVIDUAL TABLES IN TRT ARE ADDRESSABLE FROM THE TCT WHICH IN TURN IS ADDRESSABLE FROM THE TLD. (TLDDCLS - TLDTCTP)

NOTES:

THE TABLES THAT MAKE UP THE TRT ARE:

TRTLOC	-	SCREEN LOC TRANS TABLE
TRTATT	-	ATTR BYTE TRANS TABLE
TRTAID	-	AID TRANS TABLE

THE TRT OBJECT MODULE IS LINK EDITED TO THE SPFMAIN LOAD MODULE.

TSC - TABLE - SUBROUTINE COMMON TSC

PURPOSE:

THE TSC OBJECT MODULE IS USED TO BUILD THE TSC CONTROL BLOCKS IN DYNAMIC STORAGE, WHICH IN TURN ARE USED TO ADDRESS SPF COMMON SUBROUTINES.

REFERENCED VIA:

TSI (TSIDCLS - TSITSCP)

NOTES:

EACH LOGICAL SCREEN HAS A TSC WHICH IS BUILT IN GETMAINED STORAGE FROM THE TSC OBJECT MODULE BY OBJECT MODULE PMD. THE TSC CONTROL BLOCK IN DYNAMIC STORAGE IS REFERENCED BY THE PROCESSOR TASKS VIA TLDTSCP.

SEE THE DATA AREAS SECTION OF THIS MANUAL FOR FURTHER INFORMATION ABOUT THE TSC CONTROL BLOCK.

THE TSC OBJECT MODULE IS LOCATED IN THE SPFSUBS LOAD MODULE WHICH IS LOADED BY OBJECT MODULE SMD.

TSI - TABLE - SPF INTERFACES TSI

PURPOSE:

THE TSI OBJECT MODULE IS A SKELETON FOR THE PRIMARY SPF CONTROL BLOCKS.

REFERENCED VIA:

TLD (TLDDCLS - TLDTSIP)

NOTES:

FOLLOWING IS A LIST OF THE CONTROL BLOCKS CONTAINED WITHIN THE TSI OBJECT MODULE:

DCB	- DATA CONTROL BLOCKS FOR THE FOLLOWING DATASETS: SPFMENUS SPFMSGS SPFPARMS SPFPROCS SPFLOG SPFLIST SPFEDIT BACKUP DATASET 1 SPFEDIT BACKUP DATASET 2
TCT	- CONTROLLER TABLES ARRAY
TDS	- DATA SET TABLE
TFD	- FILE DEFINITION TABLES FOR ALL DATASETS LISTED UNDER DCB
TFI	- FIND MEMBER TABLES FOR THE FOLLOWING DATASETS: SPFMSGS SPFPROCS SPFMENUS
TLD0	- LOGICAL DISPLAY TABLE (CONTROLLER)
TLD1	- LOGICAL DISPLAY TABLE (PROCESSOR TASK 1)
TLD2	- LOGICAL DISPLAY TABLE (PROCESSOR TASK 2)
TPD	- PHYSICAL DISPLAY TABLE
TSV	- SPF VARIABLES TABLE
TSI	- SPF INTERFACE TABLE
TXC	- SPF EXITS CONTROL TABLE (SVC 93 AND 94 EXIT ROUTINES)

THE TSI OBJECT MODULE ALSO CONTAINS THE SPF PROLOG AND EPILOG CODE INVOKED VIA THE SPFPROC AND SPFRETRN MACROS.

THE TSI OBJECT MODULE IS LOCATED IN LOAD MODULE SPFTBLS, WHICH IS THE ONLY NON-REENTERABLE MODULE IN SPF. IT IS LOADED BY OBJECT MODULE SMD.

TT1 - TABLE - 3277 DEPENDENT TABLES TT1

PURPOSE:

TT1 CONTAINS THE TERMINAL DEPENDENT TABLES FOR THE 3277/3275 TERMINALS.

REFERENCED VIA:

INDIVIDUAL TABLES TT1 ARE ADDRESSABLE FROM THE TCT WHICH IN TURN IS ADDRESSABLE FROM THE TLD. (TLDDCLS - TLDTCTP)

NOTES:

THE TABLES THAT MAKE UP TT1 ARE:

TT1UPP	-	UPPER CASE TRANS TABLE
TT1LOW	-	LOWER CASE TRANS TABLE
TT1VAL	-	VALID OUTPUT CHARACTER TRANS TABLE
TT1BTO	-	BROWSE INVALID CHARACTER TRANS TABLE
TT1ETO	-	EDIT INVALID CHARACTER TRANS TABLE
TT1GSC	-	EDIT GENERIC STRING CHARACTER CODE TRANS TABLE
TT1GSM	-	EDIT GENERIC STRING MASTER TRANS TABLE
TT1GSS	-	EDIT GENERIC STRING SPECIAL CHARACTER TRANS TABLE

THE TT1 OBJECT MODULE IS IN REENTRANT LOAD MODULE SPF3277, WHICH IS LOADED BY OBJECT MODULE SMI OR SMC.

- TABLE - 3278 DEPENDENT TABLES

PURPOSE:

TT2 CONTAINS THE TERMINAL DEPENDENT TABLES FOR THE 3278/3276 TERMINALS.

REFERENCED VIA:

INDIVIDUAL TABLES TT2 ARE ADDRESSABLE FROM THE TCT WHICH IN TURN IS ADDRESSABLE FROM THE TLD. (TLDDCLS - TLDTCTP)

NOTES:

THE TABLES THAT MAKE UP TT2 ARE:

TT2UPP	-	UPPER CASE TRANS TABLE
TT2LOW	-	LOWER CASE TRANS TABLE
TT2VAL	-	VALID OUTPUT CHARACTER TRANS TABLE
TT2BTO	-	BROWSE INVALID CHARACTER TRANS TABLE
TT2ETO	-	EDIT INVALID CHARACTER TRANS TABLE
TT2GSC	-	EDIT GENERIC STRING CHARACTER CODE TRANS TABLE
TT2GSM	-	EDIT GENERIC STRING MASTER TRANS TABLE
TT2GSS	-	EDIT GENERIC STRING SPECIAL CHARACTER TRANS TABLE

THE TT2 OBJECT MODULE IS IN REENTRANT LOAD MODULE SPF3278, WHICH IS LOADED BY OBJECT MODULE SMI OR SMC.

TT3 - TABLE - 3278 CANADIAN(FRENCH) DEPENDENT TABLES TT3

PURPOSE:

TT3 CONTAINS THE TERMINAL DEPENDENT TABLES FOR THE 3278/3276 CANADIAN(FRENCH) TERMINALS.

REFERENCED VIA:

INDIVIDUAL TABLES TT3 ARE ADDRESSABLE FROM THE TCT WHICH IN TURN IS ADDRESSABLE FROM THE TLD. (TLDDCLS - TLDTCTP)

NOTES:

THE TABLES THAT MAKE UP TT3 ARE:

TT3UPP	-	UPPER CASE TRANS TABLE
TT3LOW	-	LOWER CASE TRANS TABLE
TT3VAL	-	VALID OUTPUT CHARACTER TRANS TABLE
TT3BTO	-	BROWSE INVALID CHARACTER TRANS TABLE
TT3ETO	-	EDIT INVALID CHARACTER TRANS TABLE
TT3GSC	-	EDIT GENERIC STRING CHARACTER CODE TRANS TABLE
TT3GSM	-	EDIT GENERIC STRING MASTER TRANS TABLE
TT3GSS	-	EDIT GENERIC STRING SPECIAL CHARACTER TRANS TABLE

THE TT3 OBJECT MODULE IS IN REENTRANT LOAD MODULE SPF3278C, WHICH IS LOADED BY OBJECT MODULE SMI OR SMC.

TUT - TUTORIAL PROCESSOR ROUTINE TUT

PURPOSE:

TUT IS INVOKED WHEN OPTION 7 IS SELECTED FROM THE PRIMARY OPTION MENU. IT DISPLAYS EACH TUTORIAL PAGE, USING THE MENU HANDLER (MHA), AND DETERMINES THE NEXT PAGE TO BE DISPLAYED. TUT IS ALSO INVOKED WHEN THE HELP PF KEY IS PRESSED AFTER A LONG MESSAGE IS DISPLAYED ON THE SECOND LINE.

INVOKED WITH:

LINK TO SPFTUTOR (FROM PMD OR CHELP)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	PARM	CHAR(8)	INPUT	MENU NAME

RETURN CODE:

0 - ALWAYS.

NOTES:

TUTORIAL PAGE SEQUENCES ARE DEFINED IN THE MENU DEFINITIONS FOR THE PAGES THEMSELVES BY A MOTHER-SISTER-DAUGHTER RELATIONSHIP. THIS PROGRAM SELECTS THE APPROPRIATE PAGE (MOTHER, SISTER, DAUGHTER OR SPECIAL REQUEST) TO BE DISPLAYED BASED ON THESE PARAMETERS, USER REQUEST, AND PREVIOUS PAGE FLOW.

THE MENU NAME "T" (TUTORIAL INTRODUCTION) IS PASSED TO TUT VIA THE SECOND PARAMETER OF THE PRIMARY OPTION MENU WHEN INVOKED FROM PMD. THE TLDHELP FIELD IS PASSED TO TUT WHEN INVOKED FROM CHELP. THIS FACILITY ALLOWS TUT TO BE USED AS A GENERAL PURPOSE DISPLAY PROGRAM FOR A SET OF TUTORIAL TYPE MENUS.

UAA - ALLOCATE NEW DATA SET ROUTINE UAA

PURPOSE:

UAA ALLOCATES A NEW DATA SET (DATA SET UTILITY OPTION).

INVOKED WITH:

CALL TO UAA (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

UAC - CATALOG OR UNCATALOG DATA SET ROUTINE UAC

PURPOSE:

UAC CATALOGS OR UNCATALOGS A DATA SET (DATA SET UTILITY OPTION).

INVOKED WITH:

CALL TO UAC (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

UAD DELETES A DATA SET (DATA SET UTILITY OPTION).

INVOKED WITH:

CALL TO UAD (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

UAI - DISPLAY DATA SET INFORMATION ROUTINE UAI

PURPOSE:

UAI DISPLAYS DATA SET INFORMATION (DATA SET UTILITY OPTION).

INVOKED WITH:

CALL TO UAI (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

UAR RENAMES A DATA SET (DATA SET UTILITY OPTION).

INVOKED WITH:

CALL TO UAR (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

UCA - CATALOG MANAGEMENT DRIVER ROUTINE UCA

PURPOSE:

UCA IS INVOKED WHEN THE USER SELECTS OPTION 4 FROM THE UTILITY SELECTION MENU (OR OPTION 3.4 FROM THE PRIMARY OPTION MENU). ITS ONLY FUNCTION IS TO DETERMINE WHETHER THE OPERATING SYSTEM IS SVS OR MVS, AND TO LINK TO THE APPROPRIATE CATALOG MANAGEMENT ROUTINE (SPFUC1 OR SPFUC2).

INVOKED WITH:

LINK TO SPFUCA (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

```
1.  TLD       <TLD>       INPUT     LOGICAL DISPLAY TABLE
```

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

UC1 PROCESSES CATALOG MANAGEMENT REQUESTS IF THE OPERATING SYSTEM IS SVS.

INVOKED WITH:

LINK TO SPFUC1 (FROM UCA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

UC2 - MVS CATALOG MANAGEMENT ROUTINE UC2

PURPOSE:

UC2 PROCESSES CATALOG MANAGEMENT REQUESTS IF THE OPERATING SYSTEM IS MVS.

INVOKED WITH:

LINK TO SPFUC2 (FROM UCA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

UDA IS INVOKED WHEN THE USER SELECTS OPTION 1 OR 2 FROM THE UTILITY SELECTION MENU (OR OPTION 3.1 OR 3.2 FROM THE PRIMARY OPTION MENU). IT DISPLAYS THE LIBRARY UTILITY MENU OR DATA SET UTILITY MENU, ALLOCATES THE SPECIFIED DATA SET, AND CALLS ONE OF THE FOLLOWING PROGRAMS TO PERFORM THE DESIRED FUNCTION: UAA, UAC, UAD, UAI, UAR UDM, UDP, UDX, OR UDZ.

INVOKED WITH:

LINK TO SPFUDA (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	PARM	CHAR(8)	INPUT	MENU NAME

RETURN CODE:

0 - ALWAYS.

NOTES:

THE MENU NAME "UDA1" FOR THE LIBRARY UTILITIES OR "UDA2" FOR THE DATA SET UTILITIES IS PASSED TO THIS PROGRAM FROM UTIL VIA THE SECOND PARAMETER OF THE "UTIL" MENU (OR FROM PMD VIA THE SECOND PARAMETER OF THE PRIMARY OPTION MENU). THIS ALLOWS UDA TO BE USED AS A GENERAL PURPOSE DRIVER FOR UTILITIES. THE SELECTIONS ON "UDA1" AND "UDA2" COULD BE REARRANGED, E.G. ALL THE SELECTIONS COULD BE PLACED ON ONE MENU OR BE REARRANGED ON THREE MENUS.

UDM - LIBRARY UTILITY MEMBER LIST ROUTINE

PURPOSE:

UDM PROCESSES MEMBER LIST REQUESTS (LIBRARY UTILITY OPTION). IT CALLS CML TO DISPLAY THE MEMBER LIST, AND PASSES THE ADDRESS OF UDMS (INVOKED VIA CML TO PROCESS EACH MEMBER SELECTED FROM THE LIST). UDM ALSO HANDLES REQUESTS TO PRINT, RENAME, DELETE, OR BROWSE A SINGLE MEMBER. IN THIS CASE, IT CALLS UDMS DIRECTLY.

INVOKED WITH:

CALL TO UDM (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

UDMS - LIBRARY UTILITY MEMBER SELECT ROUTINE UDMS

PURPOSE:

UDMS IS CALLED BY CML (ON BEHALF OF UDM) OR DIRECTLY BY UDM. IT IS A CML SELECTION ROUTINE USED TO PROCESS PRINT, DELETE, RENAME, AND BROWSE MEMBER REQUESTS.

INVOKED WITH:

CALL TO UDMS (FROM UDM OR CML)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	CODE	BIT(32)	INPUT	CML CONTROL BIT CODES
4.	NAME	CHAR(8)	IN/OUT	CML MEMBER NAME
5.	UDMSPARM	<UDACOMM>	IN/OUT	COMMON PARAMETERS FROM UDM
6.	SCODE	CHAR(1)	IN/OUT	SELECT CODE
7.	MEMBER	CHAR(8)	INPUT	SELECTED MEMBER NAME
8.	RENAME	CHAR(8)	IN/OUT	SELECTED NEW MEMBER NAME
9.	FLAGS	BIT(8)	IN/OUT	FLAGS

WHERE

UDMSPARM IS THE UDM/UDMS COMMON PARAMETER AREA (UDACOMM) WHICH IS PASSED TO UDMS DIRECTLY FROM UDM OR INDIRECTLY THROUGH CML.

RETURN CODE:

0 - ALWAYS.

NOTES:

SEE CML OBJECT MODULE DESCRIPTION FOR FURTHER EXPLANATION OF PARAMETERS.

UDP - PRINT DATA SET ROUTINE UDP

PURPOSE:

UDP PROCESSES PRINT DATA SET REQUESTS (LIBRARY UTILITY OPTION).

INVOKED WITH:

CALL TO UDP (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

UDX PROCESSES PRINT INDEX LISTING REQUESTS (LIBRARY UTILITY OPTION).

INVOKED WITH:

CALL TO UDX (FROM UDA AND UDP)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

UDZ - COMPRESS DATA SET ROUTINE UDZ

PURPOSE:

UDZ PROCESSES PDS COMPRESS REQUESTS (LIBRARY UTILITY OPTION). THE IBM UTILITY 'IEBCOPY' IS ATTACHED TO ACCOMPLISH THE COMPRESS.

INVOKED WITH:

CALL TO UDZ (FROM UDA)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	COMM	<UDACOMM>	IN/OUT	UDA COMMON AREA

RETURN CODE:

0 - ALWAYS.

NOTES:

THE LOAD MODULE NAME THAT IS ATTACHED TO PERFORM THE COMPRESS IS TAKEN FROM THE 'UDA1' MENU. IF THE NAME IS BLANK (AS DISTRIBUTED), 'IEBCOPY' IS USED. SINCE 'IEBCOBY' ABENDS UNDER MVS (DUE TO THE NONAUTHORIZED STATE OF SPF), THIS NAME MAY BE CHANGED TO A COMPRESS PROGRAM THAT DOES NOT REQUIRE AUTHORIZATION. REFER TO "FOREGROUND COMPRESS PROCEDURES UNDER MVS" IN THE INSTALLATION AND CUSTOMIZATION GUIDE.

PURPOSE:

UHC IS INVOKED WHEN THE USER SELECTS OPTION 6 FROM THE UTILITY SELECTION MENU (OR OPTION 3.6 FROM THE PRIMARY OPTION MENU). IT PRINTS OR PUNCHES SPECIFIED DATA SETS VIA A BACKGROUND JOB, OR PRINTS DATA SETS VIA "DSPRINT".

INVOKED WITH:

LINK TO SPFUHC (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

PURPOSE:

UMC IS INVOKED WHEN THE USER SELECTS OPTION 3 FROM THE UTILITY SELECTION MENU (OR OPTION 3.3 FROM THE PRIMARY OPTION MENU). IT DISPLAYS THE MOVE/COPY UTILITY MENUS, ALLOCATES THE SPECIFIED DATA SETS, AND OPENS THE "FROM" DATA SET FOR INPUT.

UMC CALLS SUBROUTINE UMCS TO PROCESS EACH MEMBER IN THE "FROM" DATA SET. IF A MEMBER LIST WAS REQUESTED, UMC CALLS CML AND PASSES THE ADDRESS OF UMCS (TO BE INVOKED VIA CML). UMC ALSO CALLS UMCS TO HANDLE A SEQUENTIAL DATA SET.

INVOKED WITH:

LINK TO SPFUMC (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

UMCS - MOVE-COPY UTILITY MEMBER SELECT ROUTINE

PURPOSE:

UMCS IS CALLED BY CML (ON BEHALF OF UMC) OR DIRECTLY BY UMC. UMCS IS A CML SELECTION ROUTINE USED TO COPY EACH MEMBER (OR AN ENTIRE SEQUENTIAL DATA SET) FROM ONE DATA SET TO ANOTHER.

INVOKED WITH:

CALL TO UMCS (FROM UMC OR CML)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	CODE	BIT(32)	INPUT	CML CONTROL BIT CODES
4.	NAME	CHAR(8)	IN/OUT	CML MEMBER NAME
5.	UMCSPARM	*	IN/OUT	COMMON PARAMETERS FROM UMC
6.	SCODE	CHAR(1)	IN/OUT	SELECT CODE
7.	MEMBER	CHAR(8)	INPUT	SELECTED MEMBER NAME
8.	RENAME	CHAR(8)	IN/OUT	SELECTED NEW MEMBER NAME
9.	FLAGS	BIT(8)	IN/OUT	FLAGS

WHERE

UMCSPARM IS THE UMC/UMCS COMMON PARAMETER AREA (UMCCOMM) WHICH IS PASSED TO UMCS DIRECTLY FROM UMC OR INDIRECTLY THROUGH CML.

RETURN CODE:

0 - ALWAYS.

NOTES:

SEE CML OBJECT MODULE DESCRIPTION FOR FURTHER EXPLANATION OF PARAMETERS.

PURPOSE:

UOL IS INVOKED WHEN THE USER SELECTS OPTION 8 FROM THE UTILITY SELECTION MENU (OR OPTION 3.8 FROM THE PRIMARY OPTION MENU). IT PERFORMS THE OUTPUT LISTING UTILITY FUNCTIONS VIA THE TSO "OUTPUT" COMMAND.

INVOKED WITH:

LINK TO SPFUOL (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	PARM	CHAR(8)	INPUT	MENU NAME

WHERE

PARM - IS THE NAME OF THE MENU THAT WILL BE DISPLAYED BY UOL.

RETURN CODE:

0 - ALWAYS

NOTES:

THE MENU NAME "UOL01" IS PASSED TO UOL FROM UTIL VIA THE SECOND PARAMETER OF THE "UTIL" MENU OR FROM PMD VIA THE SECOND PARAMETER OF THE PRIMARY OPTION MENU.

URS - RESET STATISTICS UTILITY ROUTINE

PURPOSE:

URS IS INVOKED WHEN THE USER SELECTS OPTION 5 FROM THE UTILITY SELECTION MENU (OR OPTION 3.5 FROM THE PRIMARY OPTION MENU). URS DISPLAYS THE RESET STATISTICS MENU, ALLOCATES THE SPECIFIED DATA SET, AND OPENS IT FOR INPUT.

URS CALLS SUBROUTINE URSS TO PROCESS EACH MEMBER. IF A MEMBER LIST WAS REQUESTED, URS CALLS CML AND PASSES THE ADDRESS OF URSS (TO BE INVOKED VIA CML).

INVOKED WITH:

LINK TO SPFURS (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

NONE.

URSS - RESET STATISTICS UTILITY MEMBER SELECT ROUTINE

PURPOSE:

URSS IS CALLED BY CML (ON BEHALF OF URS) OR DIRECTLY BY URS. IT IS A CML SELECTION ROUTINE USED TO UPDATE THE STATISTICS FOR A SINGLE MEMBER OF AN SPF LIBRARY DATA SET.

INVOKED WITH:

CALL TO URSS (FROM URS OR CML)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	TFD	<TFD>	IN/OUT	FILE DEFINITION TABLE
3.	CODE	BIT(32)	INPUT	CML CONTROL BIT CODES
4.	NAME	CHAR(8)	IN/OUT	CML MEMBER NAME
5.	URSSPARM	*	IN/OUT	COMMON PARAMETERS FROM URS
6.	SCODE	CHAR(1)	IN/OUT	SELECT CODE
7.	MEMBER	CHAR(8)	INPUT	SELECTED MEMBER NAME
8.	RENAME	CHAR(8)	IN/OUT	SELECTED NEW MEMBER NAME
9.	FLAGS	BIT(8)	IN/OUT	FLAGS

WHERE

URSSPARM IS THE URS/URSS COMMON PARAMETER AREA (URSSPARM) WHICH IS PASSED TO URSS DIRECTLY FROM URS OR INDIRECTLY THROUGH CML.

RETURN CODE:

0 - ALWAYS.

NOTES:

SEE CML OBJECT MODULE DESCRIPTION FOR FURTHER EXPLANATION OF PARAMETERS.

USC - SCRIPT/VS UTILITY ROUTINE

PURPOSE:

USC IS INVOKED WHEN THE USER SELECTS OPTION 9 FROM THE UTILITY SELECTION MENU (OR OPTION 3.9 FROM THE PRIMARY OPTION MENU). USC INTERFACES TO THE SCRIPT/VS PROGRAM PRODUCT.

INVOKED WITH:

LINK TO SPFUSC (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	MENU	CHAR(8)	INPUT	SCRIPT SELECTION MENU NAME

RETURN CODE:

0 - ALWAYS.

NOTES:

THE MENU NAME "SCRPTA" IS PASSED TO USC FROM UTIL VIA THE SECOND PARAMETER OF THE "UTIL" MENU OR FROM PMD VIA THE SECOND PARAMETER OF THE PRIMARY OPTION MENU.

UTIL - UTILITY DRIVER ROUTINE UTIL

PURPOSE:

UTIL IS A GENERAL PURPOSE SELECTION MENU PROCESSOR. IT IS USED TO PROCESS THE OPTION 3 AND OPTION 4 SELECTION MENUS, AND CAN BE USED TO CREATE ADDITIONAL INSTALLATION SELECTION MENUS. THE PROGRAM DISPLAYS A SELECTION MENU, AND LINKS TO THE APPROPRIATE LOAD MODULE, BASED ON THE SUBOPTION SELECTED.

INVOKED WITH:

LINK TO SPFUTIL (FROM PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE
2.	PARM	CHAR(8)	INPUT	MENU NAME

RETURN CODE:

0 - ALWAYS.

NOTES:

UTIL CAN BE BYPASSED BY ENTERING A FULLY QUALIFIED SELECTION ON THE PRIMARY OPTION MENU (E.G. "3.1", "4.5", ETC.).

FOR FURTHER INFORMATION ABOUT USING UTIL FOR CUSTOM TAILORED SELECTION MENUS, SEE THE INSTALLATION AND CUSTOMIZATION GUIDE.

PURPOSE:

UVT IS INVOKED WHEN THE USER SELECTS OPTION 7 FROM THE UTILITY SELECTION MENU (OR OPTION 3.7 FROM THE PRIMARY OPTION MENU). UVT DISPLAYS OR PRINTS THE VTOC OF A DASD VOLUME.

INVOKED WITH:

LINK TO SPFUVT (FROM UTIL OR PMD)

CALLING SEQUENCE PARAMETERS:

1.	TLD	<TLD>	INPUT	LOGICAL DISPLAY TABLE

RETURN CODE:

0 - ALWAYS.

NOTES:

THERE ARE 3 PARAMETERS PASSED TO UVT FROM THE UVT MENU WHICH CONTROL THE CONSTRUCTION OF THE VTOC LIST AND SUSEQUENT PROCESSING TIME.

```
  1. DATA SET LIST SORT CODE - CHAR(1) - "A" FOR ALPHABETIC SORT.
                                       - "P" FOR NO SORT.
  2. FORMAT CONTROL MENU - CHAR(8) - MENU NAME TO FORMAT LIST.
  3. DATA SET LIST CONTROL - CHAR(1) - "Y" FOR DATA SET LIST.
                                       "N" FOR NO DATA SET LIST.
```

THE FORMAT CONTROL MENU PROVIDES THE DETAIL OF THE VTOC LIST. THE FIRST CHARACTER OF EACH LINE OF THIS MENU DESCRIBES ITS INTENDED USE BY UVT AS FOLLOWS:

"I" - INFORMATION SECTION BEFORE THE DATA SET LIST.
"E" - TRAILER SECTION AFTER THE DATA SET LIST.
"M" - MESSAGE MODEL USED TO CONSTRUCT EACH DATA SET LINE.

RULES:

- THERE MUST BE AT LEAST 1 LINE AND NO MORE THAN 24.
- THE "I" AND "E" TYPE LINES ARE SIMILAR EXCEPT FOR THEIR LOCATION IN THE LIST. THEY CAN BE MODIFIED ACCORDING TO MENU PROCESSING RULES. THE VARIABLES CAN BE PLACED ON ANY OF THESE LINES. IF THERE ARE NO "I" OR "E" TYPE LINES THEN THE CORRESPONDING SECTION IS NOT INCLUDED IN THE LIST.
- A SINGLE "M" TYPE LINE IS REQUIRED. THIS LINE CAN BE MODIFIED ACCORDING TO MESSAGE PROCESSING RULES.

THE DISTRIBUTED VERSION OF THE FORMAT CONTROL MENU IS "UVTI". THE FIRST LINE IN THIS MENU IS A "FIELDS" STATEMENT TO ALLOW PERCENT SIGNS IN THE MENU.

SECTION 4

DIRECTORY

THIS SECTION CONTAINS LISTINGS AND CROSS REFERENCE LISTINGS OF SPF LOAD MODULES, OBJECT MODULES, MENUS, MESSAGES AND EXTERNAL SYMBOLS.

THE SECTION CONTAINS THE FOLLOWING LISTS:

```
     LOAD MODULES   - WITH INCLUDED OBJECT MODULES
     OBJECT MODULES - WITH INCLUDING LOAD MODULE
     OBJECT MODULES - WITH SPF MENUS REFERENCED
     SPF MENUS      - WITH OBJECT MODULES REFERENCING
     OBJECT MODULES - WITH SPF MESSAGES REFERENCED
     SPF MESSAGES   - WITH OBJECT MODULES REFERENCING
     OBJECT MODULES - WITH OTHER OBJECT MODULES REFERENCED
     OBJECT MODULES - WITH OTHER OBJECT MODULES REFERENCING
     OBJECT MODULES - WITH EXTERNAL SYMBOLS DEFINED
     OBJECT MODULES - WITH EXTERNAL SYMBOLS REFERENCED
     OBJECT MODULES - WITH SVC ROUTINES REFERENCED
     SVC ROUTINES   - WITH OBJECT MODULES REFERENCING
```

LOAD MODULE / OBJECT MODULE LISTING

SPF LOAD MODULES ARE LISTED ALPHABETICALLY. EACH LOAD MODULE NAME IS FOLLOWED BY THE NAME ITS OF ENTRY POINT, AND THE NAMES OF ALL OF THE OBJECT MODULES THAT ARE INCLUDED IN THE LOAD MODULE.

LOAD	ENTRY	INCLUDED
MODULE	POINT	OBJECT MODULES
SPF	SPF	SPF
SPFBRO	BRO	BRO
SPFCALCP	SPFCALCP	SPFCALCP
SPFEDIT	EDD	EDD
SPFFOR	FOR	FOR
SPFJOB	JOB	JOB
SPFMAIN	SMD	CIPARMS SIP SMA SMC SMD SMI SML TRT TKV TKW
SPFOPT	OPT	OPT
SPFPMD	PMD	PFT PMD PRS
SPFSCAN	SCN	SC
SPFSPC	SPC	SPC
SPFSUBS	TSC	BCD CAT CBC CBDSN CBF CBG CBR CBS CCB CCD CCP CCS CDA CDAIR CDATE CDC CDERR CDF CDG CDISPL CDO CDP CDT CERR CFI CHC CHELP CHPJ CHPL CIR CIV CJC CJF CJN CKVGET CKVPUT CLM CLOG CMB CML CMSG CPRINT CRELS CRESV CSB CSCROLL CSM CTA CTF CTGET CTPUT CT1 CT2 CUPARMS CVM CVSDE EBA EBE EBI EBR EBS EBX ECD ECR EDI EDO EFC EFR EFT EGN EGR EMC EML EMP EPC EPD EPF EPI EPO EPP EPR EPS EPX ERA ERC ERD ERF ERI ERN ERO ERR ERS ERX EST ETC ETS ETL MERR MHA SOP TSC
SPFTBLS	TSI	TSI
SPFTCM	TCM	TCM
SPFTMENU	MNT	MNT
SPFTSO	PTC	PTC
SPFTUTOR	TUT	TUT
SPFUCA	UCA	UCA
SPFUC1	UC1	UC1
SPFUC2	UC2	UC2
SPFUDA	UDA	UAA UAC UAD UAI UDA UAR UDM UDMS UDP UDX UDZ
SPFUHC	UHC	UHC
SPFUMC	UMC	UMC UMCS
SPFUOL	UOL	UOL
SPFURS	URS	URS URSS
SPFUSC	USC	USC
SPFUTIL	UTIL	UTIL
SPFUVT	UVT	UVT
SPF3277	TT1	TT1
SPF3278	TT2	TT2
SPF3278C	TT3	TT3

OBJECT MODULE / LOAD MODULE LISTING

SPF OBJECT MODULES ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY THE NAME OF THE INCLUDING LOAD MODULE.

OBJECT	LOAD	OBJECT	LOAD	OBJECT	LOAD
MODULE	MODULE	MODULE	MODULE	MODULE	MODULE
BCD	SPFSUBS	CTPUT	SPFSUBS	MNT	SPFTMENU
BRO	SPFBRO	CT1	SPFSUBS	OPT	SPFOPT
CAT	SPFSUBS	CT2	SPFSUBS	PFT	SPFPMD
CBC	SPFSUBS	CUPARMS	SPFSUBS	PMD	SPFPMD
CBDSN	SPFSUBS	CVM	SPFSUBS	PRS	SPFPMD
CBF	SPFSUBS	CVSDE	SPFSUBS	PTC	SPFTSO
CBG	SPFSUBS	EBA	SPFSUBS	SCN	SPFSCAN
CBR	SPFSUBS	EBE	SPFSUBS	SIP	SPFMAIN
CBS	SPFSUBS	EBI	SPFSUBS	SMA	SPFMAIN
CCB	SPFSUBS	EBR	SPFSUBS	SMC	SPFMAIN
CCD	SPFSUBS	EBS	SPFSUBS	SMD	SPFMAIN
CCP	SPFSUBS	EBX	SPFSUBS	SMI	SPFMAIN
CCS	SPFSUBS	ECD	SPFSUBS	SML	SPFMAIN
CDA	SPFSUBS	ECR	SPFSUBS	SOP	SPFSUBS
CDAIR	SPFSUBS	EDD	SPFEDIT	SPC	SPFSPC
CDATE	SPFSUBS	EDI	SPFSUBS	SPF	SPF
CDC	SPFSUBS	EDO	SPFSUBS	SPFCALCP	SPFCALCP
CDERR	SPFSUBS	EFC	SPFSUBS	SPFSC93X	IGC0009C
CDF	SPFSUBS	EFR	SPFSUBS	SPFSC94X	IGC0009D
CDG	SPFSUBS	EFT	SPFSUBS	TCM	SPFTCM
CDISPL	SPFSUBS	EGN	SPFSUBS	TKV	SPFMAIN
CDO	SPFSUBS	EGR	SPFSUBS	TKW	SPFMAIN
CDP	SPFSUBS	EMC	SPFSUBS	TRT	SPFMAIN
CDT	SPFSUBS	EML	SPFSUBS	TSC	SPFSUBS
CERR	SPFSUBS	EMP	SPFSUBS	TSI	SPFTBLS
CFI	SPFSUBS	EPC	SPFSUBS	TT1	SPF3277
CHC	SPFSUBS	EPD	SPFSUBS	TT2	SPF3278
CHELP	SPFSUBS	EPF	SPFSUBS	TT3	SPF3278C
CHPJ	SPFSUBS	EPI	SPFSUBS	TUT	SPFTUTOR
CHPL	SPFSUBS	EPO	SPFSUBS	UAA	SPFUDA
CIPARMS	SPFMAIN	EPP	SPFSUBS	UAC	SPFUDA
CIR	SPFSUBS	EPR	SPFSUBS	UAD	SPFUDA
CIV	SPFSUBS	EPS	SPFSUBS	UAI	SPFUDA
CJC	SPFSUBS	EPX	SPFSUBS	UAR	SPFUDA
CJF	SPFSUBS	ERA	SPFSUBS	UCA	SPFUCA
CJN	SPFSUBS	ERC	SPFSUBS	UC1	SPFUC1
CKVGET	SPFSUBS	ERD	SPFSUBS	UC2	SPFUC2
CKVPUT	SPFSUBS	ERF	SPFSUBS	UDA	SPFUDA
CLM	SPFSUBS	ERI	SPFSUBS	UDM	SPFUDA
CLOG	SPFSUBS	ERN	SPFSUBS	UDMS	SPFUDA
CMB	SPFSUBS	ERO	SPFSUBS	UDP	SPFUDA
CML	SPFSUBS	ERR	SPFSUBS	UDX	SPFUDA
CMSG	SPFSUBS	ERS	SPFSUBS	UDZ	SPFUDA
CPRINT	SPFSUBS	ERX	SPFSUBS	UHC	SPFUHC
CRELS	SPFSUBS	EST	SPFSUBS	UMC	SPFUMC
CRESV	SPFSUBS	ETC	SPFSUBS	UMCS	SPFUMC
CSB	SPFSUBS	ETL	SPFSUBS	UOL	SPFUOL
CSCROLL	SPFSUBS	ETS	SPFSUBS	URS	SPFURS
CSM	SPFSUBS	FOR	SPFFOR	URSS	SPFURS
CTA	SPFSUBS	JOB	SPFJOB	USC	SPFUSC
CTF	SPFSUBS	MERR	SPFSUBS	UTIL	SPFUTIL
CTGET	SPFSUBS	MHA	SPFSUBS	UVT	SPFUVT

OBJECT MODULE / MENUS LISTING

SPF OBJECT MODULES WHICH USE SPF MENUS ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY THE NAMES OF THE SPF MENUS WHICH IT USES. MENU NAMES FOLLOWED BY AN ASTERISK(*) ARE EITHER CONSTRUCTED NAMES, OR ARE NAMES THAT ARE PASSED TO THE SPECIFIED MODULE.

OBJECT MODULE	SPF MENUS
BRO	BROWSE01 BROWSE02 CML01B
EBA	EDITBR EDITBRER
ECR	EDITCRA1 EDITCRA2 EDITCRA3 EDITRPL1 EDITRPL2 EDITRPL3
EMC	EDITCPY1 EDITCPY2 EDITCPY3 EDITMOV1 EDITMOV2 EDITMOV3
EMP	EDIT01
EPO	CML01E CML01EC EDIT02
EPS	EDIT02
FOR	FOR01(*) FOR02(*) FOR03(*) FOR04(*) FOR05(*) FOR06(*) FOR07(*) FOR08(*)
JOB	JOBA(*) JOBB(*) JOBERROR JOB01(*) JOB02(*) JOB03(*) JOB04(*) JOB05(*) JOB06(*)
MNT	MNT
OPT	OPT00(*) OPT01(*) OPT01SM(*) OPT01CF(*) OPT02(*) OPT03A(*) OPT03B(*) OPT03C(*)
TUT	T(*) TERR THELP TINDEX TTUTOR
PFT	PFT01 PFT02 PFT03
PMD	APRIOPT PMDPIER
PRS	PRSTRT
PTC	PTC
UAA	UAA
UAD	UADC
UAI	UAI UAIPO UAIPOX UAIXX
UAR	UAR
UC1	UC1 UC1B
UC2	UC2 UC2B
UDA	UDA1(*) UDA2(*)
UDM	CML02
UDMS	UDMSB
UHC	UHC UHCJ
UMC	CML03 UMC1 UMC2A UMC2B
UOL	UOLB UOLCHC UOLCHCS UOL01 UOL01S
URS	CML04 URS
USC	CML01F CML01FC SCRERR SCRPRTD SCRPRTF SCRPTA SCRPTD SCRPTF USCBRO
UTIL	FORA(*) UTIL(*)
UVT	UVT UVTB UVTI

MENUS / OBJECT MODULE LISTING

SPF MENUS ARE LISTED ALPHABETICALLY. EACH MENU NAME IS FOLLOWED BY THE NAME OF THE OBJECT MODULE(S) THAT USES IT. ALL MENUS THAT START WITH "T" ARE TUTORIAL MENUS AND ARE NOT INCLUDED IN THE LIST.

SPF MENUS	OBJECT MODULE(S)	SPF MENUS	OBJECT MODULE(S)
APRIOPT	PMD	OPT01SM	OPT
BROWSE01	BRO	OPT01CF	OPT
BROWSE02	BRO	OPT02	OPT
CML01B	BRO	OPT03A	OPT
CML01E	EPO	OPT03B	OPT
CML01EC	EPO	OPT03C	OPT
CML01F	USC	PFT01	PFT
CML01FC	USC	PFT02	PFT
CML02	UDM	PFT03	PFT
CML03	UMC	PMDPIER	PMD
CML04	URS	PRSTRT	PRS
EDITBR	EBA	PTC	PTC
EDITBRER	EBA	SCRERR	USC
EDITCPY1	EMC	SCRPRTD	USC
EDITCPY2	EMC	SCRPRTF	USC
EDITCPY3	EMC	SCRPTA	USC
EDITCRA1	ECR	SCRPTD	USC
EDITCRA2	ECR	SCRPTF	USC
EDITCRA3	ECR	SETUP	MNT
EDITMOV1	EMC	UAA	UAA
EDITMOV2	EMC	UADC	UAD
EDITMOV3	EMC	UAI	UAI
EDITRPL1	ECR	UAIPO	UAI
EDITRPL2	ECR	UAIPOX	UAI
EDITRPL3	ECR	UAIXX	UAI
EDIT01	EMP	UAR	UAR
EDIT02	EPO EPS	UC1	UC1
FORA	UTIL	UC1B	UC1
FOR01	FOR	UC2	UC2
FOR02	FOR	UC2B	UC2
FOR03	FOR	UDA1	UDA
FOR04	FOR	UDA2	UDA
FOR05	FOR	UDMSB	UDMS
FOR06	FOR	UHC	UHC
FOR07	FOR	UHCJ	UHC
FOR08	FOR	UMC1	UMC
JOBA	JOB	UMC2A	UMC
JOBB	JOB	UMC2B	UMC
JOBERROR	JOB	UOLB	UOL
JOB00	JOB	UOLCHC	UOL
JOB01	JOB	UOLCHCS	UOL
JOB02	JOB	UOL01	UOL
JOB03	JOB	UOL01S	UOL
JOB04	JOB	URS	URS
JOB05	JOB	USCBRO	USC
JOB06	JOB	UTIL	UTIL
MNT	MNT	UVT	UVT
OPT00	OPT	UVTB	UVT
OPT01	OPT	UVTI	UVT

OBJECT MODULE / MESSAGES LISTING

SPF OBJECT MODULES WHICH USE SPF MESSAGES ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY THE NAMES OF THE SPF MESSAGES WHICH IT USES.

