MSD MEMO #TX--

TO: Distribution

FROM: Jeff Day

DATE: 30 January 1992

SUBJECT: Simulation Data File (SDF) Access Package

UPDATE OF: MSD Memo #TX-108-86

DATED: 06 October 1986

BY: J. Day

UPDATE OF: Shuttle Memo #12-75

OF: 07 March 1975

BY: C. Shulenberg

DISTRIBUTION:

IBM

D. Brubaker
S. Ford
T. Metcalf

Intermetrics

G. Cetrone
C. Henson
J. Pate
D. Simmons
B. Whitfield
G. Winslow

Intermetrics

P. Ansley
D. Brack
J. Crawley
R. Handley
L. Kuo
S. McCullough
J. Payne
R. Prettyman
T. Rose
J. Sobieski
R. St.John
D. Strauss
T. Varesic

Preface:

The document this memo replaces is over 4 years old, in which time, an Assembler interface program has been written to allow SDFPKG to be called from PL/I (see Section 5). In addition, it was discovered that several SDFPKG features were not documented and other areas were unclear. This updated and expanded memo seeks to rectify the situation and provide more of the needed information.

CONTENTS

Figures

1.0 Simulation Data File (SDF) Access Package

SDFPKG is an IBM-360/370 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 the data contained within the Simulation Data Files (SDFs). SDFPKG can be separately link edited and employed as a loadable and deletable service module, or it may be linked directly with other software. The latter is the case with the HAL/S-360 stand-alone diagnostic system, while the former method is used in the XPL monitor and PL/I interface module (Section 5).

1.1 Paging Area

Paging is done directly between core memory and the Partitioned Dataset (PDS) containing the SDFs generated by Phase III of the HAL/S Compilers. This is made possible by the list of TTRs (Track, Track, Records -- See Section 2 for a description) contained within the last physical record(s) of each SDF. A TTR is present for each record of the file. Reads can thus be accomplished via a FIND, POINT, READ sequence. Figure 4 of the current HAL/SDL Interface Control Document (HAL/SDL ICD) NAS9-14444 shows the physical layout of an SDF with the TTR record(s) (or page(s)) at the end of the file. The TTR record(s) contains pointers to all other file records and is itself, in turn, pointed to by one or more TTRs in the User Data area of the PDS directory entry.

SDF records (or pages) are always 1680 bytes in length. This is true even of the TTR page(s) which may contain as little as 4 bytes of data. 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 altering an assembly language macro parameter in SDFPKG or by the user program providing SDFPKG with a larger Paging Area Directory (PAD) and Paging Area. Figure 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.

Contents of Paging Area Directory (PAD) Entry

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 1: Contents of Paging Area Directory (PAD) Entry

The absolute maximum number of entries in the paging area supported by SDFPKG is 4095 and, 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 call), 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). 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 and Rescind (remove all augments) calls. The Rescind call always reduces the paging area size to the initial (nucleus) area.

The core 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 core 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.

1.2 Virtual Memory Considerations

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 core address of the data located in one of SDFPKG's virtual memory pages (if it has not been read into core, this value is 0). As described in the HAL/SDL ICD, 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 core address of the corresponding data as output. The returned core 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 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 core 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, Executable 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 core memory and the calling program may extract (or insert in update mode) data using the core address provided (see Figures 2 and 3).

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 core 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 <u>after</u> the SDFPKG call that RESV, RELS, or MODF is desirable, then one or more of these disposition parameters can be specified by a DISP (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.

SDFPKG Multi-File Control Block (FCB) Mode

Figure 2: SDFPKG Multi-File Control Block (FCB) Mode

SDFPKG One-File Control Block (FCB) Mode

Figure 3: SDFPKG One-File Control Block (FCB) Mode

Contents of File Control Block (FCB) Entry

Each FCB entry consists of the following information:

TTR - TTR of the PDS Directory Record for BLDL

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 prior to TTR records

VERSIONX - Phase III Version number of SDF member

STMTEXPT - Pointer to Block Statement Extent Cell

- Spare 2 Fullwords

The following two fields exist for every non-TTR record in the SDF member.

FCBTTRS - TTRS of each record in SDF member

FCBPDADR - Address of PAD entry holding this Record or 0 if not currently in memory

Figure 4: Contents of File Control Block (FCB) Entry

1.3 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 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 the TTR list from the last SDF member page(s), 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 (See Figures 2 and 3). The FCB (See Figure 4 for contents) is allocated from a block of memory called the FCB area and is discussed in Section 1.4. 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 core 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.

1.4 FCB Area

The FCB Area (see Figures 2 and 3 for Usage and Figure 4 for contents) 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 signalled 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 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 members in the SDF.

If the calling program does not need the flexibility of the user program allocating the FCB area, 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 x number of SDF members passed. Again, all such GETMAIN'ed areas are freed when SDFPKG is called to Terminate.

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 (see Figure 4). FCBs are thus highly variable in length.

1.5 Paging Strategy

The PAD contains an entry for each core 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 Figures 1, 2, and 3 for more information). As mentioned earlier, the reserve count is used to lock the page in its core 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 core. 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 core slot from the PDS, the core 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.

1.6 General Consideration

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

1) SDFPKG is a modular access method for SDF members built upon a demand paging virtual memory environment. It can be separately linked, loaded, and deleted or link edited into the user program.

2) All calls to SDFPKG are made through a single ENTRY point by supplying a mode number. Eighteen different mode calls (0-17) are currently provided.

3) SDFPKG employs a paging area of from 1 to the defined limit (250 is the default) of pages in size and may be dynamically expanded or contracted as the core memory situation alters.

4) SDFPKG can support simultaneous access to an unlimited number of SDF members. The area needed for FCBs can be automatically provided by SDFPKG or be controlled by the calling program.

5) SDFPKG is serially reusable. Following a Terminate call an Initialize call may be made and everything starts over again from the beginning.

6) SDFPKG allows SDF members to be either modified or merely read.

7) SDFPKG provides built-in binary search algorithms to allow efficient high-level access to data that must be searched.

8) SDFPKG FREEMAINs all storage at the Terminate call that it has GETMAIN'ed since the Initialize call.

9) SDFPKG performs one OPEN (for the HALSDF or alternate DD) at Initialize (mode 0 call) and one CLOSE (same DD) at the Terminate (mode 1) call.

