Space Flight Operations Contract

SDFPKG USER'S GUIDE

PASS 29.0/BFS 14.0

Baseline

February 8, 1999

DRD- 1.4.3.8-b

Contract NAS9-20000

SDFPKG USER'S GUIDE

Prepared by

P. Koester Date
Application Tools, PASS Build, and Reconfiguration

Approved by

T. F. Peterson Date
USA/PASS Deputy Program Manager

J. Treybig Date
Application Tools, PASS Build, and Reconfiguration

DRD- 1.4.3.8-b

Contract NAS9-20000

Change History

The SDFPKG User's Guide has been revised and issued on the following dates:

Revision	Date	Release	Sections Changed
Baseline	02/08/99	29.0/14.0	Initial Release

Table of Contents

Table of Figures

Preface

The purpose of this document is to provide the information needed by a programmer to understand and utilize the functionality of the Simulation Data File Access Package (SDFPKG). An assembly language version and a C language version of SDFPKG exist. Although both versions are covered in this document, the assembly version is the main focus, and differences between the two versions are distinguished where appropriate.

1. Simulation Data File Access Package (SDFPKG) Overview

SDFPKG is an IBM-360 assembly language program comprised of five CSECTs: SDFPKG, LOCATE, PAGMOD, NDX2PTR, and SELECT. Its function is to provide a demand paging form of access to data contained within SDFs. SDFPKG can be separately link edited and employed as a loadable and deletable service module (as with the XPL Monitor), or it may be linked directly into a load module (as with the HAL/S-360 stand-alone diagnostic system).

A C language version of SDFPKG exists to support the Flight Software Application Tools (FSWAT) which have also been reengineered in C and C++. All functionality is the same as in the Assembly language version, unless otherwise noted.

The following is a brief summary of the more important aspects of SDFPKG:

- SDFPKG is a modular access method for SDFs built upon a demand paging virtual memory foundation. It can be separately linked, loaded, and deleted.
- All calls to SDFPKG are made through a single entry point by supplying a mode number. Eighteen different mode calls (nineteen in the FSWAT C version) are currently provided (see section 9).
- SDFPKG employs a paging area of from 1 to 250 1680-byte pages in size that may be dynamically expanded or contracted as the memory situation alters (see section 2).
- SDFPKG can support simultaneous access to an unlimited number of SDFs. The area needed for File Control Blocks (FCBs) can be automatically provided by SDFPKG or be under the control of the user (see section 6).
- SDFPKG is serially reusable. Following a Terminate (mode 1) call a new Initialize (mode 0) call may be made.
- SDFPKG allows SDFs to be modified or merely read.
- SDFPKG provides built-in binary search algorithms to allow high-level access to data that must be searched.
- SDFPKG FREEMAINs all storage at the Terminate (mode 1) call that it may have GETMAINed since the Initialize (mode 0) call.
- SDFPKG performs one OPEN (for the HALSDF DD) at Initialize (mode 0) and one CLOSE (same DD) at the Terminate (mode 1) call.
- SDFPKG uses only the following O/S services: GETMAIN, FREEMAIN, FIND, BLDL, POINT, READ, WRITE, CHECK, OPEN, CLOSE.
- SDFPKG can be configured at Initialize (mode 0) so that it will perform no GETMAINs.
- SDFPKG performs complete error checking and will force an Abend in case of a legitimate user error or I/O error (see section 12). A complete set of return codes is used to signal off-nominal conditions that are not reflections of serious user error (see section).
- Almost all communication between SDFPKG and the user program is accomplished through a 120 byte "communication" table which the user provides (see section 7). This eliminates almost all parameter passing.

- SDFPKG maintains statistical information bearing on its operation plus other data of interest which the user may access at any time (see section 14).
- SDFPKG is designed to be as fast as possible without sacrificing essential error checks. With a large paging area, efficiency is competitive with implementations in which all SDFs are memory resident in their entirety.
- SDFPKG requires less than 12000 bytes of memory exclusive of FCB Area and Paging Area.

2. Paging Area

Paging is done directly between memory and the Partitioned Dataset (PDS) containing the SDFs generated by Phase III of the HAL/S Compiler.

SDF records (or pages) are always 1680 bytes in length. SDFPKG reads SDF pages from a PDS into "paging areas" which may consist of from 1 to more than 4000 1680-byte areas. The current version of SDFPKG contains a default upper limit of 250 pages. This upper limit can be increased by the user program providing SDFPKG with a larger Paging Area Directory (PAD) and Paging Area. Figure 2-1 shows the contents of the PAD. (Note: The Paging Area and PAD are two separate entities and should not be confused with each other. In addition, the maximum number of pages allowed by SDFPKG at any time is limited by the number of PAD entries allocated. The number of pages must always be less than or equal to the number of PAD entries.) Increasing the number of default PAD entries will increase the size of SDFPKG by 16 bytes per added PAD entry. At the other extreme, SDFPKG will usually function properly with a 1 page paging area (if no reserves are requested), however 2 pages is the recommended minimum.

PAGEADDR	Address of the corresponding Paging AREA entry
FCBADDR	Byte 0 contains "Page modified" Flag of Hex '80' Bytes 1-3 contains address of the File Control Block (FCB)
USECOUNT	Usage Counter
PAGENO	Page # x 8 (2 bytes)
RESUCNT	Reserve counter (2 bytes)

Figure 2-1 Contents of Paging Area Directory (PAD) Entry

The absolute maximum number of entries in the paging area supported by SDFPKG is 4095, therefore the "paging area" and PAD parameters should not exceed this value. Currently, SDFPKG may be called an unlimited number of times, but in the process of doing so, will lose track of the actual number of "Locates" (LOCCNT) after the first roll-over at Hex 'FFFFFF'. Due to the roll-over which takes place, the statistics produced by SDFPKG will now become a modulus value, i.e. (Hex 'FFFFFF') modulo (total # of locates).

The PDS containing the SDF members to be read is normally identified by a HALSDF DD card; however at the time of the initialization call to SDFPKG, an alternate DDNAME can be specified. The SDF PDS may contain catenation levels so long as the program calling SDFPKG intends only to read the data. If the user desires to "modify" an SDF (by requesting SDFPKG to operate in UPDAT mode), none of the SDFs to be updated may reside within a concatenated DD level, due to O/S restrictions.

At the time of the SDFPKG initialization (Initialize - mode 0), the calling program must specify the size of the "nucleus" paging area. This initially allocated area will then be available to, and will be exclusively controlled by, SDFPKG until the termination call (Terminate - mode 1). SDFPKG makes provisions for dynamic expansion and contraction of the paging area size (up to the limit set by the PAD whose default size is currently 250 entries) via one or more Augment (increase paging area) calls (mode 2) and Rescind (remove all augments) calls (mode 3). The Rescind call always reduces the paging area size to the initial (nucleus) area.

The memory necessary for the nucleus paging area may be allocated by SDFPKG via a GETMAIN or it may be provided by the calling program. The memory necessary for Augments, however, must always be provided by the calling program. If SDFPKG is instructed to GETMAIN the nucleus paging area, it will perform a FREEMAIN at the Terminate call as well. This is true of any GETMAINs performed by SDFPKG.

3. Virtual Memory

SDFs are built by Phase III of the HAL/S Compiler in a virtual memory environment and are manipulated by SDFPKG in the same way. In this context, SDF data possesses both a "pointer", (i.e. a record/offset address in SDFPKG's virtual memory space) and the memory address of the data located in one of SDFPKG's virtual memory pages (if it has not been read into memory, this value is 0). As described in the *HAL/SDL ICD (OB30029)*, the fullword pointers contained within the SDFs consist of a page (record) number residing in the upper 2 bytes of the 4-byte pointer followed by a displacement in the lower 2 bytes of the 4-byte pointer. The displacement is the offset (from 0 to 1679 bytes) into the page (record) referenced by the upper 2 bytes. SDF pages are numbered beginning with zero, so the pointer consisting of a fullword of zeros identifies the first byte of data in an SDF.

In the most general form of data access provided by SDFPKG, an input SDF pointer causes SDFPKG to return the memory address of the corresponding data as output. The returned memory address lies somewhere within the allocated paging area. If the necessary SDF page was already in the paging area, then this is a fast operation. If it was not, then a paging operation that is transparent to the calling program is performed as necessary. Although this process of "location" can be requested explicitly by the calling program through a Locate (mode 5) call, the program will more often employ the higher-level SDFPKG mode calls which will then perform the necessary "locates" implicitly and totally internal to SDFPKG.

Whether the locates are explicit or implicit, the important point is that almost all SDFPKG mode calls result in the memory location (and corresponding virtual memory pointer for reference purposes) of some data item being returned to the calling program. This data item may be an SDF Directory Root Cell, Block Data Cell, Symbol Data Cell, Statement Data Cell, Block Index Table Entry, Symbol Index Table Entry, Statement Index Table Entry, or merely some arbitrary SDF location (if an explicit Locate call was made). Immediately after the call, the page containing the item of interest is in memory and the calling program may extract (or insert in update mode) data using the memory address provided.

It is normally the case, and especially true when a small paging area is used, that the data located in this fashion must be considered vanished after the next SDFPKG call. When using a small paging area, a subsequent SDFPKG call of any kind may require I/O that will force the reuse of previously loaded paging area "slots". If the calling program needs to guarantee the continued existence of the located data at the advertised memory address, the RESV (Reserve) disposition parameter should be specified at the time of the initial mode call or prior to any subsequent SDFPKG calls. SDFPKG then increments a reserve count that is maintained in the PAD for the page containing the located data and ensures that this page will not be overwritten until the reserve count has been decremented to zero. At some later time, the calling program must "free" the data by making any mode call that re-locates the data item and specifies the RELS (Release) option. Since it is actually pages and not specific locations that are reserved, it is only necessary to locate any part of the page in order to free it.

If the calling program cannot determine until after the SDFPKG call that RESV, RELS, or MODF is desirable, then one or more of these disposition parameters can be specified by a Disposition (mode 6) call which retroactively applies such parameters to the preceding item located.

Programmers designing programs that use SDFPKG should be careful to limit the use of Reserves, especially if small paging areas are employed, since each reserve makes one more paging area slot unavailable for further reads. Also, all pages that are reserved should be ultimately released. A Rescind call will result in an abnormal termination (Abend 4011) if any reserved pages are detected in the augmented portion of the paging area.