OBJECT MODULE	SPF MESSAGES
BCD	B500 B501 B502 B503 B504 B505 B506 B507 B508 B510 B511 B512 B513 B514 B515 B516 B517 B518
BRO	B001 B002 B003 G018 G021 G120 G121 G122
CAT	G060 G061 G062 G063 G064 G065 G066 G067 G074 G076 G077
CBDSN	G002 G003 G004 G054 G090 G091 G092 G093 G094
CBF	B101 B102 B103 B104 B105 B106 B151 B152 B153 B154 B155 B156 B157 B158 B159 B161 B162 B163 B164 B165 B166 B167
CBR	B005 B010 B013 B014 B015 B016 B017 B018 B019 G024 G044
CCB	G140 G141 G142 G143 G144 G145 G146 G147 G150 G151 G152 G153 G154 G155 G156 G157 G158 G159 G162 G166
CDA	G007 G044 G080 G081 G082 G087 G088
CDERR	D001 D002 D003 D004 D005 D006 D007 D008 D009 D010 D011 D012 D013 D014 D015 D016 D017 D018 D019 D020 D021 D022
CDF	G083 G084 G085
CDISPL	G056 G057 G058 G059
CDO	G100 G101 G102 G103 G104 G105 G106 G107 G108 G109 G110 G111 G112 G113 G114 G115 G116 G117 G118 G119 G130 G131 G132 G133
CHC	G149 U281 U282 U283 U284 U285 U286 U287 U288 U289 U290
CKVPUT	J022 J023 J024
CLOG	P001 P002
CML	G021 G024 G030 G033 G035 G037 G038 G039 G050 G068 G069
CTA	A001 A002 A003 A004 A005 A006 A007
CTGET	S004 S008
CTPUT	S003 S008
EBA	E300 E301 E303 E310 E311 E312 E313 E314 E315 E316 E319
EBR	E302 E304 E305
EBS	E306 E307 E308 E309
ECD	E500 E501 E502 E503 E504 E505 E506 E507 E508 E510 E511 E512 E513 E514 E515 E516 E517 E518 E520 E521 E522 E523 E524 E525 E526 E527 E528 E530 E531 E532 E533 E534 E535 E536 E537 E538 E540 E541 E542 E543 E544 E545 E546 E547 E548 E550 E551 E552 E553 E554 E555 E556 E557 E558
ECR	E012 E018 E019 E056 E058 E059 E415 E416 E417 G010 G012 G018 G081
EDI	E400 E409 E414 G019
EDO	E006 E007 E008 E009 E015 G020 G031 G032 G034 G045 G048 G053
EFC	E101 E102 E103 E104 E105 E106 E150 E151 E153 E154 E155 E156 E157 E158 E159 E161 E162 E163 E164 E165 E166 E167 E168 E169 E199 E201 E202 E203 E204 E205 E206 E207 E208 E209 E211 E212 E213 E221 E222 E223 E224 E251 E253 E254 E255 E256 E257 E258 E259 E261 E262 E263 E264 E265 E266 E267 E268 E269
EFR	E002 E005 E600 E602 E612 E613 E614 E615 E616 E617 E620 E621 E624 E625 E626 E627 E630 E631 E634 E635 E636 E637 E640 E641 E642 E643 E646 E647 E650 E651 E652 E653 E654 E655 E660 E661 E662 E663 E664 E665 E670 E671 E672 E673 E676 E677 E680 E681 E682 E683 E684 E685 E690 E691 E692 E693 E694 E695 E704 E705 E706 E707 E708 E709 G024
EFT	E068
EMC	E010 E011 E013 E014 E021 E028 E059 E076 E401 E402 E403 E404 E405 E406 E407 E408 E410 E411 E412 E413 E419 G010 G018 G019 G021 G081
EMP	E000 E003 E004 E028 G010
EPC	E040 E041 E042 E043 E044 E045 E046 E047 E048 E049 E050 E051 E052 E053 E054 E055 E080 E081 E082 E083 E084 E085 E086
EPD	E035 E036 E037 E038
EPF	E016 E017 E031 E032 E033 E034 E064 E065 E066 E067 E576 E577 E579
EPI	E060 E071 E560 E561 E562 E563 E564 E565 E566 E567 E568 E570 E571 E572 E573 E574 E575 E580 E581 E582 E583 E590 E591 E592 E593 E594 E595 E596 E597 E598 E599

(CONTINUED ON NEXT PAGE)

OBJECT MODULE / MESSAGES LISTING (CONTINUED)

OBJECT MODULE	SPF MESSAGES
EPO	E022 E023 E024 G012 G018 G022
EPR	E016 E017 E028 E040 E049 E051 E052 E418 E700 E701 E702 E703 E710 E711 G010
EPS	E024 G013
EPX	G040
ERN	E069 E318
FOR	F000 F001 F002 F003 F004 F005 F006 F007 F009 F010 F011 F012 F013 F014 F015 F016 F017 F018 F019 F020 F021 F022 F023 F024 F025 F026 F027 G044
JOB	J001 J004 J006 J007 J009 J010 J011 J012 J013 J014 J015 J016 J017 J018 J019
MHA	M001 M002 M003 M005 M006 M007 M008 M009 M010 M011 M012 M014 M015 M016 M017 M018 M019
MNT	M011
OPT	O001 O002 O003 O004 O006 O007 O008 O009
PFT	O002 O003 P002 P020 P021 P022 P023 P024 P025 P026 P027 P028 P032 P033 P034 P035 P036 P037 P038
PMD	G001 P003
PRS	P011 P012 P013 P014 P015 P016 P017
PTC	G070 G071 G072 G073
SMA	S011 S012 S013 S014 S015
SMC	S001 S002 S006 S007
TUT	G028
UAA	M003 U050 U051 U052 U053 U054 U055 U056 U058 U059
UAC	U017 U130 U131 U132 U133 U134 U135 U136 U137 U138 U139
UAD	U020 U021
UAI	U049
UAR	U008 U010 U012 U013 U014 U015 U016 U017 U018 U023 U024 U025
UC1	G002 U170 U171 U172 U173 U174 U175 U176 U177 U178 U179 U181 U182 U183 U184 U185 U186 U187 U188 U189
UC2	G002 U200 U201 U202 U203 U204 U205 U207 U208 U209
UDA	D021 G006 G007 G015 G027 G054 U000 U002 U003 U006 U007 U008 U009
UDM	U004 U027 U028 U039
UDMS	G122 U002 U003 U026 U029 U030 U031 U032 U033 U034 U035 U036 U037 U038 U039 U042 U148 U149
UDP	G069 U005 U043 U044 U045 U046 U047 U048
UDX	U040 U041 U049
UDZ	U004 U140 U141 U142 U143 U144 U146
UHC	G005 G009 U091 U092 U093 U094 U095 U099 U100 U101 U103 U105 U106 U107 U160 U161 U162 U164 U165 U166 U167 U168 U169
UMC	G008 G018 U060 U061 U062 U063 U064 U065 U067 U068 U069 U070 U071 U072 U073 U074 U078 U079 U080 U088 U089 U110 U113 U114 U210
UMCS	G018 G019 G020 G031 G032 G035 U063 U066 U068 U075 U076 U077 U078 U080 U081 U082 U083 U084 U085 U086 U087 U088 U111 U112 U115 U116 U117 U118 U119
UOL	U220 U221 U222 U223 U224 U225 U228 U229
URS	U120 U121 U127 U129 U230 U231 U233
URSS	G018 G019 G031 G032 G035 U122 U123 U124 U125 U126 U128
USC	G018 G021 G120 G121 G122 U241 U250 U251 U252 U253 U254 U260 U261 U262 U263 U264 U265 U266 U267 U268 U269 U270 U271 U272 U282 U283
UTIL	G001
UVT	G006 U150 U151 U190 U191 U192 U193 U194 U195 U196 U197 U198 U199

MESSAGES / OBJECT MODULE LISTING

SPF MESSAGES ARE LISTED ALPHABETICALLY. EACH MESSAGE NAME IS FOLLOWED BY THE NAME OF THE OBJECT MODULE(S) THAT USES IT. MODULE NAMES FOLLOWED BY AN ASTERISK(*) CONSTRUCT THE MESSAGE NAME.

SPF MESSAGE	OBJECT MODULE
A001	CTA
A002	CTA
A003	CTA
A004	CTA
A005	CTA
A006	CTA
A007	CTA
B001	BRO
B002	BRO
B003	BRO
B005	CBR
B010	CBR
B013	CBR
B014	CBR
B015	CBR
B016	CBR
B017	CBR
B018	CBR
B019	CBR
B101	CBF
B102	CBF
B103	CBF
B104	CBF
B105	CBF
B106	CBF
B151	CBF
B152	CBF
B153	CBF
B154	CBF
B155	CBF
B156	CBF
B157	CBF
B158	CBF
B159	CBF
B161	CBF
B162	CBF
B163	CBF
B164	CBF
B165	CBF
B166	CBF
B167	CBF
B500	BCD
B501	BCD(*)
B502	BCD(*)
B503	BCD
B504	BCD(*)
B505	BCD(*)
B506	BCD
B507	BCD(*)
B508	BCD(*)
B510	BCD
B511	BCD(*)
B512	BCD(*)
B513	BCD
B514	BCD(*)
B515	BCD(*)
B516	BCD
B517	BCD(*)
B518	BCD(*)
D001	CDERR
D002	CDERR
D003	CDERR
D004	CDERR
D005	CDERR
D006	CDERR
D007	CDERR
D008	CDERR
D009	CDERR
D010	CDERR
D011	CDERR
D012	CDERR
D013	CDERR
D014	CDERR
D015	CDERR
D016	CDERR
D017	CDERR
D018	CDERR
D019	CDERR
D020	CDERR
D021	CDERR
D021	UDA
D022	CDERR
E000	EMP
E002	EFR
E003	EMP
E004	EMP
E005	EFR
E006	EDO
E007	EDO
E008	EDO
E009	EDO
E010	EMC
E011	EMC
E012	ECR
E013	EMC
E014	EMC
E015	EDO
E016	EPF
E016	EPR
E017	EPF
E017	EPR
E018	ECR
E019	ECR
E021	EMC
E022	EPO
E023	EPO
E024	EPO
E024	EPS
E028	EMC
E028	EMP
E028	EPR
E031	EPF
E032	EPF
E033	EPF
E034	EPF
E035	EPD
E036	EPD
E037	EPD
E038	EPD
E040	EPC
E040	EPR
E041	EPC
E042	EPC
E043	EPC
E044	EPC
E045	EPC
E046	EPC
E047	EPC
E048	EPC
E049	EPC
E049	EPR
E050	EPC
E051	EPC
E051	EPR
E052	EPC
E052	EPR
E053	EPC
E054	EPC
E055	EPC
E056	ECR
E058	ECR
E059	ECR
E059	EMC
E060	EPI
E064	EPF
E065	EPF
E066	EPF
E067	EPF
E068	EFT
E069	ERN
E071	EPI
E076	EMC
E080	EPC
E081	EPC(*)
E082	EPC
E083	EPC(*)
E084	EPC
E085	EPC(*)
E086	EPC
E101	EFC
E102	EFC
E103	EFC
E104	EFC
E105	EFC
E106	EFC
E150	EFC
E151	EFC
E153	EFC
E154	EFC
E155	EFC
E156	EFC
E157	EFC
E158	EFC
E159	EFC
E161	EFC
E162	EFC
E163	EFC

(CONTINUED ON NEXT PAGE)

MESSAGES / OBJECT MODULE LISTING (CONTINUED)

SPF MESSAGE	OBJECT MODULE	SPF MESSAGE	OBJECT MODULE	SPF MESSAGE	OBJECT MODULE
E164	EFC	E404	EMC	E552	ECD(*)
E165	EFC	E405	EMC	E553	ECD
E166	EFC	E406	EMC	E554	ECD(*)
E167	EFC	E407	EMC	E555	ECD(*)
E168	EFC	E408	EMC	E556	ECD
E169	EFC	E409	EDI	E557	ECD(*)
E199	EFC	E410	EMC	E558	ECD(*)
E201	EFC	E411	EMC	E560	EPI
E202	EFC	E412	EMC	E561	EPI(*)
E203	EFC	E413	EMC	E562	EPI(*)
E204	EFC	E414	EDI	E563	EPI
E205	EFC	E415	ECR	E564	EPI(*)
E206	EFC	E416	ECR	E565	EPI(*)
E207	EFC	E417	ECR	E566	EPI
E208	EFC	E418	EPR	E567	EPI(*)
E209	EFC	E419	EMC	E568	EPI(*)
E211	EFC	E500	ECD	E570	EPI
E212	EFC	E501	ECD(*)	E571	EPI(*)
E213	EFC	E502	ECD(*)	E572	EPI(*)
E221	EFC	E503	ECD	E573	EPI
E222	EFC	E504	ECD(*)	E574	EPI
E223	EFC	E505	ECD(*)	E575	EPI
E224	EFC	E506	ECD	E576	EPF
E251	EFC(*)	E507	ECD(*)	E577	EPF
E253	EFC(*)	E508	ECD(*)	E579	EPF
E254	EFC(*)	E510	ECD	E580	EPI
E255	EFC(*)	E511	ECD(*)	E581	EPI
E256	EFC(*)	E512	ECD(*)	E582	EPI
E257	EFC(*)	E513	ECD	E583	EPI
E258	EFC(*)	E514	ECD(*)	E590	EPI
E259	EFC(*)	E515	ECD(*)	E591	EPI
E261	EFC(*)	E516	ECD	E592	EPI
E262	EFC(*)	E517	ECD(*)	E593	EPI
E263	EFC(*)	E518	ECD(*)	E594	EPI
E264	EFC(*)	E520	ECD	E595	EPI
E265	EFC(*)	E521	ECD(*)	E596	EPI
E266	EFC(*)	E522	ECD(*)	E597	EPI
E267	EFC(*)	E523	ECD	E598	EPI
E268	EFC(*)	E524	ECD(*)	E599	EPI
E269	EFC(*)	E525	ECD(*)	E600	EFR
E300	EBA	E526	ECD	E602	EFR
E301	EBA	E527	ECD(*)	E612	EFR(*)
E302	EBR	E528	ECD(*)	E613	EFR(*)
E303	EBA	E530	ECD	E614	EFR(*)
E304	EBR	E531	ECD(*)	E615	EFR(*)
E305	EBR	E532	ECD(*)	E616	EFR(*)
E306	EBS	E533	ECD	E617	EFR(*)
E307	EBS	E534	ECD(*)	E620	EFR(*)
E308	EBS	E535	ECD(*)	E621	EFR(*)
E309	EBS	E536	ECD	E624	EFR(*)
E310	EBA	E537	ECD(*)	E625	EFR(*)
E311	EBA	E538	ECD(*)	E626	EFR(*)
E312	EBA	E540	ECD	E627	EFR(*)
E313	EBA	E541	ECD(*)	E630	EFR(*)
E314	EBA	E542	ECD(*)	E631	EFR(*)
E315	EBA	E543	ECD	E634	EFR(*)
E316	EBA	E544	ECD(*)	E635	EFR(*)
E318	ERN	E545	ECD(*)	E636	EFR(*)
E319	EBA	E546	ECD	E637	EFR(*)
E400	EDI	E547	ECD(*)	E640	EFR(*)
E401	EMC	E548	ECD(*)	E641	EFR(*)
E402	EMC	E550	ECD	E642	EFR(*)
E403	EMC	E551	ECD(*)	E643	EFR(*)

(CONTINUED ON NEXT PAGE)

SPF MESSAGE	OBJECT MODULE	SPF MESSAGE	OBJECT MODULE	SPF MESSAGE	OBJECT MODULE
E646	EFR(*)	F020	FOR	G035	CML
E647	EFR(*)	F021	FOR	G035	UMCS
E650	EFR(*)	F022	FOR	G035	URSS
E651	EFR(*)	F023	FOR	G037	CML
E652	EFR(*)	F024	FOR	G038	CML
E653	EFR(*)	F025	FOR	G039	CML
E654	EFR(*)	F026	FOR	G040	EPX
E655	EFR(*)	F027	FOR	G044	CCB
E660	EFR(*)	G001	PMD	G044	CDA
E661	EFR(*)	G001	UTIL	G044	FOR
E662	EFR(*)	G002	CBDSN	G045	EDO
E663	EFR(*)	G002	UC1	G048	EDO
E664	EFR(*)	G002	UC2	G050	CML
E665	EFR(*)	G003	CBDSN	G053	EDO
E670	EFR(*)	G004	CBDSN	G054	CBDSN
E671	EFR(*)	G005	UHC	G054	UDA
E672	EFR(*)	G006	UDA	G056	CDISPL
E673	EFR(*)	G006	UVT	G057	CDISPL
E676	EFR(*)	G007	CDA	G058	CDSIPL
E677	EFR(*)	G007	UDA	G059	CDISPL
E680	EFR(*)	G008	UMC	G060	CAT
E681	EFR(*)	G009	UHC	G061	CAT
E682	EFR(*)	G010	ECR	G062	CAT
E683	EFR(*)	G010	EMC	G063	CAT
E684	EFR(*)	G010	EMP	G064	CAT
E685	EFR(*)	G010	EPR	G065	CAT
E690	EFR(*)	G012	ECR	G066	CAT
E691	EFR(*)	G012	EPO	G067	CAT
E692	EFR(*)	G013	EPS	G068	CML
E693	EFR(*)	G015	UDA	G069	CML
E694	EFR(*)	G018	BRO	G069	UDP
E695	EFR(*)	G018	ECR	G070	PTC
E700	EPR	G018	EMC	G071	PTC
E701	EPR(*)	G018	EPO	G072	PTC
E702	EPR	G018	UMC	G073	PTC
E703	EPR(*)	G018	UMCS	G074	CAT
E704	EFR	G018	URSS	G076	CAT
E705	EFR(*)	G018	USC	G077	CAT
E706	EFR	G019	EDI	G080	CDA
E707	EFR(*)	G019	EMC	G081	CDA
E708	EFR	G019	UMCS	G081	ECR
E709	EFR(*)	G019	URSS	G081	EMC
E710	EPR	G020	EDO	G082	CDA
E711	EPR(*)	G020	UMCS	G083	CDF
F000	FOR	G021	BRO	G084	CDF
F001	FOR	G021	CML	G085	CDF
F002	FOR	G021	EMC	G087	CDA
F003	FOR	G021	USC	G088	CDA
F004	FOR	G022	EPO	G090	CBDSN
F005	FOR	G024	CBR	G091	CBDSN
F006	FOR	G024	CML	G092	CBDSN
F007	FOR	G024	EFR	G093	CBDSN
F009	FOR	G027	UDA	G094	CBDSN
F010	FOR	G028	TUT	G100	CDO
F011	FOR	G030	CML	G101	CDO
F012	FOR	G031	EDO	G102	CDO
F013	FOR	G031	UMCS	G103	CDO
F014	FOR	G031	URSS	G104	CDO
F015	FOR	G032	EDO	G105	CDO
F016	FOR	G032	UMCS	G106	CDO
F017	FOR	G032	URSS	G107	CDO
F018	FOR	G033	CML	G108	CDO
F019	FOR	G034	EDO	G109	CDO

(CONTINUED ON NEXT PAGE)

MESSAGES / OBJECT MODULE LISTING (CONTINUED)

SPF MESSAGE	OBJECT MODULE	SPF MESSAGE	OBJECT MODULE	SPF MESSAGE	OBJECT MODULE
G110	CDO	M003	UAA	S013	SMA
G111	CDO	M005	MHA	S014	SMA
G112	CDO	M006	MHA	S015	SMA
G113	CDO	M007	MHA	U000	UDA
G114	CDO	M008	MHA	U002	UDA
G115	CDO	M009	MHA	U002	UDMS
G116	CDO	M010	MHA	U003	UDA
G117	CDO	M011	MHA	U003	UDMS
G118	CDO	M011	MNT	U004	UDM
G119	CDO	M012	MHA	U004	UDZ
G120	BRO	M014	MHA	U005	UDP
G120	USC	M015	MHA	U006	UDA
G121	BRO	M016	MHA	U007	UDA
G121	USC	M017	MHA	U008	UAR
G122	BRO	M018	MHA	U008	UDA
G122	UDMS	M019	MHA	U009	UDA
G122	USC	0001	OPT	U010	UAR
G130	CDO	0002	OPT	U012	UAR
G131	CDO	0002	PFT	U013	UAR
G132	CDO	0003	OPT	U014	UAR
G133	CDO	0003	PFT	U015	UAR
G140	CCB	0004	OPT	U016	UAR
G141	CCB	0006	OPT	U017	UAC
G142	CCB	0007	OPT	U017	UAR
G143	CCB	0008	OPT	U018	UAR
G144	CCB	0009	OPT	U020	UAD
G145	CCB	P001	CLOG	U021	UAD
G146	CCB	P002	CLOG	U023	UAR
G147	CCB	P002	PFT	U024	UAR
G149	CHC	P003	PMD	U025	UAR
G150	CCB	P011	PRS	U026	UDMS
G151	CCB	P012	PRS	U027	UDM
G152	CCB	P013	PRS	U028	UDM
G153	CCB	P014	PRS	U029	UDMS
G154	CCB	P015	PRS	U030	UDMS
G155	CCB	P016	PRS	U031	UDMS
G156	CCB	P017	PRS	U032	UDMS
G157	CCB	P020	PFT	U033	UDMS
G158	CCB	P021	PFT	U034	UDMS
G159	CCB	P022	PFT	U035	UDMS
G162	CCB	P023	PFT	U036	UDMS
G166	CCB	P024	PFT	U037	UDMS
J001	JOB	P025	PFT	U038	UDMS
J004	JOB	P026	PFT	U039	UDM
J006	JOB	P027	PFT	U039	UDMS
J007	JOB	P028	PFT	U040	UDX
J009	JOB	P032	PFT(*)	U041	UDX
J010	JOB	P033	PFT(*)	U042	UDMS
J011	JOB	P034	PFT(*)	U043	UDP
J012	JOB	P035	PFT(*)	U044	UDP
J013	JOB	P036	PFT(*)	U045	UDP
J014	JOB	P037	PFT(*)	U046	UDP
J015	JOB	P038	PFT(*)	U047	UDP
J016	JOB	S001	SMC	U048	UDP
J017	JOB	S002	SMC	U049	UAI
J018	JOB	S003	CTPUT	U049	UDX
J019	JOB	S004	CTGET	U050	UAA
J022	CKVPUT	S006	SMC	U051	UAA
J023	CKVPUT	S007	SMC	U052	UAA
J024	CKVPUT	S008	CTPUT	U053	UAA
M001	MHA	S008	CTGET	U054	UAA
M002	MHA	S011	SMA	U055	UAA
M003	MHA	S012	SMA	U056	UAA

(CONTINUED ON NEXT PAGE)

MESSAGES / OBJECT MODULE LISTING (CONTINUED)

SPF MESSAGE	OBJECT MODULE
U058	UAA
U059	UAA
U060	UMC
U061	UMC
U062	UMC
U063	UMC
U063	UMCS
U064	UMC
U065	UMC
U066	UMCS
U067	UMC
U068	UMC
U068	UMCS
U069	UMC
U070	UMC
U071	UMC
U072	UMC
U073	UMC
U074	UMC
U075	UMCS
U076	UMCS
U077	UMCS
U078	UMC
U078	UMCS
U079	UMC
U080	UMC
U080	UMCS
U081	UMCS
U082	UMCS
U083	UMCS
U084	UMCS
U085	UMCS
U086	UMCS
U087	UMCS
U088	UMC
U088	UMCS
U089	UMC
U091	UHC
U092	UHC
U093	UHC
U094	UHC
U095	UHC
U099	UHC
U100	UHC
U101	UHC
U103	UHC
U105	UHC
U106	UHC
U107	UHC
U110	UMC
U111	UMCS
U112	UMCS
U113	UMC
U114	UMC
U115	UMCS
U116	UMCS
U117	UMCS
U118	UMCS
U119	UMCS
U120	URS
U121	URS
U122	URSS
U123	URSS
U124	URSS
U125	URSS
U126	URSS
U127	URS
U128	URSS
U129	URS
U130	UAC
U131	UAC
U132	UAC
U133	UAC
U134	UAC
U135	UAC
U136	UAC
U137	UAC
U138	UAC
U139	UAC
U140	UDZ
U141	UDZ
U142	UDZ
U143	UDZ
U144	UDZ
U146	UDZ
U148	UDMS
U149	UDMS
U150	UVT
U151	UVT
U160	UHC
U161	UHC
U162	UHC
U164	UHC
U165	UHC
U166	UHC
U167	UHC
U168	UHC
U169	UHC
U170	UC1
U171	UC1
U172	UC1
U173	UC1
U174	UC1
U175	UC1
U176	UC1
U177	UC1
U178	UC1
U179	UC1
U181	UC1
U182	UC1
U183	UC1
U184	UC1
U185	UC1
U186	UC1
U187	UC1
U188	UC1
U189	UC1
U190	UVT
U191	UVT
U192	UVT
U193	UVT
U194	UVT
U195	UVT
U196	UVT
U197	UVT
U198	UVT
U199	UVT
U200	UC2
U201	UC2
U202	UC2
U203	UC2
U204	UC2
U205	UC2
U207	UC2
U208	UC2
U209	UC2
U210	UMC
U220	UOL
U221	UOL
U222	UOL
U223	UOL
U224	UOL
U225	UOL
U228	UOL
U229	UOL
U230	URS
U231	URS
U233	URS
U241	USC
U250	USC
U251	USC
U252	USC
U253	USC
U254	USC
U260	USC
U261	USC
U262	USC
U263	USC
U264	USC
U265	USC
U266	USC
U267	USC
U268	USC
U269	USC
U270	USC
U271	USC
U272	USC
U281	CHC
U282	CHC
U282	USC
U283	CHC
U283	USC
U284	CHC
U285	CHC
U286	CHC
U287	CHC
U288	CHC
U289	CHC
U290	CHC

OBJECT MODULE / CALLED OBJECT MODULES LISTING

SPF OBJECT MODULES WHICH REFERENCE OR CALL OTHER OBJECT MODULES ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY THE NAMES OF THE OBJECT MODULES WHICH IT REFERENCES OR CALLS.

OBJECT MODULE	OBJECT MODULES REFERENCED OR CALLED
BRO	CBC CBR CBS CDA CDF CDO CKVGET CKVPUT CML CMSG CVSDE MERR MHA
CAT	CDAIR CMSG CSM CTPUT
CBC	CSM
CBF	CBG
CBG	CDG CSM
CBR	BCD CBC CBF CBG CBS CCP CCS CDG CDISPL CERR CSCROLL CSM
CBS	CSM
CCB	CDAIR CDERR CDG CFI CKVGET CLOG
CCD	CDATE CVSDE
CDA	CBDSN CDAIR CDERR CDF CMSG
CDAIR	CDT
CDC	CRELS CSM
CDERR	CMSG
CDF	CDAIR CDC CMSG
CDG	CDC CDO
CDISPL	CHELP CKVGET CMSG
CDO	CDC CRELS CRESV CSM
CERR	CDISPL CMSG
CHC	CDC CDP CHPJ CHPL CIV CJC CJF CJN CKVPUT CLOG CMB CMSG CSB CTA CTPUT MERR MHA
CHELP	CSM
CHPJ	CDG CDP CFI CMSG
CHPL	CAT
CIPARMS	CDC CDG CDO CDP CSM SIP
CIR	CDC CDG CDO CSM
CIV	CDATE CIR CVSDE
CJC	CDP
CKVPUT	CMSG CTPUT
CLOG	CDATE CDP CMSG CTA
CMB	CKVGET MHA
CML	CCD CDC CDG CDISPL CDO CERR CSCROLL CSM CVM
CMSG	CDG CFI
CPRINT	CDATE CDG CDP CTA CVSDE
CSB	CAT
CTA	CDERR CMSG CSM CTPUT CT1
CTF	CT2
CTGET	CLOG
CTPUT	CLOG
CT1	CDAIR CDC CDO CDT
CT2	CDAIR CDC CDP
CUPARMS	CDG SOP
EBA	CDG CLOG CMSG CSM CTA CUPARMS EBE MHA
EBE	CSM CTF EBX
EBI	CDP CSM CTA
EBR	CDG CLOG CTA CUPARMS EBE ERA ERD ERI
EBS	CDP CSM CTA CUPARMS EBI
EBX	CUPARMS
ECR	CDA CDC CDF CDO CML CPRINT CVM EDO ERR MERR MHA
EDD	EBE EMP EPO EPS EPX ETC ETS
EDI	CDG ERA ERI EX1
EDO	CDC CDF CDO CDP CLOG CSM ERD ERF EX2
EFC	ERC ERS
EFR	CDISPL CERR CSCROLL EML ERA ERD ERF ERI
EFT	CSM ERA ERC ERD ERF ERI ERN
EGN	ERC ERN
EGR	ERD ERF ERR ERS
EMC	CDA CDC CDF CDO CLOG EDI ERR MERR MHA
EML	CMSG ERA ERI
EMP	CDA CDF CDO CKVGET CKVPUT CVM EBA MERR MHA
EPC	EFT ERA ERC ERD ERF ERI ERR ERS ERX EST ETL

(CONTINUED ON NEXT PAGE)

OBJECT MODULE / CALLED OBJECT MODULES LISTING (CONTINUED)

OBJECT MODULE	OBJECT MODULES REFERENCED OR CALLED
EPD	ERA ERC ERD ERF ERI ERN
EPF	CDC CDP CLOG CML CSB CTA CVM EDO EFC EGN EGR EPP ERA ERD ERF ERI ERS
EPI	CCP CCS ECD EGN EGR
EPO	CML CMSG CVM CVSDE EPR MHA
EPP	CKVGET CKVPUT
EPR	CDA CDF CDO CML CPRINT CSM EBR EBS EBX EDI EDO EFC EFR EFT EML EPC EPD EPF EPI EPP ERA ERD ERF ERI
EPS	CMSG EPR MHA
ERA	CSM
ERC	EBS
ERD	EBS ERO ERR ERS ERX
ERI	EBS ERN ERS ERX
ERN	EGN ERC
ERO	ERA
ERR	ERS ERX
ERS	ERA
ERX	ERA
EST	ERA ERC ERI
ETC	CSM
ETS	CSM EBE
FOR	CAT CDAIR CDERR CDG CFI CKVGET CKVPUT CLOG CSM MERR MHA
JOB	CDC CDG CDP CFI CJC CJF CJN CKVGET CKVPUT CLOG CSB CSM CTA MERR MHA
MERR	MHA
MHA	CDG CDISPL CERR CFI CSM
MNT	MERR MHA
OPT	CJF CJN CKVGET CKVPUT CLOG CUPARMS MERR MHA
PFT	CDC CDP CHPJ CHPL CIV CJC CJF CJN CKVGET CKVPUT CLOG CMSG CSB CTA CTF CTPUT CUPARMS MERR MHA
PMD	CDATE CLM CSM CTF MERR MHA PFT PRS
PRS	CLOG MHA
PTC	CAT CLOG MERR MHA
SIP	CKVGET CKVPUT
SMA	CDAIR CDP CMSG CSM CTPUT
SMC	CDP CMSG CTA CTGET CTPUT CUPARMS SMA SML
SMD	CSM SMC SMI
SMI	CDAIR CMSG CSM CTA CTF CTPUT CIPARMS SMA
SML	CTGET CTPUT
SOP	CKVGET CKVPUT
SPC	CKVGET CKVPUT CSM
TUT	CSM MERR MHA
UAA	CDAIR CDERR CDO CDT CKVGET CKVPUT CLOG MERR MHA
UAC	CLOG
UAD	CDF CLOG MHA
UAI	CIV CKVPUT MHA
UAR	CBDSN CLOG MERR MHA UAC
UC1	CBC CBR CBS CDA CDC CDF CDO CDP CDT CKVGET CLOG CPRINT CTA MERR MHA
UC2	CBC CBR CBS CDC CDO CDP CKVGET CKVPUT CLOG CPRINT CTA MERR MHA
UDA	CBDSN CDA CDF CKVGET CKVPUT MERR MHA UAA UAC UAD UAI UAR UDM UDP UDX UDZ
UDM	CBC CDO CML MHA UDMS
UDMS	CBR CBS CDC CDO CERR CLOG CMSG CPRINT CSM CVM CVSDE MHA
UDP	CDO CML CPRINT UDX
UDX	CCD CDATE CDP CIR CIV CTA
UDZ	CDC CDP CLOG CRELS CRESV CTA
UHC	CBDSN CDA CDC CDF CDO CDP CHPJ CHPL CIV CJC CJF CJN CKVGET CKVPUT CLOG CSB CTA MERR MHA
UMC	CDA CDC CDF CDO CKVGET CKVPUT CLOG CML CMSG CPRINT CVM MERR MHA UMCS
UMCS	CDC CDG CDO CDP CERR CLOG CPRINT CVM
UOL	CAT CBC CBR CBS CCB CDA CDC CDF CDO CHC CKVGET CKVPUT CMB MERR MHA

(CONTINUED ON NEXT PAGE)

OBJECT MODULE / CALLED OBJECT MODULES LISTING (CONTINUED)

```
OBJECT
MODULE   OBJECT MODULES REFERENCED OR CALLED
-------- -----------------------------------------------------------------
URS      CDA CDF CDO CKVGET CKVPUT CLOG CML CMSG MERR MHA URSS
URSS     CDC CDG CDO CERR CLOG CVSDE
USC      CAT CBC CBR CBS CCB CDA CDAIR CDC CDF CDO CHC CKVPUT CMB CML
         CMSG CVSDE MERR MHA
UTIL     MERR MHA
UVT      CBC CBR CBS CDATE CDC CDO CDP CDT CMSG CSM CTA MERR MHA
```

OBJECT MODULE / CALLING OBJECT MODULES LISTING

SPF OBJECT MODULES ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY THE NAMES OF THE OBJECT MODULES WHICH CALL OR OTHERWISE REFERENCE IT.

OBJECT MODULE	OBJECT MODULES WHICH CALL OR REFERENCE
BCD	CBR
CAT	CHPL CSB FOR PTC UOL USC
CBC	BRO CBR UC1 UC2 UDM UOL USC UVT
CBDSN	CDA UAR UDA UHC
CBF	CBR
CBG	CBF CBR
CBR	BRO UC1 UC2 UDMS UOL USC UVT
CBS	BRO CBR UC1 UC2 UDMS UOL USC UVT
CCB	UOL USC
CCD	CML UDX
CCP	CBR EPI
CCS	CBR EPI
CDA	BRO ECR EMC EMP EPR UC1 UDA UHC UMC UOL URS USC
CDAIR	CAT CCB CDA CDF CT1 CT2 FOR SMA SMI UAA USC
CDATE	CCD CIV CLOG CPRINT PMD UDX UVT
CDC	CDF CDG CDO CHC CIPARMS CIR CML CT1 CT2 ECR EDO EMC EPF JOB PFT UC1 UC2 UDMS UDZ UHC UMC UMCS UOL URSS USC UVT
CDERR	CCB CDA CTA FOR UAA
CDF	BRO CDA ECR EDO EMC EMP EPR UAD UC1 UDA UHC UMC UOL URS USC
CDG	CBG CBR CCB CHPJ CIPARMS CIR CML CMSG CPRINT CUPARMS EBA EBR EDI FOR JOB MHA UMCS URSS
CDISPL	CBR CERR CML EFR MHA
CDO	BRO CDG CIPARMS CIR CML CT1 ECR EDO EMC EMP EPR UAA UC1 UC2 UDM UDMS UDP UHC UMC UMCS UOL URS URSS USC UVT
CDP	CHC CHPJ CIPARMS CJC CLOG CPRINT CT2 EBI EBS EDO EPF JOB PFT SMA SMC UC1 UC2 UDX UDZ UHC UMCS UVT
CDT	CDAIR CT1 UAA UC1 UVT
CERR	CBR CML EFR MHA UDMS UMCS URSS
CFI	CCB CHPJ CMSG FOR JOB MHA
CHC	UOL USC
CHELP	CDISPL
CHPJ	CHC PFT UHC
CHPL	CHC PFT UHC
CIPARMS	SMI
CIR	CIV UDX
CIV	CHC PFT UAI UDX UHC
CJC	CHC JOB PFT UHC
CJF	CHC JOB OPT PFT UHC
CJN	CHC JOB OPT PFT UHC
CKVGET	BRO CCB CDISPL CMB EMP EPP FOR JOB OPT PFT SIP SOP SPC UAA UC1 UC2 UDA UHC UMC UOL URS
CKVPUT	BRO CHC EMP EPP FOR JOB OPT PFT SIP SOP SPC UAA UAI UC2 UDA UHC UMC UOL URS USC
CLM	PMD
CLOG	CCB CHC CTGET CTPUT EBA EBR EDO EMC EPF FOR JOB OPT PFT PRS PTC UAA UAC UAD UAR UC1 UC2 UDMS UDZ UHC UMC UMCS URS URSS
CMB	CHC UOL USC
CML	BRO ECR EPF EPO EPR UDM UDP UMC URS USC
CMSG	BRO CAT CDA CDERR CDF CDISPL CERR CHC CHPJ CKVPUT CLOG CTA EBA EML EPO EPS PFT SMA SMC SMI UDMS UMC URS USC UVT
CPRINT	ECR EPR UC1 UC2 UDMS UDP UMC UMCS
CRELS	CDC CDO UDZ
CRESV	CDO UDZ
CSB	CHC EPF JOB PFT UHC
CSCROLL	CBR CML EFR
CSM	CAT CBC CBG CBR CBS CDC CDO CHELP CIPARMS CIR CML CTA EBA EBE EBI EBS EDO EFT EPR ERA ETC ETS FOR JOB MHA PMD SMA SMD SMI SPC TUT UDMS UVT
CTA	CHC CLOG CPRINT EBA EBI EBR EBS EPF JOB PFT SMC SMI UC1 UC2 UDX UDZ UHC UVT

(CONTINUED ON NEXT PAGE)

OBJECT MODULE / CALLING OBJECT MODULES LISTING (CONTINUED)

```
OBJECT
MODULE   OBJECT MODULES WHICH CALL OR REFERENCE
-------- -------------------------------------------------------------
CTF      EBE PFT PMD SMI
CTGET    SMC SML
CTPUT    CAT CHC CKVPUT CTA PFT SMA SMC SMI SML
CT1      CTA
CT2      CTF
CUPARMS  EBA EBR EBS EBX OPT PFT SMC
CVM      CML ECR EMP EPF EPO UDMS UMC UMCS
CVSDE    BRO CCD CIV CPRINT EPO UDMS URSS USC
EBA      EMP
EBE      EBA EBR EDD ETS
EBI      EBS
EBR      EPR
EBS      EPR ERC ERD ERI
EBX      EBE EPR
ECD      EPI
EDI      EMC EPR
EDO      ECR EPF EPR
EFC      EPF EPR
EFR      EPR
EFT      EPC EPR
EGN      EPF EPI ERN
EGR      EPF EPI
EML      EFR EPR
EMP      EDD
EPC      EPR
EPD      EPR
EPF      EPR
EPI      EPR
EPO      EDD
EPP      EPF EPR
EPR      EPO EPS
EPS      EDD
EPX      EDD
ERA      EBR EDI EFR EFT EML EPC EPD EPF EPR ERO ERS ERX EST
ERC      EFC EFT EGN EPC EPD ERN EST
ERD      EBR EDO EFR EFT EGR EPC EPD EPF EPR
ERF      EDO EFR EFT EGR EPC EPD EPF EPR
ERI      EBR EDI EFR EFT EML EPC EPD EPF EPR EST
ERN      EFT EGN EPD ERI
ERO      ERD ECR EGR EMC EPC ERD
ERS      EFC EGR EPC EPF ERD ERI ERR
ERX      EPC ERD ERI ERR
EST      EPC
ETC      EDD
ETL      EPC
ETS      EDD
EX1      EDI
EX2      EDO
MERR     BRO CHC ECR EMC EMP FOR JOB MNT OPT PFT PMD PTC TUT UAA UAR UC1
         UC2 UDA UHC UMC UOL URS USC UTIL UVT
MHA      BRO CHC CMB EBA ECR EMC EMP EPO EPS FOR JOB MERR MNT OPT PFT
         PMD PRS PTC TUT UAA UAD UAI UAR UC1 UC2 UDA UDM UDMS UHC UMC
         UOL URS USC UTIL UVT
PFT      PMD
PRS      PMD
SIP      CIPARMS
SMA      SMC SMI
SMC      SMD
SMI      SMD
SML      SMC
SOP      CUPARMS
SPFSC93X IKTTMPX1
SPFSC94X IKTTMPX2
UAA      UDA
```

(CONTINUED ON NEXT PAGE)

OBJECT MODULE / CALLING OBJECT MODULES LISTING (CONTINUED)

OBJECT	
MODULE	OBJECT MODULES WHICH CALL OR REFERENCE
UAC	UAR UDA
UAD	UDA
UAI	UDA
UAR	UDA
UDM	UDA
UDMS	UDM
UDP	UDA
UDX	UDA UDP
UDZ	UDA
UMCS	UMC
URSS	URS

OBJECT MODULE / EXTERNAL SYMBOLS DEFINED LISTING

SPF EXTERNAL SYMBOLS WHICH ARE DEFINED BY SPF OBJECT MODULES ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY A LIST OF THE EXTERNAL SYMBOLS DEFINED. ALL OBJECT MODULES IN SPF CONTAIN THEIR OWN NAME AS AN EXTERNAL SYMBOL DEFINED, AND THESE ARE NOT INCLUDED IN THE LIST.