10) SDFPKG uses only the following system services and macros: GETMAIN, FREEMAIN, FIND, BLDL, POINT, READ, WRITE, CHECK, OPEN, and CLOSE.

11) SDFPKG can be configured at Initialize so that it will perform no GETMAINs.

12) SDFPKG performs complete error checking and will force an ABEND in case of a legitimate user or I/O error (See Section 1.14). A complete set of return codes is used to signal abnormal conditions that are not reflections of serious user error (See Section 1.13).

13) Almost all communication between SDFPKG and the calling program is accomplished through a 120-byte "communication" table that is provided by the calling program (See Section 1.7). This eliminates almost all parameter passing.

14) SDFPKG maintains statistical information which has bearing on its operation as well as other data of interest which the calling program may access at any time.

15) SDFPKG is designed to be as fast as possible without sacrificing essential error checks. With a large paging area, the efficiency is competitive with implementations in which all SDF members are core resident in their entirety.

16) SDFPKG requires approximately 15,000 bytes of core, exclusive of the FCB and Paging Areas.

1.7 Communication Table (COMMTABL)

The Communication Table (COMMTABL) is a contiguous 120-byte data area that the calling program must supply. The address of COMMTABL is passed to SDFPKG during the Initialize call. The assembler DSECT overlay for COMMTABL is shown in Figure 5. The Declaration and/or structure of this communication table may vary somewhat from language to language and from language interface to language interface.

SDFPKG Communication Table (COMMTABL)

BYTE OFFSET	FIELD NAME		FIELD **SIZE**	DESCRIPTION
	COMMTABL	DSECT		SDFPKG COMMUNICATION AREA
0	APGAREA	DS	A	ADDRESS OF EXTERNAL PAGING AREA
4	AFCBAREA	DS	A	ADDRESS OF EXTERNAL FCB AREA
8	NPAGES	DS	H	# OF PAGES IN PAGING AREA OR AUGMENT
10	NBYTES	DS	H	# OF BYTES IN FCB AREA OR AUGMENT
12	MISC	DS	H	MISCELLANEOUS PURPOSES
14	CRETURN	DS	H	SDFPKG RETURN CODE
16	BLKNO	DS	H	BLOCK NUMBER (BLOCK INDEX TABLE ENTRY
	*			INDEX)
18	SYMBNO	DS	H	SYMBOL NUMBER (SYMBOL INDEX TABLE ENTRY
	*			INDEX)
20	STMTNO	DS	H	STATEMENT NUMBER (STATEMENT INDEX TABLE
	*			ENTRY INDEX)
22	BLKNLEN	DS	CL1	NUMBER OF CHARACTERS IN BLOCK NAME
	*			(BLKNAM)
23	SYMBNLEN	DS	CL1	NUMBER OF CHARACTERS IN SYMBOL NAME
	*			(SYMBNAM)
24	PNTR	DS	F	VIRTUAL MEMORY POINTER LAST LOCATED
28	ADDR	DS	A	CORE ADDRESS CORRESPONDING TO PNTR
32	SDFDDNAM	EQU	*	NAME OF ALTERNATE DD FOR SDF DATASET
32	SDFNAM	DS	CL8	NAME OF SDF TO BE SELECTED
40	CSECTNAM	DS	CL8	NAME OF CODE CSECT FOR BLOCK
48	SREFNO	DS	CL6	STATEMENT REFERENCE NUMBER (SRN)
54	INCLCNT	DS	H	INCLUDE COUNT (FOR SRN)
56	BLKNAM	DS	CL32	BLOCK NAME
88	SYMBNAM	DS	CL32	SYMBOL NAME
120	MEND			

Field Formats and Descriptions

1. PNTR -- This is the pointer (i.e. virtual memory location) to the SDF data. The pointer is in the FORMAT PPOO, where

   PP -- is a 2-byte sub-field of the SDF Virtual Page Number (record) containing the data

   OO -- is a 2-byte sub-field giving the offset into the page (record) where the data begins.

2. ADDR -- This is the actual memory location corresponding to the PNTR location of the data.

Figure 5: SDFPKG Communication Table (COMMTABL)

1.8 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 internal data pool of SDFPKG. An assembler DSECT for this data pool is available and is shown in Figure 6.

Many of the statistics are of little or no interest to the calling program or user, but the following variables may be useful:

LOCCNT - Total number of locates (explicit or implicit). This is actually:

(Hex 'FFFFFF') MODULO (# of locates)

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

SLECTCN - Total number of "real" selects, i.e. the number of times that reference was actually switched from one SDF member to another

RESERVES - Total reserve count (sum of reserve counts of all active core slots)

SDFPKG Internal Data Pool Overlay (DATABUF)