The third disposition parameter MODF (Modify) can only be used if the UPDAT mode was specified at the time of the Initialize call. MODF informs SDFPKG that the located item will be altered by the calling program. As a result, SDFPKG will rewrite the affected page back to the PDS (HALSDF or alternate DDNAME) prior to overlaying the slot with newly read pages. Again, due to O/S restrictions, SDFs which are to be altered must not lie within a concatenated dataset.

4. Paging Strategy

The PAD contains an entry for each memory slot up to the defined limit (the default is 250) with each entry containing, among other data, a reserve count and a usage count for the page (see Figure 2-1). As mentioned earlier, the reserve count is used to lock the page in its memory slot for as long as the count is non-zero. The usage count, however, keeps track of how recently that page has been accessed relative to the other pages in memory. A global count of "locates" is maintained within SDFPKG and is inserted into the usage count field of the PAD entry when the page is accessed. When an SDF page must be read into a memory slot from the PDS, the memory slot that is both unreserved and least recently accessed is overlayed by the new data. If, however, the modification flag for that PAD entry indicates that the old page is in a modified state (UPDAT mode only) then the page is written out prior to being overlayed. At the Terminate or Rescind call all modified, but as yet unwritten, pages are written out to the PDS.

5. SDF Selection

SDFPKG allows simultaneous access to an unlimited number of SDF members. This means that the paging area can contain assorted pages from a number of different SDF members. In order for SDFPKG to know which SDF member is to be referenced in support of the user's call, it is necessary for the calling program to specify or "select" the proper SDF member in one of two ways. The first method is to make an explicit Select (mode 4) call to SDFPKG with the 8 character SDF member name (##<cccccc>) as input. Until overridden, all further SDFPKG data access requests will be directed to this SDF member. The second method is called "Auto-Selection". By specifying the Auto-Select disposition parameter and including the SDF member name as an auxiliary input, SDFPKG calls will reference the specified SDF member. Auto-Selection is slightly slower than explicit selection but is very useful if the SDF members are to be referenced randomly.

When an SDF member is selected for the first time following the Initialize call, SDFPKG performs a BLDL for that PDS member, extracts certain data from the Directory Root Cell, and then incorporates all of this information into a File Control Block (FCB) for that SDF member. The FCB (see Figure 5-1) is allocated from a block of memory called the FCB area (see section 6). The new FCB is then linked into a binary tree structure that is ordered by SDF member name so that later selections can rapidly find the FCB needed to access the data in the file. With one exception (One-FCB mode), once an FCB is created, it is maintained until a Terminate call resets all SDFPKG variables and data areas. This means that the FCB area may eventually become filled with FCBs and require extension.

If the calling program knows beforehand that SDF members will be accessed in a serial fashion, or if memory space is at a premium, then SDFPKG can be instructed at the time of the Initialize call to operate in the One-FCB mode, i.e., only one FCB is kept and therefore a new Select will cause the new FCB to be built over the old one.

GTTREEPT	Pointer to next FCB Tree entry greater than this one
LTTREEPT	Pointer to next FCB Tree entry less than this one
FILENAME	8 Character SDF Member name
BLKPTR	SDFPKG Pointer to Block Index Table
SYMBPTR	Pointer to Symbol Index Table
STMTPTR	Pointer to Statement Index Table
TREEPTR	Pointer to Nested Block Tree
NODESIZE	Size of Statement Index Table Entry (4 bytes - No Statement Reference Numbers (SRNs) are Present; 12 bytes - SRNs are present)
FLAGS	SDF Flags from Directory Root Cell
NUMBLKS	Number of Blocks (Including Includes) in SDF member
NUMSYMBS	Number of symbols in SDF member
FSTSTMT	First Internal Statement Number (ISN) in SDF member
LSTSTMT	Last ISN in SDF member
LSTPAGE	Last page of SDF data
VERSIONX	Phase III Version number of SDF member
STMTEXPT	Pointer to Block Statement Extent Cell
	Spare 2 Fullwords
FCBPDADR	Address of PAD entry holding this Record or 0 if not currently in memory

Figure 5-1 Contents of File Control Block (FCB) Entry

6. FCB Area

The FCB Area is similar to the Paging Area in that an initial amount must be allocated at the time of the Initialize call. The calling program may specify what the allocation is to be or accept the default of 1024 bytes. Additionally, the calling program has to decide whether to provide SDFPKG with an FCB Area or to let SDFPKG obtain one via a GETMAIN. If the calling program supplies an FCB Area, then it must be prepared to supply additional areas (via the Augment call) whenever the current FCB Area is exhausted. This condition is signaled by a return code of 12, meaning that a select failed due to insufficient space to construct an FCB. A better method of supplying the FCB area for SDFPKG is for the user program to allocate the same number of contiguous 128-byte data blocks as there are members in the SDF PDS and then pass the address of the FCB area and the negative value of the number of FCB entries to SDFPKG. This will normally provide SDFPKG with all of the FCB area needed to process all SDF members.

If the calling program does not need the flexibility of allocating the FCB area itself, then SDFPKG can be allowed to GETMAIN the initial FCB Area, or, alternately, the MISC parameter can be set for Automatic FCB GETMAIN mode on the Initialize call. The latter case will then allow automatic GETMAINs regardless of who allocated the initial FCB area. In this mode of operation, subsequent GETMAINs of 512 bytes each will be performed as needed and will be totally transparent to the calling program. It is also possible to pass only the negative value of the number of members in the SDF and allow SDFPKG to GETMAIN an FCB area of 128 times the number of SDF members passed. Again, all such GETMAINed areas are freed when the SDFPKG Terminate call is made.

One-FCB mode is available regardless of whether the caller or SDFPKG is responsible for FCB Area allocation. It should also be noted that although the Augment call can extend either the Paging Area or FCB Area (or both simultaneously), the Rescind call only applies to the Paging Area, i.e., the FCB Area can only grow.

Each FCB requires an initial 60 bytes plus an additional 8 bytes for each page of the associated SDF member. FCBs are thus highly variable in length.

7. Communication Table (COMMTABL)

The Communication Table is a 120-byte data area that the user program must supply. The address of the COMMTABL is passed to SDFPKG on the INITIALIZE call. The DSECT for COMMTABL is shown in Figure 7-1.

FSWAT C version only: The Communication Table is called *sdfpkg_common* and is dynamically allocated by the calling program (see section 8.3). The same variable names as shown in Figure 7-1 reference the appropriate location within sdfpkg_common.

```
NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(COMMTABL)
00100             MACRO
00200             COMMTABL
00300  COMMTABL   DSECT        SDFPKG COMMUNICATION AREA
00400  APGAREA    DS     A     ADDRESS OF EXTERNAL PAGING AREA
00500  AFCBAREA   DS     A     ADDRESS OF EXTERNAL FCB AREA
00600  NPAGES     DS     H     # OF PAGES IN PAGING AREA OR AUGMENT
00700  NBYTES     DS     H     # OF BYTES IN FCB AREA OR AUGMENT
00800  MISC       DS     H     MISCELLANEOUS PURPOSES
00900  CRETURN    DS     H     SDFPKG RETURN CODE
01000  BLKNO      DS     H     BLOCK NUMBER (BLOCK NODE INDEX)
01100  SYMBNO     DS     H     SYMBOL NUMBER (SYMBOL NODE INDEX)
01200  STMTNO     DS     H     STATEMENT NUMBER (STATEMENT NODE INDEX)
01300  BLKNLEN    DS     CL1   NUMBER OF CHARACTERS IN BLOCK NAME (BLKNAM)
01400  SYMBNLEN   DS     CL1   NUMBER OF CHARACTERS IN SYMBOL NAME
                                (SYMBNAM)
01500  PNTR       DS     F     VIRTUAL MEMORY POINTER LAST LOCATED
01600  ADDR       DS     A     CORE ADDRESS CORRESPONDING TO PNTR
01610  SDFDDNAM   EQU    *     NAME OF ALTERNATE DD FOR SDF DATA SET
01700  SDFNAM     DS     CL8   NAME OF SDF TO BE SELECTED
01800  CSECTNAM   DS     CL8   NAME OF CODE CSECT FOR BLOCK
01900  SREFNO     DS     CL6   STATEMENT REFERENCE NUMBER
02000  INCLCNT    DS     H     INCLUDE COUNT (FOR SRN)
02100  BLKNAM     DS     CL32  BLOCK NAME
02200  SYMBNAM    DS     CL32  SYMBOL NAME
02300             MEND
READY
```

Figure 7-1 SDFPKG Communication Table

8. SDFPKG Calling Sequence

The following block diagram (Figure 8-1) illustrates the order and hierarchy of the mode calls needed to effectively use SDFPKG. Notice in the figure that items 5 and 6 are indented from the rest to indicate that these items may be performed multiple times within the SDFPKG calling sequence. Once SDFPKG has been Initialized, it is necessary to Select the SDF member to which the next operations will apply (Item 5). After Selecting the SDF member, the user is free to retrieve and process any data (e.g., Statement data, Symbol data, or Block data) desired (Item 6). Upon completion of the processing of the SDF data, it is necessary to Terminate SDFPKG (Item 7). Examples of actual SDF Program calls are provided in section 8.1 (Assembly language), section 8.2 (XPL language), and section 8.3 (C language).