OBJECT	
MODULE	EXTERNAL SYMBOLS DEFINED
EPC	ECLAFTER ECLBEFOR ECLBOUND ECLCOPY ECLCOLS ECLDEL ECLINSRT ECLMOVE ECLMASK ECLOVER ECLREP ECLSBOT ECLSHOW ECLSTOP ECLSL ECLSCR ECLSCL ECLSR ECLTENTR ECLTFLOW ECLTSPLT ECLTABS ECLXCLUD
EPF	EPFCANCL EPFCHG EPFFIND EPFLOC EPFNUMB EPFPROF EPFRESET EPFSAVE EPFSUBMT
EPI	EPICANCL EPIMEMB EPINUMB EPIOPTS EPIREASN EPIRESET
ETL	EPCELC
SMA	STAI
SMI	STAE

OBJECT MODULE / EXTERNAL SYMBOLS REFERENCED LISTING

SPF EXTERNAL SYMBOLS WHICH ARE REFERENCED BY SPF OBJECT MODULES ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY A LIST OF THE EXTERNAL SYMBOLS REFERENCED.

```
OBJECT
MODULE   EXTERNAL SYMBOLS REFERENCED
-------- ------------------------------------------------------------------
CIPARMS  SIP TKV
CUPARMS  SOP
PMD      PFT PRS
SMC      SMA SML
SMD      SMC SMI TKW TRT
SMI      CIPARMS SMA
TSC      BCD CAT CBC CBDSN CBF CBG CBR CBS CCB CCD CCP CCS CDA CDAIR
         CDATE CDC CDERR CDF CDG CDISPL CDO CDP CDT CERR CFI CHC CHELP
         CHPJ CHPL CIR CIV CJC CJF CJN CKVGET CKVPUT CLM CLOG CMB CML
         CMSG CPRINT CRELS CRESV CSB CSCROLL CSM CTA CTF CTGET CTPUT CT1
         CT2 CUPARMS CVM CVSDE EBA EBE EBI EBR EBS EBX ECD ECLAFTER
         ECLBEFOR ECLBOUND ECLCOLS ECLCOPY ECLDEL ECLINSRT ECLMASK
         ECLMOVE ECLOVER ECLREP ECLSBOT ECLSCL ECLSCR ECLSHOW ECLSL
         ECLSR ECLSTOP ECLTABS ECLTENTR ECLTFLOW ECLTSPLT ECLXCLUD ECR
         EDI EDO EFC EFR EFT EGN EGR EMC EML EMP EPC EPD EPF EPFCANCL
         EPFCHG EPFFIND EPFLOC EPFNUMB EPFPROF EPFRESET EPFSAVE EPFSUBMT
         EPI EPICANCL EPIMEMB EPINUMB EPIOPTS EPIREASN EPIRESET EPO EPP
         EPR EPS EPX ERA ERC ERD ERF ERI ERN ERO ERR ERS ERX EST ETC ETL
         ETS MERR MHA
UAR      UAC
UDA      UAA UAC UAD UAI UAR UDM UDP UDX UDZ
UDM      UDMS
UDP      UDX
UMC      UMCS
URS      URSS
```

OBJECT MODULE / SVC ROUTINE LISTING

SPF OBJECT MODULES WHICH REFERENCE SYSTEM SERVICES VIA SVC ARE LISTED ALPHABETICALLY. EACH OBJECT MODULE NAME IS FOLLOWED BY THE SVC NUMBER AND SYSTEM SERVICE IT REFERENCES.

OBJECT MODULE	SVC NO.	SYSTEM SERVICE
BRO	18	BLDL
CAT	1	WAIT
	2	POST
	3	EXIT
	6	LINK
	18	BLDL
	40	EXTRACT
	42	ATTACH
	62	DETACH
CBDSN	26	LOCATE
CCB	18	FIND
	19	OPEN
	20	CLOSE
CDAIR	6	LINK
	26	LOCATE
	26	UNCATALOG
	40	EXTRACT
CDC	20	CLOSE
CDERR	26	LOCATE
CDG	1	WAIT
	2	POST
CDISPL	1	WAIT
	2	POST
	13	ABEND
	13	ABEND
CDO	19	OPEN
	27	OBTAIN
	60	STAE
CFI	18	BLDL
CHC	1	WAIT
	2	POST
CHELP	6	LINK
CIPARMS	6	LINK
	18	BLDL
	21	STOW
CIV	24	DEVTYPE
	27	OBTAIN
CKVPUT	1	WAIT
	2	POST
CLM	8	LOAD
CLOG	11	TIME
CML	18	BLDL
	31	FEOV
CPRINT	11	TIME
	18	BLDL
CRELS	48	DEQ
CRESV	40	EXTRACT
	56	ENQ
	56	RESERVE
CSM	4	GETMAIN
	10	FREEMAIN
	10	GETMAIN
CTA	1	WAIT
	2	POST
CTF	1	WAIT
	2	POST
CTGET	93	TGET
	94	STFSMODE
CTPUT	93	TPUT
	94	STFSMODE
CT1	26	LOCATE
	26	UNCATALOG
	29	SCRATCH
	64	RDJFCB
CT2	26	UNCATALOG
	29	SCRATCH
CUPARMS	11	TIME
EBS	11	TIME
ECR	18	BLDL
	48	DEQ
	56	ENQ
EDO	11	TIME
	21	STOW
	60	STAE
EMC	18	BLDL
	21	STOW
	48	DEQ
	56	ENQ
EPO	18	BLDL
	48	DEQ
	56	ENQ
EPS	48	DEQ
	56	ENQ
FOR	18	FIND
	19	OPEN
	20	CLOSE
MHA	13	ABEND
PFT	1	WAIT
	2	POST
	94	GTSIZE
	94	STFSMODE
PMD	6	LINK
	8	LOAD
	9	DELETE
	11	TIME
	40	EXTRACT
SCN	10	FREEMAIN
	10	GETMAIN
	18	FIND
	19	OPEN
	20	CLOSE
SMA	1	WAIT
	13	ABEND
	42	ATTACH
	62	DETACH
	93	TPUT
	94	STLINENO
SMC	1	WAIT
	2	POST
	8	LOAD
	9	DELETE
	13	ABEND(997)
	94	STFSMODE

(CONTINUED ON NEXT PAGE)

OBJECT MODULE / SVC ROUTINE LISTING (CONTINUED)

OBJECT MODULE	SVC NO.	SYSTEM SERVICE
SMD	2	POST
	8	LOAD
	10	FREEMAIN
	10	GETMAIN
	20	CLOSE
	79	STATUS
	94	STFSMODE
	96	STAX
SMI	8	LOAD
	11	TIME
	19	OPEN
	24	DEVTYPE
	44	CHAP
	60	STAE
	64	RDJFCB
	93	TPUT
	94	GTERM
	94	GTSIZE
	94	STCC
	94	STFSMODE
	94	STLINENO
	94	STTRAN
	94	TCLEARQ
SML	1	WAIT
	2	POST
	94	STLINENO
	94	TCLEARQ
SPF	6	LINK
	10	FREEMAIN
	10	GETMAIN
	19	OPEN
	20	CLOSE
	24	DEVTYPE
SPFCALCP	1	WAIT
	6	LINK
	8	LOAD
	9	DELETE
	10	FREEMAIN
	10	GETMAIN
	10	IKJRLSA
	18	BLDL
	19	OPEN
	20	CLOSE
	42	ATTACH
	62	DETACH
	94	TCLEARQ
UAC	24	DEVTYPE
	26	CATALOG
	26	UNCATALOG
UAR	24	DEVTYPE
	30	RENAME
UCA	6	LINK
UC1	1	WAIT
	26	INDEX
	26	LOCATE
	42	ATTACH
	62	DETACH
UC2	1	WAIT
	42	ATTACH
	62	DETACH
UDMS	18	BLDL
	21	STOW
UDX	11	TIME

OBJECT MODULE	SVC NO.	SYSTEM SERVICE
UDZ	1	WAIT
	42	ATTACH
	62	DETACH
UHC	18	BLDL
UMC	18	BLDL
UMCS	18	BLDL
	21	STOW
URSS	11	TIME
	18	BLDL
	21	STOW
USC	18	BLDL
UTIL	6	LINK
UVT	11	TIME
	27	OBTAIN

SVC ROUTINE / OBJECT MODULE LISTING

SYSTEM SERVICES WHICH ARE REFERENCED BY SPF VIA SVC ARE LISTED BY SVC NUMBER AND SERVICE NAME. EACH SERVICE IS FOLLOWED BY THE NAMES OF THE OBJECT MODULES WHICH REFERENCE IT.

SVC NO.	SYSTEM SERVICE	OBJECT MODULE
1	WAIT	CAT
		CDG
		CDISPL
		CHC
		CKVPUT
		CTA
		CTF
		PFT
		SMA
		SMC
		SML
		SPFCALCP
		UC1
		UC2
		UDZ
2	POST	CAT
		CDG
		CDISPL
		CHC
		CKVPUT
		CTA
		CTF
		PFT
		SMC
		SMD
		SML
3	EXIT	CAT
4	GETMAIN	CSM
6	LINK	CAT
		CDAIR
		CHELP
		CIPARMS
		PMD
		SPF
		SPFCALCP
		UCA
		UTIL
8	LOAD	CLM
		PMD
		SMC
		SMD
		SMI
		SPFCALCP
9	DELETE	PMD
		SMC
		SPFCALCP
10	FREEMAIN	CSM
		SCN
		SMD
		SPF
		SPFCALCP
10	GETMAIN	CSM
		SCN
		SMD
		SPF
		SPFCALCP
10	IKJRLSA	SPFCALCP
11	TIME	CLOG
		CPRINT
		CUPARMS
		EBS
		EDO
		PMD
		SMI
		UDX
		URSS
		UVT
13	ABEND	CDISPL
		MHA
		SMA
		SMC
18	BLDL	BRO
		CAT
		CFI
		CIPARMS
		CML
		CPRINT
		ECR
		EMC
		EPO
		SPFCALCP
		UDMS
		UHC
		UMC
		UMCS
		URSS
		USC
18	FIND	CCB
		FOR
		SCN
19	OPEN	CCB
		CDO
		FOR
		SCN
		SMI
		SPF
		SPFCALCP
20	CLOSE	CCB
		CDC
		FOR
		SCN
		SMD
		SPF
		SPFCALCP
21	STOW	CIPARMS
		EDO
		EMC
		UDMS
		UMCS
		URSS
24	DEVTYPE	CIV
		SMI
		SPF
		UAC
		UAR

(CONTINUED ON NEXT PAGE)

SVC ROUTINE / OBJECT MODULE LISTING (CONTINUED)

SVC NO.	SYSTEM SERVICE	OBJECT MODULE
26	CATALOG	UAC
26	INDEX	UC1
26	LOCATE	CBDSN
		CDAIR
		CDERR
		CT1
		UC1
26	UNCATALOG	CDAIR
		CT1
		CT2
		UAC
27	OBTAIN	CDO
		CIV
		UVT
29	SCRATCH	CT1
		CT2
30	RENAME	UAR
31	FEOV	CML
40	EXTRACT	CAT
		CDAIR
		CRESV
		PMD
42	ATTACH	CAT
		SMA
		SPFCALCP
		UC1
		UC2
		UDZ
44	CHAP	SMI
48	DEQ	CRELS
		ECR
		EMC
		EPO
		EPS
56	ENQ	CRESV
		ECR
		EMC
		EPO
		EPS
56	RESERVE	CRESV
60	STAE	CDO
		EDO
		SMI
62	DETACH	CAT
		SMA
		SPFCALCP
		UC1
		UC2
		UDZ
64	RDJFCB	CT1
		SMI
79	STATUS	SMD
93	TGET	CTGET
93	TPUT	CTPUT
		SMA
		SMI
94	GTERM	SMI
94	GTSIZE	PFT
		SMI
94	STCC	SMI
94	STFSMODE	CTGET
		CTPUT
		PFT
		SMC
		SMD
		SMI
94	STLINENO	SMA
		SMI
		SML
94	STTRAN	SMI
94	TCLEARQ	SMI
		SML
		SPFCALCP
96	STAX	SMD

OBJECT MODULE / LOAD MODULE REFERENCED LISTING

OBJECT MODULES WHICH ISSUE LINK, LOAD, OR ATTACH SYSTEM SERVICES ARE LISTED BY SYSTEM SERVICE AND THEN OBJECT MODULE. EACH OBJECT MODULE IS FOLLOWED BY THE NAMES OF THE LOAD MODULES IT REFERENCES VIA THE SYSTEM SERVICE.

SYSTEM SERVICE	OBJECT MODULE	LOAD MODULE REFERENCED
ATTACH	CAT	(COMMAND PROCESSOR PASSED BY CALLER) EXEC IKJEFT25 SPFCALCP
	SMA	SPFPMD
	SPFCALCP	(PROGRAM SPECIFIED IN CALL COMMAND)
	UC1	IEHLIST
	UC2	IDCAMS
	UDZ	IEBCOPY
LINK	CAT	IKJPARS IKJPTGT IKJPUTL IKJSCAN IKJSTCK
	CDAIR	IKJEFD00
	CHELP	SPFTUTOR
	CIPARMS	SPFSPC
	PMD	SPFBRO SPFEDIT SPFFOR SPFJOB SPFOPT SPFTMENU SPFTSO SPFTUTOR SPFUCA SPFUDA SPFUHC SPFUMC SPFUOL SPFURS SPFUSC SPFUTIL SPFUVT
	SPF	SPFMAIN
	SPFCALCP	IKJEFD00 IKJPARS IKJPUTL IKJSTCK
	UCA	SPFUC1 SPFUC2
	UTIL	SPFFOR SPFUCA SPFUDA SPFUHC SPFUMC SPFUOL SPFURS SPFUSC SPFUVT
LOAD	SMC	SPF3277 SPF3278 SPF3278C
	SMD	SPFSUBS SPFTBLS SPFTCM
	SMI	SPF3277 SPF3278 SPF3278C
	SPFCALCP	IKJEFF18

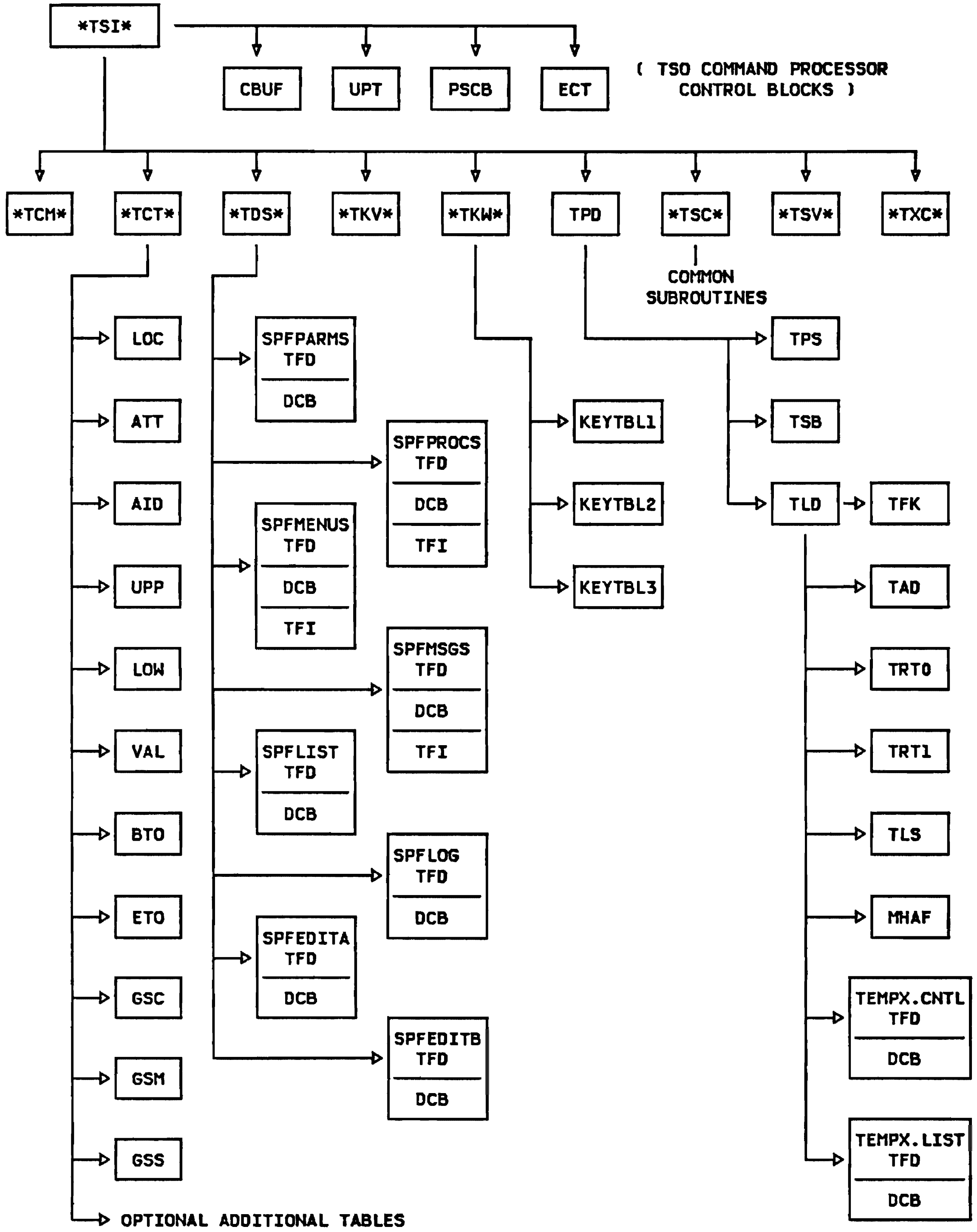
SECTION 5

DATA AREAS

THIS SECTION SHOWS THE BLOCKS, TABLES, AND COMMON AREAS THAT ARE USED IN SPF. THE FIRST TWO PAGES GIVE AN OVERVIEW OF THE PRIMARY SPF TABLES. THE OVERVIEW IS FOLLOWED BY A LIST OF ALL OF THE BLOCKS, TABLES, AND COMMON AREAS REFERENCED BY SPF AND A DESCRIPTION OF THOSE THAT ARE UNIQUE TO SPF. FINALLY, FORMATS ARE INCLUDED FOR COMPLEX AND/OR EXTENSIVELY USED SPF TABLES.

PRIMARY DATA AREAS DIAGRAM

THIS CHART SHOWS THE CENTRAL SPF TABLES AND THEIR RELATIONSHIPS. THOSE TABLES ENCLOSED IN ASTERISKS (*TCT*) OR A COPY OF THE TABLE ARE DIRECTLY ADDRESSABLE FROM THE TLD AS WELL AS BEING ADDRESSABLE AS SHOWN IN THE CHART.



PRIMARY DATA AREAS LIST

THE FOLLOWING LIST IDENTIFIES THE SPF TABLES THAT ARE IN THE PRECEEDING CHART. FOLLOWING THE NAME AND TITLE OF EACH TABLE IS THE NAME OF THE LOAD MODULE AND OBJECT MODULE WHERE THE TABLE EXISTS. LOAD MODULES SPFMAIN, SPFSUBS, SPFTCM AND SPF3277 ARE REENTRANT MODULES.

TSI	-	SPF INTERFACE	SPFTBLS - TSI
CBUF	-	COMMAND BUFFER	(PARAMETERS)
UPT	_	USER PROFILE TABLE	(PASSED)
PSCB	_	PROTECTED STEP CONTROL BLOCK	(FROM TMP)
ECT	_	ENVIRONMENTAL CONTROL TABLE	(TO SPF)
TCM	=	COMMAND TABLE	SPFTCM - TCM
TCT	-	CONTROL TABLES	SPFTBLS - TSI
LOC	-	3270 LOCATION TRANS	SPFMAIN - TRT
ATT	-	ATTRIBUTE BYTE TRANSLATE	SPFMAIN - TRT
AID	-	ATTENTION ID TRANS TBL	SPFMAIN - TRT
UPP	-	UPPER CASE INPUT TRANS TBL	SPF3277 - TT1 (*)
LOW	-	LOWER CASE INPUT TRANS TBL	SPF3277 - TT1 (*)
VAL	-	3270 TRANSLATE	SPF3277 - TT1 (*)
BTO	-	BROWSE TERM OUTPUT TRANS TBL	SPF3277 - TT1 (*)
ETO	-	EDIT TERM OUTPUT TRANS TBL	SPF3277 - TT1 (*)
GSC	-	GENERIC STRING CHARS TBL	SPF3277 - TT1 (*)
GSM	-	GENERIC STRING MASTER TBL	SPF3277 - TT1 (*)
GSS	-	GENERIC STRING SPECIAL TBL	SPF3277 - TT1 (*)
TDS	-	DATA SET INTERFACE	SPFTBLS - TSI
TFD	-	FILE DEF (SPFPARMS)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TFD	-	FILE DEF (SPFPROCS)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TFI	-	FIND MEMBER LIST	SPFTBLS - TSI
TFD	-	FILE DEF (SPFMENUS)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TFI	-	FIND MEMBER LIST	SPFTBLS - TSI
TFD	-	FILE DEF (SPFMSGS)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TFI	-	FIND MEMBER LIST	SPFTBLS - TSI
TFD	-	FILE DEF (SPFLIST)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TFD	-	FILE DEF (SPFLOG)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TFD	-	FILE DEF (SPFEDITA)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TFD	-	FILE DEF (SPFEDITB)	SPFTBLS - TSI
DCB	-	DCB	SPFTBLS - TSI
TKV	-	KEYWORD/VALUE TABLE	(GETMAINED)
TKW	-	KEYWORD TABLE	SPFMAIN - TKW
KEYTBL1	-	SCROLLING KEYWORDS	SPFMAIN - TKW
KEYTBL2	-	EDIT LINE COMMAND KEYWORDS	SPFMAIN - TKW
KEYTBL3	-	EDIT/BROWSE PRIMARY KEYWORDS	SPFMAIN - TKW
TPD	-	PHYSICAL DISPLAY	SPFTBLS - TSI
TPS	-	PHYSICAL SCREEN IMAGE	(GETMAINED)
TSB	-	SCREEN I/O BUFFER	(GETMAINED)
TLD (1)	-	LOGICAL DISPLAY	SPFTBLS - TSI
TFK (1)	-	FUNCTION KEY TABLE	SPFTBLS - TSI
TAD (1)	-	ALLOCATED DDNAME TABLE	SPFTBLS - TSI
TRT0 (1)	-	TRANSLATE AND TEST ZEROS	SPFTBLS - TSI
TRT1 (1)	-	TRANSLATE AND TEST IDENTITY	SPFTBLS - TSI
TLS (1)	-	LOGICAL SCREEN IMAGE	(THESE)
MHAF (1)	-	MENU HANDLER BUFFER	(TABLES)
TFD (1)	-	FILE DEF (TEMPX.CNTL)	(ARE)
DCB (1)	-	DCB	(GETMAINED)
TFD (1)	-	FILE DEF (TEMPX.LIST)	(AS)
DCB (1)	-	DCB	(REQUIRED)
TSC	-	SPF COMMON SUBROUTINE INTERFACE	SPFSUBS - TSC
TSV	-	SPF VARIABLES	SPFTBLS - TSI
TXC	-	SPF EXITS CONTROL TABLE	SPFTBLS - TSI

(1) MORE THAN ONE COPY CAN EXIST TO SATISFY SPLIT SCREEN REQUIREMENTS.
(*) USED FOR 3277 TERMINALS. FOR 3278 TERMINALS "SPF3278 - TT2" IS USED. FOR 3278 CANADIAN/FRENCH TERMINALS "SPF3278C - TT3" IS USED.

DATA AREAS LIST

THE FOLLOWING IS A LIST OF DATA AREAS REFERENCED BY SPF. THE TABLE NAME IS FOLLOWED BY A CODE INDICATING THE TYPE OF CONTROL BLOCK (O-OS/VS, T-TSO, S-SPF CONTROLLER, P-SPF PROCESSOR)

THE FORMAT DESCRIPTION, IS THE NAME OF THE PLS SEGMENT THAT DESCRIBES THE TABLE. THIS SEGMENT CAN BE FOUND IN PLS LISTINGS THAT REFERENCE THE TABLE. SOME TABLES, FOR EXAMPLE 256 BYTE TRANSLATE TABLES DO NOT HAVE A FORMAT DESCRIPTION. THE NAMES OF THE SOURCE MODULE, OBJECT MODULE, AND LOAD MODULE ARE SHOWN FOR TABLES WHICH ARE COMPILED.

AN ASTERISK (*) FOLLOWS TERMINAL DEPENDENT TABLES. THE TABLES USED FOR 3277 TERMINALS IS SHOWN. FOR 3278 TERMINALS REPLACE TT1 WITH TT2 AND REPLACE 3277 WITH 3278.

O - - -	OS/VS CONTROL BLOCKS
- T - -	TSO CONTROL BLOCKS
- - S -	SPF CONTROLLER TABLES
- - - P	SPF PROCESSOR TABLES

TABLE NAME	O T S P	FORMAT DESCRIPT	SOURCE MODULE	OBJECT MODULE	LOAD MODULE	TITLE
AID	- - S -	—	TRTAID	TRT	SPFMAIN	ATTENTION ID TRANSLATE TABLE
ASCB	O - - -	ASCBDCLS	—	—	—	ADDRESS SPACE CONTROL BLK
ASVT	O - - -	ASVTDCLS	—	—	—	ADDRESS SPACE VECTOR TABLE
ASXB	O - - -	ASXBDCLS	—	—	—	ADDRESS SPACE EXTENSION BLOCK
ATT	- - S -	—	TRTATT	TRT	SPFMAIN	ATTRIBUTE BYTE TRANSLATE TABLE
ATTACHL	O - - -	ATTDCLS	—	—	—	ATTACH MACRO LIST FORM
BCT	- - - P	BCTDCLS	—	—	—	BROWSE CONTROL TABLE
BLB	- - - P	BLBDCLS	—	—	—	BROWSE LINE BUFFER
BLDL	O - - -	BLDLDCLS	—	—	—	BLDL MACRO CONTROL BLOCK
BTO(*)	- - S P	—	TT1BTO	TT1	SPF3277	BROWSE TERMINAL OUTPUT TR TABLE
CBT	- - - P	CBTDCLS	—	—	—	COMMON BROWSE TABLE
CBUF	- T - -	—	—	—	—	TSO COMMAND BUFFER
CIVCOMM	- - - P	CIVCOMM	—	—	—	COMMON VTOC INFORMATION AREA
CMLCENT	- - - P	CMLDCLS	—	—	—	COMMON MEMBER LIST ENTRY
CSCB	O - - -	CSCBDCLS	—	—	—	COMMAND SCHEDULING CONTROL BLK
CSPL	- T - -	CSPLDCLS	—	—	—	COMMAND SCAN PARAMETER LIST
CVT	O - - -	CVTDCLS	—	—	—	COMMUNICATIONS VECTOR TABLE
DAIRACB	- T - -	DA34DCLS	—	—	—	DAIR ATTRIBUTE CONTROL BLOCK
DAPL	- T - -	DAPLDCLS	—	—	—	DAIR PARAMETER LIST
DA0C	- T - -	DA0CDCLS	—	—	—	DAIR DA0C BLOCK
DA00	- T - -	DA00DCLS	—	—	—	DAIR DA00 BLOCK
DA08	- T - -	DA08DCLS	—	—	—	DAIR DA08 BLOCK
DA10	- T - -	DA10DCLS	—	—	—	DAIR DA10 BLOCK
DA18	- T - -	DA18DCLS	—	—	—	DAIR DA18 BLOCK
DA2C	- T - -	DA2CDCLS	—	—	—	DAIR DA2C BLOCK
DA34	- T - -	DA34DCLS	—	—	—	DAIR DA34 BLOCK
DCB	O - - -	DCBDCLS	—	—	—	DATA CONTROL BLOCK
DEB	O - - -	DEBDCLS	—	—	—	DATA EXTENT BLOCK
DECB	O - - -	DECBDCLS	—	—	—	DATA EVENT CONTROL BLOCK
DEVT	O - - -	DEVTDCLS	—	—	—	DEVICE TABLE
DSCB	O - - -	DSCBDCLS	—	—	—	DATA SET CONTROL BLOCK
EBT	- - - P	EBTDCLS	—	—	—	EDIT BACKUP TABLE
ECB	O - - -	ECBDCLS	—	—	—	EVENT CONTROL BLOCK
ECT	- T - -	ECTDCLS	—	—	—	ENVIRONMENTAL CONTROL TABLE
EDL	- - - P	EDLDCLS	—	—	—	EDIT DISPLAY LINE
EDR	- - - P	EDRDCLS	—	—	—	EDIT RECORD
EDT	- - - P	EDTDCLS	—	—	—	EDIT TABLE
ELC	- - - P	ELCDCLS	—	—	—	EDIT LINE COMMANDS
ETO(*)	- - S P	—	TT1ETO	TT1	SPF3277	EDIT TERM OUTPUT TRANS TABLE
GSC(*)	- - S P	—	TT1GSC	TT1	SPF3277	GENERIC STRING CHARACTER TABLE
GSM(*)	- - S P	—	TT1GSM	TT1	SPF3277	GENERIC STRING MASTER TABLE
GSS(*)	- - S P	—	TT1GSS	TT1	SPF3277	GENERIC STRING SPECIAL CHAR TBL

(CONTINUED ON NEXT PAGE)

DATA AREAS LIST (CONTINUED)

TABLE NAME	O	T	S	P	FORMAT DESCRIPT	SOURCE MODULE	OBJECT MODULE	LOAD MODULE	TITLE
JSCB	O	-	-	-	JSCBDCLS	—	—	—	JOB STEP CONTROL BLOCK
KEYTBL1	-	-	S	P	—	KEYTBL1	TKW	SPFMAIN	KEYWORD TABLE 1 (SYSTEM)
KEYTBL2	-	-	S	P	—	KEYTBL2	TKW	SPFMAIN	KEYWORD TABLE 2 (EDIT LINE)
KEYTBL3	-	-	S	P	—	KEYTBL3	TKW	SPFMAIN	KEYWORD TABLE 3 (PRIMARY CMD)
KVBLOCK	-	-	S	P	—	—	—	—	KEYWORD/VALUE BLOCK
LOC	-	-	S	-	—	TRTLOC	TRT	SPFMAIN	3270 SCREEN LOCATION TRANS TBL
LOW(*)	-	-	S	P	—	TT1LOW	TT1	SPF3277	LOWER CASE TRANSLATE TABLE
LWA	O	-	-	-	LWADCLS	—	—	—	LOCAL WORK AREA
MHAF	-	-	-	P	MENUDCLS	—	—	—	MENU BUFFER
PSA	O	-	-	-	PSADCLS	—	—	—	PREFIX STORAGE AREA
PSAX	O	-	-	-	PSAXDCLS	—	—	—	PREFIX STORAGE AREA EXTENSIONS
PSCB	-	T	-	-	PSCBDCLS	—	—	—	PROTECTED STEP CONTROL BLOCK
RB	O	-	-	-	RBDCLS	—	—	—	REQUEST BLOCK
RBX	O	-	-	-	RBXDCLS	—	—	—	REQUEST BLOCK EXTENSION
SDE	-	-	S	P	SDEDCLS	—	—	—	SPF DIRECTORY ENTRY
SDWA	O	-	-	-	SDWADCLS	—	—	—	SYSTEM DIAGNOSTIC WORK AREA
STOW	O	-	-	-	STOWDCLS	—	—	—	STOW MACRO CONTROL BLOCK
TAD1	-	-	S	-	—	TLD1	TSI	SPFTBLS	ALLOCATED DDNAME TABLE FOR TLD1
TAD2	-	-	S	-	—	TLD2	TSI	SPFTBLS	ALLOCATED DDNAME TABLE FOR TLD2
TCB	O	-	-	-	TCBDCLS	—	—	—	TASK CONTROL BLOCK
TCD	-	-	-	P	TCDDCLS	—	—	—	COMMAND DEFINITION TABLE
TCM	-	-	S	P	TCMDCLS	TCM	TCM	SPFTCM	COMMAND TABLE
TCT	-	-	S	P	TCTDCLS	TCT	TSI	SPFSUBS	CONTROLLER TABLES ARRAY
TCS	-	-	-	P	TCSDCLS	—	—	—	COMMAND SCAN TABLE
TCST	-	T	-	-	TCSTDCLS	—	—	—	TERM CONTROL ADDR SPACE TABLE
TCX	O	-	-	-	TCXDCLS	—	—	—	TCAM ADDRESS VECTOR TABLE
TDS	-	-	S	-	TDSDCLS	TDS	TSI	SPFSUBS	DATA SET TABLE
TFD	-	-	S	P	TFDDCLS	—	—	—	FILE DEFINITION TABLE
TFI	-	-	S	P	TFIDCLS	—	—	—	FIND MEMBER TABLE
TFK	-	-	S	-	—	—	—	—	FUNCTION KEY TABLE
TIOT	O	-	-	-	TIOTDCLS	—	—	—	TASK INPUT/OUTPUT TABLE
TKV	-	-	S	P	—	TKV	TKV	SPFMAIN	INITIAL KEYWORD/VALUE TABLE
TKV	-	-	S	P	TKVDCLS	—	—	—	KEYWORD/VALUE TABLE
TKW	-	-	S	P	TKWDCLS	TKW	TKW	SPFMAIN	KEYWORD TABLE
TLD	-	-	S	P	TLDDCLS	—	—	—	LOGICAL DISPLAY TABLE
(TLD0)	-	-	S	P	—	TLD0	TSI	SPFTBLS	LOGICAL DISPL TBL (CONTROLLER)
(TLD1)	-	-	S	P	—	TLD1	TSI	SPFTBLS	LOGICAL DISPL TBL (SCREEN 1)
(TLD2)	-	-	S	P	—	TLD2	TSI	SPFTBLS	LOGICAL DISPL TBL (SCREEN 2)
TLS	-	-	S	P	TLSDCLS	—	—	—	LOGICAL SCREEN TABLE
(TLS1)	-	-	S	P	—	—	—	—	LOGICAL SCREEN TABLE FOR TLD1
(TLS2)	-	-	S	P	—	—	—	—	LOGICAL SCREEN TABLE FOR TLD2
TPD	-	-	S	-	TPDDCLS	TPD	TSI	SPFTBLS	PHYSICAL DISPLAY TABLE
TPS	-	-	S	-	TPSDCLS	—	—	—	PHYSICAL SCREEN TABLE
TRPT	O	-	-	-	TRPTDCLS	—	—	—	TIOC REFERENCE POINTER TABLE
TRT00	-	-	S	-	—	TLD0	TSI	SPFTBLS	ZEROS TRANS TABLE FOR TLD0
TRT01	-	-	-	P	—	TLD1	TSI	SPFTBLS	ZEROS TRANS TABLE FOR TLD1
TRT02	-	-	-	P	—	TLD2	TSI	SPFTBLS	ZEROS TRANS TABLE FOR TLD2
TRT10	-	-	S	-	—	TLD0	TSI	SPFTBLS	IDENTITY TRANS TABLE FOR TLD0
TRT11	-	-	-	P	—	TLD1	TSI	SPFTBLS	IDENTITY TRANS TABLE FOR TLD1
TRT12	-	-	-	P	—	TLD2	TSI	SPFTBLS	IDENTITY TRANS TABLE FOR TLD2
TSB	-	-	S	-	TSBDCLS	—	—	—	TPUT/TGET SCREEN BUFFER
TSC	-	-	S	P	TSCDCLS	TSC	TSC	SPFSUBS	COMMON SUBROUTINE ADDRESS TABLE
TSI	-	-	S	P	TSIDCLS	TSI	TSI	SPFTBLS	SPF INTERFACE TABLE
TSV	-	-	S	P	TSVDCLS	TSV	TSI	SPFTBLS	SPF VARIABLES TABLE
TSVKV	-	-	S	-	TSVKVDCL	—	—	—	TSV KEYWORD/VALUE TABLE
TTT	-	-	S	-	TTTDCLS	—	—	—	TERM DEPENDENT TRANS TABLES
TXC	-	-	S	P	TXCDCLS	TXC	TSI	SPFTBLS	SPF EXITS CONTROL TABLE
UCB	O	-	-	-	UCBDCLS	—	—	—	UNIT CONTROL BLOCK
UDACOMM	-	-	-	P	UDACOMM	—	—	—	DATASET UTILITY COMMON AREA
UMCCOMM	-	-	-	P	UMC@COMM	—	—	—	MOVE/COPY UTILITY COMMON AREA
UPP(*)	-	-	S	P	—	TT1UPP	TT1	SPF3277	UPPER CASE TRANSLATE TABLE
UPT	-	T	-	-	UPTDCLS	—	—	—	TSO USER PROFILE TABLE
URSSPARM	-	-	-	P	URS@COMM	—	—	—	RESET UTILITY COMMON AREA
VAL(*)	-	-	S	P	—	TT1VAL	TT1	SPF3277	VALID CHARACTER TRT TABLE

DATA AREAS DESCRIPTIONS

AID **ATTENTION ID TRANSLATE TABLE**

USED TO TRANSLATE 3270 AID'S (ATTENTION IDENTIFIERS) INTO SPF LOGICAL AID'S. FOR EXAMPLE, THE 3270 AID OF X'7C' (PROGRAM FUNCTION KEY 12) IS TRANSLATED TO SPF LOGICAL AID OF 12. THE AID IS USED IN CONJUNCTION WITH THE TFK TO ASSIGN SPECIFIC SPF FUNCTIONS TO SPECIFIC 3270 PF KEYS.