BYTE OFFSET	FIELD NAME		FIELD **SIZE**	DESCRIPTION
	DATABUF	DSECT	COMMON	DATA BUFFER
0	LOCCNT	DS	F	CURRENT LOCATE COUNTER
4	AVULN	DS	A	ADDRESS OF VULNERABLE PAD ENTRY
8	CURFCB	DS	A	ADDRESS OF CURRENT FCB
12	PADADDR	DS	A	STARTING ADDRESS OF PAD
16	ACOMMTAB	DS	A	ADDRESS OF COMMUNICATION AREA
20	ACURNTRY	DS	A	ADDRESS OF CURRENT PAD ENTRY
24	ROOT	DS	A	ADDRESS OF ROOT FCB OF FCB TREE
28	SAVEXTPT	DS	F	POINTER TO SYMBOL NODE EXTENT CELL
32	SAVFSYMB	DS	H	FIRST SYMBOL OF BLOCK
34	SAVLSYMB	DS	H	LAST SYMBOL OF BLOCK
36	NUMGETM	DS	H	NUMBER OF ENTRIES IN GETMAIN STACKS
38	NUMOFPGS	DS	H	NUMBER OF PAGES IN CURRENT PAGING AREA
40	BASNPGS	DS	H	INITIAL NUMBER OF PAGES IN PAGING AREA
42	FCBSTKLN	DS	H	NUMBER OF ENTRIES IN FCB STACKS
44	IOFLAG	DS	C	I/O IN PROGRESS INDICATOR
45	GETMFLAG	DS	C	> 0 IMPLIES AUTO GETMAINS FOR FCBS
46	GOFLAG	DS	C	> 0 IMPLIES SUCCESSFUL INITIALIZATION
47	MODFLAG	DS	C	> 0 IMPLIES UPDAT MODE ACTIVE
48	ONEFCB	DS	C	> 0 IMPLIES ONLY ONE FCB KEPT
49	FIRST	DS	C	> 0 IMPLIES TAKE FIRST SYMBOL FOUND
50		DS	2C	SPARE
52	TOTFCBLN	DS	F	TOTAL AMOUNT OF FCB SPACE IN USE
56	RESERVES	DS	F	GLOBAL (TOTAL) COUNT OF RESERVES
60	READS	DS	F	TOTAL NUMBER OF READS
64	WRITES	DS	F	TOTAL NUMBER OF WRITES
68	SLECTCNT	DS	F	TOTAL NUMBER OF 'REAL' SELECTS
72	FCBCNT	DS	F	TOTAL NUMBER OF FCBS IN EXISTENCE
76	GETMSTK1	DS	A	ADDRESS OF GETMAIN ADDRESS STACK
80	GETMSTK2	DS	A	ADDRESS OF GETMAIN LENGTH STACK
84	FCBSTK1	DS	A	ADDRESS OF FCB AREA ADDRESS STACK
88	FCBSTK2	DS	A	ADDRESS OF FCB AREA LENGTH STACK
92	MAXSTACK	DS	H	MAXIMUM NUMBER OF STACK ENTRIES
94	SDFVERS	DS	H	SDF VERSION NUMBER (OF SELECTED SDF)
96	APGEBUFF	DS	A	ADDRESS OF PAGE BUFFER
100	ADECB	DS	A	ADDRESS OF DECB
104	ECB	DS	F	EVENT CONTROL BLOCK (DECB)
108	IOTYPE	DS	H	I/O TYPE (DECB)
110	IOLENGTH	DS	H	NUMBER OF BYTES TO TRANSFER (DECB)
112	DCBADDR	DS	A	ADDRESS OF HALSDF DCB (DECB)
116	BUFLOC	DS	A	ADDRESS OF BUFFER AREA (DECB)
120	IOBADDR	DS	A	ADDRESS OF IOB (DECB)
124		MEND		

Figure 6: SDFPKG Internal Data Pool Overlay (DATABUF)

The Terminate call (mode 1) zeros out this data area so 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.

1.9 SDFPKG Calling Sequence

The following block diagram (Figure 7) 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. This indicates 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 Sections 4 (Assembly Language), 5 (XPL/Monitor), and 6 (PL/I).

SDFPKG Calling Sequence

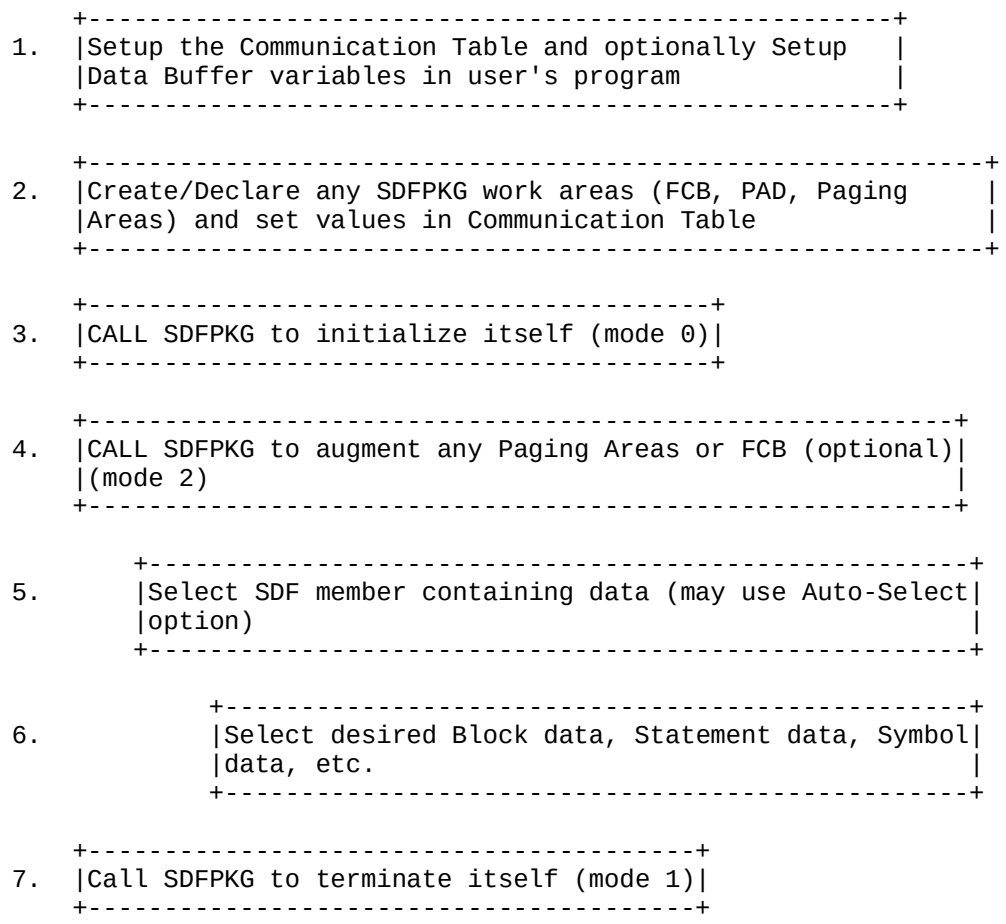
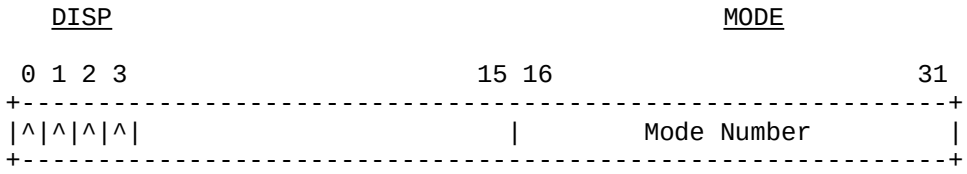


Figure 7: SDFPKG Calling Sequence

1.10 SDFPKG Mode Argument Format

Figure 8 shows the format for passing the SDFPKG Mode and option(s). As illustrated in Figure 8, the SDFPKG mode is passed in the lower halfword of the argument while the disposition options (Auto-Select, Modify, Release, and Reserve) are passed in the upper most 4 bits (Nibble).