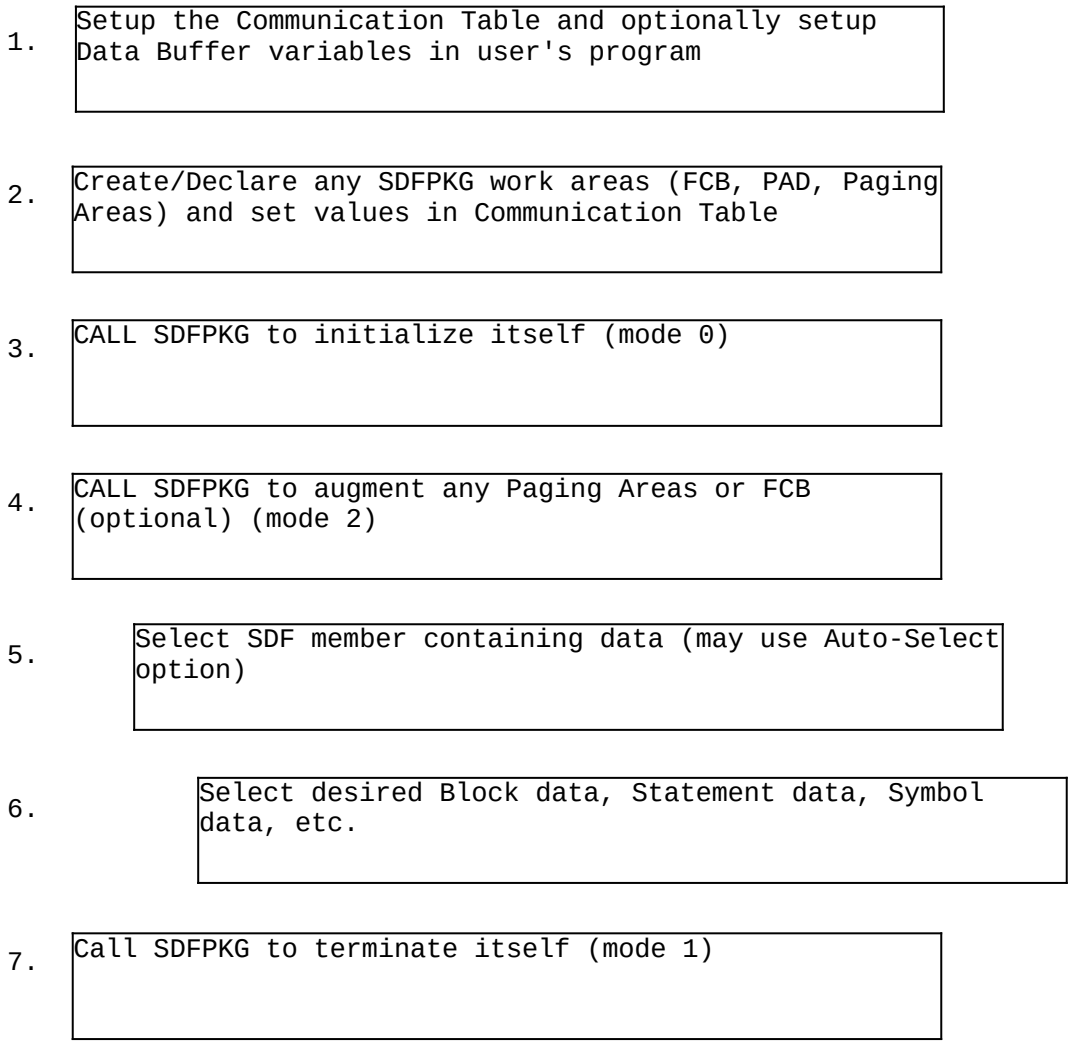


Figure 8-1 SDFPKG Generic Calling Sequence

8.1 Calling SDFPKG from an Assembly Program

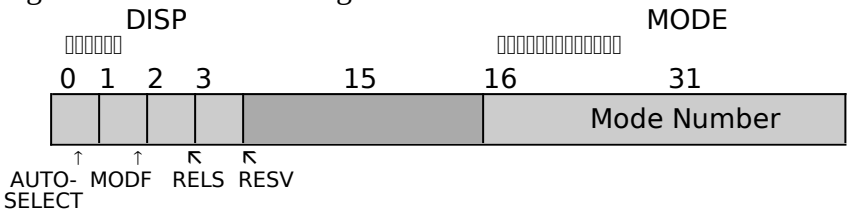
The following is an example of how SDFPKG might be loaded, called, and deleted:

```
                LOAD  EP=SDFPKG
                ST    R0,ASDFPKG        SAVE LOAD ADDRESS
                ...
                LA    R0,COMMAREA       R0=ADDRESS OF COMMUNICATION AREA
                SR    R1,R1
INITIALIZE      L     R15,ASDFPKG
CALL            BALR  R14,R15
                LTR   R15,R15           TEST RETURN CODE
                BNZ   OPENFAIL
                ...
                LA    R1,<mode#>
                O     R1,=X'<disposition parameters>'
ALL             L     R15,ASDFPKG
OTHER           BALR  R14,R15
CALLS           LTR   R15,R15           TEST RETURN CODE
                BNZ   MISCFAIL
                ...
TERMINATE       LA     R1,1             TERMINATE
CALL            L      R15,ASDFPKG
                BALR   R14,R15
                ...
                DELETE EP=SDFPKG        DELETE SDFPKG
                ...
ASDFPKG         DC     A(0)             ADDRESS OF SDFPKG
COMMAREA        DC     30F'0'           COMMUNICATION TABLE
```

In the preceding example, the DSECT for COMMAREA can be found in 'NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(COMMTABL)'.

Register 0 is used only on the INITIALIZE call and contains the address of the Communication table.

Register 1 has the following format:



Upon return the return code is available both in Register 15 and in the CRETURN location in the communication table. Register 1 generally contains the memory address of the "located" data item (also in ADDR location of the communication table).

All other registers are preserved as per standard O/S linkage conventions. Register 13, of course, must point to the user's save area at the time of the call.

Communication table variables are left unaltered unless they are explicitly output by the SDFPKG call.

8.2 Calling SDFPKG from an XPL Program

All XPL programs using SDFPKG must call it using the *MONITOR(22)* interface:

MONITOR (22,n,a);

This Monitor call causes Load, Call, and Delete of SDFPKG. This interface is used by HALSTAT, HAL/S Compilers, and other programs to call SDFPKG.

Parameters:

22 - Call SDFPKG
n - SDFPKG Mode (same form as R1 in section 8.1)
a - Address of the Communication Table (COMMTABL) interface table. This interface table is specified only in the SDFPKG initialize call (mode 0).

The following is an example of how SDFPKG may be used from an XPL program:

```
/* SDFPKG COMMUNICATION TABLE */
DECLARE APGAREA  FIXED;   /* ADDR OF EXTERNAL PAGING AREA */
DECLARE AFCBAREA FIXED;   /* ADDR OF EXTERNAL FCB AREA */
DECLARE NPAGES   BIT(16); /* # OF PGS IN PAGING AREA OR AUGMENT */
DECLARE NBYTES   BIT(16); /* # OF BYTES IN FCB AREA OR AUGMENT */
DECLARE MISC     BIT(16); /* MISC PURPOSES */
DECLARE CRETURN  BIT(16); /* SDFPKG RETURN CODE */
DECLARE BLKNO    BIT(16); /* BLOCK NUMBER */
DECLARE SYMBNO   BIT(16); /* SYMBOL NUMBER */
DECLARE STMTNO   BIT(16); /* STATEMENT NUMBER */
DECLARE BLKNLEN  BIT(8);  /* NUMBER OF CHARS IN BLOCK NAME */
DECLARE SYMBNLEN BIT(8);  /* NUMBER OF CHATS IN SYMBOL NAME */
DECLARE PNTR     FIXED;   /* VMEM POINTER LAST LOCATED */
DECLARE ADDRESS  FIXED;   /* CORE ADDRESS CORRESPONDING TO PNTR */
DECLARE SDFNAM   FIXED;   /* NAME OF SDF TO BE SELECTED */
DECLARE SDFNAMA  FIXED;   /* NAME OF SDF TO BE SELECTED */
DECLARE CSECTNAM FIXED;   /* NAME OF CODE CSECT FOR BLOCK */
DECLARE COMM40B  FIXED;
DECLARE SREFNO   FIXED;   /* STATEMENT REFERENCE NUMBER */
DECLARE COMM48B  BIT(16);
DECLARE INCLCNT  BIT(16); /* INCLUDE COUNT */
DECLARE BLKNAM   FIXED;   /* BLOCK NAME */
DECLARE COMM56B  FIXED;
DECLARE COMM56C  FIXED;
DECLARE COMM56D  FIXED;
DECLARE COMM56E  FIXED;
DECLARE COMM56F  FIXED;
DECLARE COMM56G  FIXED;
DECLARE COMM56H  FIXED;
DECLARE SYMBNAM  FIXED;   /* SYMBOL NAME */
DECLARE COMM88B  FIXED;
DECLARE COMM88C  FIXED;
DECLARE COMM88D  FIXED;
DECLARE COMM88E  FIXED;
DECLARE COMM88F  FIXED;
DECLARE COMM88G  FIXED;
DECLARE COMM88H  FIXED;

DECLARE (BLKS,SYMS,FIRST_STMT,LAST_STMT) BIT(16);
DECLARE NAME CHARACTER INITIAL('##COPY1A');
DECLARE NAMEADDR BIT(32);
DECLARE FLAGS  BIT(16);
```

```
BASED   DATABUF_FW FIXED;

/* INITIALIZE SDFPKG */
CALL MONITOR(22,0,ADDR(APGAREA));
IF (CRETURN ^= 0) THEN OUTPUT = 'SDFPKG INITIALIZATION FAILED';
COREWORD(ADDR(DATABUF_FW)) = ADDRESS;
NAMEADDR = COREWORD(ADDR(NAME)) & "FFFFFF";
SDFNAM  = COREWORD(NAMEADDR);
SDFNAMA = COREWORD(NAMEADDR+4);

/* SELECT THE SDF MEMBER */
CALL MONITOR(22,4);
OUTPUT = '--- SDFPKG SELECT '||NAME||' ---';

/* LOCATE DIRECTORY ROOT CELL BY POINTER */
OUTPUT=''; OUTPUT = '--- SDFPKG MODE 5 TEST ---';
PNTR =144;
CALL MONITOR(22,"80000005");

FLAGS = COREHALFWORD(ADDRESS);
OUTPUT = 'ROOT CELL = '|| PNTR;
IF (FLAGS & "1000") ^= 0 THEN OUTPUT = 'FC_FLAG   IS SET';
IF (FLAGS & "4000") ^= 0 THEN OUTPUT = 'ADDR_FLAG IS SET';
IF (FLAGS & "8000") ^= 0 THEN OUTPUT = 'SRN_FLAG  IS SET';
IF (FLAGS & "0008") ^= 0 THEN OUTPUT = 'SDL_FLAG  IS SET';
SYMS = COREHALFWORD(ADDRESS+18);
BLKS = COREHALFWORD(ADDRESS+16);
FIRST_STMT = COREHALFWORD(ADDRESS+52);
LAST_STMT = COREHALFWORD(ADDRESS+54);
OUTPUT = 'TOTAL # SYMBOLS = '||SYMS;
OUTPUT = 'TOTAL # BLOCKS  = '||BLKS;
OUTPUT = '1ST STMT = '||FIRST_STMT||' LAST STMT = '||LAST_STMT;

/* LOCATE SYMBOL NODE GIVEN PNTR */
PNTR = 492;
OUTPUT = ''; OUTPUT = '--- LOCATE SYMBOL NODE BY PNTR ---';
CALL MONITOR(22,"80000005"); /* LOCATE STATEMENT NODE */
COREWORD(ADDR(NAME)) = SHL(7,24) + ADDRESS;    /* SYMBOL NAME */
OUTPUT = 'SYM #10'||
      ' SYMBOL NAME = '||NAME||' PNTR = '||PNTR;

CALL MONITOR(22, "10000006");     /* RESERVE THE DATA CELL */

/* PRINT SDFPKG STATISTICS IN DATABUF */
OUTPUT = ' ';
OUTPUT = 'TOTAL NUMBER OF LOCATE  = '||DATABUF_FW(0);
OUTPUT = 'TOTAL NUMBER OF RESERVE = '||DATABUF_FW(14);
OUTPUT = 'TOTAL NUMBER OF READ    = '||DATABUF_FW(15);
OUTPUT = 'TOTAL NUMBER OF WRITE   = '||DATABUF_FW(16);
OUTPUT = 'TOTAL NUMBER OF SELECT  = '||DATABUF_FW(17);

CALL MONITOR(22, 1); /* TERMINATE SDFPKG */

EOF EOF EOF
```

8.3 Calling the FSWAT C Version of SDFPKG

All C or C++ programs using the FSWAT C Version of SDFPKG must call it using the *sdfpkg_call* procedure interface:

sdfpkg_call(mode, R);

This interface is used by BIT HALSTAT, HALEXEC 360 emulator, and the XPL debugger to call SDFPKG.