ATT **ATTRIBUTE BYTE TRANSLATE TABLE**

USED TO TRANSLATE FROM SPF LOGICAL ATTRIBUTE BYTES INTO 3270 ATTRIBUTE BYTES. SPF LOGICAL ATTRIBUTE BYTES ARE DEFINED IN TLSDCLS (SEE TLS DESCRIPTION) AND ARE STORED AS CHARACTERS IN THE TLS AND TPS. THEY ARE ALL LESS THAN X'40' AND ARE TRANSLATED TO 3270 ATTRIBUTE BYTES WHEN DATA IS FORMATTED FOR OUTPUT TO THE DISPLAY.

BCT **BROWSE CONTROL TABLE**

USED BY COMMON BROWSE TO REPRESENTS 8000 LOGICAL RECORDS FROM THE DATA SET BEING BROWSED. ONE BCT IS CREATED INITIALLY. AS MORE RECORDS ARE READ, ADDITIONAL BCT'S ARE ALLOCATED, INITIALIZED, AND CHAINED. THE PURPOSE OF THE BCT TABLES IS TO ALLOW COMPUTATION OF THE RELATIVE 'TTRN' FOR EACH RELATIVE RECORD IN THE DATA SET (EVEN FOR VARIABLE BLOCKED RECORDS). BITS IN THE TRACK BIT TABLE AND THE RECORD BIT TABLE ARE SET THE FIRST TIME THAT A LOGICAL RECORD IS READ, BASED ON THE TTRN THAT IS RETURNED FROM CGET. ONCE A RECORD HAS BEEN READ, THE TABLES ARE USED TO COMPUTE A RELATIVE TTRN FOR THE RECORD.

BLB **BROWSE LINE BUFFER**

USED BY COMMON BROWSE TO HOLD DATA FROM ONE DATA RECORD (WHICH CORRESPONDS TO ONE LINE ON THE DISPLAY). DATA IS MOVED FROM A BLB WHEN THE SCREEN IS BEING FORMATTED. BLB'S ARE GETMAINED FROM SUBPOOL 4 (AS A BLOCK) AND FORWARD AND BACKWARD CHAINED AS PART OF CBR INITILIZATION. THEY ARE FREEMAINED DURING CBR TERMINATION.

BTO **BROWSE TERMINAL OUTPUT TRANSLATE TABLE**

USED BY COMMON BROWSE TO TRANSLATE DATA AS IS IS PUT IN THE TLS FOR DISPLAY. INVALID CHARACTERS ARE TRANSLATED TO PERIODS '.' BY THIS TRANSLATION. ONE BTO IS USED FOR 3277 TERMINALS AND A DIFFERENT BTO IS USED FOR 3278 TERMINALS.

CBT **COMMON BROWSE TABLE**

USED TO COMMUNICATE INFORMATION BETWEEN THE PROGRAMS THAT MAKE UP COMMON BROWSE AND TO RETAIN INFORMATION FOR COMMON BROWSE BETWEEN CALLS TO IT (FOR EXAMPLE BETWEEN MEMBERS SELECTED FROM A MEMBER LIST). AN EXAMPLE OF THE INFORMATION SAVED IS THE CURRENT "FIND" COMMAND. THE CBT IS ALLOCATED AND INITIALIZED BY CBS (COMMON BROWSE SETUP) EITHER BEFORE CBR (COMMON BROWSE) IS CALLED, OR BY CBR ITSELF IF NECESSARY. IT IS FREED BY CBC (COMMON BROWSE CLEANUP).

CIVCOMM **COMMON VTOC INFORMATION AREA**

USED TO PASS DATA SET INFORMATION FROM CIV BACK TO ROUTINES WHICH CALL CIV. THE AREA IS PASSED TO CIV AS A PARAMETER.

CMLCENT COMMON MEMBER LIST ENTRY

COMMON MEMBER LIST ENTRIES ARE GENERATED AND CHAINED TOGETHER BY CML WHENEVER A LIST OF PARTITIONED DATA SET MEMBERS IS NEEDED, EITHER FOR A MEMBER LIST DISPLAY, OR FOR INTERNAL PROCESSING. THE ENTRIES ARE CHAINED OFF OF THE TFDCML FIELD OF A FILE DEFINITION TABLE (TFD). THE FIRST ENTRY ON THE CHAIN ALWAYS CONTAINS A NAME FIELD OF BINARY ZEROS, AND THE LAST ENTRY ON THE CHAIN ALWAYS CONTAINS A NAME FIELD OF ALL BINARY ONES. EACH CHAIN ENTRY CONSISTS OF:

- A BYTE CONTAINING STATUS BITS,
- A POINTER TO THE NEXT ENTRY,
- THE 8 CHARACTER MEMBER NAME
- 3 BYTES OF TTR INFORMATION.

EBT EDIT BACKUP TABLE

USED IN CONTROLLING EDIT BACKUP AND RECOVERY. THE EBT IS AN EXTENSION TO THE EDT (EDIT TABLE) AND IS ADDRESSED FROM THE EDT.

EDL EDIT DISPLAY LINE

USED TO DESCRIBE THE EDIT DISPLAY LINE AS IT APPEARS ON THE DISPLAY SCREEN, WITH THE COMMAND AREA ON THE LEFT, AND THE DATA AREA ON THE RIGHT.

EDR EDIT RECORD

USED TO HOLD CONTROL INFORMATION AND DATA FOR ONE EDIT RECORD. THERE IS ONE EDIT RECORD FOR EACH LINE OF DATA THAT IS BEING EDITED, AND FOR EACH SPECIAL LINE (SUCH AS TOP, BOTTOM, MASK, TABS LINES ETC.) EDIT RECORDS ARE INITIALLY ALLOCATED WHEN DATA IS READ IN BY EDO, AND ADDITIONAL EDIT RECORDS ARE ALLOCATED AS REQUIRED. EDR'S ARE FORWARD AND BACKWARD CHAINED ON THE EDIT RECORD CHAIN.

EDT EDIT TABLE

THE EDIT TABLE (EDT) IS THE MAIN COMMUNICATION AREA FOR DATA PASSED AMONG THE VARIOUS EDIT SUBROUTINES. IT IS DYNAMICALLY ALLOCATED AND INITIALIZATED BY ETS (EDIT TABLE SETUP) AND DELETED BY ETC (EDIT TABLE CLEANUP). IT IS ADDRESSED BY REGISTER 3 IN ALL EDIT ROUTINES (AND THUS DOES NOT HAVE TO BE PASSED AS A PARAMETER WHEN CALLING EDIT ROUTINES).

ELC EDIT LINE COMMANDS

USED TO DESCRIBE AN EDIT LINE COMMAND DEFINITION. THE COMMAND DEFINITIONS THEMSELVES ARE IN THE MODULE ETL (EDIT LINE COMMAND TABLE.)

ETO EDIT TERM OUTPUT TRANS TABLE

USED BY EDIT TO TRANSLATE DATA AS IT IS PUT INTO THE TLS FOR DISPLAY. INVALID CHARACTERS ARE TRANSLATED TO ATTRIBUTE BYTES BY THIS TRANSLATION. ONE ETO IS USED FOR 3277 TERMINALS AND A DIFFERENT ETO IS USED FOR 3278 TERMINALS.

GSC GENERIC STRING CHARACTER TABLE

USED TO DEFINE THE BIT MASK ASSOCIATED WITH A PARTICULAR SPECIAL CHARACTER. THE BIT MASK IDENTIFIES THE TYPE(S) OF CHARACTERS THAT ARE TO BE REPRESENTED BY THE SPECIAL CHARACTER. A CODE FROM THE GSS TABLE IS USED AS AN INDEX INTO THIS TABLE. THIS TABLE CONSISTS OF BIT MASKS THAT ARE MATCHED AGAINST BYTES IN THE GSM TO SEE IF A PARTICULAR CHARACTER FITS INTO A PARTICULAR CATAGORY.

GSM GENERIC STRING MASTER TABLE

USED TO ASSIGN TO EACH POSSIBLE CHARACTER, ONE OR MORE BITS WHICH DESCRIBES THE CHARACTERS. POSSIBILITIES INCLUDE ALPHA, NUMERIC, SPECIAL, INVALID, ETC. THE TABLE IS USED PRIMARILY IN PROCESSING PICTURE STRINGS IN BROWSE AND EDIT, BUT CAN BE USED AS A TRANSLATE TABLE TO DETERMINE WHETHER DATA CONTAINS INVALID OR LOWER CASE CHARACTERS (EDIT USES THE GSM FOR THIS FUNCTION).

GSS GENERIC STRING SPECIAL CHAR TBL

USED IN PROCESSING PICTURE STRINGS WITH THE TR (TRANSLATE) AND TRT (TRANSLATE AND TEST) INSTRUCTIONS.

- ALL ALPHABETIC/NUMERIC CHARACTERS TRANSLATE INTO THEMSELVES.
- SPECIAL PICTURE STRING CHARACTERS TRANSLATE INTO CODES WHICH ARE USED AS AN INDEX INTO THE GSC TABLE. IN THE TABLES THAT ARE DISTRIBUTED, SPECIAL CHARACTERS ARE "=", ".", "<", ">", "#", "$", "@", "¬" AND "-".
- ALL INVALID CHARACTERS AND SPECIAL CHARACTERS THAT ARE NOT DEFINED AS BEING SPECIAL CHARACTERS TRANSLATE INTO X'FF'.

KEYTBL1 KEYWORD TABLE 1 (SYSTEM)

USED TO ASSIGN 1 BYTE INTERNAL CODES TO 1 TO 8 BYTE CHARACTER SYMBOLS. TABLE 1 CONTAINS SYSTEM WIDE SYMBOLS. SEE ALSO TKW.

KEYTBL2 KEYWORD TABLE 2 (EDIT LINE)

USED TO ASSIGN 1 BYTE INTERNAL CODES TO 1 TO 8 BYTE CHARACTER SYMBOLS. TABLE 2 CONTAINS EDIT LINE COMMAND SYMBOLS. SEE ALSO TKW.

KEYTBL3 KEYWORD TABLE 3 (PRIMARY CMD)

USED TO ASSIGN 1 BYTE INTERNAL CODES TO 1 TO 8 BYTE CHARACTER SYMBOLS. TABLE 3 CONTAINS BROWSE AND EDIT PRIMARY COMMAND SYMBOLS. SEE ALSO TKW.

KVBLOCK KEYWORD/VALUE BLOCK

USED TO DESCRIBE THE KEYWORDS AND VALUES THAT ARE PROCESSED BY THE CKVGET AND CKVPUT COMMON SUBROUTINES. IT CONTAINS A LIST OF VARIABLE LENGTH ENTRIES, ONE FOR EACH KEYWORD. EACH ENTRY CONSISTS OF THE FOLLOWING ITEMS IN ORDER:

- VALUE LENGTH - FIXED(8)
- KEYWORD LENGTH - FIXED(8)
- KEYWORD - CHARACTER(KEYWORD LENGTH)

THE END OF THE LIST IS INDICATED BY A BYTE SET TO '00'X.
SEE ALSO TKV.

LOC **3270 SCREEN LOCATION TRANS TBL**

USED TO ASSIST IN TRANSLATING FROM RELATIVE LOCATIONS ON A 3270 DISPLAY SCREEN TO AN EBCDIC LOCATION AS USED IN 3270 ORDERS.

LOW **LOWER CASE TRANSLATE TABLE**

USED TO TRANSLATE DATA THAT IS READ IN FROM THE TERMINAL. THE ONLY CHARACTERS ACTUALLY CHANGED BY THIS TRANSLATION ARE THE 3270 DUP, FIELD MARK, AND GRAPHIC ESCAPE CHARACTERS WHICH ARE CHANGED TO BLANK.

MHAF **MENU BUFFER**

THE MENU HANDLER BUFFER IS SET UP BY MHA AND AND IS SAVED AND RESTORED BY CHELP. INFORMATION IN THIS AREA IS PRESERVED ACROSS CALLS TO MHA. MENUACTN - (MENU ACTION ENTRY) IS A DSECT THAT DESCRIBES MENU HANDLER ACTION STATEMENTS IN THE MHA BUFFER. THERE IS ONE MHAFACTN ENTRY FOR EACH ACTION STATEMENT

SDE **SPF DIRECTORY ENTRY**

USED TO HOLD INFORMATION THAT IS COLLECTED AND MAINTAINED BY SPF ABOUT A PDS MEMBER. THE SDE IS KEPT IN THE USER AREA OF A PARTITIONED DATA SET DIRECTORY ENTRY.

TAD **ALLOCATED DDNAME TABLE**

USED TO HOLD THE NAMES OF ALLOCATED DDNAMES. WHEN CDAIR ALLOCATES A FILE, THE DDNAME IS ADDED TO THE TAD, AND WHEN IT FREE THE FILE, THE NAME IS REMOVED. THE PURPOSE OF THE TABLE IS TO ALLOW FILES TO BE FREED AT TASK TERMINATION TIME IF THEY HAVE NOT BEEN PREVIOUSLY FREED. THIS SITUATION WILL OCCUR IF THE TASK TERMINATES ABNORMALLY.

TCD **COMMAND DEFINITION TABLE**

USED TO DESCRIBE PRIMARY COMMAND USED IN BROWSE AND EDIT. THE COMMAND DEFINITIONS (WHICH ARE COMPILED INTO MODULES ECD - EDIT, AND BCD - BROWSE) ARE USED BY CCP (COMMON COMMAND PARSE) TO CATCH CERTAIN COMMAND SYNTAX OR PARAMETER ERRORS AND TO REORDER THE COMMAND PARAMETERS IN THE TCS (COMMAND SCAN TABLE) SO THAT LATER PROCESSING OF THE COMMAND IS MADE EASIER.

TCM **COMMAND TABLE**

THE COMMAND TABLE (TCM) IS USED TO DESCRIBE COMMANDS TO BE INVOKED UNDER SPF VIA THE COMMMON ATTACH ROUTINE (CAT). COMMANDS MAY BE CLASSIFIED IN ONE OF FOUR WAYS AS SPECIFIED BY THE TCM TYPE FIELD DESCRIBED BELOW. SPF LOOKS UP EACH COMMAND IN THE TCM BEFORE INVOKING THE COMMAND. ANY COMMAND NAME NOT FOUND IN THE TABLE WILL BE TREATED AS INDICATED BY THE FINAL ENTRY IN THE TCM. USE OF THE TCM IMPROVES SYSTEM PERFORMANCE BY ELIMINATING THE OVERHEAD OF SEARCHING LINKLIB FOR COMMANDS THAT ARE IN LPA AND FLAGGED AS COMMAND PROCESSORS, COMMANDS THAT ARE FLAGGED AS CLISTS, OR COMMANDS THAT ARE FLAGGED AS INVALID. IF IT IS UNKNOWN WHETHER A GIVEN COMMAND NAME IS A COMMAND PROCESSOR OR A CLIST, THE ENTRY MAY BE FLAGGED AS A "BLDL" TYPE TO CAUSE SPF TO SEARCH THE LINKLIST USING BLDL. THE TCM IS ASSEMBLED AND LINKEDITED BY ITSELF. THIS MAKES IT POSSIBLE TO CREATE A TAILORED VERSION OF TCM FOR A SUBSET OF USERS, THUS RESTRICTING THE COMMANDS THAT ARE AVAILABLE TO THEM. THESE USERS WOULD HAVE A SEPARATE LOGON PROCEDURE WHICH WOULD INCLUDE A STEPLIB CONTAINING THE MODIFIED TCM.

TCT **CONTROLLER TABLES ARRAY**

THE CONTROLLER TABLES ARRAY (TCT) CONTAINS THE ADDRESSES OF 13 TABLES THAT ARE USED BY THE CONTROLLER TASK. THESE TABLES ARE USED FOR TRANSLATE AND TRANSLATE-AND-TEST FUNCTIONS. THE FIRST FIVE TABLE ADDRESSES POINT TO THE STATIC TRANSLATE TABLES (TRT). SEE THE TRT OBJECT MODULE DESCRIPTION. THE LAST EIGHT TABLE ADDRESSES POINT TO THE TERMINAL DEPENDENT TABLES (TTT). SEE THE TT1 OR TT2 OBJECT MODULE DESCRIPTION.

TCS **COMMAND SCAN TABLE**

IS CREATED BY CCS (COMMON COMMAND SCAN) IN PROCESSING BROWSE AND EDIT PRIMARY COMMANDS. THE TCS IS AN ARRAY OF ENTRIES, EACH REPRESENTING ONE PARAMETER IN THE COMMAND.

TDS **DATA SET TABLE**

THE SPF DATA SETS TABLE CONSISTS OF POINTERS TO EIGHT SPF TFD'S THAT ARE USED THROUGHOUT SPF. THE TDS AND THE EIGHT TFD'S ALONG WITH THE DCB'S FOR THE EIGHT DATA SETS ARE ALL COMPILED INTO THE TSI OBJECT MODULE WHICH IS LINK EDITED INTO THE SPFTBLS LOAD MODULE.

TFD **FILE DEFINITION TABLE**

USED TO COMMUNICATE FILE DEFINITION INFORMATION BETWEEN COMMON SUBROUTINES. SOME TFD'S ARE COMPILED. FOR EXAMPLE, THE TFD'S FOR THE MENUS, MSGS, PROCS AND PARMS FILES. MOST ARE CREATED DYNAMICALLY, INITIALIZED WITH ZEROS, AND FILLED WITH APPRIOPRIATE INFORMATION. TFD'S ARE PASSED TO COMMON ALLOCATE FOR ALLOCATING A FILE, COMMON OPEN FOR OPENING IT, COMMON GET TO READ A RECORD, COMMON PUT TO WRITE A RECORD, COMMON CLOSE TO CLOSE THE FILE, AND FINALLY COMMON FREE TO FREE THE FILE.

TFI **FIND MEMBER TABLE**

USED TO CONTAIN BLDL INFORMATION IN MAIN MEMORY SO THAT THE PDS DIRECTORY ON DISK NEED NOT BE REFERENCED EACH TIME THAT A MEMBER IS TO BE READ. THERE IS A TFI TABLE FOR THE SPFMENUS DATASET, FOR THE SPFMSGS DATASET, AND FOR THE SPFPROCS DATASET. THE TFI IS ADDRESSED BY A FIELD IN THE TFD FOR THE DATA SET. A TFI TABLE IS MADE UP OF TWO BLDL LISTS (LISTS THAT CAN BE PASSED TO THE BLDL MACRO TO READ A PDS DIRECTORY ENTRY). THE FIRST LIST IS FOR ONE MEMBER, AND IS USED WHEN A MEMBER IS SELECTED THAT IS NOT IN THE SECOND LIST. THE SECOND LIST CONTAINS COMMONLY USED MEMBERS. THE FIRST TIME THAT A FIND IS REQUESTED (BY CALLING CFI), A BLDL IS DONE FOR THE SECOND LIST. THEREAFTER, WHEN A FIND IS REQUESTED, THE SECOND LIST IS SCANNED, AND IF THE MEMBER NAME IS FOUND, NO I/O TO THE DATASET DIRECTORY IS REQUIRED. IF A MEMBER IS NOT FOUND, FOR EXAMPLE TUTORIAL MEMBERS ARE NOT INCLUDED IN THE SECOND LIST, A SINGLE MEMBER BLDL IS DONE INTO THE FIRST (SINGLE MEMBER) LIST.

TFK **FUNCTION KEY TABLE**

USED IN TRANSLATING FROM A 3270 ATTENTION ID, INTO AN SPF PROGRAM FUNCTION CODE. THE CODE IN TURN IS USED IN CONJUNCTION WITH BITS IN THE TLD TO DETERMINE WHETHER OR NOT THE FUNCTION IS ENABLED OR NOT.

TKV KEYWORD/VALUE TABLE

THE KEYWORD/VALUE TABLE IS USED TO REMEMBER USER PARAMETERS DURING AN SPF SESSION. THE TKV IS SAVED IN THE SPFPARMS DATA SET FROM SESSION TO SESSION. FOR A NEW USER, THE TKV IS INITIALIZED FROM OBJECT MODULE TKV, WHICH IS IN THE SPFMAIN LOAD MODULE. DURING THE SESSION, TKV ENTRIES MAY BE RETRIEVED, UPDATED, ADDED OR DELETED VIA THE CKVGET AND CKVPUT COMMON SUBROUTINES. THE TKV CONSISTS OF THREE PARTS: THE HEADER, THE FIXED SECTION AND THE VARIABLE SECTION. ENTRIES IN THE FIXED SECTION ARE NEVER DELETED FROM THE TABLE. ENTRIES IN THE VARIABLE SECTION ARE DELETED BY CKVPUT IF THE VALUE BECOMES BLANK. THE MAXIMUM LENGTH OF THE TKV IS DETERMINED FROM THE BLKSIZE OF THE SPFPARMS DATA SET. INSTALLATIONS THAT REQUIRE A LARGER TKV THAN THAT SPECIFIED IN THE SPF INSTALLATION PROCEDURE MAY CREATE AN SPFPARMS DATA SET WITH A LARGER BLKSIZE. THIS MAY BE NECESSARY IF THE INSTALLATION ADDS ADDITIONAL KEYWORDS TO THE BACKGROUND AND FOREGROUND PROCS AND MENUS.

TKW KEYWORD TABLE

THERE ARE THREE KEYWORD TABLES, KEYTBL1, KEYTBL2, AND KEYTBL3. THEY ARE USED TO ASSOCIATE WORDS OF 1 TO 8 CHARACTERS WITH INTERNAL CODES. THEY ALLOW SPF PROGRAMS TO BE CODED INDEPENDENT OF ACTUAL COMMANDS AND KEYWORDS THAT ARE ENTERED BY AN SPF USER. THEY ALSO IMPROVE THE INTERNAL EFFICIENCY OF SPF SINCE A SINGLE ONE BYTE INTERNAL CODE CAN BE USED INSTEAD OF SEVERAL CHARACTER LITERALS. AND THEY PERMIT THE KEYWORDS TO BE EASILY CHANGED WITHOUT CHANGING ANY PROGRAMS. TWO INTERNAL PROCEDURES ARE USED TO REFERENCE ENTRIES IN THE KEYWORD TABLES.

SYSCODE - IS USED TO RETRIEVE A CODE FROM A KEYWORD TABLE WHEN THE WORD IS KNOWN

SYSWORD - IS USED TO RETRIEVE A WORD FROM A KEYWORD TABLE WHEN ITS INTERNAL CODE IS KNOWN.

THE KEYWORD SCHEME ALLOWS MORE THAN ONE KEYWORD TO BE ASSOCIATED WITH A CODE. FOR EXAMPLE, THE WORDS C, CHG, AND CHANGE, USED AS EDIT PRIMARY COMMANDS ALL HAVE THE SAME INTERNAL CODE. THE EDIT PROGRAM AFTER CALLING SYSCODE TO TRANSFORM THE COMMAND INTO AN INTERNAL CODE NEED ONLY CHECK FOR A SINGLE CODE TO DETERMINE IF ANY FORM OF THE CHANGE COMMAND WAS ENTERED. THE KEYWORD TABLES ARE ASSEMBLED TOGETHER IN THE TKW. THEY ARE ADDRESSED ONLY BY THE SYSCODE AND SYSWORD ROUTINES. AN ENTRY IN THE TLD CONTAINS THE ADDRESS OF THE TKW WHICH CONTAINS THE ADDRESSES OF THE THREE TABLES.

TLD LOGICAL DISPLAY TABLE

USED TO CONTAIN INFORMATION ASSOCIATED WITH ONE LOGICAL DISPLAY (AND WITH ONE OS TASK). THERE ARE TWO LOGICAL DISPLAY TABLES, ONE FOR EACH LOGICAL SCREEN. SOME OF THE INFORMATION IS PASSED BETWEEN THE CONTROLLER AND A PROCESSOR AND OTHER DATA IS USED EITHER WITHIN THE CONTROLLER OR WITHIN A PROCESSOR.

TLS LOGICAL SCREEN TABLE

THE LOGICAL SCREEN IMAGE IS A 1920, 2560 OR 3440 BYTE AREA THAT IS FORMATTED EXACTLY AS THE FORMAT DISPLAY IS TO APPEAR. WHERE ATTRIBUTE BYTES ARE TO BE LOCATED, LOGICAL SPF ATTRIBUTE INTERNAL CODES MUST BE USED. THE INTERNAL ATTRIBUTE CODES THAT ARE SUPPORTED BY SPF ARE:

SYMBOL	BINARY	HEX	DESCRIPTION
TLSON	'00000100'B	'04'X	OUTPUT NON-DISPLAY
TLSOL	'00000101'B	'05'X	OUTPUT LOW INTENSITY
TLSOH	'00000111'B	'07'X	OUTPUT HIGH INTENSITY
TLSIAN	'00010000'B	'10'X	INPUT ASIS NON-DISPLAY
TLSIAL	'00010001'B	'11'X	INPUT ASIS LOW INTENSITY
TLSIAH	'00010011'B	'13'X	INPUT ASIS HIGH INTENSITY
TLSIBN	'00010100'B	'14'X	INPUT CAPS (BLANK) NON-DISPLAY
TLSIBL	'00010101'B	'15'X	INPUT CAPS (BLANK) LOW INTENS
TLSIBH	'00010111'B	'17'X	INPUT CAPS (BLANK) HIGH INTENS

TPD PHYSICAL DISPLAY TABLE

THE PHYSICAL DISPLAY TABLE (TPD) CONSISTS OF VALUES AND POINTERS TO OTHER TABLES THAT ARE USED IN MANAGING THE PHYSICAL DISPLAY. THE TPD IS COMPILED INTO THE TSI OBJECT MODULE WHICH IS LINK EDITED INTO LOAD MODULE SPFTBLS.

TPS PHYSICAL SCREEN TABLE

THE PHYSICAL SCREEN IS AN IMAGE OF THE SCREEN THAT THE USER IS VIEWING. DATA IS MERGED INTO THE TPS FROM THE TLS (IN SINGLE SCREEN MODE, OR FROM BOTH TLS'S (IN SPLIT SCREEN MODE).

TRT0 ZEROS TRANS TABLE

USED AS A GENERAL PURPOSE 256 BYTE TABLE INITIALIZED TO ZEROS. THE TRT0 IS ASSOCIATED WITH A TLD AND THUS WITH A SINGLE TASK. IT IS USED PRIMARILY WHEN A PROGRAM WANTS TO CHANGE ONE OR A FEW BYTES IN THE TABLE, AND THEN DO TRT (TRANSLATE AND TEST) INSTRUCTIONS TO SCAN FOR PARTICULAR CHARACTERS. EACH PROGRAM USING THE TRT0 IS RESPONSIBLE FOR RESTORING IT TO ITS INITIAL STATE (ALL ZEROS) BEFORE CALLING OTHER PROGRAMS OR RETURNING.

TRT1 IDENTITY TRANS TABLE

USED AS A GENERAL PURPOSE 256 BYTE TABLE. EACH BYTE CONTAINS ITS OWN VALUE (I.E. BYTE 0 IS X'00', BYTE 1 IS X'01', BYTE 255 IS X'FF', ETC). THE TRT1 IS ASSOCIATED WITH A TLD AND THUS WITH A SINGLE TASK. IT IS USED PRIMARILY WHEN A PROGRAM WANTS TO TRANSLATE DATA. BYTES IN THE TABLE CAN BE CHANGED AND THEN TRT (TRANSLATE AND TEST) INSTRUCTIONS CAN BE DONE TO TRANSLATE PARTICULAR CHARACTERS TO OTHER CHARACTERS. EACH PROGRAM USING THE TRT1 IS RESPONSIBLE FOR RESTORING IT TO ITS INITIAL STATE (IDENTITY) BEFORE CALLING OTHER PROGRAMS OR RETURNING.

TSB TPUT/TGET SCREEN BUFFER

TSB IS THE BUFFER USED FOR BOTH TPUT AND TGET.

TSC COMMON SUBROUTINE ADDRESS TABLE

TSC IS INCLUDED IN THE SPFSUBS LOAD MODULE AND SERVES AS AN INTERFACE TO THE COMMON SUBROUTINES THAT ARE INCLUDED IN THAT LOAD MODULE. THE BEGINNING OF THE TSC SERVES AS THE ENTRY POINT OF THE SPFSUBS LOAD MODULE. WHEN SPFSUBS IS LOADED, ITS ENTRY POINT, AND THUS THE ADDRESS OF THE TSC, IS STORED IN THE TSI.

TSI SPF INTERFACE TABLE

THE SPF INTERFACE TABLE (TSI) IS THE CENTRAL INTERFACE POINT FOR SPF. IT CONTAINS POINTERS TO OTHER SIGNIFICANT TABLES AND IN TURN IS POINTED AT BY THE LOGICAL DISPLAY TABLES (TLD). TSI IS COMPILED AS PART OF THE TSI OBJECT MODULE, AND IS LINK EDITED AS PART OF SPFTBLS. ITS INITIALIZATION IS COMPLETED BY SPF MAIN INITIALIZATION (SMI).

TSV SPF VARIABLES TABLE

THE SPF VARIABLES TABLE CONTAINS PARAMETERS, CODES, AND VARIABLES THAT ARE USED THROUGHOUT SPF. DURING SPF INITIALIZATION, PORTIONS OF THE TSV ARE OVERLAID WITH INFORMATION FROM THE SPFPARMS DATA SET (IF A MEMBER EXISTS FOR THIS USER).

TTT TERM DEPENDENT TRANS TABLES

IS AN ADDRESS ARRAY CONTAINING POINTERS TO TERMINAL DEPENDENT TRANSLATE TABLES. ITS ADDRESS IS OBTAINED FROM THE ENTRY POINT OF THE TERMINAL DEPENDENT LOAD MODULES (SPF3277,SPF3278, OR SPF3278C). THE ADDRESSES OF INDIVIDUAL TABLES ARE MOVED TO THE TCT. THEY ARE NOT REFERENCED DIRECTLY FROM THE "TTT".

TXC SPF EXITS CONTROL TABLE

THE TXC IS USED FOR COMMUNICATE WITH THE SPF EXIT ROUTINES THAT OPERATE AS PART OF SVC 93 AND SVC 94.

UDACOMM DATASET UTILITY COMMON AREA

USED TO PASS INFORMATION BETWEEN UDA AND ITS SUBROUTINES.

UMCCOMM MOVE/COPY UTILITY COMMON AREA

USED TO PASS INFORMATION BETWEEN UMC AND ITS SUBROUTINES.

UPP UPPER CASE TRANSLATE TABLE

USED TO TRANSLATE LOWER CASE CHARACTERS TO UPPER CASE. THIS TABLE IS USED TO TRANSLATE DATA THAT IS READ IN FROM THE TERMINAL (INSTEAD OF TABLE LOW), IF THE SPF ATTRIBUTE BYTE FOR THE DISPLAY FIELD INDICATES THAT TRANSLATION TO UPPER CASE IS TO BE DONE. IN ADDITION TO TRANSLATING LOWER CASE ALPHABETIC CHARACTERS, THE 3270 DUP, FIELD MARK, AND GRAPHIC ESCAPE CHARACTERS ARE CHANGED TO BLANK.

URSSPARM RESET UTILITY COMMON AREA

USED TO PASS INFORMATION BETWEEN URS AND ITS SUBROUTINES.

VAL VALID CHARACTER TRT TABLE

USED TO DETERMINE WHETHER DATA CONTAINS CHARACTERS THAT ARE INVALID FOR THE TERMINAL CURRENTLY IN USE. VAL CONTAINS AN X'FF' IN EACH CHARACTER POSITION THAT CORRESPONDS TO AN INVALID CHARACTER AND X'00' FOR EACH VALID CHARACTER. IT CAN BE USED WITH THE TRANSLATE AND TEST (TRT) INSTRUCTION TO SCAN A CHARACTER STRING FOR AN INVALID CHARACTER. ONE VAL IS USED FOR 3277 TERMINALS AND A DIFFERENT VAL IS USED FOR 3278 TERMINALS.

SPF TABLES FORMATS

THE PAGES THAT FOLLOW CONTAIN THE FORMATS OF BLOCKS, TABLES, AND COMMON AREAS THAT ARE COMPLEX, OR ARE USED EXTENSIVELY IN SPF. FORMATS OF OTHER TABLES AND BLOCKS CAN BE FOUND IN THE PLS COMPILATION LISTINGS.

BCT	- BROWSE CONTROL TABLE
CBT	- COMMON BROWSE TABLE
CIVCOMM	- CIV COMMON AREA
EDR	- EDIT RECORD
EDT	- EDIT TABLE
ELC	- EDIT LINE COMMAND
MHAF	- MENU HANDLER BUFFER
SDE	- SPF DIRECTORY ENTRY
TCS	- COMMON COMMAND SCAN TABLE
TCT	- CONTROLLER TABLES ARRAY
TDS	- SPF DATA SETS TABLE
TFD	- FILE DEFINITION TABLE
TFI	- FIND MEMBER TABLE
TKV	- KEYWORD-VALUE TABLE
TLD	- LOGICAL DISPLAY TABLE
TLS	- LOGICAL SCREEN TABLE
TPD	- PHYSICAL DISPLAY TABLE
TSC	- SUBROUTINE COMMON TABLE
TSI	- SPF INTERFACE TABLE
TSV	- SPF VARIABLES TABLE
UDACOMM	- UDA COMMON AREA

THE FIELD DESCRIPTIONS THAT ARE PART OF EACH DETAILED DESCRIPTION ARE IN THREE COLUMNS. THE COLUMNS ARE:

OFFSET - THE NUMERIC ADDRESS OF THE FIELD RELATIVE TO THE BEGINNING OF THE AREA. THE FIRST NUMBER IS THE OFFSET IN DECIMAL, THE SECOND IS THE HEXADECIMAL EQUIVALENT. THE BIT OFFSET WITHIN A BYTE IS SHOWN FOLLOWING THE HEXIDECIMAL OFFSET.

FIELD NAME - THE NAME AND FORMAT IN PLS FORMAT, AS IT WOULD APPEAR IN COMPILER LISTINGS. '*' INDICATES AN UNNAMED FIELD. THE FORMATS THAT ARE USED ARE SHOWN IN THE EXAMPLE BELOW.

FIELD DESCRIPTION - A DESCRIPTION OR TITLE FOR THE FIELD.