Mode Argument Format



BITS		PURPOSE
0	-	Auto-Select
1	-	MODF (Modify)
2	-	RELS (Rescind/Release)
3	-	RESV (Reserve)
4-15	-	Unused
16-31	-	Mode Number

Figure 8: Mode Argument Format

Upon return from SDFPKG, the return code is available in the CRETURN location in the Communication Table (COMMTABL), while ADDR in the same table generally contains the core memory address of the "located" data item.

It should be noted that the Communication Table variables are not altered unless they are explicitly output by the SDFPKG call.

1.11 SDFPKG Mode Numbers

The following is a summary of the currently supported SDFPKG Mode calls:

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 an SDFPKG 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 Internal Statement Number (ISN).
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 Statement Reference Number (SRN).
15	Locate a Block Index Table Entry given the Block Number.
16	Locate a Symbol Index Table Entry given the Symbol Number.
17	Locate a Statement Index Table Entry given the Internal Statement Number (ISN).

1.12 SDFPKG - HAL/SDL ICD Terminology Mapping

The definitions and layouts for the various data blocks contained within an SDF and returned by SDFPKG can be found in the current version of the HAL/SDL Interface Control Document (ICD). In the past, different terminology was employed between the HAL/SDL ICD and SDFPKG, however now an effort is being made to standardize the SDF table and cell names, so Figure 9 may be helpful in resolving the new names.

Naming Convention Cross-Reference Table

HAL/S Compiler Specification and SDFPKG Terminology	HAL/S SDL ICD Terminology	Standardized Table/Cell Names	HAL/SDL ICD Figure No.	DSECT Template Names (Section 7)
Directory Root Cell	Simulation Table or Directory Header	Master Directory Cell	Section 1.2.1.2.1 (Figure 5)	PAGEZERO
Directory Root Cell	Simulation Table or Directory Header	Directory Root Cell	Section 1.2.1.2.1 (Figure 5)	DROOTCEL
Block Data Cell	HAL Block List Member	Block Data Cell	Section 1.2.1.2.1.2.2 (Figure 7)	BLKTCELL
Symbol Data Cell	Symbol Data Entry	Symbol Data Cell	Section 1.2.1.2.2.2 (Figure 11)	SYMBDC ARRADATA STRCDATA
Statement Data Cell (Executable)	Statement Data Entry	Executable State-ment Data Cell	Section 1.2.1.2.3.2.1 (Figure 14)	STMTDC
Statement Data Cell (Declare)	Statement Data Entry	Declare Statement Data Cell	Section 1.2.1.2.3.2.2 (Figure n/a)	(n/a)
Block Node	Block Index Table Entry	Block Index Table Entry	Section 1.2.1.2.1.2.1 (Figure 6)	BLCKNODE
Symbol Node	Symbol Names and Pointers Table Entry	Symbol Index Table Entry	Section 1.2.1.2.2.1 (Figure 10)	SYMBNODE
Statement Node	Statement Names and Pointers Table Entry	Statement Index Table Entry	Section 1.2.1.2.3.1 (Figure 12)	STMTNOD0 (No SRNs) STMTNOD1 (With SRNs)

Figure 9: Naming Convention Cross-Reference Table

Naming Convention Cross-Reference Table

HAL/S Compiler Specification and SDFPKG Terminology	HAL/S SDL ICD Terminology	Standardized Table/Cell Names	HAL/SDL ICD Figure No.	DSECT Template Names (Section 7)
Statement Block Extent Cell	Statement Block Extent Cell	Block Statement Extent Cell	Section 1.2.1.2.1.3 (Figure 9)	STMTEXTF STMTEXTV
Symbol Block Extent Cell	Symbol Block Extent Cell	Block Symbol Extent Cell	Section 1.2.1.2.1.2.3 (Figure 8)	SYMEXTF SYMEXTV

Figure 9: Naming Convention Cross-Reference Table

1.13 SDFPKG Return Codes

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

Return Code	Relevant Mode Numbers	Error Description
0	all	Error Message: 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	Error Message: 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 JCL stream to make sure the HALSDF or alternate name DD card(s) is/are provided and that it points to a valid SDF dataset.
8*	4*	Error Message: Select failure: SDF not found in PDS. Description: The SDFPKG Select was unsuccessful since the specified SDF member could not be found in the PDS. Effects: SDFPKG was unable to locate the

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 4* Error Message: 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 1.4 and 1.15 for more information)

1611, 12 Error Message: 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 Error Message: Symbol name or SRN not found.

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 Error Message: 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 Error Message: 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.

Effects: SDFPKG cannot locate the specified SRN using its binary search, therefore Statement Data Cells cannot be located using SRNs.

Recommendation: Resequence 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 Error Message: 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.

* If the Auto-Select option (Select SDF member implicitly) has been requested, then modes 5, and 7 through 17 (Locate Modes) can result in return codes 8 and 12 as well.

1.14 SDFPKG ABEND Codes

The SDFPKG ABEND codes are listed and described below:

ABEND Code	Relevant Mode Numbers	ABEND Description
4001	all	**Error Message:** 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	**Error Message:** SYNAD error on HALSDF or alternate DD. **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

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 Error Message: 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 Error Message: 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

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 Error Message: 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 Error Message: 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

capability is actually desired; if so, then change the SDFPKG Initialization call to include the UPDAT option.

4009 1-17 Error Message: 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 Error Message: 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 Error Message: Paging Area Rescind failure -- one or more pages are reserved.

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 Error Message: 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 Error Message: 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 Error Message: 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.

successfully issued prior to the "Set Disposition" request.

4015 0, 2 Error Message: 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 Error Message: Bad mode number input.

Description: SDFPKG encountered an invalid Mode number (i.e., one not between 0 and 17).

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 Error Message: 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

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 Error Message: GETMAIN failure in SDFPKG Initialize.

Description: SDFPKG encountered an error during a GETMAIN while attempting to Initialize itself.