Parameters:

mode - SDFPKG Mode (same form as R1 in section 8.1)
R - Output register array

The calling program must initialize a few things before calling sdfpkg_call, like the sdfpkg_common (COMMTABL) data area, and the DDNAME where the SDF files are located. The calling program of course must also initialize the input data required for each particular SDFPKG mode call. The user can reference any of the SDFPKG common or internal data via a C++ interface 'SDF Processor' (SdfProc for short). Details on the usage of SdfProc appear in the FSWAT *C++ Reuseable Component User's Guide*. Here is an example C calling program:

```
extern unsigned char *sdfpkg_common;  /* common data area for i/o */
extern unsigned int R[16];            /* registers used for output */
void sdfpkg_call(int, unsigned int*); /* main sdfpkg routine */
extern char sdfpkg_ddname[9];         /* ddname for sdf files */
static void call_sdf(int mode)
{
   if (mode == 0)
      /* allocate 120 bytes for the sdfpkg common data area */
      sdfpkg_common = (unsigned char *)calloc(120, 1);
   /* if auto-select (1st bit), call sdfpkg with SELECT first */
   if (mode < 0)
      sdfpkg_call(4, R);
   sdfpkg_call(mode, R);
}
void main(int argc, char *argv[])
{
   int i;
   char sdf_name[8];
   call_sdf(0); /* initialize */
   /* select */
   strcpy(sdfpkg_ddname, "HALSDF");
   strcpy(sdf_name, "##TEST");
   for (i = 0; i < 8; i++)
       BYTE_AT(SDFNAM+i) = sdf_name[i];
   call_sdf(4);
   /* etc….. */
}
```

9. SDFPKG Mode Numbers

Mode #	Function
0	Initialize (must be the first call to SDFPKG).
1	Terminate (must be the last call to SDFPKG).
2	Augment Paging Area and/or FCB area.
3	Rescind all Paging Area augments.
4	Select SDF explicitly.
5	Locate a virtual memory pointer.
6	Set disposition parameters (MODF, RESV, RELS) for the last "located" data item.
7	Locate Directory Root Cell.
8	Locate a Block Data Cell given the block number.
9	Locate a Symbol Data Cell given the symbol number.
10	Locate a Statement Data Cell given the statement number.
11	Locate a Block Data Cell given the block name.
12	Locate a Symbol Data Cell given the block name and symbol name.
13	Locate a Symbol Data Cell given the symbol name only.(Must be preceded by a mode 8, 11, or 12 call.)
14	Locate a Statement Data Cell given the SRN (Statement Reference Number).
15	Locate a Block Node given the block number.
16	Locate a Symbol Node given the symbol number.
17	Locate a Statement Node given the statement number.
18	Deselect an SDF. (FSWAT C version only)

10. SDF Data Organization

Cells
- Are contiguous
- Contain data concerning 1 item
- Are addressable only by Pointer
- Cannot cross page boundary (= 1680 bytes)

Tables
- Consist of multiple entries
- May cross page boundaries, but an individual entry may not
- May be accessed both by Pointer and by halfword indexes for each entry

Block Data
- Provides information about:
  - Compilation unit
  - Blocks
    - Functions
    - Procedures
    - Programs
    - COMPOOLs
    - Tasks
    - Updates
  - Included Data
- Consists of:
  - Block Index Table
  - Block Data Cells

Symbol Data
- Provides attribute information about symbols in a compilation unit
- Consists of:
  - Symbol Index Table
  - Symbol Data Cells
  - Block Symbol Extent Cells
  - Constant Value Cells
  - Replace Text Cells
  - Procedure/Function Formal Parameter Cells
  - Name Terminal Initialization Cells
  - Variable Reference Cells

Statement Data
- Provides attribute information about statements in a compilation unit
- Consists of:
  - Statement Index Table
  - Statement Data Cells
  - Block Statement Extent Cells
  - Expression Variables Cells
  - Procedure/Function Invocation Cells

Note 1) Definitions and layouts for the various data blocks contained within an SDF can be found in the HAL/SDL Interface Control Document (OB30029):

Terminology	ICD Figure No.
Directory Root Cell	Fig. 2 - 11
Block Data Cell	Fig. 2 - 23
Symbol Data Cell	Fig. 2 - 30
Statement Data Cell	Fig. 2 - 60
Block Index Table	Fig. 2 - 19
Symbol Index Table	Fig. 2 - 29
Statement Index Table	Fig. 2 - 57

Note 2) A "Node" refers to an entry in an index table, for instance a Block Node is an entry in the Block Index Table.

Note 3) DSECTs for the pertinent data blocks can be found as members of the 'NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB' dataset (Assembly version only) :

Terminology	Member Name
Directory Root Cell	DROOTCEL
Block Data Cell	BLKTCELL
Symbol Data Cell	SYMBDC
Statement Data Cell	STMTDC
Block Node	BLCKNODE
Symbol Node	SYMBNODE
Statement Node (no SRNs)	STMTNOD0
Statement Node (SRNs)	STMTNOD1

For convenience, listings of these DSECTs follow. Although these are for the Assembly version of SDFPKG, the data offset and content is the same as is referenced in the FSWAT C version. This information is useful to the calling program so that the format of the data returned by SDFPKG can be understood.

```
'NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(DROOTCEL)'
             MACRO
             DROOTCEL
DROOTCEL     DSECT   DIRECTORY ROOT CELL
SDFFLAGS     DS  2C
LASTPAGE     DS  H   # OF LAST PAGE IN SDF FILE
SDFDATE      DS  F   DATE OF CREATION
SDFTIME      DS  F   TIME OF CREATION
LASTDPGE     DS  H   # OF LAST DIRECTORY PAGE
COMPOOLS     DS  H   # OF INCLUDED COMPOOLS
BLKNODES     DS  H   # OF BLOCK NODES
SYMNODES     DS  H   # OF SYMBOL NODES
FBNPTR       DS  A   POINTER TO FIRST BLOCK NODE
LBNPTR       DS  A   POINTER TO LAST BLOCK NODE
INSTRCNT     DS  H   NO.  OF EMITTED MACHINE INSTRUCTIONS
FREEBYTE     DS  H   TOTAL AMOUNT OF FREE SPACE IN SDF
DLSTHEAD     DS  H   LIST HEAD FOR DECLARED VARS (BY ADDR)
RLSTHEAD     DS  H   LIST HEAD FOR REMOTE VARS (BY ADDR)
FSNPTR       DS  A   POINTER TO FIRST SYMBOL NODE
LSNPTR       DS  A   POINTER TO LAST SYMBOL NODE
CUBTCPTR     DS  A   PTR TO COMP. UNIT BLOCK DATA CELL
BTREEPTR     DS  A   POINTER TO ROOT OF BLOCK TREE
FSTMTNUM     DS  H   FIRST STATEMENT NUMBER
LSTMTNUM     DS  H   LAST STATEMENT NUMBER
EXECSTMT     DS  H   # OF EXECUTABLE STATEMENTS
STMTNODE     DS  H   # OF STATEMENT NODES
FSTNPTR      DS  A   POINTER TO FIRST STATEMENT NODE
LSTNPTR      DS  A   POINTER TO LAST STATEMENT NODE
SNELPTR      DS  A   POINTER TO STATEMENT NODE EXTENT LIST
FIRSTSRN     DS  CL8
LASTSRN      DS  CL8
CUBTCNUM     DS  H   BLOCK NUMBER OF UNIT BLOCK
COMPUNIT     DS  H   COMPILATION UNIT ID CODE
TITLEPTR     DS  F   VIRTUAL MEMORY POINTER TO TITLE INFOR
USERDATA     DS  CL8 FREE FOR USER DATA
SYMBCNT      DS  F   ACTUAL NUMBER OF SYMBOLS IN COMP.
MACROCNT     DS  F   TOTAL SIZE OF MACRO TEXT (BYTES)
LITSCNT      DS  F   TOTAL NUMBER OF LITERAL STRING
XREFCNT      DS  F   ACTUAL NUMBER OF XREF ENTRIES
DRCLEN       EQU  *_DROOTCEL
             MEND
```

```
'NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(SYMBDC)'
          MACRO
          SYMBDC
SYMBDC    DSECT    SYMBOL DATA CELL
BLOCKNUM  DS   H   BLOCK NUMBER
EXTDOFF   DS   C   OFFSET TO EXTENSION DATA
XREFOFF   DS   C   OFFSET TO XREF DATA
ARRAYOFF  DS   C   OFFSET TO ARRAYNESS DATA
STRUCTOF  DS   C   OFFSET TO STRUCTURE DATA
CLASS     DS   C   SYMBOL CLASS
TYPE      DS   C   SYMBOL TYPE
FLAG1     DS   C   FLAG BYTE ONE
FLAG2     DS   C   FLAG BYTE TWO
FLAG3     DS   C   FLAG BYTE THREE
FLAG4     DS   C   FLAG BYTE 4
SYMBLEN   DS   C   LENGTH OF SYMBOL NAME
RELADDR   DS   CL3 RELATIVE CORE ADDRESS
SBLKID    DS   H   UNIQUE BLOCK ID
TEMPL#    EQU  *   HALFWORD FOR MAJOR STRUCTURE TEMPLATE SYMBOL #
DENSEOFF  EQU  *   DENSE BIT STRING OFFSET (ONE BYTE)
CHARLEN   EQU  *   HALFWORD FOR CHARACTER STRING LENGTH
ROWS      DS   C   NUMBER OF ROWS (LENGTH)
BITLEN    EQU  *   BYTE FOR BIT STRING LENGTH
COLUMNS   DS   C   NUMBER OF COLUMNS (LENGTH OF VECTOR)
LOCK#     DS   C   LOCK GROUP # OF VARIABLE (IF LOCKED)
BYTESIZE  DS   CL3 TOTAL NUMBER OF BYTES USED BY SYMBOL
NAMECONT  DS   OC  SYMBOL NAME CONTINUATION
SDCLEN    EQU  * - SYMBDC
           MEND

'NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(BLKTCELL)'
         MACRO
         BLKTCELL
BLKTCELL DSECT          BLOCK DATA C
RTREEPTR DS    A        RIGHT TREE POINTE
LTREEPTR DS    A        LEFT TREE POINTER
FNESTPTR DS    F        FIRST NESTED BLOC
LNESTPTR DS    F        NEXT BLOCK AT SAME
EXTPTR   DS    A        SYMBOL EXTENT CELL PO
         DS    F        SPARE
BLKFLGS  DS    C        FLAGS APPLICABLE TO THE
         DS    C        SPARE
BLKNDX   DS    H        BLOCK INDEX
BLKID    DS    H        BLOCK (STACK) ID
BLKCLASS DS    C        CLASS OF BLOCK
BLKTYPE  DS    C        TYPE OF BLOCK
FSYMB#   DS    H        FIRST SYMBOL NUMBER
LSYMB#   DS    H        LAST SYMBOL NUMBER
FSTMT#   DS    H        FIRST STATEMENT NUMBER
LSTMT#   DS    H        LAST STATEMENT NUMBER
POSTDCL  DS    H        STMT # OF FIRST POST-DECLARE STMT
STAKLIST DS    H        LIST HEAD FOR STACK VARS (BY ADDR)
BNAMELEN DS    C        LENGTH OF BLOCK NAME
BLKNAME  DS    0C       NAME OF BLOCK (1 TO 32 CHARACTERS)
BTCELLEN EQU   *-BLKTCELL
         MEND
```

```
‘NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(STMTDC)’
           MACRO
           STMTDC
STMTDC     DSECT      STATEMENT DATA CELL
BNUM       DS  H      BLOCK NUMBER
STMTTYPE   DS  H      STATEMENT TYPE
NUMLABLS   DS  CL1    NUMBER OF LABELS
NUMLHS     DS  CL1    NUMBER OF LEFT-HAND SIDES
           MEND

‘NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(BLCKNODE)’
           MACRO
           BLCKNODE
BLCKNODE   DSECT     BLOCK NODE TEMPLATE
CSECTNAME  DS  CL8   CSECT NAME OF BLOCK
BLOCKPTR   DS  A     POINTER TO BLOCK TREE CELL
           MEND

‘NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(SYMBNODE)’
           MACRO
           SYMBNODE
SYMBNODE   DSECT     SYMBOL NODE
SYMBNAME   DS   CL8  FIRST EIGHT CHARACTERS OF SYMBOL NAME
SDCPTR     DS   A    POINTER TO SYMBOL DATA CELL
           MEND

‘NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(STMTNOD0)’
           MACRO
           STMTNOD0
STMTNOD0   DSECT       STATEMENT NODE(SRN_FLAG=0)
STDCPTR    DS    A     POINTER TO STATEMENT DATA CELL
           MEND

‘NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(STMTNOD1)’
           MACRO
           STMTNOD1
STMTNOD1   DSECT       STATEMENT NODE(SRN_FLAG=1)
SRN        DS   CL6    STATEMENT REFERENCE NUMBER
INCOUNT    DS   CL2    INCLUDE COUNT
STDCPTR1   DS   A      POINTER TO STATEMENT DATA CELL
           MEND
```