EXAMPLE OF FIELD DESCRIPTIONS:

OFFSET		FIELD		FIELD	
DEC	HEX	NAME AND FORMAT		DESCRIPTION	
0	0	1 TABLE		/* TABLE NAME	*/
0	0	2 *,		/* UNNAMED GROUP OF FIELDS	*/
0	0	3 FIELD1	CHAR(8),	/* CHAR STRING (8 BYTES)	*/
8	8	3 FIELD2	BIT(64),	/* BIT STRING (8 BYTES)	*/
16	10	3 FIELD3	FIXED(31),	/* FIXED NUMBER (4 BYTES)	*/
20	14	3 FIELD4	PTR(31),	/* ADDRESS (4 BYTES)	*/
24	18	3 FIELD5	BIT(8),	/* BIT STRING (1 BYTE)	*/
24	18.0	4 BIT1	BIT(1),	/* 1ST BIT IN FIELD 5	*/
24	18.1	4 BIT7	BIT(7),	/* NEXT 7 BITS IN FIELD 5	*/
25	19	3 FIELD6	PTR(24),	/* ADDRESS (3 BYTES)	*/
28	1C	3 FIELD7	FIXED(15),	/* FIXED NUMBER (2 BYTES)	*/
32	20	3 FIELD8	FIXED(8),	/* FIXED NUMBER (1 BYTE)	*/
33	21	3 FIELD9	CHAR(*);	/* CHAR STRING (VARIABLE)	*/

BCT - BROWSE CONTROL TABLE BCT

```
 OFFSET     FIELD                                  FIELD
DEC  HEX      NAME                                   DESCRIPTION
---- ----   -----------------------------------     -------------------------
   0    0   1 BCT      BASED,                    /* BROWSE CONTROL TBL        */
   0    0     2 BCTCNTL,                         /*  CONTROL PART OF CNTL TBL */
   0    0       3 BCTNEXTP PTR(31),              /*    PTR TO NEXT BROWSE BUF */
   4    4       3 BCTPREVP PTR(31),              /*    PTR TO PREV BROWSE BUF */
   8    8       3 BCTF#    FIXED(31),            /*    FIRST LINE # FOR TBL   */
  12    C       3 BCTL#    FIXED(31),            /*    LAST LINE # FOR TBL    */
  16   10       3 BCTLREC# FIXED(31),            /*    FIRST LINE OF LAST REC */
  20   14       3 BCTFTTRN,                      /*    FIRST TTRN OF CNTL TBL */
  20   14         4 BCTFTRK  FIXED(15),          /*      TRK (TT) FOR         */
  22   16         4 BCTFREC  FIXED(8),           /*      REC (R)              */
  23   17         4 *        FIXED(8),           /*      N = 0 ALWAYS         */
  24   18       3 BCTLTTRN,                      /*    LAST TTRN OF CNTL TBL  */
  24   18         4 BCTLTRK  FIXED(15),          /*      TRK (TT) FOR         */
  26   1A         4 BCTLREC  FIXED(8),           /*      REC (R)              */
  27   1B         4 *        FIXED(8),           /*      N = 0 ALWAYS         */
  28   1C     2 BCTTTBL CHAR(1000),              /*  NEW TRACK BIT TABLE      */
1028  404     2 BCTRTBL CHAR(1000);              /*  NEW RECORD BIT TABLE     */
```

```
 OFFSET      FIELD                                  FIELD
DEC  HEX       NAME                                   DESCRIPTION
___  ___    ___________________________________    ____________________________

  0    0    1 CBT      BASED,                  /*                           */
  0    0      2 * BOUNDARY(WORD),              /*                           */
  0    0        3 *,                           /* CAPS/ASIS BYTE(BIT)       */
  0  0.0          4 *         BIT(7),          /*   ON -> CAPS MODE         */
  0  0.7          4 CBTCAPS   BIT(1),          /*   OFF -> ASIS MODE        */
  1    1        3 *,                           /* COLS BYTE(BIT)            */
  1  1.0          4 *         BIT(7),          /*   ON -> COLS              */
  1  1.7          4 CBTCOL    BIT(1),          /*   OFF -> NOCOLS           */
  2    2        3 *,                           /* HEX BYTE(BIT)             */
  2  2.0          4 *         BIT(7),          /*   ON -> HEX               */
  2  2.7          4 CBTHEX    BIT(1),          /*   OFF -> NOHEX            */
  3    3        3 *,                           /* HEX CHAR FORMAT BYTE(BIT) */
  3  3.0          4 *         BIT(7),          /*   ON -> HEX CHARS FORMAT  */
  3  3.7          4 CBTHCHAR BIT(1),           /*   OFF -> HEX DATA FORMAT  */
  4    4      2 * BOUNDARY(WORD),              /*                           */
  4    4        3 CBTPADCH    FIXED(8),        /* PAD CHAR FOR SHORT RECS   */
  5    5        3 *           BIT(24),         /* ** RESERVED **            */
  8    8      2 *          PTR(31),            /* ** RESERVED **            */
 12    C      2 CBTBCTP    PTR(31),            /* BCT PTR                   */
 16   10      2 CBTBLBP    PTR(31),            /* BLB PTR                   */
 20   14      2 CBTCBLP    PTR(31),            /* FIRST CBL PTR             */
 24   18      2 CBTCBLBP PTR(31),              /* CURRENT BLB PTR           */
 28   1C      2 CBTCMDP    PTR(31),            /* COMMAND INPUT PTR         */
 32   20      2 CBTCSRLP PTR(31),              /* CURSOR LINE BUFF (BLB) PTR*/
 36   24      2 CBTER1P    PTR(31),            /* ERROR 1 PARM PTR          */
 40   28      2 CBTER2P    PTR(31),            /* ERROR 2 PARM PTR          */
 44   2C      2 CBTER3P    PTR(31),            /* ERROR 3 PARM PTR          */
 48   30      2 CBTER4P    PTR(31),            /* ERROR 4 PARM PTR          */
 52   34      2 CBTER5P    PTR(31),            /* ERROR 5 PARM PTR          */
 56   38      2 CBTER6P    PTR(31),            /* ERROR 6 PARM PTR          */
 60   3C      2 CBTER7P    PTR(31),            /* ERROR 7 PARM PTR          */
 64   40      2 CBTER8P    PTR(31),            /* ERROR 8 PARM PTR          */
 68   44      2 CBTER9P    PTR(31),            /* ERROR 9 PARM PTR          */
 72   48      2 CBTGETLP PTR(31),              /* GET LINE BUFF (BLB) PTR   */
 76   4C      2 CBTIOAP    PTR(31),            /* I/O ARRAY PTR             */
 80   50      2 CBTLBCTP PTR(31),              /* LAST BCT PTR              */
 84   54      2 CBTCSRP    PTR(31),            /* CURSOR POINTER            */
 88   58      2 CBTTFDP    PTR(31),            /* TFD PTR                   */
 92   5C      2 CBTTLSP    PTR(31),            /* TLS PTR                   */
 96   60      2 *          PTR(31),            /*   ** RESERVED **          */
100   64      2 *          PTR(31),            /*   ** RESERVED **          */
104   68      2 *          PTR(31),            /*   ** RESERVED **          */
108   6C      2 CBTBCTSZ FIXED(31),            /* BCT SIZE                  */
112   70      2 CBTCMDSZ FIXED(31),            /* COMMAND INPUT SIZE (LINE2)*/
116   74      2 CBTCSR#    FIXED(31),          /* CURSOR (LINE) NUMBER      */
120   78      2 CBTCSROS FIXED(31),            /* CURSOR OFFSET             */
124   7C      2 CBTCTRK    FIXED(31),          /* CURRENT TRACK             */
128   80      2 CBTCREC    FIXED(31),          /* CURRENT RECORD            */
132   84      2 CBTCNUM    FIXED(31),          /* CURRENT NUMBER            */
136   88      2 CBTDUPD    FIXED(31),          /* DISPLAY UNITS / DISPLAY   */
140   8C      2 CBTEOF#    FIXED(31),          /* LINE NUMB OF EOF          */
144   90      2 CBTERCOD FIXED(31),            /* ERROR CODE                */
148   94      2 CBTFD#     FIXED(31),          /* FIRST DISPLAY LINE NUMB   */
152   98      2 CBTFIO#    FIXED(31),          /* FIRST I/O ARRAY LINE NUMB */
156   9C      2 CBTGET#    FIXED(31),          /* GET REQUEST LINE NUMB     */
160   A0      2 CBTHIGH# FIXED(31),            /* HIGHEST LINE # READ       */
164   A4      2 CBTIODIM FIXED(31),            /* I/O ARRAY DIMENSION       */
168   A8      2 CBTIOX     FIXED(31),          /* I/O ARRAY INDEX           */
172   AC      2 CBTLRECL FIXED(31),            /* LRECL (OF DATA SET)       */
176   B0      2 CBTLD#     FIXED(31),          /* LAST DISPLAYED LINE #     */
180   B4      2 CBTLEFTC FIXED(31),            /* LEFT (DISPLAY) COLUMN     */
184   B8      2 CBTLIO#    FIXED(31),          /* LAST I/O ARRAY LINE NUMB  */
188   BC      2 CBTPCSR    FIXED(31),          /* PREVIOUS CSR PTR          */
192   C0      2 CBTSTRCT FIXED(31),            /* STRING COUNT (FIND ALL)   */
196   C4      2 CBTLINCT FIXED(31),            /* LINE COUNT (FIND ALL)     */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET      FIELD                                  FIELD
DEC   HEX      NAME                                   DESCRIPTION
____  ____   ___________________________________    _________________________

200   C8     2 CBTSCNCT FIXED(31),              /* SCAN COUNT (FIND ALL)      */
204   CC     2 CBTWDCNT FIXED(31),              /* WORD CNT FOR INPUT SCAN    */
208   D0     2 CBTSCSRB FIXED(31),              /* SCROLL CURSOR BACKUP       */
212   D4     2 CBTCMCOD FIXED(31),              /* CONFIRMATION MSG CODE      */
216   D8     2 CBTPL#   FIXED(31),              /* PREVIOUS LABEL NUMBER      */
220   DC     2 CBTLBACT FIXED(15),              /* LEFT BOUND ACTUAL          */
222   DE     2 CBTRBACT FIXED(15),              /* RIGHT BOUND ACTUAL         */
224   E0     2 CBTSCROL,                        /* SCROLL PARMS (FOR CSCROLL)*/
224   E0       3 CBTSCURL FIXED(31),            /* CURRENT LINE NUMBER        */
228   E4       3 CBTSMAXL FIXED(31),            /* MAXIMUM LINE NUMBER        */
232   E8       3 CBTSPLEN FIXED(31),            /* PAGE LENGTH (NO. LINES)    */
236   EC       3 CBTSCURC FIXED(31),            /* CURRENT COL NUMBER         */
240   F0       3 CBTSMAXC FIXED(31),            /* MAXIMUM COLUMN NUMBER      */
244   F4       3 CBTSPCOL FIXED(31),            /* PAGE WIDTH (NO. COLUMNS)   */
248   F8     2 CBTTCS,                          /* TCS (CMD SCAN) INTERFACE   */
248   F8       3 *        PTR(31),              /*   COMMAND INPUT PTR        */
252   FC       3 *        FIXED(31),            /*   COMMAND INPUT SIZE       */
256   100      3 *        FIXED(31),            /*   COMMAND INPUT SIZE       */
260   104      3 *        PTR(31),              /*   TCD ENTRY PTR            */
264   108      3 *        (12),                 /*   INPUT PARM WORD ARRAY    */
264   108        4 *        CHAR(16),           /*    TCSWDS ENTRY            */
456   1C8    2 *,                               /*                            */
456   1C8      3 CBTCSTAT,                      /*   CURRENT STATUS           */
456   1C8.0      4 CBTFCTOP BIT(1),             /*     FIND - TOP OF DATA     */
456   1C8.1      4 CBTFCBOT BIT(1),             /*     FIND - BOTTOM OF DATA */
456   1C8.2      4 CBTCFND  BIT(1),             /*     CURR TIME STR FOUND    */
456   1C8.3      4 *        BIT(5),             /*     * RESERVED             */
457   1C9      3 CBTPSTAT,                      /*   PREVIOUS STATUS          */
457   1C9.0      4 CBTFPTOP BIT(1),             /*     FIND - TOP OF DATA     */
457   1C9.1      4 CBTFPBOT BIT(1),             /*     FIND - BOTTOM OF DATA */
457   1C9.2      4 CBTPFND  BIT(1),             /*     PREV TIME STR FOUND    */
457   1C9.3      4 *        BIT(5),             /*     * RESERVED             */
458   1CA      3 CBTDIR   FIXED(8),             /*   FIND DIRECTION           */
459   1CB      3 CBTTYP   FIXED(8),             /*   FIND TYPE                */
460   1CC      3 CBTDIRW  CHAR(8),              /*   FIND DIRECTION           */
468   1D4      3 CBTTYPW  CHAR(8),              /*   FIND TYPE                */
476   1DC      3 CBTLB    FIXED(15),            /*   LEFT BOUND               */
478   1DE      3 CBTRB    FIXED(15),            /*   RIGHT BOUND              */
480   1E0      3 CBTSTRSZ FIXED(15),            /*   FIND STRING SIZE         */
482   1E2      3 CBTSTRST BIT(8),               /*   FIND STRING STATUS       */
482   1E2.0      4 CBTSTRF  BIT(1),             /*      STRING IS DEFINED     */
482   1E2.1      4 CBTHEXF  BIT(1),             /*      STRING IS HEX FLAG    */
482   1E2.2      4 CBTPICTF BIT(1),             /*      STRING IS PICT FLAG   */
482   1E2.3      4 CBTTEXTF BIT(1),             /*      STRING IS TEXT FLAG   */
482   1E2.4      4 *        BIT(4),             /*                            */
483   1E3      3 *        BIT(8),               /*   ** RESERVED **           */
484   1E4      3 *,                             /*   FIND STRINGS             */
484   1E4        4 CBTSTR   CHAR(40),           /*     USED BY FIND CMD       */
524   20C        4 CBTSTRIA CHAR(46),           /*     INPUT FORMAT           */
570   23A        4 CBTSTROA CHAR(46),           /*     OUTPUT FORMAT          */
616   268    2 CBTTTRN,                         /* TTRN FROM CGET             */
616   268      3 CBTTT    FIXED(15),            /*    TRACK                   */
618   26A      3 CBTR     FIXED(8),             /*    RECORD                  */
619   26B      3 CBTN     FIXED(8),             /*    NUMB                    */
620   26C    2 CBTSPNUM   FIXED(8),             /* BROWSE SUBPOOL NUMBER      */
621   26D    2 * BIT(8),                        /*                            */
621   26D.0    3 CBTCSRFG BIT(1),               /* CURSOR SET FLAG            */
621   26D.1    3 CBTDBOT  BIT(1),               /* DISPLAY - BOTTOM OF DATA   */
621   26D.2    3 CBTEOF   BIT(1),               /*                            */
621   26D.3    3 CBTPCFG  BIT(1),               /* PRIMARY COMMAND FLAG BIT   */
621   26D.4    3 CBTLIST  BIT(1),               /* LISTING (FBA,FBM) FLAG BIT*/
621   26D.5    3 CBTTRKOF BIT(1),               /* TRACK OVERFLOW CONDITION   */
621   26D.6    3 *        BIT(2),               /*                            */
622   26E    2 CBTENBL CHAR(8),                 /*                            */
630   276    2 CBTPCCOD CHAR(1),                /* PRIMARY COMMAND CODE       */
631   277    2 *       CHAR(23);                /* ** RESERVED **             */
```

CIVCOMM - CIV COMMON AREA

```
 OFFSET       FIELD                                 FIELD
DEC  HEX        NAME                                  DESCRIPTION
---  ----   ---------------------------------       --------------------------

  0     0   1 CIVCOMM  BASED BDY(WORD),           /*                            */
  0     0     2 CIVCNRX  FIXED(31),               /* NUMBER OF EXTENTS          */
  4     4     2 CIVCNRXU FIXED(31),               /* NUMBER OF EXTENTS IN USE   */
  8     8     2 CIVCTOTQ FIXED(31),               /* TOTAL SPACE QUAN           */
 12     C     2 CIVCTOTU FIXED(31),               /* TOTAL SPACE USED           */
 16    10     2 CIVCNRBU FIXED(31),               /* NUMBER OF USED DIR BLKS    */
 20    14     2 CIVCNRM  FIXED(31),               /* NUMBER OF MEMBERS          */
 24    18     2 CIVCSPCL FIXED(31),               /* LEN OF WORD IN CIVCSPCU    */
 28    1C     2 CIVCDSOB BIT(16),                 /* DSORG DSCB BITS            */
 28    1C.0     3 CIVCIS   BIT(1),                /*   INDEX SEQUENTIAL         */
 28    1C.1     3 CIVCPS   BIT(1),                /*   PHYSICAL SEQUENTIAL      */
 28    1C.2     3 CIVCDO   BIT(1),                /*   DIRECT                   */
 28    1C.3     3 *        BIT(3),                /*   *** UNREFERENCED ***     */
 28    1C.6     3 CIVCPO   BIT(1),                /*   PARTITIONED              */
 28    1C.7     3 CIVCUM   BIT(1),                /*   UNMOVABLE                */
 29    1D.0     3 *        BIT(4),                /*   *** UNREFERENCED ***     */
 29    1D.4     3 CIVCVS   BIT(1),                /*   VSAM                     */
 29    1D.5     3 *        BIT(3),                /*   OTHER - 3-UNOPENED PS    */
                                                  /*           7-SYSCTLG        */
 30    1E     2 CIVCRECB BIT(8),                  /* RECFM DSCB BITS            */
 30    1E.0     3 CIVCRFTP BIT(2),                /*   RECORD FORMAT TYPE       */
 30    1E.2     3 CIVCRFT  BIT(1),                /*   TRACK OVERFLOW           */
 30    1E.3     3 CIVCRFB  BIT(1),                /*   BLOCKED                  */
 30    1E.4     3 CIVCRFS  BIT(1),                /*   STANDARD OR SPANNED      */
 30    1E.5     3 CIVCRFCL BIT(2),                /*   PRINT CONTROL TYPE       */
 30    1E.7     3 *        BIT(1),                /*   *** UNREFERENCED ***     */
 31    1F     2 CIVCFLGS BIT(8),                  /* FLAGS                      */
 31    1F.0     3 CIVCLOAD BIT(1),                /*   LOAD MODULE DATASET      */
 31    1F.1     3 CIVCSPF  BIT(1),                /*   SPF STATS IN ONE MEMBER  */
 31    1F.2     3 CIVCSUL  BIT(1),                /*   USER LABEL DATASET       */
 31    1F.3     3 *        BIT(5),                /*   ** RESERVED **           */
 32    20     2 CIVCDSO  CHAR(8),                 /* DSORG                      */
 40    28     2 CIVCCD   CHAR(8),                 /* CREATION DATE              */
 48    30     2 CIVCED   CHAR(8),                 /* EXPIRATION DATE            */
 56    38     2 CIVCOPTJ CHAR(1),                 /* OPTCD (J OR BLANK)         */
 57    39     2 *        CHAR(3),                 /* ** RESERVED **             */
 60    3C     2 *        CHAR(8),                 /* ** RESERVED **             */
 68    44     2 CIVCALOC ,                        /* ALLOC TKV PARMS            */
 68    44       3 CIVCVOL  CHAR(6),               /*   VOLUME SERIAL            */
 74    4A       3 CIVCSPCU CHAR(8),               /*   SPACE UNIT: 'CYLINDER',  */
                                                  /*   'BLOCK   ', OR 'TRACK   '*/
 82    52       3 CIVCRECF CHAR(6),               /*   RECORD FORMAT            */
 88    58       3 CIVCBLK  FIXED(15),             /*   BLOCK SIZE               */
 90    5A       3 CIVCLREC FIXED(15),             /*   LRECL                    */
 92    5C       3 CIVCEXT1 FIXED(31),             /*   SIZE OF FIRST EXTENT     */
 96    60       3 CIVCSECQ FIXED(31),             /*   SECONDARY QUANTITY       */
100    64       3 CIVCNRB  FIXED(31);             /*   NUMBER OF DIR BLKS       */
```

```
 OFFSET      FIELD                                FIELD
DEC  HEX      NAME                                 DESCRIPTION
---  ---    ----------------------------------    --------------------------

  0    0    1 EDR      BASED,                 /* EDIT LINE CONTROL          */
  0    0      2 EDRBASE,                      /*  BASE PART OF EDR          */
  0    0        3 EDRNEXTP PTR(31),           /*   NEXT EDR PTR             */
  4    4        3 EDRPREVP PTR(31),           /*   PREV EDR PTR             */
  8    8        3 EDRALTCT,                   /*   ALTERNATE AREA CONTROL   */
  8    8          4 *       BIT(8),           /*   TYPE OF RECORD           */
  8    8.0          5 EDRXCLUD BIT(1),        /*     PTR IN EDRXFP/EDRXLP   */
  8    8.1          5 EDRCMD   BIT(1),        /*     CMD IN EDRCAREA        */
  8    8.2          5 *        BIT(6),        /*     ** RESERVED **         */
  9    9          4 EDRALTP PTR(24),          /*   ALTERNATE AREA PTR       */
 12    C        3 EDRNUMB  FIXED(32),         /*   INTERNAL ASCENDING NUMB */
 16   10        3 EDRCNTL,                    /*                            */
 16   10          4 EDRSPECL FIXED(8),        /*   SPECIAL RECORD INDEX     */
 17   11          4 EDRSOURC BIT(8),          /*   SOURCE OF RECORD         */
 17   11.0          5 EDRORIG  BIT(1),        /*     ORIGINAL               */
 17   11.1          5 EDRIMOVE BIT(1),        /*     INTERNAL MOVE          */
 17   11.2          5 EDRICOPY BIT(1),        /*     INTERNAL COPY/REPEAT   */
 17   11.3          5 EDREMOVE BIT(1),        /*     EXTERNAL MOVE          */
 17   11.4          5 EDRECOPY BIT(1),        /*     EXTERNAL COPY          */
 17   11.5          5 EDRTEXTI BIT(1),        /*     TEXT INSERTED          */
 17   11.6          5 EDRTYPEI BIT(1),        /*     TYPED INSERTED         */
 17   I1.7          5 *        BIT(1),        /*     ** RESERVED **         */
 18   12          4 EDRGEN   BIT(8),          /*   GENERAL AND DISPL FLAGS */
 18   12.0          5 EDRTOP   BIT(1),        /*     TOP (RECORD)           */
 18   12.1          5 EDRBOT   BIT(1),        /*     BOTTOM (RECORD)        */
 18   12.2          5 EDRSTD   BIT(1),        /*     STANDARD (RECORD)      */
 18   12.3          5 EDRTEMP  BIT(1),        /*     TEMPORARY (RECORD)     */
 18   12.4          5 EDRLDINT BIT(1),        /*     LINE DATA INTENSIFY    */
 18   12.5          5 EDRLDPRO BIT(1),        /*     LINE DATA PROTECTED    */
 18   12.6          5 EDRNOTAB BIT(1),        /*     NO TABS (ATTR BYTES)   */
 18   12.7          5 EDRLCPRO BIT(1),        /*     LINE CMD PROTECTED     */
 19   13          4 EDRCHGST BIT(8),          /*   RECORD CHANGED STATUS    */
 19   13.0          5 EDRCHG   BIT(1),        /*     CHANGED                */
 19   13.1          5 EDRTYPED BIT(1),        /*     DATA OVERTYPED         */
 19   13.2          5 EDRCHGED BIT(1),        /*     CMD CHG OR OVERLAY CHG*/
 19   13.3          5 EDRSCOLS BIT(1),        /*     COLUMNS SHIFTED        */
 19   13.4          5 EDRSDATA BIT(1),        /*     DATA SHIFTED           */
 19   13.5          5 EDRTEXTC BIT(1),        /*     TEXT CHANGE            */
 19   13.6          5 *        BIT(1),        /*     ** RESERVED **         */
 19   13.7          5 EDRRENUM BIT(1),        /*     LINE RENUMBERED        */
 20   14      2 EDRDATA  CHAR(*);             /*   RECORD DATA              */
```

EDRALT - EDIT RECORD ALTERNATE

```
 OFFSET      FIELD                                FIELD
DEC  HEX      NAME                                 DESCRIPTION
---  ---    ----------------------------------    --------------------------

  0    0    1 EDRALT BASED (EDRALTP),         /* ALT EDR AREA (IF X | CMD)*/
  0    0      2 EDRXLP    PTR(31),            /*   PTR TO LAST X'ED OF BLK*/
  4    4      2 EDRXFP    PTR(31),            /*   PTR TO 1ST X'ED OF BLK */
  8    8      2 EDRXCNT   FIXED(31),          /*   CNT OF X'ED IN BLOCK   */
 12    C      2 EDRCAREA CHAR(6),             /*   CMD AREA               */
 18   12      2 EDRCNAME CHAR(6),             /*   CMD NAME               */
 24   18      2 EDRCSUFF FIXED(31),           /*   CMD SUFFIX             */
 28   1C      2 EDRCELCP PTR(31),             /*   LINE COMMAND PTR       */
 32   20      2 EDRCMDNP PTR(31),             /*   PTR TO NEXT CMD EDR    */
 36   24      2 EDRCMDPP PTR(31);             /*   PTR TO PREV CMD EDR    */
 OFFSET      FIELD                                FIELD
DEC  HEX      NAME                                 DESCRIPTION
---  ---  ---------------------------------   ---------------------------

  0    0  1 EDT      BASED,                   /* EDIT TABLE                 */
           /* — EDT#DCLS ——————————————————————————————————————————————————— */
  0    0    2 EDTTLDP  POINTER(31),           /* PTR TO TLD (LOGICAL DISPL)*/
  4    4    2 EDTEDTP  POINTER(31),           /* PTR TO EDT (EDIT TABLE)    */
  8    8    2 EDTPTR1  POINTER(31),           /* PTR FOR EXIT ROUTINES      */
 12    C    2 EDTPTR2  POINTER(31),           /* PTR FOR EXIT ROUTINES      */
 16   10    2 EDTTOPP  POINTER(31),           /* PTR TO TOP EDR             */
 20   14    2 EDTBOTP  POINTER(31),           /* PTR TO BOTTOM EDR          */
 24   18    2 EDTEDROP POINTER(31),           /* PTR TO EDRDATA OFFSET      */
 28   1C    2 EDTLRECP POINTER(31),           /* PTR TO EDRDATA LENGTH      */
 32   20    2 EDTERCDP POINTER(31),           /* PTR TO ERROR CODE WORD     */
 36   24    2 EDTMGCDP POINTER(31),           /* PTR TO MSG CODE WORD       */
 40   28    2 EDTDSNSP POINTER(31),           /* PTR TO D.S NAME STRUCT     */
 44   2C    2 EDTMEMBP POINTER(31),           /* PTR TO MEMBER NAME         */
 48   30    2 EDTSDEP  POINTER(31),           /* PTR TO SPF DIR ENTRY (IN)  */
 52   34    2 EDTBLDLP POINTER(31),           /* PTR TO BLDL AREA (OUT)     */
 56   38    2 EDTIDTTP POINTER(31),           /* PTR TO IN/DATA TRANS TBL   */
 60   3C    2 EDTODTTP POINTER(31),           /* PTR TO OUT/DATA TRANS TBL  */
 64   40    2 EDTITTP  POINTER(31),           /* PTR TO IN/TERM TRANS TBL   */
 68   44    2 EDTOTTP  POINTER(31),           /* PTR TO OUT/TERM TRANS TBL  */
 72   48    2 EDTITFDP POINTER(31),           /* PTR TO PRIM INPUT TFD      */
 76   4C    2 EDTOTFDP POINTER(31),           /* PTR TO PRIM OUTPUT TFD     */
 80   50    2 EDTCTFDP POINTER(31),           /* PTR TO COPY INPUT TFD      */
 84   54    2 EDTRTFDP POINTER(31),           /* PTR TO REPL OUTPUT TFD     */
 88   58    2 EDTFREEP POINTER(31),           /* PTR TO FREE CHAIN EDR'S    */
 92   5C    2 EDTDELP  POINTER(31),           /* PTR TO DELETE CHAIN EDR'S  */
 96   60    2 EDTMASKP POINTER(31),           /* PTR TO MASK LINE           */
100   64    2 EDTTABSP POINTER(31),           /* PTR TO TABS LINE           */
104   68    2 EDTXMSGP POINTER(31),           /* PTR TO 'EXCLUDE' LINE MSG  */
108   6C    2 EDTHEADP POINTER(31),           /* PTR TO 2 LINE HEADER       */
112   70    2 *        CHARACTER(16),         /*   EXTRA SPACE              */
           /* — EDTBDCLS ——————————————————————————————————————————————————— */
128   80    2 EDTEBTP  POINTER(31),           /*   BACKUP TABLE POINTER     */
132   84    2 EDTB     BIT(8),                /*   BACKUP FLAGS             */
132   84.0    3 EDTBINIT BIT(1),              /*     BACKUP INITIALIZED     */
132   84.1    3 EDTBST   BIT(1),              /*     BACKUP STARTED         */
132   84.2    3 EDTBERR  BIT(1),              /*     BKUP/RCVR ERROR        */
132   84.3    3 *        BIT(1),              /*                            */
132   84.4    3 EDTBR    BIT(1),              /*     RECOVERY IN PROGRESS   */
132   84.5    3 *        BIT(3),              /*                            */
133   85    2 EDTBCODE CHARACTER(1),          /*   EBS BACKUP FUNCTION CODE*/
134   86    2 EDTBSUSC CHARACTER(1),          /*   EBS SUSPEND CODE         */
135   87    2 *        CHARACTER(1),          /*   ** RESERVED **           */
136   88    2 EDTEBUSZ FIXED(31),             /*   BACKUP DATASET BLKSIZE   */
140   8C    2 *        CHARACTER(20),         /*   EXTRA SPACE FOR BACKUP   */
           /* — EDTCDCLS ——————————————————————————————————————————————————— */
          /* LINE COMMAND VARIABLES                                         */
160   A0    2 EDABSUF  FIXED(31),             /*   AFTER/BEFORE SUFFIX      */
164   A4    2 EDCMDCNT FIXED(31),             /*   COMMAND COUNT            */
168   A8    2 EDSUFFIX FIXED(31),             /*   SUFFIX FROM LINE CMD     */
172   AC    2 EDFBSUF  FIXED(31),             /*   FIRST OF BLOCK - SUFFIX  */
176   B0    2 EDCNAME  CHARACTER(6),          /*   COMMAND NAME             */
182   B6    2 EDFBCODE FIXED(8),              /*   FIRST OF BLOCK - CODE    */
183   B7    2 *        CHARACTER(1),          /*   ** RESERVED **           */
184   B8    2 *        CHARACTER(16),         /*   ** RESERVED **           */
          /* PRIMARY COMMAND VARIABLES                                      */
200   C8    2 EDEPCP   POINTER(31),           /*   EPC (PRIM CMD DEF) PTR   */
204   CC    2 EDTTCS,                         /*   TCS (CMD SCAN ARRAY)     */
204   CC      3 *        POINTER(31),         /*   PTR TO COMMAND INPUT     */
208   D0      3 *        POINTER(31),         /*   SIZE OF COMMAND INPUT    */
212   D4      3 *        FIXED(31),           /*   PRIMARY CMD WORD COUNT   */
216   D8      3 *        POINTER(31),         /*   PTR TO TCD ENTRY         */
220   DC      3 *        (12),                /*   PRIMARY CMD WORDS(ARRAY)*/
220   DC        4 * CHARACTER(16),            /*     ARRAY (TCSWDS ENTRIES)*/
412  19C    2 EDPCCODE FIXED(31),             /*   PRIMARY COMMAND ERR      */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET      FIELD                                   FIELD
DEC   HEX      NAME                                    DESCRIPTION
----  ----   -----------------------------------    -------------------------
416   1A0      2 EDPCCSRP PTR(31),             /*    CODE, CURSOR POSITION  */
420   1A4      2 EDPCER1P PTR(31),             /*    " AND ERROR PARM PTRS  */
424   1A8      2 EDPCER2P PTR(31),             /*    "  PASSED FROM EPI TO  */
428   1AC      2 EDPCER3P PTR(31),             /*    "   EPF                */
432   1B0      2 EDTPCMD  CHARACTER(1),        /*   EDIT PRIMARY COMMAND    */
433   1B1      2 *        CHARACTER(3),        /*   ** RESERVED **          */
436   1B4      2 *        CHARACTER(12),       /*   EXTRA SPACE             */
               /* — EDTDDCLS ————————————————————————————————————————————— */
             /* CURSOR FIELDS                                               */
448   1C0      2 EDCSRCO  FIXED(31),           /*   CSR (LINE) CMD OFFSET   */
452   1C4      2 EDCSRDO  FIXED(31),           /*   CSR (LINE) DATA OFFSET  */
456   1C8      2 EDCSRREL FIXED(31),           /*   CURSOR REL LINE (0 | 1) */
460   1CC      2 EDCSRRO  FIXED(31),           /*   CSR (EDRDATA) REC OFFSET*/
464   1D0      2 EDPCSR   FIXED(31),           /*   PREVIOUS CURSOR LOCATION*/
468   1D4      2 EDTCSRP  POINTER(31),         /*   CURSOR POINTER          */
472   1D8      2 *        BIT(32),             /*   ** RESERVED **          */
472   1D8.0      3 EDTCSRFC BIT(1),            /*     CSR SET BY FIND/CHG   */
472   1D8.1      3 *        BIT(31),           /*     ** RESERVED **        */
476   1DC      2 EDTCSRTX FIXED(31),           /*   CURSOR TEXT OFFSET      */
             /* DISPLAY FIELDS                                              */
480   1E0      2 EDFEDLP  POINTER(31),         /*   FIRST EDIT LINE POINTER */
484   1E4      2 EDRELNUM FIXED(31),           /*   REL NUM OF 1ST DISPL EDR*/
488   1E8      2 EDMAXLNS FIXED(31),           /*   MAX (DISPLAY) LINES     */
492   1EC      2 EDCURLNS FIXED(31),           /*   CURR LINES (ON SCREEN)  */
496   1F0      2 EDTPSLNS FIXED(31),           /*   LINES ON TPS (NOT HDR)  */
500   1F4      2 EDINSCNT FIXED(31),           /*   INSERT LINE COUNT       */
504   1F8      2 EDTRDHDR CHAR(1),             /*   REDISPLAY HEADER Y | N  */
505   1F9      2 *        CHAR(3),             /*   REDISPLAY HEADER Y | N  */
508   1FC      2 *        CHARACTER(20),       /*   EXTRA SPACE             */
               /* — EDTFDCLS ————————————————————————————————————————————— */
528   210      2 *        BIT(8),              /*   FIND/CHG STATUS BITS    */
528   210.0      3 EDFCCHG  BIT(1),            /*       CHG CMD LAST ACTION */
528   210.1      3 EDFCFIND BIT(1),            /*       FIND CMD LAST ACTION*/
528   210.2      3 *        BIT(6),            /*    **  RESERVED **        */
529   211      2 EDFCSBIT BIT(8),              /* STRING BITS               */
529   211.0      3 EDFCS1F  BIT(1),            /*    ON -> STRING 1 DEFINED */
529   211.1      3 EDFCX1F  BIT(1),            /*    ON -> STRING 1 IS HEX  */
529   211.2      3 EDFCP1F  BIT(1),            /*    ON -> STRING 1 IS PICT */
529   211.3      3 EDFCT1F  BIT(1),            /*    ON -> STRING 1 IS TEXT */
529   211.4      3 EDFCS2F  BIT(1),            /*    ON -> STRING 2 DEFINED */
529   211.5      3 EDFCX2F  BIT(1),            /*    ON -> STRING 2 IS HEX  */
529   211.6      3 *        BIT(2),            /*    **  RESERVED **        */
530   212      2 *        CHAR(2),             /*   ** RESERVED **          */
532   214      2 EDFCS1P  POINTER(31),         /*   STR1 PTR                */
536   218      2 EDFCS1SZ FIXED(31),           /*   STR1 SIZE               */
540   21C      2 EDFCS1IP POINTER(31),         /*   STR1 INPUT AREA PTR     */
544   220      2 EDFCS1OP POINTER(31),         /*   STR1 OUTPUT AREA PTR    */
548   224      2 EDFCS2P  POINTER(31),         /*   STR2 PTR                */
552   228      2 EDFCS2SZ FIXED(31),           /*   STR2 SIZE               */
556   22C      2 EDFCS2IP POINTER(31),         /*   STR2 INPUT AREA PTR     */
560   230      2 EDFCDIR  FIXED(8),            /*   F/C DIRECTION KEY CODE  */
561   231      2 EDFCLMT  FIXED(8),            /*   F/C LIMIT KEY CODE      */
562   232      2 EDFCTYP  FIXED(8),            /*   F/C TYPE KEY CODE       */
563   233      2 *        FIXED(8),            /*   ** RESERVED **          */
564   234      2 EDFCDIRW CHARACTER(8),        /*   F/C DIRECTION KEY WORD  */
572   23C      2 EDFCLMTW CHARACTER(8),        /*   F/C LIMIT KEY WORD      */
580   244      2 EDFCTYPW CHARACTER(8),        /*   F/C TYPE KEY WORD       */
588   24C      2 EDFCLBU  FIXED(31),           /*   F/C LEFT BOUND USED     */
592   250      2 EDFCRBU  FIXED(31),           /*   F/C RIGHT BOUND USED    */
596   254      2 EDFCLB   FIXED(31),           /*   F/C LEFT BOUND (ENTERED)*/
600   258      2 EDFCRB   FIXED(31),           /*   F/C RIGHT BOUND   (")   */
604   25C      2 EDFCPCRO FIXED(31),           /*   F/C PREV CSR OFFSET     */
608   260      2 EDFCLNS  (4) FIXED(31),       /*   F/C LINE COUNTERS       */
624   270      2 EDFCSTRS (4) FIXED(31),       /*   F/C STRINGS COUNTERS    */
640   280      2 *        CHARACTER(16),       /*   EXTRA SPACE             */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET      FIELD                                   FIELD
DEC  HEX      NAME                                    DESCRIPTION
---- ----    --------------------------------        ----------------------------
             /* — EDTGDCLS ---------------------------------------------------- */
 656  290    2 EDTPARMV,                       /* TKV PARMS:                  */
 656  290      3 EDTPRJ0 CHARACTER(8),         /*                             */
 664  298      3 EDTLIB0 CHARACTER(8),         /*                             */
 672  2A0      3 EDTTYP0 CHARACTER(8),         /*                             */
 680  2A8      3 EDTMPARM,                     /* MENU PARAMETERS:            */
 680  2A8        4 EDTMPROJ CHARACTER(8),      /*  PROJECT NAME               */
 688  2B0        4 EDTMLIBS,                   /*  LIBRARY NAMES              */
 688  2B0          5 EDTMLIB  (4) CHARACTER(8), /*                            */
 720  2D0        4 EDTMTYPE CHARACTER(8),      /*  TYPE QUALIFIER             */
 728  2D8        4 EDTMMEMB CHARACTER(8),      /*  MEMBER NAME                */
 736  2E0        4 EDTMOFID CHARACTER(62),     /*  "OTHER" FILE ID:           */
 736  2E0          5 EDTMDSN  CHARACTER(56),   /*    "OTHER" DATASET NAME     */
 792  318          5 EDTMVOL  CHARACTER(6),    /*    "OTHER" VOLUME SERIAL    */
 798  31E        4 EDTMPSWD CHARACTER(8),      /*  DATASET PASSWORD           */
 806  326        4 EDTMPNAM CHARACTER(8),      /*  PROFILE NAME               */
 814  32E        4 EDTMUOPT CHARACTER(2),      /*  REASON CODE                */
 816  330        4 EDTMREC  CHARACTER(2),      /*  RECOVERY MODE              */
 818  332        4 EDTMANUM CHARACTER(2),      /*  AUTONUM MODE               */
 820  334        4 EDTMPRT  CHARACTER(2),      /*  PRINT MODE                 */
 822  336        4 EDTMSTAT CHARACTER(2),      /*  STATS MODE                 */
 824  338        4 EDTMUSER CHARACTER(8),      /*  USER FIELD                 */
 832  340    2 EDTCOFID CHARACTER(62),         /* CURRENT "OTHER" FILE ID     */
 894  37E    2 *        CHARACTER(2),          /* ** RESERVED **              */
 896  380    2 EDTQNAME CHARACTER(8)           /* QNAME FOR ENQ/DEQ:          */
                        BOUNDARY(DWORD),       /*   "SPFDSN  "                */
 904  388    2 EDTRNAME CHARACTER(52)          /* RNAME FOR ENQ/DEQ:          */
                        BOUNDARY(DWORD),       /*                             */
 904  388      3 EDTRDSN  CHARACTER(44),       /*   DSN                       */
 948  3B4      3 EDTRMEM  CHARACTER(8),        /*   MEMBER                    */
 956  3BC    2 *        CHARACTER(4),          /* ** RESERVED **              */
 960  3C0    2 EDDSTYPE CHARACTER(8),          /* DATASET TYPE ('ASM',ETC)    */
 968  3C8    2 EDMEMNAM CHARACTER(8),          /* MEMBER NAME FROM CMD        */
 976  3D0    2 *        CHARACTER(56),         /* ** RESERVED **              */
1032  408    2 EDCMEMNM CHARACTER(8),          /* EXTEND COMMAND MEM NAME     */
1040  410    2 EDCFLINE CHARACTER(8)           /* EXTEND COPY 1ST LINE        */
                        BOUNDARY(DWORD),       /*                             */
1048  418    2 EDCLLINE CHARACTER(8)           /* EXTEND COPY LAST LINE       */
                        BOUNDARY(DWORD),       /*                             */
1056  420    2 EDCRSPEC CHARACTER(1),          /* EXTEND COPY RANGE INDIC     */
                                               /* ' '=NO,S=STD,C=COB,R=REL    */
1057  421    2 *        CHARACTER(3),          /* ** RESERVED **              */
1060  424    2 EDCRDSN  CHARACTER(44),         /* CREATE/REPLACE DSN          */
1104  450    2 EDCFRLIN FIXED(31),             /* FIRST WHEN EDCRSPEC=R       */
1108  454    2 EDCLRLIN FIXED(31),             /* LAST WHEN EDCRSPEC=R        */
1112  458    2 EDTMENU  CHARACTER(8),          /* MENU FOR EMP TO DISPLAY     */
1120  460    2 EDTASIZE FIXED(31),             /* SIZE OF EDT & ASSOC AREA    */
1124  464    2 EDTHELP  CHARACTER(8),          /* GENHELP MENU FROM HEADER    */
1132  46C    2 EDTIDABP PTR(31),               /* CMD INPUT AREA PTR          */
1136  470    2 EDTINSIZ FIXED(31),             /* CMD INPUT AREA SIZE         */
1140  474    2 EDCMFLIN CHARACTER(8),          /* EXTEND COPY MENU 1ST LINE   */
1148  47C    2 EDCMLLIN CHARACTER(8),          /* EXTEND COPY MENU LAST LIN   */
1156  484    2 *        CHARACTER(12),         /* EXTRA SPACE                 */
             /* — EDTMDCLS ---------------------------------------------------- */
1168  490    2 EDERCODE CHARACTER(4),          /*  EDIT ERROR CODE            */
1172  494    2 EDMGCODE CHARACTER(4),          /*  EDIT MESSAGE CODE          */
1176  498    2 EDTSMSGP POINTER(31),           /*  POINTER TO SHORT MSG       */
1180  49C    2 EDTLMSGP POINTER(31),           /*  POINTER TO LONG MSG        */
1184  4A0    2 EDABCOD  CHARACTER(4),          /*  ABEND CODE FOR ERR MSGS    */
1188  4A4    2 EDSHFERC FIXED(31),             /*  SHIFT ERR CNT FOR MSGS     */
1192  4A8    2 EDER1P   POINTER(31),           /*  GENERAL ERROR POINTER      */
1196  4AC    2 EDER2P   POINTER(31),           /*  GENERAL ERROR POINTER      */
1200  4B0    2 EDER3P   POINTER(31),           /*  GENERAL ERROR POINTER      */
1204  4B4    2 EDER4P   POINTER(31),           /*  GENERAL ERROR POINTER      */
1208  4B8    2 EDER5P   POINTER(31),           /*  GENERAL ERROR POINTER      */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET      FIELD                               FIELD
 DEC  HEX     NAME                                DESCRIPTION
----  ---   ------------------------------      ------------------

1212  4BC      2 EDER6P   POINTER(31),        /*  GENERAL ERROR POINTER   */
1216  4C0      2 EDER7P   POINTER(31),        /*  GENERAL ERROR POINTER   */
1220  4C4      2 EDER8P   POINTER(31),        /*  GENERAL ERROR POINTER   */
1224  4C8      2 EDER9P   POINTER(31),        /*  GENERAL ERROR POINTER   */
1228  4CC      2 EDER10P  POINTER(31),        /*  GENERAL ERROR POINTER   */
1232  4D0      2 EDER11P  POINTER(31),        /*  GENERAL ERROR POINTER   */
1236  4D4      2 EDER12P  POINTER(31),        /*  GENERAL ERROR POINTER   */
1240  4D8      2 EDER13P  POINTER(31),        /*  GENERAL ERROR POINTER   */
1244  4DC      2 EDER14P  POINTER(31),        /*  GENERAL ERROR POINTER   */
1248  4E0      2 *        CHARACTER(16),      /*  EXTRA AREA              */
               /* — EDTPDCLS ——————————————————————————————————————————— */
1264  4F0      2 EDTPROF  CHAR(8),            /* PROFILE (D.S.TYPE) NAME  */
1272  4F8      2 EDTEOPN  CHAR(24),           /* PROFILE OPTIONS NEW      */
1296  510      2 EDTEOPC  CHAR(24),           /* PROFILE OPTIONS CURRENT  */
1320  528      2 EDTMASKC CHAR(1),            /* MASK CHANGED 'Y' OR 'N'  */
1321  529      2 EDTTABSC CHAR(1),            /* TABS CHANGED 'Y' OR 'N'  */
1322  52A      2 EDTMISCC CHAR(1),            /* MISC CHANGED 'Y' OR 'N'  */
1323  52B      2 EDTPROFL CHAR(1),            /* PROF LRECL   (1 TO 255)  */
1324  52C      2 EDTPROFR CHAR(1),            /* PROF RECFM   'F' OR 'V'  */
1325  52D      2 EDTPROFD CHAR(1),            /* PROF DEFAULT 'C' OR      */
                                              /*   'C' - USE CURR AS DEF  */
                                              /*   ELSE USE STD DEFAULT   */
1326  52E      2 *        CHARACTER(2),       /* EXTRA AREA               */
               /* — EDTRDCLS ——————————————————————————————————————————— */
             /* EDR VARIABLES                                           */
1328  530      2 EDGMSIZE FIXED(31),          /* GETMAIN (EDR BLOCK) SIZE */
1332  534      2 EDEDRSZ  FIXED(31),          /* EDR SIZE (LRECL + BASE)  */
1336  538      2 EDTEDRO  FIXED(31),          /* EDR OFFSET(LENGTH-EDRBASE)*/
1340  53C      2 EDTLRECL FIXED(31),          /* LRECL (DATA SIZE IN EDR) */
1344  540      2 EDTEDRCT FIXED(31),          /* EDR COUNT (ON CHAIN)     */
1348  544      2 EDPXNUMB FIXED(31),          /* PREV XCLUDED EDR NUMB    */
1352  548      2 EDTSTDCT FIXED(31),          /* STD EDR COUNT (ON CHAIN) */
1356  54C      2 EDTDELTA FIXED(31),          /* SEQUENCE NUMBERING DELTA */
1360  550      2 EDRDAP   POINTER(31),        /* EDR DISPLAY ARRAY PTR    */
1364  554      2 EDTEDRBS CHARACTER(20),      /* ORED INTO EDRBASE BY EDI */
1384  568      2 EDCOLS,                      /*                          */
1384  568        3 EDCOL    (8,2) FIXED(31),  /* COLUMN (TYPE, L|R) ARRAY */
1448  5A8      2 EDRP     (30) POINTER(31),   /* EDR (ED REC) PTRS ARRAY  */
                              -5A8 EDTOP      /* 1-> TOP (COMMAND AREA)   */
                              -5AC EDBOT      /* 2-> BOTTOM (COMMAND AREA) */
                              -5B0            /* 3-> INSERT               */
                              -5B4            /* 4-> EXCLUDED             */
                              -5B8            /* 5-> CHANGED              */
                              -5BC            /* 6-> CHANGE NOT DONE      */
                              -5C0            /* 7-> SHIFT INCOMPLETE     */
                              -5C4            /* 8-> BOUNDS               */
                              -5C8            /* 9-> COLUMNS              */
                              -5CC            /* 10->MASK                 */
5EC  EDCSRINF                 -5D0            /* 11->TABS                 */
                              -5D4            /* 12->INPUT TAB ERROR      */
                              -5D8            /* 13->MESSAGE LINE         */
                               5DC            /* 14->OPTION LINE 1        */
                               5E0            /* 15->OPTION LINE 2        */
                              -5E4            /* 16->OPTION LINE 3        */
                               5E8            /* 17->TEXT ENTRY           */
1568  620      2 EDCAREA  (20) CHARACTER(6),  /* CMD AREA ARRAY           */
1688  698      2 *        CHARACTER(24),      /*                          */
               /* — EDTSDCLS ——————————————————————————————————————————— */
1712  6B0      2 *         BIT(64),           /*  EDIT STATUS/CNTL BITS:  */
1712  6B0        3 EDCSTAT  BIT(8),           /*    CURRENT STATUS BITS:  */
1712  6B0.0        4 *        BIT(5),         /*                          */
1712  6B0.5        4 EDCTOP   BIT(1),         /*       F/C TOP OF DATA    */
1712  6B0.6        4 EDCBOT   BIT(1),         /*       F/C BOTTOM OF DATA */
1712  6B0.7        4 EDCWRAP  BIT(1),         /*       F/C WRAP AROUND    */
1713  6B1        3 EDPSTAT  BIT(8),           /*    PREVIOUS STATUS BITS: */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET     FIELD                                FIELD
DEC   HEX    NAME                                 DESCRIPTION
----  ----  ---------------------------------    ------------------

1713  6B1.0       4 *        BIT(5),           /*                           */
1713  6B1.5       4 EDPTOP   BIT(1),           /*       F/C TOP OF DATA     */
1713  6B1.6       4 EDPBOT   BIT(1),           /*       F/C BOTTOM OF DATA */
1713  6B1.7       4 EDPWRAP  BIT(1),           /*       F/C WRAP AROUND     */
1714  6B2       3 *        BIT(8),             /*    MISC STATUS BITS:      */
1714  6B2.0       4 EDBNDMOD BIT(1),           /*      BOUNDS MODIFIED      */
1714  6B2.1       4 *        BIT(1),           /*      ** RESERVED **       */
1714  6B2.2       4 EDCSRSET BIT(1),           /*      CSR (TLDCSRP) SET    */
1714  6B2.3       4 EDNEWMEM BIT(1),           /*      NEW MEMBER           */
1714  6B2.4       4 EDTCHGED BIT(1),           /*      ACTIVE SOURCE CHGED */
1714  6B2.5       4 EDTSAVED BIT(1),           /*      DATA HAS BEEN SAVED */
1715  6B3       3 *        BIT(8),             /*    ** RESERVED **         */
1716  6B4       3 EDCMBITS BIT(16),            /*  COPY/MOVE STATUS BITS:   */
1716  6B4.0       4 EDSBEFOR BIT(1),           /*    BEFORE DESTN DEFINED   */
1716  6B4.1       4 EDSAFTER BIT(1),           /*    AFTER DESTN DEFINED    */
1716  6B4.2       4 EDSOVER  BIT(1),           /*    OVER DESTN DEFINED     */
1716  6B4.3       4 EDSOVER1 BIT(1),           /*    SECOND OVER EFINED     */
1716  6B4.4       4 EDSOVER2 BIT(1),           /*    SECOND OVER EFINED     */
1716  6B4.5       4 *        BIT(3),           /*    ** RESERVED **         */
1717  6B5.0       4 EDSCOPY1 BIT(1),           /*    COPY 1 DEFINED         */
1717  6B5.1       4 EDSCOPY2 BIT(1),           /*    COPY 2 DEFINED         */
1717  6B5.2       4 EDSMOVE1 BIT(1),           /*    MOVE 1 DEFINED         */
1717  6B5.3       4 EDSMOVE2 BIT(1),           /*    MOVE 2 DEFINED         */
1717  6B5.4       4 EDSBLOCK BIT(1),           /*    BLOCK CMD DECODED      */
1717  6B5.5       4 *      BIT(3),             /*    ** RESERVED **         */
1718  6B6       3 *        BIT(8),             /*    ** RESERVED **         */
1719  6B7       3 *        BIT(8),             /*    ** RESERVED **         */
1720  6B8     2 *,                             /*                           */
1720  6B8       3 EDHTCHAR FIXED(8),           /* HARDWARE TABS CHAR (*)    */
1721  6B9       3 EDCTCHAR FIXED(8),           /* SOFTWARD CURSOR CHAR (-)  */
1722  6BA       3 EDC2CHAR FIXED(8),           /* SOFTWARD CURSOR CHAR (_)  */
1723  6BB       3 EDLBCHAR FIXED(8),           /* LEFT BOUND CHAR  (<)      */
1724  6BC       3 EDRBCHAR FIXED(8),           /* RIGHT BOUND CHAR (>)      */
1725  6BD       3 *        CHAR(3),            /* ** RESERVED **            */
1728  6C0       3 EDTDSORG CHARACTER(1),       /* DSORG (P-PDS, S-SEQ)      */
1729  6C1       3 EDTRECFM CHARACTER(1),       /* RECFM (F-FIX, V-VAR)      */
1730  6C2       3 EDTSDEX,                     /* SPF DIR ENTRY DATA        */
1730  6C2         4 EDTSSMEM CHARACTER(1),     /*   SPF STATS EXIST 'Y','N' */
1731  6C3         4 EDTVLCUR FIXED(8),         /*   VERSION LEVEL - CURR    */
1732  6C4         4 EDTMLCUR FIXED(8),         /*   MOD LEVEL - CURR        */
1733  6C5       3 EDTABEND CHARACTER(1),       /* EDO ABEND OCCURRED (Y/N)  */
1734  6C6       3 *        CHAR(2),            /* ** RESERVED **            */
1736  6C8       3 EDTPARMI CHARACTER(1),       /* EDTPARMV INITIALIZED(Y/N) */
1737  6C9       3 EDTRDTOP CHARACTER(1),       /* REDISPLAY TOP 2 LINES     */
                                               /*   ('Y' - YES, ELSE NO)    */
1738  6CA       3 *        CHAR(6),            /* ** RESERVED **            */
1744  6D0       3 EDTEDIFG CHAR(8),            /* CODES SET BY EDI BASED    */
                                               /*  ON THE DATA IT READS     */
                                               /*   " "-DATA IGNORED        */
                                               /*   "Y"-YES,"N"-NO          */
                                               /*   "M"-MAYBE               */
                                               /*    (NUMERIC NOT ASCENDING)*/
1744  6D0         4 EDTEDIIC CHARACTER(1),     /*  INVALID CHAR  'Y' OR 'N' */
1745  6D1         4 EDTEDILC CHARACTER(1),     /*  LOWER CASE    'Y' OR 'N' */
1746  6D2         4 EDTEDIUC CHARACTER(1),     /*  UPPER CASE    'Y' OR 'N' */
1747  6D3         4 EDTEDIML CHARACTER(1),     /*  VALID MOD LVL 'Y' OR 'N' */
1748  6D4         4 EDTEDIRC CHARACTER(1),     /*  VALID REASON  'Y' OR 'N' */
1749  6D5         4 EDTEDICN CHARACTER(1),     /*  COBOL NUMB 'Y' 'N' 'M'   */
1750  6D6         4 EDTEDI6N CHARACTER(1),     /*  STD 6 NUMB 'Y' 'N' 'M'   */
1751  6D7         4 EDTEDI8N CHARACTER(1),     /*  STD 8 NUMB 'Y' 'N' 'M'   */
1752  6D8       3 EDTRESET CHARACTER(1),       /*   EGR RESET TYPE CODE     */
                                               /*     'I'-INIT, 'F'-FINAL   */
1753  6D9       3 EDTNUMBR CHARACTER(1),       /*   EGN NUMBER TYPE CODE    */
                                               /*     'N'-NUMB, 'R'-RENUM   */
                                               /*     ' '-UNNUM             */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET       FIELD                                  FIELD
DEC   HEX       NAME                                   DESCRIPTION
____  _____   ____________________________________    __________________________

1754  6DA       3 EDTEDODF CHARACTER(1),          /*   EDO: DEL/FREE EDRS(Y,N) */
                                                  /*      ('Y' - YES, ELSE NO) */
1755  6DB       3 EDTEFROD CHAR(1),               /*   EFR - ORIGINAL DATA READ*/
                                                  /*      ('Y' - YES, ELSE NO) */
1756  6DC       3 EDTREASN CHAR(2),               /*   REASON CODE             */
                                                  /*      BLANK, OR CODE       */
1758  6DE       3 *        CHAR(2),               /* ** RESERVED **            */
1760  6E0     2 EDTEOPB  CHARACTER(24),           /* EOP BACKUP FOR CHECKING   */
1784  6F8     2 *        CHARACTER(24);           /* EXTRA AREA                */
```