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 Error Message: 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

the Symbol information.

4021 all Error Message: 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 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.

4022 4* Error Message: 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.

1.15 SDFPKG Mode Calls

A more detailed description of the different SDFPKG modes as well as their inputs and outputs are described in the following sub-sections.

General Notes:

1) If the Auto-Select option is requested for mode 5, and 7 through 17 calls 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.

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

3) Multiple Augments may be performed for either Paging Area or FCB Area

4) The Paging Area cannot be Augmented past the maximum length of the PAD (the default length is 250)

5) None of the augmented FCB Areas can be Rescinded.

6) All Paging Area augments are removed by a single Rescind.

1.15.1 Mode 0 - Initialize SDFPKG

A. Input: R0 Address of communication table

MODE, R1 0

MISC*1 -- SDFPKG is to automatically GETMAIN the FCB Area as needed

2 -- UPDAT mode (MODF parameter is legal)

4 -- Alternate DDNAME is contained within communication table field SDFDDNAM (actually the same field normally occupied by the SDF member name)

8 -- One-FCB mode (old FCB is overlayed by new FCB each time a Select is performed)

16 -- FIRST mode (Symbol name search returns with the first name found)

32 -- Alternate PAD is supplied.

FCB area.

NBYTES Zero, the number of bytes in the initial FCB Area, or the negative value of the number of 128-byte FCB entries being allocated (generally this is the same as the number of members in the SDF).

ADDR Zero or the address of the calling program supplied external PAD Area.

PNTR Zero or the number of 16-byte PAD entries provided by the calling program

The following table describes the result of different settings for the APGAREA, AFCBAREA, NPAGES, NBYTES, ADDR, and PNTR input parameters mentioned above.

1)	ADDR	PNTR	RESULTS
a)	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.
b)	0	PNTR = 0	SDFPKG will use its internal default (currently 250) for the number PAD entries. In this situation, the number of "slots" in the Paging Area is limited to 250.

2)	APGAREA	NPAGES	RESULTS
a)	Address	1 <= N <= PAD Length	SDFPKG will accept the paging area of N pages provided by the calling program and will not GETMAIN an area of its own
b)	0	1 <= N <= PAD Length	SDFPKG will perform its own GETMAIN to build a paging area of N pages (if N=0, SDFPKG defaults to 2)

3)	AFCBAREA	NBYTES	RESULTS
a)	Address	NB > 0, MISC 1	SDFPKG will accept the external FCB area of NB bytes provided by the calling program and will not GETMAIN

			FCB Area depletion through a return code of 12.
c)	Address	NB > 0, MISC = 1	SDFPKG will accept the external FCB area of NB bytes provided by the calling program and will GETMAIN additional space as needed (in 512-byte extensions)
d)	Address	NB < 0, MISC = 1	SDFPKG will accept the external FCB area of NB number of 128-byte entries provided by the calling program and will GETMAIN additional space as needed (in 512-byte extensions).
e)	0	NB > 0, MISC = 1	SDFPKG will perform a GETMAIN to build an initial FCB area of NB bytes and will GETMAIN additional area as needed (in 512-byte extensions)
f)	0	NB < 0	SDFPKG will perform a GETMAIN to build an initial FCB area of NB number of 128-byte FCB entries, and will not GETMAIN any additional area.
g)	0	NB = 0, MISC = 1	SDFPKG will GETMAIN an initial 1024-byte internal FCB area and will then GETMAIN additional area as needed (in 512-byte extensions).

B. Output: CRETURN, R15 — 0 - Operation was successful
4 - DCB OPEN failure

ADDR, R1	Address of the SDFPKG internal data area.
APGAREA, AFCBAREA, NBYTES	0
NPAGES	Length of PAD (default is 250) - Paging Area nucleus size, i.e. the number of pages that can yet be added via an augment call. Since this is the initialization call, no pages have been added, and therefore all of it can be added.

1.15.2 Mode 1 - Terminate SDFPKG

possible paging area size is limited by the number of entries in the PAD.

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

A. Input: MODE, R1 — 2

APGAREA — Zero or the address of the external FCB Area augment

NPAGES — Zero or the number of pages in the Paging Area augment

NBYTES — Zero or the number of bytes in the FCB Area Augment

1)	APGAREA	NPAGES	RESULT
a)	Address	1 <= N <= PAD Length	Current Paging Area will be extended by the N page external area
b)	0	-	No action

2)	AFCBAREA	NBYTES	RESULT
a)	Address	NB > 0	The NB byte area will be added to the available FCB Area
b)	0	-	No action

B. Output: CRETURN, R15 — 0 - Operation was successful

APGAREA, AFCBAREA, NBYTES — 0

NPAGES — The number of pages which can yet be added to the Paging Area

1.15.4 Mode 3 - Rescind Paging Area Augments

A. Input: MODE, R1 — 3

B. Output: CRETURN, R15 — 0 - Operation was successful

APGAREA, AFCBAREA, NBYTES — 0

NPAGES — The number of pages which can yet be added

	SDFNAME	8 character SDF name, e.g. ##NAVIGA
B. Output:	CRETURN, R15	0 - Select was successful 8 - BLDL was unsuccessful (member not found) 12 - FCB Area was exhausted (This return code is valid only if the user's program is supplying the FCB Areas)

1.15.6 Mode 5 - Locate Pointer

A. Input:	DISP	{SELECT, MODF, RESV, RELS}
	MODE, R1	5
	PNTR	Virtual memory pointer to be located
B. Output:	CRETURN, R15	0 - Operation was successful
	ADDR, R1	Core address corresponding to the "located" pointer

1.15.7 Mode 6 - Set Disposition Parameters

A. Input:	DISP	{MODF, RESV, RELS}
	MODE, R1	6
B. Output:	CRETURN, R15	0 - Operation was successful

1.15.8 Mode 7 - Locate Directory Root Cell

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

B. Output: CRETURN, R15 0 - Operation was successful

ADDR, R1 Core address of the Block Data Cell

PNTR Pointer to the Block Data Cell

BLKNLEN Number of characters in the Block name

CSECTNAM Name of the code CSECT of the Block requested

BLKNAM Block Name (up to 32 characters)