11. SDFPKG Return Codes

SDFPKG return codes are returned both in Register 15 and in the Communication Table (COMMTABL) variable CRETURN.

Return Code	Relevant Mode #s		Description
0	all	Reason:	Operation successful.
		Description:	The requested SDFPKG operation was successful.
		Effects:	The requested SDFPKG operation was successful and the data is ready for processing by the user's program.
		Recommendation:	Not Applicable.
4	0	Reason:	HALSDF DD Open Unsuccessful.
		Description:	The dataset(s) specified by the HALSDF or alternate name DD card(s) could not be opened by SDFPKG.
		Effects:	SDFPKG cannot initialize itself and cannot access the SDF data; therefore, the SDF information is unavailable to the user's program.
		Recommendation:	Check the user's setup code (i.e. JCL) to make sure the HALSDF or alternate name DD card(s) is/are provided and that it points to a valid SDF dataset.
8[1]	4, 18	Reason:	Select (or Deselect in FSWAT C version) failure: SDF not found in PDS.
		Description:	The SDFPKG Select (or Deselect in FSWAT C version) was unsuccessful since the specified SDF member could not be found in the PDS.
		Effects:	SDFPKG was unable to locate the specified SDF member in the PDS; therefore the data

[1] If the Auto-Select option has been requested, then modes 5, and 7 - 17 can result in return codes 8 and 12 also.

			for that SDF member is unavailable to the user's program.
		Recommendation:	Check to see that all of the SDF datasets are concatenated to the HALSDF or alternate DD card(s). Alternately, check the SDF member name being passed to SDFPKG to make sure it is correct and spelled correctly and that it is truly contained in the SDF PDS.
12[2]	4	Reason:	FCB Area Exhausted.
		Description:	The FCB Area is full and the FCB Automatic GETMAIN option was not specified at the time SDFPKG was Initialized.
		Effects:	SDFPKG cannot select the requested SDF member since it has insufficient space to build the FCB for the member.
		Recommendation:	Provide SDFPKG with more FCB Area using the Augment operation. Alternately, SDFPKG may be Initialized with a larger FCB Area or allowed to automatically GETMAIN the FCB area as needed. Another option is to Initialize SDFPKG using One-FCB Mode. (See sections 13.1and 13.3 for more information)
16	11, 12	Reason:	Block name not found.
		Description:	The specified Block name could not be found in the currently selected SDF member.
		Effects:	The information for the requested Block will not be available to the user's program.
		Recommendation:	Check to see that the Block name is spelled correctly and that it is contained within the currently selected SDF member.
20	12, 13	Reason:	Symbol name or SRN not found.

[2] If the Auto-Select option has been requested, then modes 5, and 7 - 17 can result in return codes 8 and 12 also.

		Description:	The specified Symbol name or SRN could not be found in the SDF member.
		Effects:	The information for the requested Symbol or SRN will not be available to the user's program.
		Recommendation:	Check to see that the Symbol name is spelled correctly or that the correct SRN was specified and that it is contained within the currently selected SDF member.
24	10, 14	Reason:	Statement is non-executable.
		Description:	The Statement for which the ISN or SRN was specified is not an executable statement and does not possess an Executable Statement Data Cell; it may, however, be a Declare statement and possess a Declare Statement Data Cell.
		Effects:	This statement is not executable; however, Data may be available for this statement in the form of a Declare Statement Data Cell.
		Recommendation:	If this is not a Declare statement, then make sure the correct ISN or SRN was specified and that it applies to the currently selected SDF member.
28	14	Reason:	SDF does not have SRNs.
		Description:	The compiled source member either did not contain SRNs or was compiled without the SRN option being specified. The SDF member does not contain any SRN information.
		Effects:	The SDF does not contain any SRN information, therefore statements cannot be located using SRN numbers.
		Recommendation:	Recompile the source member so that SRN information is available in the SDF member

			or use ISNs to locate the appropriate Statement Data Cell.
32	14	Reason:	SRNs not in increasing order; search will not be attempted.
		Description:	The SRNs specified in the SDF member are not in increasing order and cannot be used by SDFPKG in a binary search to locate the Statement Data Cell.
		Effects:	SDFPKG cannot locate the specified SRN using its binary search, therefore Statement Data Cells cannot be located using SRNs.
		Recommendation:	Re-sequence the source member and recompile it so that the SRNs are in increasing order, or use ISNs to locate the Statement Data Cells.
36	10, 17	Reason:	ISN is outside legal range.
		Description:	The specified statement number is outside the range that is legal for the currently selected SDF member.
		Effects:	SDFPKG cannot locate the specified ISN because it is not contained within the SDF member; therefore, the Statement Data Cell information will be unavailable for the requested ISN.
		Recommendation:	Check to see that the correct ISN was specified and that it exists within the currently selected SDF member.

12. SDFPKG Abend Codes

The SDFPKG Abend codes are listed and described below.

Abend Code	Relevant Mode #s		Description
4001	all	Reason:	Paging Area exhausted, i.e. all pages are reserved.
		Description:	The Paging Area is exhausted, i.e. all pages are reserved and further I/O is impossible.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot retrieve any more SDF data.
		Recommendation:	Check to see that any Reserved SDF data is Released as soon as possible. Alternately, the user may allocate a larger Paging Area.
4002	all	Reason:	SYNAD error on HALSDF or alternate DD. (Assembly version only)
		Description:	SDFPKG received a SYNAD error while attempting to use the dataset specified by the HALSDF or alternate DD card(s). This may be the result of an SDF member Open/Close or Read/ Write operation failure or SDFPKG being improperly initialized with user specified options.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot retrieve any more SDF data.
		Recommendation:	Check the user specified SDFPKG initialization options to make sure that they are correct and provide SDFPKG with the correct inputs. In addition, check to see that the correct SDF dataset(s) was/were specified and that they can be accessed without producing an error.
4003	5-17	Reason:	Reserve count overflow -- too many reserves for one page.

		Description:	SDFPKG has encountered a Reserve count overflow, i.e. the user's program has specified too many Reserve operations for one SDF page.
		Effects:	SDFPKG and the user's program Abend since SDFPKG can no longer keep an accurate count of the number of Reserves applied to a single SDF page.
		Recommendation:	Identify the area of the user's program that continues to Reserve a single page without ever Releasing it.
4004	5-17	Reason:	Reserve count underflow -- too many Releases for one page.
		Description:	SDFPKG has encountered a Reserve count underflow, i.e. the user's program has specified too many Release operations for one SDF page.
		Effects:	SDFPKG and the user's program Abend.
		Recommendation:	Identify the area of the user's program that Releases a single page without having ever Reserved it or Releases it more times than it was Reserved.
4005	5	Reason:	Bad SDF virtual memory pointer.
		Description:	SDFPKG was given an invalid virtual memory pointer, i.e. a pointer that does not exist within this SDF member.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot retrieve the SDF data located at the specified virtual memory pointer.
		Recommendation:	Check to make sure the specified virtual memory pointer was retrieved or calculated correctly in the user's program and that it applies to the currently selected SDF member.