ELC   - EDIT LINE COMMAND   ELC

```
 OFFSET      FIELD                                      FIELD
DEC  HEX      NAME                                       DESCRIPTION
---  ---   ----------------------------------          ------------------------
  0   0    1 ELC      BASED,                  /*                           */
  0   0      2 ELCCODE  FIXED(8),             /*   COMMAND CODE            */
  1   1      2 ELCPASS  FIXED(8),             /*   PASS TO BE EXEC (1-3)   */
  2   2      2 ELCTYPE  BIT(8),               /*   COMMAND TYPE BITS       */
  2   2.0      3 ELCAFTER BIT(1),             /*      AFTER COMMAND        */
  2   2.1      3 ELCBEFOR BIT(1),             /*      BEFORE COMMAND       */
  2   2.2      3 ELCCOPY  BIT(1),             /*      COPY COMMAND         */
  2   2.3      3 ELCMOVE  BIT(1),             /*      MOVE COMMAND         */
  2   2.4      3 ELCMULTI BIT(1),             /*      MULTI-LINE TYPE CMD  */
  2   2.5      3 ELCBLOCK BIT(1),             /*      BLOCK TYPE COMMAND   */
  2   2.6      3 *        BIT(1),             /*      ** RESERVED **       */
  2   2.7      3 ELCOVER  BIT(1),             /*      OVER COMMAND         */
  3   3      2 ELCCSR   BIT(8),               /*   CURSOR POSITIONING BITS */
  3   3.0      3 ELCNEXT  BIT(1),             /*      NEXT (AFTER LAST) EDR*/
  3   3.1      3 ELCLAST  BIT(1),             /*      LAST EDR             */
  3   3.2      3 ELCFIRST BIT(1),             /*      FIRST EDR            */
  3   3.3      3 ELCPREV  BIT(1),             /*      PREV (TO FIRST) EDR  */
  3   3.4      3 *        BIT(3),             /*      * RESERVED           */
  3   3.7      3 ELCPLUS1 BIT(1),             /*      PLUS 1 EDR AT DISPLAY*/
  4   4      2 ELCSUFFX FIXED(8),             /*   SUFFIX DEFAULT VALUE    */
  5   5      2 ELCLMASK BIT(8),               /*   LINE (TYPE) MASK        */
  6   6      2 ELCPRSUB FIXED(8);             /*   PROCESS SUBROUTINE INDEX*/
  OFFSET      FIELD                            FIELD
 DEC   HEX     NAME                             DESCRIPTION
----  -----  ------------------------------    ------------------------------

   0     0   1 MHAF BASED,                   /* MHA BUFFER FOR MENU           */
   0     0     2 MHAFSPLN,                   /* BUFFER SUBPOOL & LENGTH       */
   0     0       3 MHAFSP   FIXED(8),        /*   SUBPOOL                     */
   1     1       3 MHAFLN   FIXED(24),       /*   LENGTH                      */
   4     4     2 MHAFNAME CHAR(8),           /* NAME OF MENU IN BUFFER        */
  12     C     2 MHAFHELP CHAR(8),           /* PRIMARY HELP NAME             */
  20    14     2 MHAFPARM (103) PTR(31),     /* SAVED MHA PARAM LIST          */
 432   1B0     2 MHAFMME  PTR(31),           /* PTR TO END OF MODEL MENU + 1  */
 436   1B4     2 MHAFACTS PTR(31),           /* PTR TO START OF ACTION TABLE  */
 440   1B8     2 MHAFACTE PTR(31),           /* LAST ENTRY OF ACTION TABLE    */
 444   1BC     2 MHAFNRP  FIXED(31),         /* NUMBER OF SUBSTITUTION PARAMS */
 448   1C0     2 MHAFACTT (100)              /* ACTION STMT TABLE AREA        */
                 CHAR(LENGTH(MHAFACTN))      /*                               */
                 BOUNDARY(WORD),             /*                               */
3648   E40     2 MHAFADA  CHAR(*);           /* AREA TO STORE ACTION DATA     */
                                             /* ACTUAL SIZE IF MHAFADA IS     */
                                             /* DETERMINED BY CONSTANT IN     */
                                             /* MHA NAMED MHAFADAL            */
```

MHAFACTN – ACTION STATEMENT TABLE ENTRY

```
  OFFSET      FIELD                            FIELD
 DEC   HEX     NAME                             DESCRIPTION
----  -----  ------------------------------    ------------------------------

   0     0   1 MHAFACTN CHAR(32)             /* ACTION STATEMENT TABLE ENTRY  */
               BOUNDARY(WORD) BASED,         /*  (ONE PER ACTION STATEMENT)   */
   0     0     2 MHAFAPLP FIXED(15),         /* PARAMETER LIST POSITION       */
   2     2     2 MHAFAPLN FIXED(8),          /* PARAMETER LENGTH              */
   3     3     2 MHAFABTS BIT(8),            /* BIT FLAGS                     */
   3   3.0       3 MHAFAFXD BIT(1),          /*   FIXED  (ELSE CHAR)          */
   3   3.1       3 MHAFCURS BIT(1),          /*   CURSOR PARAM                */
   3   3.2       3 MHAFNOCU BIT(1),          /*   NOCURSOR PARAM              */
   3   3.3       3 MHAFRJ   BIT(1),          /*   INITR PARAM (RIGHT JUSTIFY) */
   3   3.4       3 *        BIT(4),          /*   SPARE                       */
   4     4     2 MHAFAFCP FIXED(15),         /* INPUT FIELD CHAR POSITION     */
   6     6     2 *        CHAR(2),           /* SPARE                         */
   8     8     2 MHAFAIP  PTR(31),           /* ADDR OF INIT DATA             */
  12     C     2 MHAFALP  PTR(31),           /* ADDR OF LIST DATA             */
  16    10     2 MHAFARP  PTR(31),           /* ADDR OF RETURN DATA           */
  20    14     2 MHAFALN  FIXED(15),         /* NUMBER OF LIST VALUES         */
  22    16     2 MHAFARN  FIXED(15),         /* NUMBER OF RETURN VALUES       */
  24    18     2 MHAFAKEY CHAR(8);           /* KEY NAME USED BY FOR & JOB    */
```

SDE - SPF DIRECTORY ENTRY SDE

```
 OFFSET     FIELD                                FIELD
DEC  HEX      NAME                                 DESCRIPTION
---  ---    ------------------------------       -------------------------
  0    0  1 SDE      BASED,                  /* SPF DIRECTORY ENTRY       */
  0    0    2 SDEVERS  FIXED(8),             /*   VERSION LEVEL           */
  1    1    2 SDEMOD   FIXED(8),             /*   MOD LEVEL               */
  2    2    2 SDERESV  FIXED(16),            /*   * RESERVED              */
  4    4    2 SDECDATE CHAR(4),              /*   CREATION DATE           */
  8    8    2 SDEMDATE CHAR(4),              /*   DATE LAST MODIFIED      */
 12    C    2 SDEMTIME CHAR(2),              /*   TIME LAST MODIFIED      */
 14    E    2 SDECLINE FIXED(16),            /*   CURRENT NUMBER OF LINES */
 16   10    2 SDEILINE FIXED(16),            /*   INITIAL NUMBER OF LINES */
 18   12    2 SDEMLINE FIXED(16),            /*   NUMBER OF MODIFIED LINES*/
 20   14    2 SDEID    CHAR(7),              /*   USER ID                 */
 27   1B    2 SDEBLANK CHAR(3);              /*   * RESERVED (BLANKS)     */
```

TCS - COMMAND SCAN TABLE

```
 OFFSET       FIELD                                  FIELD
DEC   HEX       NAME                                   DESCRIPTION
___   ____    ____________________________________    _________________________

  0     0     1 TCS       BASED,                  /*                          */
  0     0       2 TCSCIP  PTR(31),                /*   PTR TO COMMAND INPUT    */
  4     4       2 TCSCISZ FIXED(31),              /*   COMMAND INPUT SIZE      */
  8     8       2 TCSWDCT FIXED(31),              /*   COMMAND INPUT WORD COUNT*/
 12     C       2 TCSTCDP PTR(31),                /*   COMMAND DEF ENTRY PTR   */
 16    10       2 TCSWDS (12) CHAR(16);           /*   PRIMARY CMD WORDS(ARRAY)*/
```

TCSWD - COMMAND SCAN TABLE ENTRY

```
 OFFSET       FIELD                                  FIELD
DEC   HEX       NAME                                   DESCRIPTION
___   ____    ____________________________________    _________________________

  0     0     1 TCSWD BASED(ADDR(TCSWDS)),        /*                          */
  0     0         3 TCSWDSZ  FIXED(8),            /*      SIZE OF TOTAL WORD   */
  1     1         3 TCSWDP    PTR(24),            /*      PTR TO TOTAL WORD    */
  4     4         3 TCSSTRSZ FIXED(8),            /*      SIZE OF STRING       */
  5     5         3 TCSSTRP  PTR(24),             /*      PTR TO STRING        */
  8     8         3 TCSTYPE  BIT(8),              /*      FLAG BITS            */
  8   8.0           4 TCSQUOT  BIT(1),            /*        QUOTED STRING ("|')*/
  8   8.1           4 *        BIT(1),            /*        ** RESERVED **     */
  8   8.2           4 TCSNUMB  BIT(1),            /*        NUMERIC WORD       */
  8   8.3           4 TCSHEX   BIT(1),            /*        HEX STRING         */
  8   8.4           4 TCSPICT  BIT(1),            /*        PICTURE STRING     */
  8   8.5           4 TCSTEXT  BIT(1),            /*        TEXT STRING        */
  8   8.6           4 *        BIT(2),            /*        * RESERVED         */
  9     9         3 TCSCODE FIXED(8),             /*     COMMAND WORD KEYCODE*/
 10     A         3 TCSERR BIT(8),                /*   ERROR FLAG BITS         */
 10   A.0           4 TCSIVSIZ BIT(1),            /*     HEX STRING ODD SIZE   */
 10   A.1           4 TCSIVHEX BIT(1),            /*     NON HEX DIGITS IN STR */
 10   A.2           4 TCSIVNCQ BIT(1),            /*     INVAL NO CLOSING QUOTE*/
 10   A.3           4 *        BIT(5),            /*     ** RESERVED **        */
 11     B         3 *      CHAR(1),               /*     ** RESERVED **        */
 12     C         3 *         FIXED(31);          /*     ** RESERVED **        */
```

TCT - CONTROL TABLES TABLE TCT

```
 OFFSET       FIELD                                   FIELD
DEC   HEX      NAME                                    DESCRIPTION
___   ___   _________________________________      _________________________

  0     0   1 TCT       BASED,                  /* TRANSLATE TABLES TABLE    */
  0     0     2 *          CHAR(8),             /*   TCT IDENTIFICATION      */
  8     8     2 TCTTRTPS,                       /*   PTRS FROM TRT           */
  8     8       3 TCTLOCP PTR(31),              /*     LOC TBL PTR           */
 12     C       3 TCTATTP PTR(31),              /*     ATT TBL PTR           */
 16    10       3 TCTAIDP PTR(31),              /*     AID TBL PTR           */
 20    14       3 *       PTR(31),              /*     ** RESERVED **        */
 24    18       3 *       PTR(31),              /*     ** RESERVED **        */
 28    1C     2 TCTTTTPS,                       /*   PTRS FROM TTT           */
 28    1C       3 TCTUPPP PTR(31),              /*     UPP TBL PTR           */
 32    20       3 TCTLOWP PTR(31),              /*     LOW TBL PTR           */
 36    24       3 TCTVALP PTR(31),              /*     VAL TBL PTR           */
 40    28       3 TCTBTOP PTR(31),              /*     BTO TBL PTR           */
 44    2C       3 TCTETOP PTR(31),              /*     ETO TBL PTR           */
 48    30       3 TCTGSCP PTR(31),              /*     GSC TBL PTR           */
 52    34       3 TCTGSMP PTR(31),              /*     GSM TBL PTR           */
 56    38       3 TCTGSSP PTR(31),              /*     GSS TBL PTR           */
 60    3C       3 TCTEDIP PTR(31),              /*     EDI TBL PTR           */
 64    40       3 TCTEDOP PTR(31),              /*     EDO TBL PTR           */
 68    44       3 *       PTR(31),              /*     ** RESERVED **        */
 72    48       3 *       PTR(31);              /*     ** RESERVED **        */
```

TDS - DATA SET TABLE

OFFSET		FIELD	FIELD
DEC	HEX	NAME	DESCRIPTION
0	0	1 TDS BASED,	/* */
0	0	2 * CHAR(8),	/* TDS IDENTIFICATION */
8	8	2 TDSPARMP PTR(31),	/* PARMS TFD PTR */
12	C	2 TDSPROCP PTR(31),	/* PROCS TFD PTR */
16	10	2 TDSMENUP PTR(31),	/* MENUS TFD PTR */
20	14	2 TDSMSGSP PTR(31),	/* MSGS TFD PTR */
24	18	2 TDSLISTP PTR(31),	/* LIST TFD PTR */
28	1C	2 TDSLOGP PTR(31),	/* LOG TFD PTR */
32	20	2 TDSEBUAP PTR(31),	/* EDIT BACKUP "A" TFD PTR */
36	24	2 TDSEBUBP PTR(31),	/* EDIT BACKUP "B" TFD PTR */
40	28	2 * PTR(31),	/* ** RESERVED ** */
44	2C	2 * PTR(31);	/* ** RESERVED ** */

```
 OFFSET      FIELD                                 FIELD
DEC  HEX      NAME                                  DESCRIPTION
___  ____   ________________________________     ________________________

  0   0    1 TFD      BASED BOUNDARY(WORD),     /*                          */
  0   0      2 TFDHDRID CHARACTER(4),           /* 'TFD:'                   */
  4   4      2 TFDDDN   (4) CHARACTER(8),       /* ARRAY OF DDNAMES         */
 36  24      2 TFDSTAT1 BIT(8),                 /* CTA/CTF STATUS BITS:     */
 36  24.0      3 TFDREST  BIT(1),               /*   RESTART FLAG           */
 36  24.1      3 TFDALLOC BIT(1),               /*   ALLOCATED (TRIED)      */
 36  24.2      3 TFDPREAL BIT(1),               /*   PREALLOCATED           */
 36  24.3      3 TFDOLD   BIT(1),               /*   ALLOCATED OLD          */
 36  24.4      3 TFDRECVR BIT(1),               /*   RECOVER FROM ABEND     */
 36  24.5      3 TFDDCBAB BIT(1),               /*   DCB ABEND OCCURRED     */
 36  24.6      3 TFDCTAX  BIT(1),               /*   CTA FAILED             */
 36  24.7      3 *        BIT(1),               /*   RESERVED               */
 37  25      2 TFDSTAT2 BIT(8),                 /* INPUT STATUS FOR CDO/CDC */
 37  25.0      3 TFDDCBI  BIT(1),               /*   DCB PRE-INITIALIZED    */
 37  25.1      3 TFDVAL   BIT(1),               /*   VALIDITY CK REQUESTED  */
 37  25.2      3 TFDPDS   BIT(1),               /*   PDS OK SWITCH          */
 37  25.3      3 TFDSEQ   BIT(1),               /*   SEQ OK SWITCH          */
 37  25.4      3 TFDRFU   BIT(1),               /*   RECFM=U OK SWITCH      */
 37  25.5      3 TFDRFV   BIT(1),               /*   RECFM=V OK SWITCH      */
 37  25.6      3 TFDRFF   BIT(1),               /*   RECFM=F OK SWITCH      */
 37  25.7      3 TFDNOBUF BIT(1),               /*   BYPASS BUFFER GETMAIN  */
 38  26      2 TFDSTAT3 BIT(8),                 /* CDO/CDC/CRESV/CRELS SW   */
 38  26.0      3 TFDOPN   BIT(1),               /*   ON IF OPENED BY CDO    */
 38  26.1      3 TFDESEQ  BIT(1),               /*   EMPTY INPUT SEQ DS     */
 38  26.2      3 TFDRESV  BIT(1),               /*   RESERVE REQUEST SWITCH */
 38  26.3      3 TFDLENQ  BIT(1),               /*   LINK EDIT ENQ REQUIRED */
 38  26.4      3 TFDRFMO  BIT(1),               /*   RECFM OVERRIDE SWITCH  */
 38  26.5      3 TFDDEQS  BIT(1),               /*   CRELS DEQ SYSTEMS SWITCH*/
 38  26.6      3 *        BIT(2),               /*   RESERVED               */
 39  27      2 TFDSTAT4 BIT(8),                 /* CDG/CDP STATUS BITS:     */
 39  27.0      3 TFDRREQ  BIT(1),               /*   CDG READ-REQUEST SW    */
 39  27.1      3 TFDUPPL  BIT(1),               /*   CDG UPDATE-IN-PLACE SW */
 39  27.2      3 TFDNOTE  BIT(1),               /*   CDG/CDP NOTE REQUIRED  */
 39  27.3      3 TFDECOD5 BIT(1),               /*   CDG ENTRY CODE 5 SW    */
 39  27.4      3 *        BIT(4),               /*   RESERVED               */
 40  28      2 TFDDCBP  POINTER(31),            /* DCB POINTER              */
 40  28        3 TFDDCB#  FIXED(8),             /*   DCB NUMBER         *OOS*/
 41  29        3 *        POINTER(24),          /*                          */
 44  2C      2 TFDDSNP  (4) POINTER(31),        /* TFDDSNS POINTER          */
 60  3C      2 TFDMENUP POINTER(31),            /* TFDMENUD POINTER         */
 64  40      2 TFDDECB  CHARACTER(24),          /* DECB                     */
 88  58      2 TFDXLP   POINTER(31),            /* DCB EXIT LIST POINTER    */
 92  5C      2 TFDCML   POINTER(31),            /* POINTER TO MEMBER LIST   */
 96  60      2 TFDBLDLP POINTER(31),            /* PTR TO BLDL LIST, OR 0   */
                                                /* CDG/CDP PARAMETERS:      */
100  64      2 TFDECODE FIXED(15),              /*   ENTRY CODE             */
102  66      2 TFDRECL  FIXED(15),              /*   RECORD LENGTH          */
104  68      2 TFDRECP  POINTER(31),            /*   RECORD POINTER         */
108  6C      2 TFDTTRN  CHARACTER(4),           /*   1ST OR CURRENT TTRN    */
112  70      2 TFDBUFA  POINTER(31),            /*   BUFFER A ADDRESS       */
116  74      2 TFDBO    FIXED(15),              /*   BUFFER OFFSET          */
118  76      2 TFDBLKS  FIXED(15),              /*   ACTUAL BLOCK SIZE READ */
120  78      2 TFDLRECL FIXED(15),              /* SPF LOGICAL REC LENGTH   */
122  7A      2 TFDBLKSZ FIXED(15),              /* SAME AS DCBBLKSI         */
124  7C      2 TFDMAXLN FIXED(15),              /* MAXIMUM LRECL ALLOWED    */
126  7E      2 TFDRECFM BIT(8),                 /* RECFM (SAME AS DCBRECFM): */
126  7E.0      3 TFDRFTYP BIT(2),               /*   TYPE OF RECORD FORMAT  */
                                                /*     SEE CONSTANTS BELOW  */
126  7E.2      3 TFDRFT     BIT(1),             /*   TRACK OVERFLOW         */
126  7E.3      3 TFDRFB     BIT(1),             /*   BLOCKED                */
126  7E.4      3 TFDRFS     BIT(1),             /*   STANDARD OR SPANNED    */
126  7E.5      3 TFDRFCTL BIT(2),               /*   PRINT CONTROL TYPE     */
                                                /*     SEE CONSTANTS BELOW  */
126  7E.7      3 TFDRFK     BIT(1),             /*   KEY LENGTH SPECIFIED   */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET     FIELD                                     FIELD
DEC  HEX      NAME                                      DESCRIPTION

127  7F     2 TFDMACRF BIT(16),                    /* MACRF (SAME AS DCBMACRF): */
127  7F       3 * BIT(8),                          /*   INPUT SWITCHES          */
127  7F.0       4 *      BIT(2),                   /*                           */
127  7F.2       4 TFDMRRD BIT(1),                  /*     READ                  */
127  7F.3       4 *      BIT(2),                   /*                           */
127  7F.5       4 TFDMRPT1 BIT(1),                 /*     POINT                 */
127  7F.6       4 *      BIT(2),                   /*                           */
128  80       3 * BIT(8),                          /*   OUTPUT SWITCHES         */
128  80.0       4 *      BIT(2),                   /*                           */
128  80.2       4 TFDMRWRT BIT(1),                 /*     WRITE                 */
128  80.3       4 *      BIT(2),                   /*                           */
128  80.5       4 TFDMRPT2 BIT(1),                 /*     POINT                 */
128  80.6       4 *      BIT(2),                   /*                           */
129  81     2 TFDDSORG BIT(8),                     /* DSORG (SAME AS DA08DSO):  */
129  81.0     3 TFDIS    BIT(1),                   /*   INDEX SEQUENTIAL        */
129  81.1     3 TFDPS    BIT(1),                   /*   PHYSICAL SEQUENTIAL     */
129  81.2     3 TFDDO    BIT(1),                   /*   DIRECT                  */
129  81.3     3 TFDLG    BIT(1),                   /*   BTAM/QTAM LINE GROUP    */
129  81.4     3 TFDDAMQ  BIT(1),                   /*   QTAM DA MESSAGE QUEUE   */
129  81.5     3 TFDPPMQ  BIT(1),                   /*   QTAM PROB PROG MSG QUE  */
129  81.6     3 TFDPO    BIT(1),                   /*   PARTITIONED             */
129  81.7     3 TFDUM    BIT(1),                   /*   UNMOVABLE               */
                                                   /* REQUESTED DISPOSITION:   */
130  82     2 TFDDSP1  BIT(8),                     /*   STATUS (LIKE DA08DSP1) */
131  83     2 TFDDSP2  BIT(8),                     /*   DISP (LIKE DA18DPS2)   */
132  84     2 TFDSTAT5 BIT(8),                     /* STATUS BITS FOR CDA/CDF:  */
132  84.0     3 TFDOTHER BIT(1),                   /*   "OTHER" DATASET         */
132  84.1     3 TFDSPVOL BIT(1),                   /*   VOL SER SPECIFIED       */
132  84.2     3 TFDALLCT BIT(1),                   /*   CDA DID AN ALLOC        */
132  84.3     3 TFDALIAS BIT(1),                   /*   DSN CHANGED TO REAL DSN */
132  84.4     3 *        BIT(4),                   /*   RESERVED                */
133  85     2 TFDDISP  BIT(8),                     /* CURRENT DISP STATUS       */
134  86     2 TFDOPENT BIT(8),                     /* OPEN OPTIONS              */
135  87     2 TFDCLOST BIT(8),                     /* CLOSE OPTIONS             */
136  88     2 TFDMEMB  CHARACTER(8),               /* SPECIFIED MEMBER NAME     */
144  90     2 TFDPASSW CHARACTER(8),               /* PASSWORD                  */
152  98     2 TFDUNIT  CHARACTER(8),               /* UNIT TYPE FROM CDAIR      */
160  A0     2 TFDVOL   CHARACTER(6),               /* VOLUME SERIAL             */
166  A6     2 *        CHARACTER(2);               /* ** RESERVED **            */
```