1.15.10 Mode 9 - Locate Symbol Data Cell given Symbol Number

A. Input: DISP {SELECT, MODF, RESV, RELS}

MODE, R1 9

SYMBNO Symbol Number

B. Output: CRETURN, R15 0 - Operation was successful

ADDR, R1 Core Address of the Symbol Data Cell

PNTR Pointer to the Symbol Data Cell

SYMBNLEN Number of characters in the Symbol name

SYMBNAM Symbol name (up to 32 characters)

BLKNO Block Number of the Block to which the Symbol belongs

1.15.11 Mode 10 - Locate Statement Data Cell Given Internal Statement Number (ISN)

A. Input: DISP {SELECT, MODF, RESV, RELS}

MODE, R1 10

STMTNO ISN

B. Output: CRETURN, R15 0 - Locate was successful
24 - Statement is non-executable (i.e.

SREFNO, INCLCNT — SRN and Include Count (if the SDF has them)

BLKNO — Block number of the Block to which the statement belongs

1.15.12 Mode 11 - Locate Block Data Cell given Block Name

A. Input:

DISP — {SELECT, MODF, RESV, RELS}

MODE, R1 — 11

BLKNLEN — Number of the characters in Block name

BLKNAM — Block name

B. Output:

CRETURN, R15 — 0 - Locate was successful
16 - No Block was found with the given name

ADDR, R1 — Core Address of the Block Data Cell

PNTR — Pointer to the Block Data Cell

BLKNO — Block Number

CSECTNAM — Name of the Block's code CSECT

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

A. Input:

DISP — {SELECT, MODF, RESV, RELS}

MODE, R1 — 12

BLKNLEN — Number of characters in the Block name

SYMBNLEN — Number of characters in the Symbol name

SYMBNAM — Symbol name

B. Output:

CRETURN, R15 — 0 - Locate was successful
16 - Block was not found
20 - Block was found but the Symbol was not found

ADDR, R1 — Core address of the Symbol Data Cell

1.15.14 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 to set the Block in which the Symbol resides. Successive Mode 13 calls are legal.

A.	Input:	DISP	{MODF, RESV, RELS}
		MODE, R1	13
		SYMBNLEN	Number of characters in the Symbol name
		SYMBNAM	Symbol name
B.	Output:	CRETURN, R15	0 - Locate was successful 20 - Symbol was not found
		ADDR, R1	Core address of the Symbol Data Cell
		PNTR	Pointer to the Symbol Data Cell
		SYMBNO	Symbol number

1.15.15 Mode 14 - Locate Statement Data Cell given Statement Reference Number (SRN)

A.	Input:	DISP	{SELECT, MODF, RESV, RELS}
		MODE, R1	14
		SREFNO, INCLCNT	Augmented SRN, i.e., SRN + Include Count
B.	Output:	CRETURN, R15	0 - Locate was successful 20 - SRN was not found 24 - Statement is non-executable (i.e. Comment or Declare statement, and may have a Declare Statement Data Cell) 28 - SDF does not have SRNs 32 - SRNs are not in increasing order
		ADDR, R1	Core address of the Executable Statement Data Cell
		PNTR	Pointer to the Executable Statement Data

A. Input: DISP {SELECT, MODF, RESV, RELS}

MODE, R1 15

BLKNO Block Number

B. Output: CRETURN, R15 0 - Operation was successful

ADDR, R1 Core Address of the Block Index Table Entry

PNTR Pointer to the Block Index Table Entry

CSECTNAM Name of the Block's code CSECT

1.15.17 Mode 16 - Locate Symbol Index Table Entry given Symbol Number

A. Input: DISP {SELECT, MODF, RESV, RELS}

MODE, R1 16

SYMBNO Symbol number

B. Output: CRETURN, R15 0 - Operation was successful

ADDR, R1 Core address of the Symbol Index Table Entry

PNTR Pointer to the Symbol Index Table Entry

1.15.18 Mode 17 - Locate Statement Index Table Entry given Statement Number

A. Input: DISP {SELECT, MODF, RESV, RELS}

MODE, R1 17

STMTNO ISN

B. Output: CRETURN, R15 0 - Locate was successful
36 - ISN is outside legal range

ADDR, R1 Core address of the Statement Index Table Entry

PNTR Pointer to the Statement Index Table Entry

2.0 SDFPKG Definitions

BLDL - The BLDL macro-instruction is used to place PDS (Partitioned Data Set) 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.

CHECK - When processing a dataset, you can test for completion of a READ or WRITE request by using the CHECK 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 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.

Core Memory - The address space (memory) allocated for the job. This is the virtual memory supported by the MVS Operating System (OS) and is transparent to the job and the user.

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

DATABUF - Data Buffer - This Table lives within SDFPKG and contains

calling program must issue a FIND macro instruction. 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. There are 2 ways the FIND macro can be used to direct the system to the correct member: (1) by specifying the address of an area containing the name of the member, or (2) by specifying the address of the TTR field of the entry in a build list that was previously created using the BLDL macro. In the first case, the system searches the PDS directory for the relative Track address (TTR), while in the second case, no search is needed since the relative track address is in the build list.

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

OPEN - The OPEN 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. See Section 1.0 for more information.

POINT - The POINT 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-

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.

Real Memory - Real non virtual memory as defined by JCL and the MVS O/S.

SDF - Simulation Data File(s). This is the new name for the Simulation Tables.

Simulation Data Files - See SDF.

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

Simulation Tables - See SDF.

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.

TTR - Track Track Record - Relative Address/Disk I/O - Two kinds of relative addresses can be used to refer to records in a direct-access dataset: relative block addresses and relative track addresses.

The relative block address is a 3-byte binary number that indicates the position of the block relative to the first block of the dataset. Allocation of noncontinuous sets of blocks does not affect the number. The first block of a dataset always has a relative block address of 0.

The relative track address has the form TTR, where:

TT- is a 2-byte binary number specifying the position of the track relative to the first track allocated for the dataset. The TT for the first track is 0. Allocation of noncontinuous sets of tracks does not affect the

Virtual Memory - The data paging system implemented in SDFPKG.

WRITE - The WRITE 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.