4006	8, 15	Reason:	Bad block number specified.
		Description:	SDFPKG has encountered an invalid Block number, i.e. a Block number that is outside the range that is valid for this SDF member.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot locate the specified Block within the currently selected SDF member.
		Recommendation:	Identify the area of the user's program that is requesting the Block information and make sure that it is requesting the information from the correct SDF member and that the Block number is being retrieved or calculated correctly and that the limit checking for any loops is correct.
4007	9, 16	Reason:	Bad symbol number specified.
		Description:	SDFPKG has encountered an invalid Symbol number, i.e. a Symbol number that is outside the range that is valid for this SDF member.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot locate the specified Symbol within the currently selected SDF member.
		Recommendation:	Identify the area of the user's program that is requesting the Symbol information and make sure that it is requesting the information from the correct SDF member and that the Symbol number is being retrieved or calculated correctly and that the limit checking for any loops is correct.
4008	5-17	Reason:	MODF specified but SDFPKG not Initialized with UPDAT option.
		Description:	A call to SDFPKG specified the MODF (Modify) option; however the UPDAT

			(Update) option was not specified when SDFPKG was Initialized.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot perform the requested update, i.e. write out the Modified SDF page.
		Recommendation:	Check the user's program to make sure that Update capability is actually desired; if so, then change the SDFPKG Initialization call to include the UPDAT option.
4009	1-17	Reason:	First call to SDFPKG was not Initialize.
		Description:	The user's program did not Initialize SDFPKG prior to calling it for the first time.
		Effects:	SDFPKG and the user's program Abend since SDFPKG is not Initialized and cannot retrieve any SDF data.
		Recommendation:	Identify the area of the user's program that fails to initialize SDFPKG and add the code to properly initialize it.
4010	5-17	Reason:	No SDF currently selected.
		Description:	An SDF member was not selected prior to requesting Block, Symbol, or Statement information or prior to locating an SDF virtual memory pointer. It is very possible that a previous SDF Select may have failed.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot identify which SDF member is to be used to retrieve the SDF data.
		Recommendation:	Identify the area of the user's program that is requesting the information from SDFPKG and make sure that an SDF member is actually selected prior to requesting the SDF data.

4011	3	Reason:	Paging Area Rescind failure -- one or more pages are reserved.
		Description:	One or more pages were still Reserved when SDFPKG received a Paging Area Rescind request.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot Rescind the Paging Area.
		Recommendation:	Identify the area of the user's program that is requesting the Paging Area Rescind and make sure that all SDF pages have been Released prior to attempting the Rescind.
4012	3	Reason:	Paging Area Rescind failure -- no external area established.
		Description:	SDFPKG encountered a Paging Area Rescind request when no external (Augment) paging area had been established.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot Rescind that which it does not have.
		Recommendation:	Identify the area of the user's program that is requesting the Paging Area Rescind and either delete the request or perform at least one Paging Area Augment prior to this request.
4013	0, 2	Reason:	Bad Paging Area Specification.
		Description:	A Paging Area was incorrectly specified to SDFPKG by the user's program.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot use the specified Paging Area.
		Recommendation:	Identify the area of the user's program that is specifying the Paging Area and correct the error.

4014	6	Reason:	Set disposition parameters called prior to any "locate"-type request.
		Description:	SDFPKG received a request to set the disposition parameters prior to any "locate"-type requests being made.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot set the disposition of that which it does not have.
		Recommendation:	Identify the area of the user's program that is making the "Set Disposition" request and make sure that a "locate"-type request has been successfully issued prior to the "Set Disposition" request.
4015	0, 2	Reason:	Bad FCB Area Specification.
		Description:	An FCB Area was incorrectly specified to SDFPKG by the user's program.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot use the specified FCB Area.
		Recommendation:	Identify the area of the user's program that is specifying the FCB Area and correct the error.
4016	n/a	Reason:	Bad mode number input.
		Description:	SDFPKG encountered an invalid Mode number (i.e., one not between 0 and 17 in the Assembly version, or 0 and 18 in the FSWAT C version).
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot identify the mode desired by the user.
		Recommendation:	Identify the area of the user's program that is making the invalid request and correct it.
4017	0	Reason:	Multiple calls to SDFPKG Initialize.

		Description:	SDFPKG has encountered multiple requests to Initialize itself prior to a Terminate request.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot initialize itself more than once without an intervening Terminate request.
		Recommendation:	Identify the area of the user's program that is requesting the second SDFPKG Initialize and correct it either by deleting the second Initialize request or adding a Terminate request prior to the second SDFPKG Initialize request.
4018[3]	4	Reason:	FCB Area exhausted - automatic GETMAIN was unsuccessful. (Assembly version only)
		Description:	The SDFPKG FCB Area was exhausted and an automatic GETMAIN was attempted by SDFPKG, but was unsuccessful in obtaining any more space.
		Effects:	SDFPKG and the user's program Abend since SDFPKG does not have the FCB space to open any more SDF members and, so, cannot retrieve any more SDF data.
		Recommendation:	Identify the area of the user's program that Initializes SDFPKG and provide it with a larger initial FCB area that is allocated by the user's program or increase the Region size for the user's program.
4019	0	Reason:	GETMAIN failure in SDFPKG Initialize. (Assembly version only)
		Description:	SDFPKG encountered an error during a GETMAIN while attempting to Initialize itself.

[3] If the Auto-Select option has been requested, then modes 5, and 7 - 17 can result in this Abend code as well.

		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot Initialize itself.
		Recommendation:	Identify the area of the user's program performing the SDFPKG Initialize and allocate all of the FCB and PAD areas in the user's program, or increase the Region size for the user's program.
4020	13	Reason:	HAL Block not "set" prior to locate Symbol Data Cell using only Symbol name.
		Description:	The user did not specify a HAL Block prior to attempting to locate a Symbol Data Cell using only the Symbol name.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot be sure which Symbol to retrieve as there may be multiple symbols of the same name in different Blocks. Note that there may be symbols of the same name in the same Block if TEMPORARY data is involved.
		Recommendation:	Identify the area of the user's program that is requesting the Symbol information and make sure that the appropriate Block is specified prior to asking for the Symbol information.
4021	all	Reason:	Internal SDFPKG storage area overflow.
		Description:	SDFPKG has encountered an internal storage area overflow.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot retrieve any more SDF data.
		Recommendation:	Identify the area of the user's program that performs the SDFPKG Initialize and allocate all of the FCB and PAD areas in the user's program, or increase the Region size for the user's program.

4022[4]	4	Reason:	New SDF Select requested when pages of last selected SDF are Reserved.
		Description:	After having been initialized in One-FCB mode (i.e., MISC=8 was specified during SDFPKG Initialization), SDFPKG encountered a request to Select a new SDF member when one or more pages of the currently selected SDF member were still Reserved.
		Effects:	SDFPKG and the user's program Abend since SDFPKG cannot overlay the current FCB until all pages have been Released.
		Recommendation:	Identify the area of the user's program that is attempting to perform a Select of a new SDF member and make sure that all pages Reserved in the previously Selected member have been Released, or change the SDFPKG initialization to allow multiple FCBs.
4023	43	Reason:	SDF length mismatch. (Assembly version only)
		Description:	An incompatible (old) SDF file version was found in the SDF when it was Selected.
		Effects:	SDFPKG and the user's program Abend since the data is incompatible with the current version of SDFPKG.
		Recommendation:	Either change the DDNAME to a more recent version of the SDF file if it exists, or recompile the HAL/S source to produce an updated SDF file.

[4] If the Auto-Select option has been requested, then modes 5, and 7 - 17 can result in this Abend code as well.

13. SDFPKG Mode Calls

Note 1) If the Auto-Select option is requested for mode 5 or 7-17, then return codes 8 and 12 apply additionally. Also, the SDFNAM field of the communication area must be set to the 8-character SDF name. FSWAT C version only: The calling program must handle the Auto-Select option by calling sdfpkg_call(4) if the option is set (see section 8.3).

Note 2) The MISC field of the communication area is used only as an input to INITIALIZE.

13.1 Mode 0 - Initialize SDFPKG

A. Input:

R0	Address of communication table	
MODE	0	
MISC[5]	1 →	SDFPKG is to automatically GETMAIN FCB Area as needed
	2 →	UPDAT mode (MODF parameter legal)
	4 →	Alternate DDNAME contained within communication table field SDFDDNAM (actually the same field normally occupied by the SDF name)
	8 →	ONEFCB mode (old FCB is overlain by new FCB)
	16 →	FIRST symbol mode (take 1st symbol when found regardless of type in mode 12 or 13). It is possible that more than one symbol with the specified name may exist within the specified block (like an EQUATE label, TEMPORARY data, or within a STRUCTURE template), so the calling program must determine if the symbol found by SDFPKG is the desired one.
	32 →	Alternate PAD is supplied. (Assembly version only)
APGAREA	Zero or address of user-supplied external paging area	
AFCBAREA	Zero or address of user-supplied external FCB area	
NPAGES	Zero or number of pages in nucleus paging area	
NBYTES	Zero or number of bytes in initial FCB Area	
ADDR	Zero or the address of the external PAD Area supplied by the calling program. (Assembly version only)	
PNTR	Zero or the number of 16-byte PAD entries provided by the calling program. (Assembly version only)	

The following table describes the result of different settings for the last mentioned four input parameters.