TFDMENUD - DATA SET MENU INFORMATION

```
 OFFSET     FIELD                                     FIELD
DEC  HEX      NAME                                      DESCRIPTION

  0    0   1 TFDMENUD BASED,                       /* DATASET MENU INFORMATION  */
  0    0     2 TFDPROJ  CHARACTER(8),              /*   PROJECT NAME            */
  8    8     2 TFDLIBS  CHARACTER(32),             /*   PROJECT LIBRARIES:      */
  8    8       3 TFDLIB   (4) CHARACTER(8),        /*     LIB1,LIB2,LIB3,LIB4   */
 40   28     2 TFDTYPE  CHARACTER(8),              /*   DATASET TYPE            */
 48   30     2 TFDMEM   CHARACTER(8),              /*   PDS MEMBER NAME         */
 56   38     2 TFDODSN  CHARACTER(56),             /*   "OTHER" DSN(MEMBER)     */
112   70     2 TFDOVOL  CHARACTER(6),              /*   "OTHER" DATASET VOLUME  */
118   76     2 TFDPSWD  CHARACTER(8);              /*   PASSWORD                */
```

```
 OFFSET      FIELD                                   FIELD
DEC   HEX     NAME                                    DESCRIPTION
----  ----  -------------------------------------   -------------------------
  0     0   1 TLD      BASED,                     /* LOGICAL DISPLAY          */
  0     0     2 TLDTBLID CHAR(3),                 /*   TABLE ID 'TLD'         */
  3     3     2 TLDID    CHAR(1),                 /*   TLD ID ('0','1','2')   */
  4     4     2 TLDNEXTP PTR(31),                 /* NEXT TLD PTR             */
  8     8     2 TLDRC    FIXED(31),               /* RETURN CODE              */
 12     C     2 TLDPCODE CHAR(6),                 /* PROLOG CODE INTERFACE    */
 18    12     2 TLDECODE CHAR(6),                 /* EPILOG CODE INTERFACE    */
 24    18     2 *        PTR(31),                 /* ** RESERVED **           */
 28    1C     2 *        PTR(31),                 /* ** RESERVED **           */
 32    20     2 TLDSA    (10) FIXED(32),          /* SAVEAREA INTERFACE       */
 72    48     2 *,                                /* CONTROL/PROCESS INTERFACE */
 72    48       3 TLDTCBP  PTR(31),               /*   TCB (TASK CNTL BLK) PTR */
 76    4C       3 TLDDRECB FIXED(32),             /*   DISPLAY REQUEST ECB    */
 80    50       3 TLDPRECB FIXED(32),             /*   PROCESS REQUEST ECB    */
 84    54       3 TLDTCECB FIXED(32),             /*   TASK COMPLETION ECB    */
 88    58       3 TLDCCI   FIXED(31),             /*   INTERFACE FOR CONTROL  */
 92    5C       3 TLDSXECB PTR(31),               /*   STAX POST ECB          */
 96    60       3 TLDSXTCB PTR(31),               /*   STAX POST TCB          */
100    64       3 *        PTR(31),               /*   ** RESERVED **         */
104    68     2 TLDSTBLS,                         /* SPF SYSTEM TABLES PTRS   */
104    68       3 TLDTCMP  PTR(31),               /*   TCM (CMD TABLE) ENTRY PT*/
108    6C       3 TLDTCTP  PTR(31),               /*   TCT (CONT TBLS TBL) PTR */
112    70       3 TLDTDSP  PTR(31),               /*   TDS (DATA SETS) PTR    */
116    74       3 TLDTKVP  PTR(31),               /*   TKV (KEY/VALUE TBL) PTR */
120    78       3 TLDTKWP  PTR(31),               /*   TKW (KEY-WORD TBLS) PTR */
124    7C       3 TLDTSCP  PTR(31),               /*   TSC (COMMON SUBS) PTR  */
128    80       3 TLDTSIP  PTR(31),               /*   TSI (SYS INTERFACE) PTR */
132    84       3 TLDTSVP  PTR(31),               /*   TSV (SPF VARIABLES) PTR */
136    88       3 TLDTXCP  PTR(31),               /*   TXC (TERM EXIT TBL) PTR */
140    8C       3 *        PTR(31),               /*   ** RESERVED **         */
144    90     2 *,                                /* SPF PROCESSOR TABLES PTRS */
144    90       3 TLDMHABP PTR(31),               /*   MENU HANDLER BUFFER PTR */
148    94       3 TLDTFDCP PTR(31),               /*   TFD CONTROL CARD PTR   */
152    98       3 TLDTFDLP PTR(31),               /*   TFD LISTING PTR        */
156    9C       3 TLDTFDEP PTR(31),               /*   TFD EDIT BACKUP PTR    */
160    A0       3 TLDTFKP  PTR(31),               /*   TFK (FUNCT/KEY) PTR    */
164    A4       3 TLDTLSP  PTR(31),               /*   TLS (LOGIC SCREEN) PTR  */
168    A8       3 TLDTRT0P PTR(31),               /*   TRT ZEROS TBL PTR      */
172    AC       3 TLDTRT1P PTR(31),               /*   TRT IDENTITY TBL PTR   */
176    B0       3 TLDTADP  PTR(31),               /*   TAD (ALLOC DDNAMES) PTR */
180    B4       3 *        PTR(31),               /*   ** RESERVED **         */
184    B8       3 TLDUSER1 PTR(31),               /*   USER FIELD # 1         */
188    BC       3 TLDUSER2 PTR(31),               /*   USER FIELD # 2         */
192    C0       3 TLDUSER3 PTR(31),               /*   USER FIELD # 3         */
196    C4     2 *,                                /* DISPLAY/SCREEN INTERFACE */
196    C4       3 TLDFUNC  BIT(64),               /*   KEY FUNCTION BITS      */
196    C4.0       4 TLDNOPK  BIT(1),              /*     NOP                  */
196    C4.1       4 TLDREDK  BIT(1),              /*     REDISPLAY            */
196    C4.2       4 TLDSPLK  BIT(1),              /*     SPLIT                */
196    C4.3       4 TLDSWPK  BIT(1),              /*     SWAP                 */
196    C4.4       4 TLDCSRK  BIT(1),              /*     CURSOR               */
196    C4.5       4 TLDPRTHK BIT(1),              /*     PRINT HIGH           */
196    C4.6       4 TLDPRTLK BIT(1),              /*     PRINT LOW            */
196    C4.7       4 *        BIT(1),              /*     ** RESERVED **       */
197    C5.0       4 TLDLNEK  BIT(1),              /*     LINE COMMAND         */
197    C5.1       4 TLDCMDK  BIT(1),              /*     PRIMARY COMMAND      */
197    C5.2       4 TLDNOPMK BIT(1),              /*     NOP WITH MESSAGE     */
197    C5.3       4 TLDHLPK  BIT(1),              /*     HELP                 */
197    C5.4       4 TLDRETK  BIT(1),              /*     RETURN               */
197    C5.5       4 *        BIT(3),              /*     ** RESERVED **       */
198    C6.0       4 TLDENDK  BIT(1),              /*     END                  */
198    C6.1       4 TLDENTK  BIT(1),              /*     ENTER                */
198    C6.2       4 TLDSMMK  BIT(1),              /*     ENTER/SESS MGR MODE  */
198    C6.3       4 *        BIT(5),              /*     ** RESERVED **       */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET      FIELD                                FIELD
DEC   HEX      NAME                                 DESCRIPTION
----  ----  -----------------------------------     --------------------------

  0    0    1 TKV BASED,                          /*                          */
  0    0      2 TKVHEAD,                          /* TKV HEADER               */
  0    0        3 TKVID     CHAR(8),              /*   IDENTIFICATION         */
  0    0          4 *         CHAR(6),            /*     TABLE ID             */
  6    6          4 TKVVID    CHAR(2),            /*     SPF VERSION ID       */
  8    8        3 TKVLEN    FIXED(15),            /*   TOTAL LENGTH OF TKV    */
 10    A        3 TKVUSED   FIXED(15),            /*   OFFSET TO LAST USED BYTE*/
 12    C        3 TKVFIXED FIXED(15),             /*   OFFSET PAST LAST FIXED */
 14    E        3 TKVFLAGS BIT(16),               /*   FLAGS                  */
 14    E.0        4 *         BIT(6),             /*     ** RESERVED **       */
 14    E.6        4 TKVNOUPD BIT(1),              /*     ON -> NOT WRITEABLE  */
 14    E.7        4 TKVNV2    BIT(1),             /*     ON -> NOT VER 2      */
                                                  /*       (VER 2.2 OR GREATER)*/
 15    F.0        4 *         BIT(7),             /*     ** RESERVED **       */
 15    F.7        4 TKVFULL   BIT(1),             /*     ON -> TKV OVERFILLED */
 16   10        3 *         FIXED(31),            /*   ** RESERVED **         */
 20   14      2 TKVENTS   CHAR(*),                /* FIRST KEYWORD/VALUE ENTRY */
```

TKVENTRY - KEYWORD-VALUE TABLE ENTRY (TKVDCLS)

```
 OFFSET      FIELD                                FIELD
DEC   HEX      NAME                                 DESCRIPTION
----  ----  -----------------------------------     --------------------------

  0    0    1 TKVENTRY BASED,                     /* FORMAT OF KEYWD-VAL ENTRY */
  0    0      2 TKVEHEAD,                         /* ENTRY HEADER             */
  0    0        3 TKVVALLN FIXED (8),             /* LENGTH OF VALUE          */
  1    1        3 TKVNAMLN FIXED (8),             /* LENGTH OF KEYWORD        */
  2    2      2 TKVNAME   CHAR (*);               /* KEYWORD NAME             */
 OFFSET      FIELD                                   FIELD
DEC  HEX      NAME                                     DESCRIPTION
---  ---   ----------------------------------         ---------------------------

277  115      3 TLDTFKID CHAR(1),              /*    TFK ID (A,B,C)            */
278  116      3 TLDTCBID CHAR(1),              /*    TCB ID (0,1,2)        *OOS*/
279  117      3 TLDFORBT CHAR(1),              /*    OPT 4/5 FLAG (F,B)    *CMS*/
280  118      3 TLDPMENU CHAR(8),              /*    PREV MENU FROM CDISPL     */
288  120      3 TLDPMSG  CHAR(4),              /*    LAST MSG FROM CMSG        */
292  124      3 TLDCSMCA PTR(31),              /*    CSM AUTO STOR ADDR        */
296  128      3 TLDTMBP  PTR(31),              /*    TASK MGT BLOCK PTR    *CMS*/
300  12C      3 TLDBATCT FIXED(8),             /*    BATCH NAME GEN NUM    *CMS*/
301  12D      3 *        CHAR(3),              /*    ** RESERVED **            */
304  130      3 *        PTR(31),              /*    ** RESERVED **            */
308  134      3 *        PTR(31),              /*    ** RESERVED **            */
312  138      3 *        PTR(31),              /*    ** RESERVED **            */
316  13C      3 *        PTR(31);              /*    ** RESERVED **            */
```

```
 OFFSET     FIELD                               FIELD
DEC   HEX    NAME                                DESCRIPTION
----  ----  ---------------------------------   ----------------

199   C7.0       4 TLDSCRK  BIT(4),          /*    SCROLL KEYS          */
199   C7.0         5 TLDSCRUK BIT(1),        /*      SCROLL UP          */
199   C7.1         5 TLDSCRDK BIT(1),        /*      SCROLL DOWN        */
199   C7.2         5 TLDSCRLK BIT(1),        /*      SCROLL LEFT        */
199   C7.3         5 TLDSCRRK BIT(1),        /*      SCROLL RIGHT       */
199   C7.4       4 TLDFNDK  BIT(1),          /*    REPEAT FIND          */
199   C7.5       4 TLDCHGK  BIT(1),          /*    REPEAT CHANGE        */
199   C7.6       4 *        BIT(34),         /*    ** RESERVED **       */
204   CC       3 TLDENBL  BIT(64),           /*  ENABLED KEY FUNCT BITS */
204   CC.0       4 TLDNOPE  BIT(1),          /*    NOP                  */
204   CC.1       4 TLDREDE  BIT(1),          /*    REDISPLAY            */
204   CC.2       4 TLDSPLE  BIT(1),          /*    SPLIT                */
204   CC.3       4 TLDSWPE  BIT(1),          /*    SWAP                 */
204   CC.4       4 TLDCSRE  BIT(1),          /*    CURSOR               */
204   CC.5       4 TLDPRTHE BIT(1),          /*    PRINT HIGH           */
204   CC.6       4 TLDPRTLE BIT(1),          /*    PRINT LOW            */
204   CC.7       4 *        BIT(1),          /*    ** RESERVED **       */
205   CD.0       4 TLDLNEE  BIT(1),          /*    LINE COMMAND         */
205   CD.1       4 TLDCMDE  BIT(1),          /*    PRIMARY COMMAND      */
205   CD.2       4 TLDNOPME BIT(1),          /*    NOP WITH MESSAGE     */
205   CD.3       4 TLDHLPE  BIT(1),          /*    HELP                 */
205   CD.4       4 TLDRETE  BIT(1),          /*    RETURN               */
205   CD.5       4 *        BIT(3),          /*    ** RESERVED **       */
206   CE.0       4 TLDENDE  BIT(1),          /*    END                  */
206   CE.1       4 TLDENTE  BIT(1),          /*    ENTER                */
206   CE.2       4 TLDSMME  BIT(1),          /*    ENTER/SESS MGR MODE  */
206   CE.3       4 *        BIT(5),          /*    ** RESERVED **       */
207   CF.0       4 TLDSCRE  BIT(4),          /*    SCROLL KEYS          */
207   CF.0         5 TLDSCRUE BIT(1),        /*      SCROLL UP          */
207   CF.1         5 TLDSCRDE BIT(1),        /*      SCROLL DOWN        */
207   CF.2         5 TLDSCRLE BIT(1),        /*      SCROLL LEFT        */
207   CF.3         5 TLDSCRRE BIT(1),        /*      SCROLL RIGHT       */
207   CF.4       4 TLDFNDE  BIT(1),          /*    REPEAT FIND          */
207   CF.5       4 TLDCHGE  BIT(1),          /*    REPEAT CHANGE        */
207   CF.6       4 *        BIT(34),         /*    ** RESERVED **       */
212   D4       3 TLDAID   BIT(8),            /*  LOGICAL ATTENTION ID   */
213   D5       3 *        BIT(16),           /*  MISC FUNCTION BITS     */
213   D5.0       4 TLDALARM BIT(1),          /*    ALARM BIT            */
213   D5.1       4 TLDTUT   BIT(1),          /*    TUTORIAL FLAG (TUT)  */
213   D5.2       4 TLDNODSP BIT(1),          /*    NO DISPLAY REQUESTED */
213   D5.3       4 TLDTFKLK BIT(1),          /*    TFK LOCK BIT         */
213   D5.4       4 TLDMERRC BIT(1),          /*    MERR CALLING MHA     */
213   D5.5       4 TLDPRIOP BIT(1),          /*    PRIM OPT FLAG (PMD)  */
213   D5.6       4 TLDSAFLG BIT(1),          /*    SCROLL AMT FLAG      */
213   D5.7       4 TLD998   BIT(1),          /*    998 ABEND RESTART FLAG*/
214   D6.0       4 TLDSTAX  BIT(1),          /*    CAT STAX FLAG        */
214   D6.1       4 TLDSTAX6 BIT(1),          /*    OPTION 6 STAX FLAG   */
214   D6.2       4 TLDDFREE BIT(1),          /*    CAT DD FREE SWITCH   */
214   D6.3       4 *        BIT(5),          /*    ** RESERVED **       */
215   D7       3 *        BIT(8),            /*  ** RESERVED **         */
216   D8       3 TLDCSR   FIXED(31),         /*  CURSOR (REL-LOC)       */
220   DC       3 TLDCPSRL FIXED(31),         /*  CURR PHYS SCR REL-LOC  */
224   E0       3 TLDCPSSZ FIXED(31),         /*  CURR PHYS SCREEN USED  */
228   E4       3 TLDMLSSZ FIXED(31),         /*  MAX LOGIC SCREEN SIZE  */
232   E8       3 TLDCLSSZ FIXED(31),         /*  CURR LOGIC SCREEN SIZE */
236   EC       3 TLDMDTCT FIXED(31),         /*  MODIFIED DATA TAG CNT  */
240   F0       3 *        FIXED(31),         /*  ** RESERVED **         */
244   F4     2 *,                            /* MISCELLANEOUS INTERFACE */
244   F4       3 TLDPOPTN CHAR(8),           /*  RETURN (PRIMARY OPTION) */
252   FC       3 TLDHELP  CHAR(8),           /*  CURR HELP (MEMBER) NAME */
260   104      3 TLDSCAMT CHAR(4),           /*  CURRENT SCROLL AMOUNT  */
264   108      3 TLDREPCT FIXED(31),         /*  REPEAT END COUNT       */
268   10C      3 TLDIDABP PTR(31),           /*  PRIM INPUT FLD ATTR BYTE*/
272   110      3 TLDINSIZ FIXED(31),         /*    PTR AND DATA LEN     */
276   114      3 TLDEBUID CHAR(1),           /*  EDIT BACKUP TFD ID     */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET     FIELD                                 FIELD
DEC  HEX      NAME                                  DESCRIPTION
___  ___   ____________________________________  _________________________
  0    0   1 TPD      BASED                      /* PHYSICAL DISPLAY TABLE    */
                      BOUNDARY(DWORD),           /*                           */
  0    0     2 *        CHAR(8),                 /* TPD IDENTIFICATION        */
  8    8     2 *        BOUNDARY(DWORD),         /* TIMES (IN MICRO-SECONDS)  */
  8    8       3 TPDLASTT CHAR(8),               /*   LAST TIME (FROM STCK)   */
 16   10       3 TPDTPUTT CHAR(8),               /*   TPUT TIME (TOTAL)       */
 24   18       3 TPDPROCT CHAR(8),               /*   PROCESSING TIME (TOTAL) */
 32   20       3 TPDTGETT CHAR(8),               /*   TGET TIME (TOTAL)       */
 40   28     2 TPDAID   BIT(8),                  /* LOGICAL ATTENTION AID     */
 41   29     2 *        BIT(8),                  /* CONTROL BITS              */
 41   29.0     3 TPDRDISP BIT(1),                /*   REDISPLAY SCREEN        */
 41   29.1     3 TPDRSTPS BIT(1),                /*   RESTORE TPS             */
 41   29.2     3 TPDRDLIO BIT(1),                /*   REDISPLAY AFTER LINE I/O*/
 41   29.3     3 TPDMSGON BIT(1),                /*   SMC ERR/INFO MSG FLAG   */
 41   29.4     3 TPDMSGAL BIT(1),                /*   SMC ERR/INFO MSG ALARM  */
 41   29.5     3 TPDFSINT BIT(1),                /*   FULL SCREEN INIT    *CMS*/
 41   29.6     3 *        BIT(2),                /*   ** RESERVED **          */
 42   2A     2 *        CHAR(2),                 /* ** RESERVED **            */
 44   2C     2 TPDCSR   FIXED(31),               /* CURSOR (REL-LOC)          */
 48   30     2 *,                                /* PHYS SCREEN INTERFACE     */
 48   30       3 TPDTPSSZ FIXED(31),             /*   TPS (PHYS SCREEN) SIZE  */
 52   34       3 TPDTPSP  PTR(31),               /*   TPS (PHYS SCREEN) PTR   */
 56   38       3 *        FIXED(31),             /*   ** RESERVED **          */
 60   3C     2 *,                                /* SCREEN BUFFER INTERFACE   */
 60   3C       3 TPDTSBSZ FIXED(31),             /*   TSB (SCREEN BUFF) SIZE  */
 64   40       3 TPDTSBP  PTR(31),               /*   TSB (SCREEN BUFF) PTR   */
 68   44       3 *        FIXED(31),             /*   ** RESERVED **          */
 72   48     2 *,                                /* LOGICAL DISPL INTERFACE   */
 72   48       3 TPDTLDC  FIXED(31),             /*   TLD (LOGIC DISPL) CNT   */
 76   4C       3 TPDFTLDP PTR(31),               /*   FIRST TLD PTR           */
 80   50       3 TPDPTLDP PTR(31),               /*   PRIMARY TLD PTR         */
 84   54       3 TPDDTLDP PTR(31),               /*   DUMMY (SMC'S) TLD PTR   */
 88   58       3 *        FIXED(31),             /*   ** RESERVED **          */
 92   5C     2 *,                                /* PHYSICAL SCREEN I/O COUNTS*/
 92   5C       3 TPDPUTCT FIXED(31),             /*   TPUT COUNT              */
 96   60       3 TPDPUTSZ FIXED(31),             /*   TOTAL TPUT BYTES        */
100   64       3 TPDGETCT FIXED(31),             /*   TGET COUNT              */
104   68       3 TPDGETSZ FIXED(31),             /*   TOTAL TGET BYTES        */
108   6C     2 TPDMSGP  PTR(31),                 /* SMC ERROR/INFO MSG PTR    */
112   70     2 TPDCSMGP PTR(31),                 /* CLEAR SCREEN TPUT DATA    */
116   74     2 TPDCSMGL PTR(31),                 /* CLEAR SCREEN TPUT DATA LEN*/
120   78     2 *        PTR(31),                 /* ** RESERVED **            */
124   7C     2 *        PTR(31);                 /* ** RESERVED **            */
```

```
 OFFSET      FIELD                                FIELD
DEC  HEX       NAME                                DESCRIPTION
___  ____  ___________________________________    ___________________________

  0    0   1 TLS      BASED,                     /* LOGICAL SCREEN              */
  0    0     2 TLSHEAD,                          /*  HEADER AREA OF DISPLAY     */
  0    0       3 TLST     CHAR(80),              /*   TITLE LINE                */
  0    0         4 TLSTAB   BIT(8),              /*     TITLE ATTR BYTE         */
  1    1         4 TLSTITLE CHAR(55),            /*     TITLE DATA              */
  1    1           5 TLSTFUNC CHAR(7),           /*       FUNCTION (EDIT/BRO) */
  8    8           5 TLSTDSN  CHAR(48),          /*       DATASET NAME          */
 56   38         4 TLSMSG   CHAR(24),            /*     MESSAGE DATA            */
 56   38           5 *        CHAR(7),           /*         (DASHES)            */
 63   3F           5 TLSCLABL CHAR(7),           /*       COLUMN LABEL          */
 70   46           5 *        CHAR(1),           /*         (BLANK)             */
 71   47           5 TLSLCOL  CHAR(3),           /*       LEFT COLUMN           */
 74   4A           5 *        CHAR(1),           /*         (BLANK)             */
 75   4B           5 TLSRCOL  CHAR(3),           /*       RIGHT COLUMN          */
 78   4E           5 *        CHAR(2),           /*         (BLANKS)            */
 80   50       3 TLSI     CHAR(80),              /*   INPUT LINE                */
 80   50         4 *        CHAR(62),            /*     PRIMARY INPUT AREA      */
142   8E         4 TLSS,                         /*     SCROLL FIELDS           */
142   8E           5 TLSSLAB  BIT(8),            /*       LABEL ATTR BYTE       */
143   8F           5 TLSSLABL CHAR(11),          /*       SCROLL LABEL          */
154   9A           5 TLSSAAB  BIT(8),            /*       AMOUNT ATTR BYTE      */
154   9A.0           6 *        BIT(2),          /*         *                   */
154   9A.2           6 TLSSAMDT BIT(1),          /*         MOD DATA TAG        */
154   9A.3           6 *        BIT(5),          /*         *                   */
155   9B           5 TLSSAMT  CHAR(4),           /*       SCROLL AMOUNT         */
159   9F           5 TLSSFAB  BIT(8),            /*       FINAL ATTR BYTE       */
160   A0     2 TLSBODY  CHAR(3280),              /*  BODY AREA OF DISPLAY       */
160   A0       3 TLSHELP  CHAR(80),              /*   HELP LINE                 */
160   A0         4 TLSHLPAB BIT(8),              /*     HELP ATTR BYTE          */
161   A1         4 TLSHLPDA CHAR(77),            /*     HELP DATA               */
238   EE         4 TLSHLPA2 BIT(8),              /*     HELP END ATTR BYTE      */
239   EF         4 TLSHLPEN CHAR(1),             /*     HELP END DATA           */
240   F0       3 TLSTEXT  CHAR(*);               /*   MAIN TEXT AREA            */
```

OFFSET DEC	 HEX	FIELD NAME		FIELD DESCRIPTION	
400	190	2 TSCPTR(098) PTR(31),	/*	ADDRESS OF EPP	*/
404	194	2 TSCPTR(099) PTR(31),	/*	ADDRESS OF EPO	*/
408	198	2 TSCPTR(100) PTR(31),	/*	ADDRESS OF EPS	*/
412	19C	2 TSCPTR(101) PTR(31),	/*	ADDRESS OF ETX	*/
416	1A0	2 TSCPTR(102) PTR(31),	/*	ADDRESS OF ETC	*/
428	1AC	2 TSCPTR(105) PTR(31),	/*	ADDRESS OF EPR	*/
432	1B0	2 TSCPTR(106) PTR(31),	/*	ADDRESS OF EDI	*/
436	1B4	2 TSCPTR(107) PTR(31),	/*	ADDRESS OF EDO	*/
440	1B8	2 TSCPTR(108) PTR(31),	/*	ADDRESS OF EFR	*/
444	1BC	2 TSCPTR(109) PTR(31),	/*	ADDRESS OF EPD	*/
448	1C0	2 TSCPTR(110) PTR(31),	/*	ADDRESS OF EPC	*/
452	1C4	2 TSCPTR(111) PTR(31),	/*	ADDRESS OF EPI	*/
456	1C8	2 TSCPTR(112) PTR(31),	/*	ADDRESS OF EPF	*/
460	1CC	2 TSCPTR(113) PTR(31),	/*	ADDRESS OF EFC	*/
468	1D4	2 TSCPTR(115) PTR(31),	/*	ADDRESS OF EGN	*/
472	1D8	2 TSCPTR(116) PTR(31),	/*	ADDRESS OF EGR	*/
476	1DC	2 TSCPTR(117) PTR(31),	/*	ADDRESS OF EML	*/
480	1E0	2 TSCPTR(118) PTR(31),	/*	ADDRESS OF EFT	*/
484	1E4	2 TSCPTR(119) PTR(31),	/*	ADDRESS OF EST	*/
488	1E8	2 TSCPTR(120) PTR(31),	/*	ADDRESS OF EBI	*/
492	1EC	2 TSCPTR(121) PTR(31),	/*	ADDRESS OF EBS	*/
496	1F0	2 TSCPTR(122) PTR(31),	/*	ADDRESS OF EBX	*/
500	1F4	2 TSCPTR(123) PTR(31),	/*	ADDRESS OF EBE	*/
504	1F8	2 TSCPTR(124) PTR(31),	/*	ADDRESS OF EBA	*/
508	1FC	2 TSCPTR(125) PTR(31),	/*	ADDRESS OF EBR	*/
528	210	2 TSCPTR(130) PTR(31),	/*	ADDRESS OF ECD	*/
532	214	2 TSCPTR(131) PTR(31),	/*	ADDRESS OF ETL	*/
548	224	2 TSCPTR(135) PTR(31),	/*	ADDRESS OF ERA	*/
552	228	2 TSCPTR(136) PTR(31),	/*	ADDRESS OF ERC	*/
556	22C	2 TSCPTR(137) PTR(31),	/*	ADDRESS OF ERD	*/
560	230	2 TSCPTR(138) PTR(31),	/*	ADDRESS OF ERF	*/
564	234	2 TSCPTR(139) PTR(31),	/*	ADDRESS OF ERI	*/
568	238	2 TSCPTR(140) PTR(31),	/*	ADDRESS OF ERN	*/
572	23C	2 TSCPTR(141) PTR(31),	/*	ADDRESS OF ERO	*/
576	240	2 TSCPTR(142) PTR(31),	/*	ADDRESS OF ERR	*/
580	244	2 TSCPTR(143) PTR(31),	/*	ADDRESS OF ERS	*/
584	248	2 TSCPTR(144) PTR(31),	/*	ADDRESS OF ERX	*/

OFFSET DEC	HEX	FIELD NAME	FIELD DESCRIPTION
0	0	1 TSC BASED BOUNDARY(WORD),	/* TSC (SYS COMMON SUBS) TBL */
0	0	2 TSCID CHAR(8),	/* IDENTIFICATION "TSCX" */
8	8	2 TSCLEN FIXED(31),	/* LENGTH OF TCS */
12	C	2 TSCPTR(001) PTR(31),	/* ADDRESS OF CAT */
16	10	2 TSCPTR(002) PTR(31),	/* ADDRESS OF CBC */
20	14	2 TSCPTR(003) PTR(31),	/* ADDRESS OF CBDSN */
24	18	2 TSCPTR(004) PTR(31),	/* ADDRESS OF CBF */
28	1C	2 TSCPTR(005) PTR(31),	/* ADDRESS OF CBG */
32	20	2 TSCPTR(006) PTR(31),	/* ADDRESS OF CBR */
36	24	2 TSCPTR(007) PTR(31),	/* ADDRESS OF CBS */
40	28	2 TSCPTR(008) PTR(31),	/* ADDRESS OF CCB */
44	2C	2 TSCPTR(009) PTR(31),	/* ADDRESS OF CCD */
48	30	2 TSCPTR(010) PTR(31),	/* ADDRESS OF CCP */
52	34	2 TSCPTR(011) PTR(31),	/* ADDRESS OF CCS */
56	38	2 TSCPTR(012) PTR(31),	/* ADDRESS OF CDA */
60	3C	2 TSCPTR(013) PTR(31),	/* ADDRESS OF CDAIR */
64	40	2 TSCPTR(014) PTR(31),	/* ADDRESS OF CDATE */
68	44	2 TSCPTR(015) PTR(31),	/* ADDRESS OF CDC */
72	48	2 TSCPTR(016) PTR(31),	/* ADDRESS OF CDERR */
76	4C	2 TSCPTR(017) PTR(31),	/* ADDRESS OF CDF */
80	50	2 TSCPTR(018) PTR(31),	/* ADDRESS OF CDG */
84	54	2 TSCPTR(019) PTR(31),	/* ADDRESS OF CDISPL */
88	58	2 TSCPTR(020) PTR(31),	/* ADDRESS OF CDO */
92	5C	2 TSCPTR(021) PTR(31),	/* ADDRESS OF CDP */
96	60	2 TSCPTR(022) PTR(31),	/* ADDRESS OF CDT */
100	64	2 TSCPTR(023) PTR(31),	/* ADDRESS OF CERR */
104	68	2 TSCPTR(024) PTR(31),	/* ADDRESS OF CFI */
108	6C	2 TSCPTR(025) PTR(31),	/* ADDRESS OF CHC */
112	70	2 TSCPTR(026) PTR(31),	/* ADDRESS OF CHELP */
116	74	2 TSCPTR(027) PTR(31),	/* ADDRESS OF CHPJ */
120	78	2 TSCPTR(028) PTR(31),	/* ADDRESS OF CHPL */
124	7C	2 TSCPTR(029) PTR(31),	/* ADDRESS OF CIR */
128	80	2 TSCPTR(030) PTR(31),	/* ADDRESS OF CIV */
132	84	2 TSCPTR(031) PTR(31),	/* ADDRESS OF CJC */
136	88	2 TSCPTR(032) PTR(31),	/* ADDRESS OF CJF */
140	8C	2 TSCPTR(033) PTR(31),	/* ADDRESS OF CJN */
144	90	2 TSCPTR(034) PTR(31),	/* ADDRESS OF CKVGET */
148	94	2 TSCPTR(035) PTR(31),	/* ADDRESS OF CKVPUT */
152	98	2 TSCPTR(036) PTR(31),	/* ADDRESS OF CLM */
156	9C	2 TSCPTR(037) PTR(31),	/* ADDRESS OF CLOG */
160	A0	2 TSCPTR(038) PTR(31),	/* ADDRESS OF CMB */
168	A8	2 TSCPTR(040) PTR(31),	/* ADDRESS OF CML */
172	AC	2 TSCPTR(041) PTR(31),	/* ADDRESS OF CMSG */
176	B0	2 TSCPTR(042) PTR(31),	/* ADDRESS OF CPRINT */
180	B4	2 TSCPTR(043) PTR(31),	/* ADDRESS OF CRELS */
184	B8	2 TSCPTR(044) PTR(31),	/* ADDRESS OF CRESV */
188	BC	2 TSCPTR(045) PTR(31),	/* ADDRESS OF CSB */
192	C0	2 TSCPTR(046) PTR(31),	/* ADDRESS OF CSCROLL */
196	C4	2 TSCPTR(047) PTR(31),	/* ADDRESS OF CSM */
200	C8	2 TSCPTR(048) PTR(31),	/* ADDRESS OF CTA */
204	CC	2 TSCPTR(049) PTR(31),	/* ADDRESS OF CTF */
208	D0	2 TSCPTR(050) PTR(31),	/* ADDRESS OF CTGET */
212	D4	2 TSCPTR(051) PTR(31),	/* ADDRESS OF CTPUT */
216	D8	2 TSCPTR(052) PTR(31),	/* ADDRESS OF CT1 */
220	DC	2 TSCPTR(053) PTR(31),	/* ADDRESS OF CT2 */
224	E0	2 TSCPTR(054) PTR(31),	/* ADDRESS OF CUPARMS */
228	E4	2 TSCPTR(055) PTR(31),	/* ADDRESS OF CVM */
232	E8	2 TSCPTR(056) PTR(31),	/* ADDRESS OF CVSDE */
252	FC	2 TSCPTR(061) PTR(31),	/* ADDRESS OF BCD */
256	100	2 TSCPTR(062) PTR(31),	/* ADDRESS OF MERR */
260	104	2 TSCPTR(063) PTR(31),	/* ADDRESS OF MHA */
328	148	2 TSCPTR(080) PTR(31),	/* ADDRESS OF EX1 */
332	14C	2 TSCPTR(081) PTR(31),	/* ADDRESS OF EX2 */
392	188	2 TSCPTR(096) PTR(31),	/* ADDRESS OF ETS */
396	18C	2 TSCPTR(097) PTR(31),	/* ADDRESS OF EMP */

(CONTINUED ON NEXT PAGE)

OFFSET		FIELD		FIELD
DEC	HEX	NAME		DESCRIPTION
136	88	2 *,	/*	SPOOL CLASS *CMS*/
136	88	3 TSISPCL1 CHAR(1),	/*	FOR TLD1 *CMS*/
137	89	3 TSISPCL2 CHAR(1),	/*	FOR TLD2 *CMS*/
138	8A	3 * CHAR(2),	/*	** RESERVED ** *CMS*/
140	8C	2 TSIDDNN FIXED(31),	/*	DDNAME NUMBER (SPFXXXXX)*/
144	90	2 * (16) FIXED(31);	/*	** RESERVED ** */

```
 OFFSET       FIELD                                FIELD
DEC   HEX       NAME                                 DESCRIPTION

  0     0     1 TSI       BASED,               /* SYSTEM INTERFACE          */
  0     0       2 *         CHAR(8),           /*   TSI IDENTIFICATION      */
  8     8       2 TSIVRMZ  CHAR(8),            /*   SPF VER/REL/MOD/ZAP LVL */
 16    10       2 TSISYSP,                     /*   TSI PTRS IN TLD         */
 16    10         3 TSITCMP  PTR(31),          /*     CMD TABLE ENTRY POINT */
 20    14         3 TSITCTP  PTR(31),          /*     CONTLER TBLS TBL PTR  */
 24    18         3 TSITDSP  PTR(31),          /*     DATA SET TABLE PTR    */
 28    1C         3 TSITKVP  PTR(31),          /*     KEY/VAL TABLE PTR     */
 32    20         3 TSITKWP  PTR(31),          /*     KEYWORD TABLE PTR     */
 36    24         3 TSITSCP  PTR(31),          /*     COMMON SUB PTRS       */
 40    28         3 TSITSIP  PTR(31),          /*     SYS INTERFACE PTR     */
 44    2C         3 TSITSVP  PTR(31),          /*     SYS VARIABLES PTR     */
 48    30         3 TSITXCP  PTR(31),          /*     TERM EXIT TABLE PTR   */
 52    34         3 *        PTR(31),          /*     ** RESERVED **        */
 56    38       2 TSITPDP  PTR(31),            /*   PHYSICAL DISPLAY PTR    */
 60    3C       2 TSITFKP  PTR(31),            /*   CURRENT MASTER TFK PTR  */
 64    40       2 TSIDEVNP PTR(31),            /*   DEVICE NAME TABLE PTR   */
 68    44       2 TSITSOPL,                    /*   TSO PARM LIST           */
 68    44         3 TSICBUFP PTR(31),          /*     COMMAND BUFFER PTR    */
 72    48         3 TSIUPTP  PTR(31),          /*     USER PROFILE TBL PTR  */
 76    4C         3 TSIPSCBP PTR(31),          /*     PROT STEP CNTL BLK PTR*/
 80    50         3 TSIECTP  PTR(31),          /*     ENVIRON CNTL TBL PTR  */
 84    54       2 TSILDCBP PTR(31),            /*   SPFLIB DCB PTR          */
 88    58       2 TSIPFXP  PTR(31),            /*   DATA SET PREFIX PTR     */
 92    5C       2 TSIDAIRP PTR(31),            /*   PTR TO LOADED 'IKJEFD00'*/
 96    60       2 TSISDWAP PTR(31),            /*   SYS DIAGNOSTIC W.A. PTR */
100    64       2 TSITLMP  PTR(31),            /*   LOAD MODULE TAB PTR *CMS*/
104    68       2 TSICCFBP PTR(31),            /*   CCF EXT INT BUF PTR *CMS*/
108    6C       2 TSICDAF# FIXED(15),          /*   CDA DYNMC DDNAME NO *CMS*/
110    6E       2 TSI#FML  FIXED(8),           /*   NUM MODE LETS AVAIL *CMS*/
111    6F       2 *         FIXED(8),          /*   ** RESERVED **          */
112    70       2 *,                           /*   OPERATING SYSTEM        */
112    70.0       3 TSIMVS   BIT(1),           /*     ON -> MVS EXECUTING   */
112    70.1       3 TSISVS   BIT(1),           /*     ON -> SVS EXECUTING   */
112    70.2       3 TSIMVT   BIT(1),           /*     ON -> MVT EXECUTING   */
112    70.3       3 TSIOOS   BIT(1),           /*     ON -> OTHER OS EXEC   */
112    70.4       3 TSIAUTH  BIT(1),           /*     ON -> SPF AUTHORIZED  */
112    70.5       3 *        BIT(1),           /*     ** RESERVED **    *OOS*/
112    70.6       3 *        BIT(2),           /*     ** RESERVED **        */
113    71       2 *,                           /*   TERMINAL ACCESS         */
113    71.0       3 TSITCAM  BIT(1),           /*     ON -> TCAM INTERFACE  */
113    71.1       3 TSIVTAM  BIT(1),           /*     ON -> VTAM INTERFACE  */
113    71.2       3 TSIVMALT BIT(1),           /*     ON -> NDS ALT SZ  *CMS*/
113    71.3       3 TSIALT   BIT(1),           /*     ON -> NDS ALT SZ      */
113    71.4       3 TSITRACE BIT(1),           /*     ON -> TERM I/O TRACE  */
113    71.5       3 *        BIT(3),           /*     ** RESERVED **        */
114    72       2 TSIMODES CHAR(1),            /*   SPF MODES               */
114    72.0       3 TSIMSGMN BIT(1),           /*     ON -> MSG/MENU TESTING*/
114    72.1       3 TSISTEST BIT(1),           /*     OFF-> DO SPF STAX     */
114    72.2       3 *        BIT(1),           /*     ** RESERVED **        */
114    72.3       3 TSISDUMP BIT(1),           /*     ON -> ALLOW MAIN ABEND*/
114    72.4       3 TSIPDUMP BIT(1),           /*     ON -> ALLOW PROC ABEND*/
114    72.5       3 TSITREQ  BIT(1),           /*     ON -> TERM TRACE REQ  */
114    72.6       3 *        BIT(2),           /*     ** RESERVED **        */
115    73       2 TSITFKID CHAR(1),            /*   CURRENT MASTER TFK ID   */
116    74       2 TSIUSRID CHAR(8),            /*   TSO USERID (8 CHARS)    */
124    7C       2 TSISDATE CHAR(4),            /*   SESSION START DATE      */
128    80       2 TSISTIME CHAR(4),            /*   SESSION START TIME      */
132    84       2 *,                           /*   EDIT BACKUP TFD CONTROL */
132    84         3 TSIEBUAF FIXED(8),         /*     "A" TFD IN USE        */
133    85         3 TSIEBUBF FIXED(8),         /*     "B" TFD IN USE        */
134    86         3 *         FIXED(15),       /*     ** RESERVED **        */
```