3.0 Calling SDFPKG by an Assembler Program

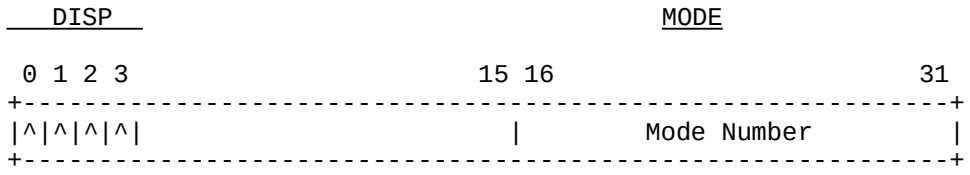
3.1 SDFPKG Register Conventions

When calling SDFPKG from Assembler or an interface program the following Register Conventions are used.

Register 0 is used only on the Initialize call and contains the address of the Communication Table.

Figure 10 shows the format for passing the SDFPKG Mode and option(s) in Register 1. As illustrated in Figure 10, the SDFPKG mode is passed in the lower half-word of register 1 while the disposition options (Auto-Select, Modify, Release, and Reserve) are passed in the upper most 4 bits (Nibble) of Register 1.

Register 1 Format



BITS		PURPOSE
0	-	Auto-Select
1	-	MODF (Modify)
2	-	RELS (Rescind/Release)
3	-	RESV (Reserve)
4-15	-	Unused
16-31	-	Mode Number

Figure 10: Register 1 Format

Upon return from SDFPKG, the return code is available both in Register 15 and in the CRETURN location in the Communication Table (COMMTABL). Register 1, as well as ADDR in the Communication Table, generally contains the core memory address of

3.2 Example Assembler Program

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

```
*  INITIALIZE CALL
          LOAD  EP=SDFPKG
          ST    R0,ASDFPKG          SAVE LOAD ADDRESS
*                ....
          LA    R0,COMMAREA         R0=ADDRESS OF COMMUNICATION AREA
          SR    R1,R1               ZERO REGISTER 1
          L     R15,ASDFPKG         LOAD ADDRESS OF SDFPKG IN R15
          BALR  R14,R15             RETURN ADDR + BRANCH TO SDFPKG
          LTR   R15,R15             TEST RETURN CODE
          BNZ   OPENFAIL            IF NONZERO RETURN CODES, OPEN
*                                   FAILED
*                ...
*  ALL OTHER CALLS
          LA    R1,<mode>
          O     R1,=X'<disposition parameters>'
          LA    R15,ASDFPKG         LOAD ADDRESS OF SDFPKG IN R15
          BALR  R14,R15             LOAD RETURN ADDRESS + BRANCH TO
*                                   SDFPKG
          LTR   R15,R15             TEST RETURN CODE
          BNZ   MISCFAIL            IF NOT 0 THEN FAILED
*                ...
*  TERMINATE CALL
          LA    R1,1                TERMINATE
          L     R15,ASDFPKG         LOAD ADDR OF SDFPKG IN R15
          BALR  R14,R15             LOAD RETURN ADDR IN R14  &
*                                   BRANCH TO SDFPKG
*                 ...
          DELETE EP=SDFPKG          DELETE SDFPKG
*
ASDFPKG   DC    30F'0'              ADDRESS OF SDFPKG
COMMAREA  DC    30F'0'              COMMUNICATION TABLE
```

Register 0 is used only on the Initialize call and contains the address of the Communication Table (COMMTABL).

4.0 Calling SDFPKG From an XPL Program

All XPL programs using SDFPKG must call it using the MONITOR (22) interface. For completeness, this Monitor Call is documented in Section 4.1 below.

4.1 SDFPKG Monitor Call

F=MONITOR (22,n,a); This Monitor call causes Load, Call, and Delete of SDFPKG. This interface is used by HALSTAT, DASS, and other programs to call SDFPKG.

Parameter Purpose

22 - Call SDFPKG
n - SDFPKG Mode (0-17)
a - Address of the Communication Table (COMMTABL) interface table. This interface table is specified only in the SDFPKG initialize call (mode 0).