ADDR	PNTR	Result
1) Address	0 < PNTR ≤ 4095	SDFPKG will accept the external PAD provided by the calling program. The number of entries contained in the PAD will limit the maximum number of slots in the Paging Area. The Paging Area cannot be longer than the PAD, though it may be shorter.
2) 0	0	SDFPKG will use the default of 250 for the number of PAD entries.
APGAREA	**NPAGES**	**Result**
1) Address	1 ≤ N ≤ 250	SDFPKG will accept users paging area of N pages and

[5] More than one MISC option may be specified by combining them additively.

			will not GETMAIN an area
2)	0	$1 \leq N \leq 250$	SDFPKG will perform its own GETMAIN to build a paging area of N pages (if N=0, SDFPKG assumes 2)
3)	Address	N=0	Abend 4013
AFCBAREA		NBYTES	Result
4)	Address	NB > 0 MISC ≠ 1	SDFPKG will accept users external FCB area of NB bytes and will not GETMAIN any areas. User program will be notified of FCB Area depletion by a return code of 12
5)	Address	NB > 0 MISC = 1	SDFPKG will accept users external FCB area of NB bytes but will GETMAIN additional area automatically if needed (512-byte extensions)
6)	0	NB > 0 MISC irrelevant	SDFPKG will perform a GETMAIN to build an initial FCB Area of NB bytes (if NB = 0, SDFPKG assumes 1024) and will GETMAIN additional area automatically if needed (in 512-byte or larger extensions)
7)	Address	NB=0	Abend 4015

B. Output:

R15 CRETURN	0 → Operation successful 4 → DCB OPEN failure
R1 ADDR	Address of SDFPKG internal data area (see Section 14)
APGAREA AFCBAREA NBYTES	0
NPAGES	Length of PAD (250 default) - nucleus size, i.e. the number of pages that can yet be added via an augment

13.2 Mode 1 - Terminate SDFPKG

A. Input: MODE 1

B. Output:

R15 CRETURN	0 → Operation successful
APGAREA AFCBAREA NBYTES	0
NPAGES	250. The maximum possible Paging Area size

13.3 Mode 2 - Augment Paging Area and/or FCB Area

A. Input:

MODE	2
APGAREA	Zero or address of external Paging Area augment
AFCBAREA	Zero or address of external FCB Area augment
NPAGES	Zero or number of pages in Paging Area augment
NBYTES	Zero or number of bytes in FCB Area augment

APGAREA		NPAGES	Result
1)	Address	$1 \leq N \leq$ max	Current Paging Area will be extended by the N page external area
2)	0	nonzero	Abend 4013

3)	-	0	No action
AFCBAREA		NBYTES	Result
4)	Address	NB > 0	The NB byte area will be added to the available FCB Area
5)	0	nonzero	Abend 4015
6)	-	0	No action

General Notes:

a. Multiple augments can be done for either Paging Area or FCB Area.
b. The Paging Area cannot be augmented past its maximum value (currently 50).
c. None of the augmented FCB Areas can be rescinded.
d. All Paging Area augments are removed by a single rescind.

B. Output:

R15, CRETURN	0 → Operation successful
APGAREA, AFCBAREA, NBYTES	0
NPAGES	Number of pages which can yet be added to the Paging Area

13.4 Mode 3 - Rescind Paging Area Augments

A. Input: MODE 3

B. Output:

R15, CRETURN	0 → Operation successful
APGAREA, AFCBAREA, NBYTES	0
NPAGES	Number of pages which can yet be added to the Paging Area.

13.5 Mode 4 - Select an SDF (Explicitly)

A. Input:

DISP	0 (Auto-Select parameter should not be specified)
MODE	4
SDFNAM	8 character SDF name, e.g. ##NAVIGA

B. Output:

R15, CRETURN	0 → Operation successful 8 → BLDL unsuccessful (member not found) 12→ FCB Area is exhausted (only if user is supplying FCB Areas)

13.6 Mode 5 - Locate Pointer

A. Input:

DISP	{SELECT, MODF, RESV, RELS}
MODE	5
PNTR	Virtual memory pointer to be located

B. Output:

R15, CRETURN	0 → Operation successful
R1, ADDR	Memory address corresponding to "located" pointer

13.7 Mode 6 - Set Disposition Parameters

A. Input:	DISP	{MODF, RESV, RELS}
	MODE	6
B. Output:	R15 CRETURN	0 → Operation successful

13.8 Mode 7 - Locate Directory Root Cell

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	7
B. Output:	R15 CRETURN	0 → Operation successful
	R1 ADDR	Memory address of Directory Root Cell
	PNTR	Virtual memory pointer to Directory Root Cell

13.9 Mode 8 - Locate Block Data Cell given Block Number

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	8
	BLKNO	Block Number
B. Output:	R15 CRETURN	0 → Operation successful
	R1 ADDR	Memory address of Block Data Cell
	PNTR	Pointer to Block Data Cell
	BLKNLEN	Number of characters in block name
	CSECTNAM	Name of code CSECT of block
	BLKNAM	Block Name (up to 32 EBCDIC characters)

13.10 Mode 9 - Locate Symbol Data Cell given Symbol Number

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	9
	SYMBNO	Symbol Number
B. Output:	R15 CRETURN	0 → Operation successful
	R1 ADDR	Memory Address of Symbol Data Cell
	PNTR	Pointer to Symbol Data Cell
	SYMBNLEN	Number of characters in symbol name
	SYMBNAM	Symbol Name (up to 32 EBCDIC characters)

	BLKNO	Block Number of block to which symbol belongs

13.11 Mode 10 - Locate Statement Data Cell given Statement Number

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	10
	STMTNO	Statement Number
B. Output:	R15 CRETURN	0 → Operation successful 24 → Statement is non-executable 36 → Statement number is outside legal range
	R1 ADDR	Memory Address of Statement Data Cell
	PNTR	Pointer to Statement Data Cell
	SREFNO INCLCNT	Augmented SRN (if the SDF has them)
	BLKNO	Block Number of block to which statement belongs

13.12 Mode 11 - Locate Block Data Cell given Block Name

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	11
	BLKNLEN	Number of characters in block name
	BLKNAM	Block name
B. Output:	R15 CRETURN	0 → Operation successful 16 → No block found with name given
	R1 ADDR	Memory Address of Block Data Cell
	PNTR	Pointer to Block Data Cell
	BLKNO	Block Number
	CSECTNAM	Name of block's code CSECT

13.13 Mode 12 - Locate Symbol Data Cell given Block Name and Symbol Name

Algorithm: If FIRST option is set, then return the symbol.
Otherwise, look at symbol class:
If EQUATE EXTERNAL, unqualified STRUCTURE or TEMPLATE, then skip symbol and continue search, otherwise return the symbol.

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	12
	BLKNLEN	Number of characters in block name
	BLKNAM	Block name
	SYMBNLEN	Number of characters in symbol name
	SYMBNAM	Symbol name
B. Output:	R15 CRETURN	0 → Operation successful 16 → Block not found 20 → Block found but symbol not found
	R1 ADDR	Memory Address of Symbol Data Cell

	PNTR	Pointer to Symbol Data Cell
	BLKNO	Block Number
	SYMBNO	Symbol number
	CSECTNAM	Name of block's code CSECT

13.14 Mode 13 - Locate Symbol Data Cell given Only Symbol Name

Note: A Mode 13 call must have been preceded at some point by a Mode 8, 11, or 12 call. Successive mode 13 calls are legal.

Algorithm: If FIRST option is set, then return the symbol.
Otherwise, look at symbol class:
If EQUATE EXTERNAL, unqualified STRUCTURE or TEMPLATE, then skip symbol and continue search, otherwise return the symbol.

A. Input:	DISP	{MODF, RESV, RELS}
	MODE	13
	SYMBNLEN	Number of characters in symbol name
	SYMBNAM	Symbol name
B. Output:	R15 CRETURN	0 → Operation successful 20 → Symbol not found
	R1 ADDR	Memory Address of Symbol Data Cell
	PNTR	Pointer to Symbol Data Cell
	SYMBNO	Symbol number

13.15 Mode 14 - Locate Statement Data Cell given SRN

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	14
	SREFNO INCLCNT	Augmented SRN, i.e. SRN + Include Count
B. Output:	R15 CRETURN	0 → Operation successful 20 → SRN not found 24 → Statement is non-executable 28 → SDF does not have SRNs 32 → SRNs are non-monotonic
	R1 ADDR	Memory address of Statement Data Cell
	PNTR	Pointer to Statement Data Cell
	BLKNO	Block number of block to which statement belongs
	STMTNO	Statement number

13.16 Mode 15 - Locate Block Node given Block Number

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	15
	BLKNO	Block number
B. Output:	R15 CRETURN	0 → Operation successful
	R1 ADDR	Memory address of Block Node
	PNTR	Pointer to Block Node
	CSECTNAM	Name of block's code CSECT

13.17 Mode 16 - Locate Symbol Node given Symbol Number

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	16
	SYMBNO	Symbol Number
B. Output:	R15 CRETURN	0 → Operation successful
	R1 ADDR	Memory address of Symbol Node
	PNTR	Pointer to Symbol Node

13.18 Mode 17 - Locate Statement Node given Statement Number

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE	17
	STMTNO	Statement Number
B. Output:	R15 CRETURN	0 → Operation successful 36 → Statement number is outside legal range
	R1 ADDR	Memory address of Statement Node
	PNTR	Pointer to Statement Node
	SREFNO INCLCNT	Augmented SRN (if the SDF has SRNs)

13.19 Mode 18 - Deselect an SDF (FSWAT C version only)

Note: This mode can be useful when many SDF members are to be processed. Deselecting members facilitates the reusage of the FCB area, minimizing the need for augmentation (mode 2) and the possibility of running out of space.

A. Input:	DISP	0
	MODE	18
	SDFNAM	8 character SDF name
B. Output:	R15 CRETURN	0 → Operation successful

⌋ 8 → Deselect unsuccessful (member not found)

14. SDFPKG Statistics

Upon return from the Initialize (mode 0) call to SDFPKG, register 1 together with the ADDR field of the communication table will point to the DATABUF internal data pool of SDFPKG.

'NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(DATABUF)' contains a DSECT for this data pool (Assembly version only) and is shown in Figure 14-1.

FSWAT C version only: The internal data pool is called *sdfpkg_internal* and is dynamically allocated by SDFPKG. Only the following variables from Figure 14-1 are implemented and available to the user: LOCCNT, NUMGETM, NUMOFPGS, BASNPGS, GETMFLAG, GOFLAG, MODFLAG, ONEFCB, FIRST, TOTFCBLN, RESERVES, READS, WRITES, SLECTCNT, FCBCNT, SDFVERS.