(CONTINUED ON NEXT PAGE)

```
 OFFSET      FIELD                                   FIELD
DEC  HEX       NAME                                    DESCRIPTION
---  ---    ----------------------------------       ------------------------------

115  73.4     3 $FNDE    BIT(1),               /*   REPEAT FIND              */
115  73.5     3 $CHGE    BIT(1),               /*   REPEAT CHANGE            */
115  73.6     3 *        BIT(2),               /*   ** RESERVED **           */
116  74       3 *        BIT(32),              /*   ** RESERVED **           */
                                               /*                            */
                                               /* THE FOLLOWING IS SAVED IN */
                                               /* IN THE TKV                 */
120  78     2 $TSVKV   BOUNDARY(WORD),         /*                            */
120  78       3 $JCHAR   CHAR(1),              /* UNIQUE JOB CHARACTER       */
121  79       3 $SMM     CHAR(1),              /* SESS MGR MODE (Y,N)        */
122  7A       3 *        CHAR(1),              /* ** RESERVED **             */
123  7B       3 $MODE    CHAR(1),              /* CHAR MODE (M-MONO,D-DUAL) */
124  7C       3 $KEYS    CHAR(2),              /* NUM PF KEYS ('12'|'24')    */
126  7E       3 *        FIXED(15),            /* ** RESERVED **             */
128  80       3 $CHARLM  CHAR(8),              /* CHAR SET LOAD MOD NAME     */
136  88       3 *        FIXED(31),            /* ** RESERVED **             */
140  8C       3 $DATE    CHAR(4),              /* DATE & TIME PARMS LAST     */
144  90       3 $TIME    CHAR(4),              /*   STORED (TIME MACRO FORM)*/
148  94       3 $LST,                          /* SPFLIST INFORMATION        */
148  94         4 $LSTPQTY FIXED(31),          /*   PRIMARY QUANTITY (PAGES)*/
152  98         4 $LSTSQTY FIXED(31),          /*   SECONDARY QUANTITY (PGS)*/
156  9C         4 $LSTLPP  FIXED(31),          /*   LINES PER PAGE           */
160  A0         4 $LSTCHAR CHAR(1),            /*   DATA SET UNIQUE CHAR     */
161  A1         4 $LSTIDSP CHAR(1),            /*   INITIAL DISP (FROM PMD) */
162  A2         4 $LSTFDSP CHAR(1),            /*   FINAL DISP (FROM OPT)    */
163  A3         4 $LSTKEPT CHAR(1),            /*   KEPT STATUS 'Y'->YES     */
164  A4         4 *        FIXED(31),          /*   ** RESERVED **           */
168  A8       3 $LOG,                          /* SPFLOG INFORMATION         */
168  A8         4 $LOGPQTY FIXED(31),          /*   PRIMARY QUANTITY (PAGES)*/
172  AC         4 $LOGSQTY FIXED(31),          /*   SECONDARY QUANTITY (PGS)*/
176  B0         4 $LOGLPP  FIXED(31),          /*   LINES PER PAGE           */
180  B4         4 $LOGCHAR CHAR(1),            /*   DATA SET UNIQUE CHAR     */
181  B5         4 $LOGIDSP CHAR(1),            /*   INITIAL DISP (FROM PMD) */
182  B6         4 $LOGFDSP CHAR(1),            /*   FINAL DISP (FROM OPT)    */
183  B7         4 $LOGKEPT CHAR(1),            /*   KEPT STATUS 'Y'->YES     */
184  B8         4 *        FIXED(31),          /*   ** RESERVED **           */
188  BC       3 $USESTAT,                      /* USAGE STATISTICS           */
188  BC         4 $SESSION FIXED(31),          /*   COUNT OF SPF SESSIONS    */
192  C0         4 $INCNT   FIXED(31),          /*   COUNT INPUT OPERATIONS   */
196  C4         4 $INBYTES FIXED(31),          /*   INPUT BYTE COUNT(TOTAL) */
200  C8         4 $OUTCNT  FIXED(31),          /*   COUNT OUTPUT OPERATIONS */
204  CC         4 $OUTBYTE FIXED(31),          /*   OUTPUT BYTE COUNT(TOTAL)*/
208  D0         4 $PROCTIM FIXED(31),          /*   PROCESSING (TOTAL SECS) */
212  D4         4 $USERTIM FIXED(31),          /*   USER-THINK (TOTAL SECS) */
216  D8         4 $LOGCNT  FIXED(31),          /*   LOG COUNT                */
220  DC         4 *        FIXED(31),          /*   ** RESERVED **           */
224  E0         4 *        FIXED(31),          /*   ** RESERVED **           */
228  E4       3 $TFK77   CHAR(36),             /* MASTER TFK FOR 3277        */
264  108      3 $TFK7812 CHAR(36),             /* MASTER TFK FOR 3278 12PFK */
300  12C      3 $TFK7824 CHAR(36),             /* MASTER TFK FOR 3278 24PFK */
336  150      3 $EBU,                          /* EDIT BACKUP RECOVER CNTL   */
336  150        4 $EBUA    FIXED(8),           /*   BACKUP "A" CONTROL       */
337  151        4 $EBUB    FIXED(8),           /*   BACKUP "B" CONTROL       */
338  152        4 *        FIXED(15),          /*   ** RESERVED **           */
340  154      3 $CHAR    CHAR(8),              /* CHAR SET NAME              */
348  15C      3 *        CHAR(1),              /* END OF TSV/TKV AREA        */
349  15D    2 *        CHAR(3),                /* ** RESERVED **             */
352  160    2 *        (4) FIXED(31);          /* ** RESERVED **             */
```

```
 OFFSET      FIELD                                FIELD
DEC  HEX      NAME                                 DESCRIPTION
___  ____   __________________________________    _________________________

  0    0   1 TSV     BASED,                      /* SYSTEM VARIABLES          */
  0    0     2 *          CHAR(8),               /* TSV IDENTIFICATION        */
  8    8     2 $MINTGET FIXED(32) UNSIGNED,      /* MINIMUM TGET DELAY TIME   */
 12    C     2 $SWAPLIN FIXED(15),               /* SWAP/SPLIT LINE CONTROL   */
 14    E     2 $SPCOUNT FIXED(15),               /* PRINT SCREEN CONTROL      */
 16   10     2 $SBAINCR FIXED(15),               /* 'SBA' OPTIMIZE INCR(4-255)*/
 18   12     2 $RAINCR  FIXED(15),               /* 'RA' OPTIMIZE INCR (1-255)*/
 20   14     2 $SETPAGE CHAR(4),                 /* FULL SCROLL VALUE FOR 'P' */
 24   18     2 $SETHALF CHAR(4),                 /* FULL SCROLL VALUE FOR 'H' */
 28   1C     2 $TCAMWCC CHAR(1),                 /* WCC - STD TPUT - TCAM     */
 29   1D     2 $VTAMWCC CHAR(1),                 /* WCC - STD TPUT - VTAM     */
 30   1E     2 $TCSWCC  CHAR(1),                 /* WCC - CLEAR SCREEN - TCAM */
 31   1F     2 $VCSWCC  CHAR(1),                 /* WCC - CLEAR SCREEN - VTAM */
 32   20     2 $TFSON   CHAR(2),                 /* SPEC SBA ADDRS TO TCAM TO */
 34   22     2 $TFSOFF  CHAR(2),                 /*   TURN FULL SCRN ON OR OFF*/
 36   24     2 $LINES   FIXED(8),                /* NUMB OF LINES FOR DISPLAY */
 37   25     2 *        CHAR(3),                 /* ** RESERVED **            */
 40   28     2 *        FIXED(31),               /* ** RESERVED **            */
 44   2C     2 $TPL,                             /* SPF TEMPLIST INFORMATION  */
 44   2C       3 $TPLPQTY FIXED(31),             /*   PRIMARY QUANTITY (BLKS) */
 48   30       3 $TPLSQTY FIXED(31),             /*   SECONDARY QUANTITY (BKS)*/
 52   34       3 $TPLBLKS FIXED(31),             /*   BLOCK SIZE              */
 56   38       3 *        FIXED(31),             /*   ** RESERVED **          */
 60   3C     2 $TPC,                             /* SPF TEMPCNTL INFORMATION  */
 60   3C       3 $TPCPQTY FIXED(31),             /*   PRIMARY QUANTITY (BLKS) */
 64   40       3 $TPCSQTY FIXED(31),             /*   SECONDARY QUANTITY (BKS)*/
 68   44       3 $TPCBLKS FIXED(31),             /*   BLOCK SIZE              */
 72   48       3 *        FIXED(31),             /*   ** RESERVED **          */
 76   4C     2 $EBUDS,                           /* SPF EDIT BACKUP INFO      */
 76   4C       3 $EBUPQTY FIXED(31),             /*   PRIMARY QUANTITY (BLKS) */
 80   50       3 $EBUBLKS FIXED(31),             /*   BLOCK SIZE              */
 84   54       3 $EBUSQTY FIXED(31),             /*   SECONDARY QUANTITY (BKS)*/
 88   58       3 *        FIXED(31),             /*   ** RESERVED **          */
 92   5C     2 *,                                /* LIST/LOG VALUES (NON TKV) */
 92   5C       3 $LSTBLKS FIXED(31),             /*   LIST BLOCK SIZE         */
 96   60       3 $LOGBLKS FIXED(31),             /*   LOG BLOCK SIZE          */
100   64       3 $LOGPG   FIXED(15),             /*   LOG PAGE NUMBER         */
102   66       3 $LOGLN   FIXED(15),             /*   LOG LINE NUMBER         */
104   68       3 $LOGDATE CHAR(4),               /*   LOG CURRENT DATE        */
108   6C       3 $LOGFLAG CHAR(1),               /*   LOG FLAG ('A' | ' ')    */
109   6D       3 *        CHAR(3),               /*   ** RESERVED **          */
112   70     2 $ENBL    BIT(64),                 /* INIT KEY/FUN ENABLE BITS  */
112   70.0     3 $NOPE    BIT(1),                /*   NOP                     */
112   70.1     3 $REDE    BIT(1),                /*   REDISPLAY               */
112   70.2     3 $SPLE    BIT(1),                /*   SPLIT                   */
112   70.3     3 $SWPE    BIT(1),                /*   SWAP                    */
112   70.4     3 $CSRE    BIT(1),                /*   CURSOR                  */
112   70.5     3 $PRTHE   BIT(1),                /*   PRINT HIGH              */
112   70.6     3 $PRTLE   BIT(1),                /*   PRINT LOW               */
112   70.7     3 *        BIT(1),                /*   ** RESERVED **          */
113   71.0     3 $LNEE    BIT(1),                /*   LINE COMMAND            */
113   71.1     3 $CMDE    BIT(1),                /*   PRIMARY COMMAND         */
113   71.2     3 $NOPME   BIT(1),                /*   NOP WITH MESSAGE        */
113   71.3     3 $HLPE    BIT(1),                /*   HELP                    */
113   71.4     3 $RETE    BIT(1),                /*   RETURN                  */
113   71.5     3 *        BIT(3),                /*   ** RESERVED **          */
114   72.0     3 $ENDE    BIT(1),                /*   END                     */
114   72.1     3 $ENTE    BIT(1),                /*   ENTER                   */
114   72.2     3 $SMME    BIT(1),                /*   ENTER/SESS MGR MODE     */
114   72.3     3 *        BIT(5),                /*   ** RESERVED **          */
115   73.0     3 $SCRUE   BIT(1),                /*   SCROLL UP               */
115   73.1     3 $SCRDE   BIT(1),                /*   SCROLL DOWN             */
115   73.2     3 $SCRLE   BIT(1),                /*   SCROLL LEFT             */
115   73.3     3 $SCRRE   BIT(1),                /*   SCROLL RIGHT            */
```

(CONTINUED ON NEXT PAGE)

UDACOMM - UDA COMMON AREA

```
 OFFSET      FIELD                                      FIELD
DEC   HEX     NAME                                       DESCRIPTION
____  _____  _______________________________________   _________________________

  0     0    1 UDACOMM  BASED                      /*                             */
                      BOUNDARY(WORD),              /*                             */
  0     0      2 UDACIVCP PTR(31),                 /* CIV COMMON AREA PTR         */
  4     4      2 UDAMENUP PTR(31),                 /* MENU PARMS PTR (0 IN TFD)   */
                                                   /*   (CDA USES TFDDSNS)        */
  8     8      2 UDAMID   CHAR(4),                 /* MSG ID FROM SUBROUTINE      */
 12     C      2 UDAMCL   FIXED(31),               /* MSG CURSOR LOC FROM SUB     */
 16    10      2 UDAP,                             /* MSG PARMS FOR MERR          */
 16    10        3 UDAP1    CHAR(80),              /*   MSG PARM 1 FROM SUB       */
 96    60        3 UDAP2    CHAR(80),              /*   MSG PARM 2 FROM SUB       */
176    B0        3 UDAP3    CHAR(80),              /*   MSG PARM 3 FROM SUB       */
256   100      2 UDAOPT   CHAR(1),                 /* SELECTED OPTION FROM MENU   */
257   101      2 UDABITS  BIT(8),                  /* BIT FLAGS                   */
257   101.0      3 UDASPF   BIT(1),                /*   ON IF SPF DATASET         */
257   101.1      3 UDARGN   BIT(1),                /*   GENERATE MENU INDICATOR   */
257   101.2      3 UDAERR   BIT(1),                /*   ERROR INDICATOR           */
257   101.3      3 UDARN    BIT(1),                /*   RENAME FLAG (FOR UAC)     */
257   101.4      3 UDACML   BIT(1),                /*   CML REQUEST (FOR UDMS)    */
257   101.5      3 UDAALARM BIT(1),                /*   ALARM FOR '0'X MSG        */
257   101.6      3 *        BIT(2),                /*   ** RESERVED **            */
258   102      2 *        CHAR(2),                 /* ** RESERVED **              */
260   104      2 UDAREN   CHAR(8),                 /* RENAME FIELD FROM MENU      */
268   10C      2 UDAMEM#  FIXED(31),               /* PARM NR OF MEMBER NAME      */
272   110      2 UDADSN#  FIXED(31),               /* PARM NR OF DATASET NAME     */
276   114      2 UDAZPGM  CHAR(8),                 /* "IEBCOPY" (FOR UDZ)         */
284   11C      2 *,                                /* UDM/UDMS VALUES             */
284   11C        3 UDACP    CHAR(8),               /*   "PPRINTED"                */
292   124        3 UDACR    CHAR(8),               /*   "RRENAMED"                */
300   12C        3 UDACD    CHAR(8),               /*   "DDELETED"                */
308   134        3 UDACB    CHAR(8),               /*   "B       "                */
316   13C        3 UDAPCT   FIXED(31),             /*   COUNT OF PRINTED MEMS     */
320   140        3 UDARCT   FIXED(31),             /*   COUNT OF RENAMED MEMS     */
324   144        3 UDADCT   FIXED(31),             /*   COUNT OF DELETED MEMS     */
328   148        3 UDAOTFDP PTR(31),               /*   PTR TO OUTPUT TFD         */
332   14C        3 UDACBRP  PTR(31),               /*   CBR WORK AREA PTR         */
336   150        3 *        FIXED(31),             /*   ** RESERVED **            */
340   154      2 *        FIXED(31),               /* ** RESERVED **              */
344   158      2 *        FIXED(31),               /* ** RESERVED **              */
348   15C      2 *        FIXED(31),               /* ** RESERVED **              */
352   160      2 *        FIXED(31);               /* ** RESERVED **              */
                                                   /*                             */
               /* NOTES:                                                          */
               /*   1. IF UDAMID = '0'X THEN                                      */
               /*        UDAP1 = SHORT MSG, UDAP2 = LOGN MSG,                      */
               /*        UDAP3(1:8) = TUT HELP MENU, AND                          */
               /*        UDAALARM = ALARM STATE                                   */
```

ENQ/DEQ LOGIC AND SHARED DASD SUPPORT

PASSWORD PROTECTED DATASETS

IT IS NOT RECOMMENDED THAT PASSWORD PROTECTED DATASETS BE PLACED ON SHARED DASD VOLUMES. SPF USERS UPDATING PASSWORD PROTECTED DATASETS ON SHARED DASD SHOULD BE CAUTIONED TO FILL IN THE PASSWORD FIELD ON THE SPF MENU TO AVOID BEING PROMPTED FOR A PASSWORD BY OPEN SINCE THE VOLUME IS RESERVED DURING OPEN PROCESSING.

DATA SET INTEGRITY ENQUE

THE SPF EDITOR, LIBRARY UTILITY, MOVE/COPY UTILITY, AND RESET STATISTICS UTILITY ALLOCATE PARTITIONED DATA SETS FOR "SHARED" USE EVEN THOUGH THE DATA SET MAY BE MODIFIED (BY ADDING, REPLACING, RENAMING, OR DELETING MEMBERS). THIS ALLOWS MORE THAN ONE SPF USER TO MODIFY OTHER MEMBERS IN THE SAME DATA SET WITHOUT TYING UP THE ENTIRE DATA SET.

TO ENSURE THAT DATA SET INTEGRITY IS MAINTAINED, SPF ISSUES RESERVE/DEQ MACROS AT THE APPROPRIATE TIME. THE RESERVE MACRO PREVENTS SHARED DASD CONFLICTS IN ACCESSING THE VOLUME AND ALSO CAUSES AN ENQ T8 BE ISSUED. THIS IS AN UNCONDITIONAL ENQ; IF THE RESOURCE IS IN USE SPF IS PLACED IN THE WAIT STATE UNTIL IT IS FREED. THE DEQ MACRO DEQUEUES THE RESOURCE AND ALSO CAUSES THE VOLUME TO BE RELEASED (IF IT IS A SHARED VOLUME). THE DATA SET INTEGRITY ENQUE IS DONE BY SPF OBJECT MODULES CRESV AND CRELS.

THE SEQUENCE OF EVENTS IS:

```
     ISSUE SPFDSN RESERVE
     ISSUE LINK EDITOR RESERVE OR ENQUE IF LOAD MODULE
       OPEN DATA SET FOR OUTPUT
       WRITE THE MEMBER (FOR ADD OR REPLACE MEMBER)
       ISSUE STOW MACRO (ADD, REPLACE, RENAME, OR DELETE)
       CLOSE THE OUTPUT DCB
     ISSUE SPFDSN DEQ
     ISSUE LINK EDITOR DEQ IF LOAD MODULE
```

THE RESERVE MACRO PARAMETERS FOR THE SPFDSN ENQUE ARE:

```
     TYPE  = SYSTEMS RESERVE
     QNAME = 'SPFDSN  '
     RNAME = DATA SET NAME RIGHT-PADDED WITH BLANKS
     RNAME LENGTH = 44
```

THE LINK EDITOR ENQUE IS ISSUED ONLY FOR RECORD FORMAT U DATASETS.

```
     IF THE VOLUME IS A SHARED DASD VOLUME:
     TYPE  = SYSTEMS RESERVE
     QNAME: 'SYSIEWLP'
     RNAME: DATA SET NAME RIGHT PADDED WITH BLANKS
     RNAME LENGTH: 44


     IF THE VOLUME IS NOT A SHARED DASD VOLUME:
     TYPE  = SYSTEM ENQ
     QNAME: 'SYSIEWLP'
     RNAME: DATA SET NAME RIGHT PADDED WITH BLANKS
     RNAME LENGTH: 44
```

(CONTINUED ON NEXT PAGE)

APPENDIX A. SYSTEM INTERFACE NOTES

| SESSION MANAGER INTERFACE

| MODULES SPFSC93X AND SPFSC94X ARE SPF EXIT ROUTINES FOR SVC 93 (IGC0009C) AND SVC 94 (IGC0009D). THE SESSION MANAGER EXIT ROUTINES IKTTMPX1 AND IKTTMPX2 REFERENCE THESE MODULES. THE SPF EXITS PROVIDE A BETTER INTERFACE BETWEEN SPF AND THE SESSION MANAGER, BUT ARE NOT REQUIRED TO OPERATE SPF WITH THE SESSION MANAGER. THE SPF EXITS ARE FUNCTIONLESS IF THE SESSION MANAGER IS NOT INSTALLED.

| REFER TO THE INSTALLATION AND CUSTOMIZATION MANUAL FOR INFORMATION ON THE INSTALLATION OF THESE MODULES. REFER TO THE MODULE DESCRIPTIONS AND THE MICROFICHE OF "SPFSC93X" AND "SPFSC94X" FOR A DETAILED DESCRIPTION OF THEIR OPERATION.

| THE SESSION MANAGER PROGRAM NUMBER IS 5740-XE2.

| PROGRAM CONTROL FACILITY (PCF) INTERFACE

| PCF COMMAND AUTHORIZATION

| SPF SUPPORTS PCF COMMAND AUTHORIZATION BY INTERFACING WITH PCF VIA "IKJSCAN". BEFORE CALLING "IKJSCAN", SPF PLACES 'SPF' IN THE PRIMARY COMMAND FIELD OF THE ECT (ENVIRONMENTAL CONTROL TABLE). THIS INDICATES TO PCF THAT THE COMMAND BEING SCANNED IS NOT A SUBCOMMAND AND COMMAND AUTHORIZATION SHOULD BE PERFORMED. IF "IKJSCAN" RETURNS WITH RC=52 (INDICATING THE USER IS NOT AUTHORIZED TO USE THE COMMAND), THEN SPF DISPLAYS AN ERROR MESSAGE.

| PCF DATA SET AUTHORIZATION

| PCF DATA SET "READ ONLY" AUTHORIZATION IS BASED ON THE ASSUMPTION THAT PROGRAMS WISHING TO UPDATE A DATA SET WILL FIRST ALLOCATE THE DATA SET WITH DISPOSITION "OLD". HOWEVER, THE SPF EDITOR, LIBRARY UTILITY, MOVE/COPY UTILITY, AND RESET STATISTICS UTILITY ALLOCATE PARTITIONED DATA SETS WITH DISPOSITION "SHARE" AND THEN ISSUES A "RESERVE" TO SERIAL THE UPDATING OF THE MEMBERS.

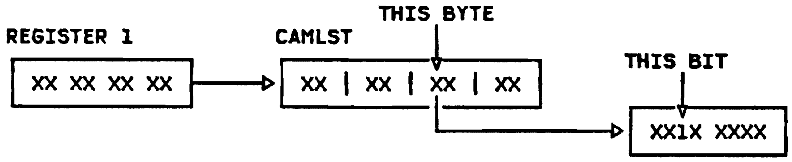
| IN ORDER TO INFORM PCF THAT A "SHARE" ALLOCATION IS BEING DONE FOR UPDATING PURPOSES, SPF SETS THE FIRST BYTE OF THE ECTSCMD FIELD IN THE ECT TO '01'X. PCF TESTS FOR THIS FLAG AND PERFORMS THE AUTHORIZATION CHECK AS IF THE DATA SET WERE ALLOCATED "OLD".

| PCF'S PROGRAM NUMBER IS 5798-CLW.

| HIERACHICAL STORAGE MANAGER (HSM) INTERFACE

| FOR SPF TO ALLOCATE A CATALOGED DATA SET IT ISSUES A LOCATE SYSTEM SERVICE. IF THE DATA SET IS UNDER HSM CONTROL, THE VOLUME "MIGRAT" IS NORMALLY RETURNED. IN ORDER TO CAUSE A "RECALL" AND GET THE REAL VOLUME SERIAL, SPF SETS THE THIRD BIT IN THE THIRD BYTE (CAMOPTN3) OF THE PARAMETER LIST (CAMLST) PASSED TO LOCATE (DIAGRAMMED BELOW). THIS IS DONE FOR ALL SPF ISSUED LOCATES, WHETHER OR NOT HSM IS INSTALLED.

| HSM'S PROGRAM NUMBER IS 5740-XRB.



ENQ/DEQ LOGIC AND SHARED DASD SUPPORT (CONTINUED)

SPF EDITOR MEMBER NAME ENQ

THE SPF EDITOR ISSUES ANOTHER ENQ WHEN DATA IS SELECTED FOR EDITING TO DETECT WHETHER ANOTHER USER IS CURRENTLY EDITING THE SAME PDS MEMBER. IN THIS CASE, IT IS A CONDITIONAL ENQ. IF THE MEMBER IS BEING EDITED BY ANOTHER USER, A MESSAGE IS DISPLAYED. OTHERWISE, THE MEMBER IS FETCHED FOR EDITING. THE CORRESPONDING DEQ IS ISSUED WHEN THE USER ENDS (OR CANCELS) THE EDIT SESSION.

FOR THIS CASE, THE ENQ/DEQ PARAMETERS ARE:

```
     TYPE  = SYSTEM ENQ
     QNAME = 'SPFDSN  '
     RNAME (FIRST 44 CHARACTERS) = FULLY QUALIFIED DATA SET NAME,
                                   RIGHT-PADDED WITH BLANKS
     RNAME (NEXT 8 CHARACTERS) = MEMBER NAME (BLANKS IF RECFM=PS)
     RNAME LENGTH = 52
```

NOTE THAT THIS ENQ/DEQ CANNOT DETECT ANOTHER USER ON A DIFFERENT CPU WHO MAY BE EDITING THE SAME MEMBER VIA SHARED DASD.

TSO/TCAM INTERFACE (CONTINUED)

IN ADDITION TO USING THESE SIGNAL STRINGS TO DISTINGUISH SPF-GENERATED OUTPUT, THE MESSAGE HANDLER MACROS RESET THE TSO/TCAM LINE COUNT WHENEVER A SIGNAL STRING IS ENCOUNTERED, AS FOLLOWS:

> FOR SIGNAL STRING (A) THE LINE COUNT IS SET TO 2. THIS CAUSES ANY SUBSEQUENT LINE MESSAGE (SUCH AS A BROADCAST) TO BE DISPLAYED ON LINE 3 OF THE SCREEN.
>
> FOR SIGNAL STRING (B) THE LINE COUNT IS SET TO THE POSITION INDICATED BY THE SECOND SBA (XXXX). THIS ALLOWS SPF TO CONTROL THE STARTING LOCATION FOR SUBSEQUENT LINE MESSAGES, SUCH AS A 'SUBMIT' COMMAND CONFIRMATION MESSAGE OR THE BEGINNING OF A FOREGROUND PROCESSING SESSION.

THE SPF MESSAGE HANDLER MACROS USE THE IEDQFSCR OPTION BYTE TO KEEP TRACK OF THE MESSAGE SEQUENCE.

IEDQFSCR BITS	MEANING
...1	RESERVED FOR USE BY TSO FULLSCR MACRO
.... 1...	EXPFLS REDISPLAY FLAG
.... .1..	EXPFLS BROADCAST FLAG
.... ..1.	EXPFLS FULL SCREEN FLAG
....1	RESERVED FOR USE BY TSO FULLSCR MACRO

THE OPTION BYTE MUST BE INITIALIZED TO ZERO IN THE ASSSEMBLY OF THE TCAM MCP.

TSO/VTAM INTERFACE

THE FOLLOWING IS A DESCRIPTION OF TSO/VTAM SYSTEM SERVICES THAT ARE USED TO RUN SPF ON A TSO/VTAM SYSTEM. REFER TO THE INSTALLATION AND CUSTOMIZATION GUIDE FOR INFORMATION ON INSTALLATION CONSIDERATIONS.

THE NON-SPF-GENERATED MESSAGE SITUATION, DESCRIBED IN THE TSO/TCAM INTERFACE NOTES, IS GENERALLY HANDLED BY TSO/VTAM. THERE ARE NO TERMINAL ACCESS METHOD MODIFICATIONS THAT MUST BE MADE AS IN TSO/TCAM. HOWEVER, SPF MUST SIGNAL TSO/VTAM WHEN SPF ENTERS OR LEAVES FULL SCREEN MODE.

MODULE SMI ISSUES A "STFSMODE ON,INITIAL=YES" SYSTEM SERVICE BEFORE THE FIRST SPF FULL SCREEN TPUT. THIS SYSTEM SERVICE IS ALSO USED TO DETERMINE THE TERMINAL ACCESS METHOD. A NON-ZERO RETURN CODE INDICATES THAT SPF IS RUNNING UNDER TSO/TCAM.

WHEN AN SPF MODULE WANTS TO ENTER NORMAL TSO LINE I/O MODE, CONTROL FIRST PASSES TO MODULE SML. IT ISSUES THE "STLINENO MODE=OFF,LINENO=XX" SYSTEM SERVICE TO SET THE LINE NUMBER. THE REQUESTING MODULE THEN CAN ISSUE LINE I/O TPUTS AND TGETS TO THE TERMINAL (OR ATTACH A PROGRAM THAT DOES). WHEN THE MODULE WANTS TO RETURN TO SPF FULL SCREEN OPERATIONS, ALL IT HAS TO DO IS REQUEST A FULL SCREEN OUTPUT IN ITS NORMAL WAY (CALL TO SUBROUTINE MHA OR CDISPL). MODULE SMC WILL GET CONTROL AND ISSUE A FULL SCREEN TPUT. THIS WILL CAUSE TSO/VTAM TO GENERATE A SIMULATED PA2 INTERRUPT. A PA2 INTERRUPT IS HANDLED BY SMC BY FIRST ISSUING A "STFSMODE ON" SYSTEM SERVICE AND THEN REDISPLAYING THE COMPLETE SCREEN.

FINAL TERMINATION (MODULE PFT) ISSUES A "STFSMODE OFF" SYSTEM SERVICE.

A MAJOR OPERATING DIFFERENCE BETWEEN TSO/VTAM AND TSO/TCAM IS WHEN A NON-SPF-GENERATED MESSAGE IS SENT TO THE DISPLAY, TSO/VTAM CLEARS THE SCREEN BEFORE DISPLAYING THE MESSAGE IF THE FULL SCREEN MODE HAS NOT BEEN TURNED OFF BY A "STFSMODE" OR "STLINENO" SYSTEM SERVICE.

APPENDIX B. TERMINAL I/O NOTES

TSO/TCAM INTERFACE

THE FOLLOWING IS A DESCRIPTION OF MODIFICATIONS TO THE TSO/TCAM MESSAGE HANDLER THAT ARE REQUIRED TO RUN SPF ON A TSO/TCAM SYSTEM. REFER TO THE INSTALLATION AND CUSTOMIZATION GUIDE FOR INFORMATION ON THE INSTALLATION OF THESE MODIFICATIONS.

DURING OPERATION OF SPF, CERTAIN NON-SPF-GENERATED MESSAGES MAY BE SENT TO THE TERMINAL, SUCH AS BROADCAST MESSAGES FROM THE SYSTEM OPERATOR OR OTHER TSO USERS. THESE MESSAGES WILL BE QUEUED BY TCAM AND DISPLAYED WHEN THE USER HITS AN INTERRUPT KEY, SUCH AS "ENTER" OR ONE OF THE PF KEYS. THAT SAME INTERRUPT, HOWEVER, WILL GENERALLY CAUSE SPF TO GENERATE FULL SCREEN OUTPUT. UNLESS INTERCEPTED, THAT OUTPUT WILL BE SENT TO THE TERMINAL, CAUSING AN IMMEDIATE OVERLAY OF THE BROADCAST MESSAGE.

THE SPF MODIFICATIONS TO THE STANDARD TSO/TCAM MESSAGE HANDLER ARE DESIGNED TO CORRECT THIS PROBLEM. WHEN SPF FULL SCREEN OUTPUT IS FOLLOWED BY NON-SPF LINE OUTPUT:

1. THE LINE MESSAGE IS DISPLAYED WITH HIGH INTENSITY, AND THE AUDIBLE ALARM IS SOUNDED (IF INSTALLED).

2. A SIMULATED ATTENTION (SIMATTN) IS GENERATED. THIS CAUSES THREE ASTERISKS (***) TO BE DISPLAYED ON THE BOTTOM LINE OF SCREEN, AND PREVENTS FURTHER OUTPUT FROM BEING SENT TO THE TERMINAL UNTIL THE USER HITS AN INTERRUPT KEY.

3. WHEN SPF FULL SCREEN OUTPUT FOLLOWS THE LINE MESSAGE, A SIMULATED PA2 INTERRUPT IS GENERATED. THIS CAUSES SPF TO RE-DISPLAY THE ENTIRE SCREEN CONTENTS. (SINCE SPF DOES NOT KNOW ABOUT THE LINE MESSAGE, A GARBLED SCREEN MAY RESULT UNTIL THE REDISPLAY OCCURS.)

THE FULLSCR EXPFLS=YES OR THE SPFSCRN AND SPFMCHK MESSAGE HANDLER MACROS PROVIDE THE PROGRAM LOGIC FOR THESE FUNCTIONS. THESE MACROS GENERATE CODE THAT IS ABLE TO DISTINGUISH SPF-GENERATED OUTPUT FROM NON-SPF-GENERATED OUTPUT. THIS IS DONE BY MEANS OF A "SIGNAL STRING" WHICH BEGINS EVERY SPF-GENERATED FULL SCREEN OUTPUT MESSAGE.

THERE ARE TWO TYPES OF SPF SIGNAL STRINGS:

```
    A.  SPF FULL SCREEN SIGNAL STRING:
              HEX  '11 5D7F 11 XXXX ........'

    B.  SPF EXIT FULL SCREEN SIGNAL STRING:
              HEX  '11 5D7E 11 XXXX ........'
```

WHERE: THE HEX 11'S ARE 3270 SET BUFFER ADDRESS (SBA) ORDERS,
THE HEX 5D7F REPRESENTS SCREEN LOCATION 1919 (DECIMAL),
THE HEX 5D7E REPRESENTS SCREEN LOCATION 1918 (DECIMAL), AND
XXXX REPRESENTS THE ACTUAL BUFFER ADDRESS DESIRED BY SPF.

SINCE TWO SBA'S IN A ROW IS A VALID BUT MEANINGLESS SEQUENCE, IT IS ASSUMED THAT NON-SPF-GENERATED MESSAGES WILL NOT CONTAIN EITHER OF THESE SIGNAL STRINGS.

(CONTINUED ON NEXT PAGE)

TERMINAL INPUT/OUTPUT ERROR CODES

THE MODULE SMC IS RESPONSIBLE FOR MOST TERMINAL I/O DONE BY SPF. IT USES THE COMMON SUBROUTINES CTPUT AND CTGET, WHICH IN TURN ISSUE TPUT AND TGET SYSTEM SERVICE REQUESTS. THE TPUT AND TGET RETURN CODES ARE CHECKED AND THE TGET INPUT DATA IS VERIFIED BY SMC. IF AN ERROR IS DETECTED, SMC SENDS A MESSAGE ALONG WITH CODES TO THE DISPLAY.

THE FOLLOWING IS A LIST SMC TERMINAL I/O ERROR MESSAGES AND CODES.

1. "** SPF SCREEN OUTPUT ERROR - CODE = 41 - TPUT RC = XXXX **", WHERE "XXXX" IS A TPUT RETURN CODE OTHER THAN 0 OR 8.

2. "** SPF SCREEN INPUT ERROR - CODE = 21 - TGET RC = XXXX **", WHERE "XXXX" IS A TGET RETURN CODE OTHER THAN 0, 4, OR 8.

3. "** SPF SCREEN INPUT ERROR - CODE = NN **", WHERE "NN" VALUES ARE:
   - "22" - INPUT STREAM SIZE GREATER THAN INPUT BUFFER SIZE OR 0.
   - "24" - INVALID INPUT AID.
   - "25" - INPUT CURSOR LOCATION NOT WITHIN PHYSICAL SCREEN.
   - "26" - INPUT STREAM SIZE INVALID FOR INPUT AID.
   - "28" - INPUT BUFFER FIELD SIZE GREATER THAN CORRESPONDING PHYSICAL SCREEN FIELD SIZE (INVALID AMOUNT OF INPUT DATA).
   - "29" - 1ST BYTE OF INPUT BUFFER FIELD NOT WITHIN EITHER LOGICAL SCREEN WHICH IS ON THE PHYSICAL SCREEN (INPUT DATA FROM INVALID SCREEN POSITION).
   - "2A" - 1ST BYTE OF INPUT BUFFER FIELD NOT AN SBA (INVALID INPUT DATA).
   - "31" - PHYSICAL SCREEN FIELD SIZE GREATER THAN 255 (INPUT DATA FROM INVALID SCREEN POSITION).
   - "32" - PHYSICAL SCREEN FIELD SIZE IS 0 (INPUT DATA FROM INVALID SCREEN POSITION).
   - "33" - BYTE PRECEEDING THE PHYSICAL SCREEN FIELD IS NOT AN INPUT ATTRIBUTE (INPUT DATA FROM INVALID SCREEN POSITION).
   - "34" - BYTE PRECEEDING THE PHYSICAL SCREEN FIELD IS PAST THE END OF THE PHYSICAL SCREEN (INPUT DATA FROM INVALID SCREEN POSITION).
   - "38" - INPUT BUFFER FIELD SIZE GREATER THAN 255 (INVALID AMOUNT OF INPUT DATA).

   NOTES: THE PHYSICAL SCREEN SIZE IS DETERMINED BY SPF DURING INITIALIZATION. THE INPUT BUFFER SIZE IS A VARIABLE BASED ON THE PHYSICAL SCREEN SIZE. THE LOGICAL SCREEN IS THE SAME SIZE AS THE PHYSICAL SCREEN AND IT IS WHAT THE PROCESSOR TASK USES FOR SCREEN I/O. ONLY A PART OF THE LOGICAL SCREEN IS ON THE PHYSICAL SCREEN WHEN SPF IS RUNNING IN SPLIT SCREEN MODE. A INPUT BUFFER FIELD IS FROM AN SBA TO THE NEXT SBA OR THE END. A PHYSICAL SCREEN FIELD IS FROM THE LOCATION INDICATED IN THE INPUT BUFFER SBA TO THE NEXT ATTRIBUTE BYTE IN THE PHYSICAL SCREEN.

APPENDIX C. ERROR CODES

ABEND CODES

ABENDS OF THE SPF CONTROLLER AND PROCESSOR TASKS ARE CONTROLLED BY STAE AND STAI EXIT ROUTINES AND SPF EXECUTION MODES WHICH ARE SET VIA THE "SPF TEST" COMMAND (SEE THE INSTALLATION AND CUSTOMIZATION GUIDE).

UNDER NORMAL SITUATIONS (WHEN PROCESSOR AND CONTROLLER DUMPS ARE NOT REQUESTED VIA THE "SPF TEST" COMMAND) THE FOLLOWING OCCURS:

- WHEN A PROCESSOR TASK ABENDS, NO DUMP IS TAKEN, THE CONTROLLER REATTACHES THE PROCESSOR MAIN DRIVER (SPFPMD), AND THE PRIMARY OPTION MENU IS REDISPLAYED FOR THAT LOGICAL SCREEN.
- WHEN THE CONTROLLER TASK ABENDS, NO DUMP IS TAKEN, SPF TERMINATES, AND CONTROL RETURNS TO TSO.

THE CONTROLLER AND PROCESSOR TASKS WILL ISSUE THE ABEND SYSTEM SERVICE AND ALLOW DUMPS UNDER CERTAIN SITUATIONS. THE SPF MODULES THAT ISSUE THE ABEND AND THE ASSOCIATED CODES AND REASONS FOLLOW:

CDISPL - USER CODE = "111" OR "222" - TO PRODUCE THESE ABENDS THE USER MUST REQUEST PROCESSOR DUMPS VIA THE "SPF TEST" COMMAND AND ENTER ONE OF THE FOLLOWING COMMANDS IN THE FIRST 8 BYTES OF THE FIRST INPUT FIELD ON A LOGICAL SCREEN.

- "ABEND" - TERMINATES SPF WITH CODE "111".
- "CRASH" - TERMINATES SPF WITH CODE "222" AND PREVENTS TASK TERMINATION FROM CLOSING THE EDIT BACKUP/RECOVERY DATA SETS.

MHA - USER CODE = "998" - THIS ABEND IS USED BY MHA TO PASS CONTROL TO THE CONTROLLER TASK WHEN MHA IS UNABLE TO DISPLAY THE MENU REQUESTED BY THE PROCESSOR TASK. THE CONTROLLER REATTACHES SPFPMD AND THE PRIMARY OPTION MENU IS REDISPLAYED FOR THAT LOGICAL SCREEN.

SMA - THERE ARE TWO USER ABENDS POSSIBLE FROM SMA AS FOLLOWS:

1. USER CODE = PROCESSOR ABEND CODE OR ATTACH RETURN CODE - THIS ABEND IS ISSUED WHEN THE ATTACH OF SPFPMD FAILS OR SPFPMD ABENDS BEFORE THE PRIMARY OPTION MENU IS DISPLAYED IN THE FOLLOWING SITUATIONS:.
   A. FOR THE FIRST LOGICAL SCREEN - ALWAYS.
   B. FOR THE SECOND LOGICAL SCREEN - WHEN THEN THE USER HAS REQUESTED PROCESSOR DUMPS VIA THE "SPF TEST" COMMAND. NORMALLY, FOR THIS SITUATION, ONLY MESSAGES ARE DISPLAYED ON THE FIRST LOGICAL SCREEN.
2. USER CODE = PROCESSOR ABEND CODE - THIS ABEND IS ISSUED WHEN THE PROCESSOR TASK ABENDS AS FOLLOWS:
   A. ANYTIME AFTER THE PRIMARY OPTION MENU IS DISPLAYED IF THE USER HAS REQUESTED CONTROLLER DUMPS VIA THE "SPF TEST" COMMAND. NORMALLY, THE CONTROLLER REATTACHES SPFPMD AND THE PRIMARY OPTION MENU IS REDISPLAYED FOR THAT LOGICAL SCREEN.
   B. ANYTIME AFTER THE FINAL TERMINATION PROCESS BEGINS.

SMC - USER CODE = "997" - WHEN SMC DETECTS A TPUT RETURN CODES OTHER THAN 0 OR 8, A MESSAGE WILL BE DISPLAYED (DESCRIBED IN THE TERMINAL I/O ERROR CODES), AND THEN SMC WILL ATTEMPT A FULL SCREEN REDISPLAY. IF THE REDISPLAY FAILS TWICE THIS ABEND WILL BE ISSUED.

OTHERS - SYSTEM CODE = "0C1" - IN SEVERAL SPF COMMON SUBROUTINES PARAMETER VERIFICATION FAILURE WILL RESULT IN THE EXECUTION OF A "00"X, WHICH IN TURN CAUSES AN "0C1" ABEND. THIS INDICATES THE CALLER HAS PASSED AN INVALID VALUE FOR A PARAMETER WHICH IS REQUIRED FOR CONTINUED EXECUTION. THIS SHOULD NEVER OCCUR IN THE DISTRIBUTED SYSTEM.

LY20-2339-2

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Time Sharin ion 3270 Display Support and Structured Programming Facility Versic Program Logic Manual Printed in U.S.A. LY20-2339-2