Much of this data is of little or no interest to a user program, but the following variables may be useful:

LOCCNT	Total number of locates (explicit or implicit)
READS	Total number of reads from the SDF PDS
WRITES	Total number of writes to the SDF PDS (UPDAT mode only)
NUMGETM	Total number of GETMAINs performed by SDFPKG
NUMOFPGS	Current Paging Area size
BASNPGS	Size of "nucleus" Paging Area
TOTFCBLN	Total size of all FCBs
FCBCNT	Number of FCBs in FCB Area
SLECTCNT	Total number of "real" selects, i.e., the number of times that reference was actually switched from one SDF to another.
RESERVES	Total reserve count (sum of reserve count of all active memory slots)

The Terminate call (mode 1) zeros out this data area so that the values must be extracted prior to the call. These parameters are maintained dynamically and may be accessed at any time between the Initialize and Terminate calls.

```
'NCAMCM.PASS.CURRENT.DIAGNSTC.MACLIB(DATABUF)'
00100                MACRO
00100                DATABUF
00300  DATABUF       DSECT COMMON DATA BUFFER
00400  LOCCNT        DS    F   CURRENT LOCATE COUNTER
00500  AVULN         DS    A   ADDRESS OF VULNERABLE PAD ENTRY
00600  CURFCB        DS    A   ADDRESS OF CURRENT FCB
00700  PADADDR       DS    A   STARTING ADDRESS OF PAD
00800  ACOMMTAB      DS    A   ADDRESS OF COMMUNICATION AREA
00900  ACURNTRY      DS    A   ADDRESS OF CURRENT PAD ENTRY
01000  ROOT          DS    A   ADDRESS OF ROOT FCB OF FCB TREE
01100  SAVEXTPT      DS    F   POINTER TO SYMBOL NODE EXTENT CELL
01200  SAVFSYMB      DS    H   FIRST SYMBOL OF BLOCK
01300  SAVLSYMB      DS    H   LAST SYMBOL OF BLOCK
01400  NUMGETM       DS    H   NUMBER OF ENTRIES IN GETMAIN STACKS
01500  NUMOFPGS      DS    H   NUMBER OF PAGES IN CURRENT PAGING AREA
01600  BASNPGS       DS    H   INITIAL NUMBER OF PAGES IN PAGING AREA
01700  FCBSTKLN      DS    H   NUMBER OF ENTRIES IN FCB STACKS
01800  IOFLAG        DS    C   I/O IN PROGRESS INDICATOR
01900  GETMFLAG      DS    C   > 0 IMPLIES AUTO GETMAINS FOR FCBS
02000  GOFLAG        DS    C   > 0 IMPLIES SUCCESSFUL INITIALIZATION
02100  MODFLAG       DS    C   1   IMPLIES UPDAT MODE ACTIVE
02200  ONEFCB        DS    C   > 0 IMPLIES ONLY ONE FCB KEPT
02205  FIRST         DS    C   1   IMPLIES TAKE FIRST SYMBOL FOUND
02300                DS    2C  SPARE
02400  TOTFCBLN      DS    F   TOTAL AMOUNT OF FCB SPACE IN USE
02500  RESERVES      DS    F   GLOBAL (TOTAL) COUNT OF RESERVES
02600  READS         DS    F   TOTAL NUMBER OF READS
02700  WRITES        DS    F   TOTAL NUMBER OF WRITES
02800  SLECTCNT      DS    F   TOTAL NUMBER OF 'REAL' SELECTS
02900  FCBCNT        DS    F   TOTAL NUMBER OF FCBS IN EXISTENCE
03000  GETMSTK1      DS    A   ADDRESS OF GETMAIN ADDRESS STACK
03100  GETMSTK2      DS    A   ADDRESS OF GETMAIN LENGTH STACK
03200  FCBSTK1       DS    A   ADDRESS OF FCB AREA ADDRESS STACK
03300  FCBSTK2       DS    A   ADDRESS OF FCB AREA LENGTH STACK
03400  MAXSTACK      DS    H   MAXIMUM NUMBER OF STACK ENTRIES
03500  SDFVERS       DS    H   SDF VERSION NUMBER (OF SELECTED SDF)
03600  APGEBUFF      DS    A   ADDRESS OF PAGE BUFFER
03700  ADECB         DS    A   ADDRESS OF DECB
03800  ECB           DS    F   EVENT CONTROL BLOCK (DECB)
03900  IOTYPE        DS    H   I/O TYPE (DECB)
04000  IOLENGTH      DS    H   NUMBER OF BYTES TO TRANSFER (DECB)
04100  DCBADDR       DS    A   ADDRESS OF HALSDF DCB (DECB)
04200  EUFLOC        DS    A   ADDRESS OF BUFFER AREA (DECB)
04300  IOBADDR       DS    A   ADDRESS OF IOB (DECB)
04400                MEND
```

Figure 14-1 SDFPKG Internal Data Pool

15. Glossary

Abend

IBM-360 terminology for an abnormal program termination.

BLDL

The BLDL Assembly macro instruction is used to place PDS directory information in main storage. The data is in a build list, which must be constructed before the BLDL macro instruction is issued. The format of the list is similar to that of the directory. For each member name in the list, the system supplies the address of the member and any additional (user data) information contained in the directory entry. Note that if there is more than one member name in the list, the member names must be in alphabetical order, regardless of whether the members are from the same or different libraries (PDS files). BLDL can be used to optimize the retrieval of directory information using the FIND macro instruction.

Calling program

The user's program that calls SDFPKG and is responsible for manipulating the data that SDFPKG returns. The calling program can be in Assembly, C/C++, or XPL language.

CHECK

When processing a dataset, you can test for completion of a READ or WRITE request by using the CHECK Assembly macro instruction. The system tests for errors and exception conditions in the Data Event Control Block (DECB). Successive CHECK macro instructions issued for the same dataset must be issued in the same order as the associated READ and WRITE macro instructions.

CLOSE

The CLOSE Assembly macro instruction is used to terminate the processing of a dataset and to release it from a DCB. The volume dispositioning information that is to result from closing the dataset can also be specified. The volume dispositioning options that may be specified are the same as those that can be specified for the end-of-volume conditions in the OPEN macro instruction or the JCL DD statement. Before issuing the CLOSE macro, a CHECK macro must be issued for all DECBs that have outstanding I/O requests from the WRITE macro instruction.

COMMTABL

Communication Table - This contiguous 120-byte Table of Variables is used to pass data to and receive data from SDFPKG.

DATABUF

Data Buffer - This Table lives within SDFPKG and contains the current status information. The address of this table is returned in the ADDR variable by SDFPKG after an initialization call.

DD / DDNAME

Data Definition is a statement in JCL that describes the file and its attributes to be used in a specific operation. The DDNAME is an identifier that refers to the JCL DD statement.

DSECT

An Assembly language "Dummy Section" which symbolically describes the logical structure of a memory area.

FCB

File Control Block - This SDFPKG Data structure contains all the information needed to Read or Write an SDF page to disk.

FIND

An Assembly macro instruction that determines the starting address of a specific member. The system places the correct address in the Data Control Block (DCB) so that a subsequent input or output operation begins processing at that point.

FREEMAIN

An Assembly language macro instruction that releases one or more areas of virtual storage previously allocated by GETMAIN.

GETMAIN

An Assembly language macro instruction that requests the OS to allocate one or more areas of virtual storage.

HALSDF

The DDNAME normally used for the SDF dataset when running SDFPKG, the HAL/S compilers, etc.

HAL/S-FC

The version of the HAL/S Compiler which runs on the Space Shuttle Flight Computers (AP-101/S).

HAL/S-360

The version of the HAL/S Compiler which runs on any IBM 360/370 or compatible machine.

Include Count

This is a sequential number, starting with 1, that is given to each "card" within a data group that is included into the main source member.

ISN or **ISNs**

Internal Statement Number(s) - This is the sequential number that is assigned by the compiler to every program statement.

JCL

Job Control Language used to code statements which describe a job to the OS and inform the system how the job is to be processed.

OPEN

The OPEN Assembly macro instruction is used to complete a Data Control Block (DCB) that is needed to "open" an associated dataset. The method of processing (accessing) and the end-of-volume dispositioning instructions may also be specified.

OS or **O/S**

Operating system - in our case, this is OS-MVS/SP (Multiple Virtual Systems/System Product), MVS/XA (MVS/Extended Architecture), and MVS/ESA (MVS/Enterprise System Architecture).

PAD

Paging Area Directory - This SDFPKG Data Structure is used to keep track of the address of a "Page" in the Paging Area and to correlate which FCB is associated with that Page.

Page

A slot of memory of a predetermined length which is used to hold data that is read in from disk in a virtual memory system. A group of these slots is referred to as the paging area.

Paging

The act of manipulating Pages by reading in and writing out the desired data that is referenced in a virtual memory system.

PDS

Partitioned Dataset - A file that can contain one or more members (as opposed to a sequential dataset which is a singular file). The HALSDF file is a PDS.

POINT

The POINT Assembly macro instruction causes repositioning of a magnetic tape or direct access volume to a specified block (not record). The next read or write operation will then begin at this block. If a write operation follows the POINT macro instruction, the operation will begin at the previous block (assuming the dataset is opened in UPDAT mode). In a multi-volume dataset, the calling program must insure that the volume referred to is the volume currently being processed. If a write operation follows the POINT instruction and the dataset is not opened in UPDAT mode, then all of the tracks immediately following the written block will be erased.

READ

The READ Assembly macro instruction retrieves a data block (not record) from an input dataset and places it in a designated area of main storage. To allow overlap of the input operation with processing, the system returns control to the calling program before the read operation is completed. The DECB (Data Event Control Block) created for the read operation must be tested for successful completion before the record is processed or the DECB is reused. If a dataset is being read, the block is brought into main storage and the address of the block is returned to the calling program in the DECB.

SDF

Simulation Data File(s) produced by the HAL/S-FC compiler.

SDFPKG

Simulation Data File Access Package - This is the SDF access program that is described within this document.

SRN or **SRNs**

Statement Reference Number(s) - This is the 6 character sequence number that appears on the program source "card" in columns 73-78. This SRN, in combination with the Include Count, creates a unique identifier for every source statement in the program.

User program

This is synonymous with the calling program.

Virtual Memory

A data paging system, as implemented in SDFPKG, which allows for a small amount of memory to be used to store data, and the remaining data is in a file on the disk. Data

references by a program may require disk accesses to load the data, which is transparent to the user.

WRITE

The WRITE Assembly macro instruction places a data block (not record) in an output dataset from a designated area of main memory. The WRITE macro instruction can also be used to return an updated record to a dataset. To allow overlap of output operations with processing, the system returns control to the calling program before the write operation is completed. The DECB (Data Event Control Block) created for the write operation must be tested for successful completion before the DECB can be reused.

16. Index

T

U

V