4.2 XPL Example Program

The following are examples of how SDFPKG might be called from an XPL Program such as HALSTAT.

```
/*########################*/
/*                        */
/*   SDFPKG XPL EXAMPLE   */
/*                        */
/*########################*/


/*==============================*/
/*  LIMITING SIZES AND LENGTHS   */
/*==============================*/

 DECLARE
   /* PHYSICAL BLOCK SIZE OF SDF PAGES */
    PAGE_SIZE              LITERALLY '1680',

   /* MAX NUMBER OF CSECT TYPES */
    MAX_CSECT_TYPES        LITERALLY '19',
```

```
    ADDR_FIXER             FIXED,
    ADDR_FIXED_LIMIT  FIXED,
    ADDR_ROUNDER      FIXED,
    TMP                    FIXED,
    TMP1                   FIXED;

/*--------------------------*/
/*  SDF GLOBAL PARAMETERS   */
/*--------------------------*/

 COMMON
   /* NUMBER OF FIRST EXECUTABLE STATEMENTS */
    FIRST_STMT             BIT(16),

   /* NUMBER OF LAST STATEMENT */
    LAST_STMT              BIT(16),

   /* TOTAL NUMBER OF SYMBOL INDEX TABLE ENTRIES */
    #SYMBOLS               BIT(16),

   /* TOTAL NUMBER OF STATEMENT INDEX TABLE ENTRIES */
    #STMTS            BIT(16),

   /* NUMBER OF INCLUDED COMPOOLS */
    #EXTERNALS             BIT(16),

   /* NUMBER OF HAL BLOCKS */
    #PROCS            BIT(16),

    REF_STAT               BIT(16),
    KEY_BLOCK              BIT(16),
    KEY_SYMB               BIT(16),
    #GAPS             BIT(16),
    #PSEUDO_GAPS      BIT(16),
    MIN_GAP           BIT(16),
    #OVERLAPS              BIT(16),
    #REFS1            BIT(16),
    #REFS2            BIT(16),
    SAVE_BASE              BIT(16),
    SAVE_NDX               FIXED,
    SELECTED_UNIT          BIT(16),
    LAST_MAP_ADDR          FIXED,
    LAST_GAP_ADDR          FIXED,
    LAST_OVERLAP_ADDR FIXED,
    GAP_SIZE               FIXED,
    TOTAL_ERRORS      FIXED,
    TOTAL_STACK_WALKS FIXED,
```

```
   LIBRARY          BIT(16),
   QCON             BIT(16),
   LIBRARY_DATA     BIT(16),
   MAX_PHASE             BIT(16),
   MAP_LEVEL             BIT(16),
   MAX_MAP_LEVEL         BIT(16),
   FILE_LEVEL            BIT(16),
   #U_CMDS          BIT(16),
   CARD_NO          FIXED,
   GSD_LEVEL             BIT(16),
   MSG_LEVEL             BIT(16),
   GFORMAT          BIT(16),

   MFORMAT          BIT(16),
   TOTAL_SYM_REC         FIXED,
   TOTAL_SYM_REC_INSERT   FIXED,
   TOTAL_RLD_REC         FIXED,
   SDFPKG_LOCATES        FIXED,
   SDFPKG_READS     FIXED,
   SDFPKG_SLECTCNT       FIXED,
   SDFPKG_FCBAREA        FIXED,
   SDFPKG_PGAREA         BIT(16),
   SDFPKG_NUMGETM        BIT(16),
   FREE_STRING_SIZE  FIXED,
   MIN_STRING_SIZE       FIXED,
   LHS_RHS_SPILL_OFFSET   BIT(16),

  /* NUMBER OF LABELS                      */
   #LABELS          BIT(16),

  /* NUMBER OF LEFT-HAND SIDES         */
   #LHS                  BIT(16);

/*----------------------------------------*/
/*  DECLARES FOR SDFPKG COMMUNICATION AREA  */
/*----------------------------------------*/

COMMON
   COMM_TAB(29)     FIXED;

COMMON BASED
   COMMTABL_BYTE         BIT(8),
   COMMTABL_HALFWORD BIT(16),
   COMMTABL_FULLWORD FIXED;

COMMON
   COMMTABL_ADDR         FIXED;
```

```
   SYMBNO            LITERALLY 'COMMTABL_HALFWORD (9)',
   STMTNO            LITERALLY 'COMMTABL_HALFWORD (10)',
   BLKNLEN           LITERALLY 'COMMTABL_BYTE (22)',
   SYMBNLEN               LITERALLY 'COMMTABL_BYTE (23)',
   PNTR                   LITERALLY 'COMMTABL_FULLWORD (6)',
   ADDRESS           LITERALLY 'COMMTABL_FULLWORD (7)',
   SDFNAM            LITERALLY 'COMMTABL_ADDR + 32',
   CSECTNAM               LITERALLY 'COMMTABL_ADDR + 40',
   SREFNO            LITERALLY 'COMMTABL_ADDR + 48',
   INCLCNT           LITERALLY 'COMMTABL_HALFWORD (27)',
   BLKNAM            LITERALLY 'COMMTABL_ADDR + 56',
   SYMBNAM           LITERALLY 'COMMTABL_ADDR + 88';


/*-----------------------------------------------------------*/
/*  DECLARES FOR SDFPKG INTERNAL DATA BUFFER  (DATABUF)   */
/*-----------------------------------------------------------*/

COMMON BASED
   DATABUF_BYTE      BIT(8),
   DATABUF_HALFWORD  BIT(16),
   DATABUF_FULLWORD  FIXED;

DECLARE
   LOCCNT            LITERALLY 'DATABUF_FULLWORD (0)',
   AVULN                  LITERALLY 'DATABUF_FULLWORD (1)',
   CURFCB            LITERALLY 'DATABUF_FULLWORD (2)',
   PADADDR           LITERALLY 'DATABUF_FULLWORD (3)',
   ACOMMTAB               LITERALLY 'DATABUF_FULLWORD (4)',
   ACURNTRY               LITERALLY 'DATABUF_FULLWORD (5)',
   ROOT                   LITERALLY 'DATABUF_FULLWORD (6)',
   SAVEXTPT               LITERALLY 'DATABUF_FULLWORD (7)',
   SAVFSYMB               LITERALLY 'DATABUF_HALFWORD (16)',
   SAVLSYMB               LITERALLY 'DATABUF_HALFWORD (17)',
   NUMGETM           LITERALLY 'DATABUF_HALFWORD (18)',
   NUMOFPGS               LITERALLY 'DATABUF_HALFWORD (19)',
   BASNPGS           LITERALLY 'DATABUF_HALFWORD (20)',
   FCBSTKLN               LITERALLY 'DATABUF_HALFWORD (21)',
   IOFLAG            LITERALLY 'DATABUF_BYTE (44)',
   GETMFLAG               LITERALLY 'DATABUF_BYTE (45)',
   GOFLAG            LITERALLY 'DATABUF_BYTE (46)',
   MODFLAG           LITERALLY 'DATABUF_BYTE (47)',
   ONEFCB            LITERALLY 'DATABUF_BYTE (48)',
   FIRST                  LITERALLY 'DATABUF_BYTE (49)',
   TOTFCBLN               LITERALLY 'DATABUF_FULLWORD (13)',
   RESERVES               LITERALLY 'DATABUF_FULLWORD (14)',
   READS                  LITERALLY 'DATABUF_FULLWORD (15)',
```

```
COMMON BASED
   MISC_NAMES             CHARACTER;

DECLARE
  /* LOAD MODULE NAME */
   LMOD              LITERALLY 'MISC_NAMES (0)',

  /* TITLE  (TYPE II COMMON) */
   TITLE                  LITERALLY 'MISC_NAMES (1)',

  /* NAME OF LAST SELECTED SDF */
   SDF_NAME               LITERALLY 'MISC_NAMES (2)',

   GOT_NAME               LITERALLY 'MISC_NAMES (3)',
   MEM_TYPE               LITERALLY 'MISC_NAMES (4)',
   MAX_MISC_NAMES         LITERALLY '4';


  /*---------------------------------------*/
  /* SET UP THE SDFPKG COMMUNICATION AREA  */
  /*---------------------------------------*/

   COREWORD (ADDR (COMMTABL_BYTE))     = ADDR (COMM_TAB);
   COREWORD (ADDR (COMMTABL_HALFWORD)) = ADDR (COMM_TAB);
   COREWORD (ADDR (COMMTABL_FULLWORD)) = ADDR (COMM_TAB);
   COMMTABL_ADDR = ADDR (COMM_TAB);

  /*-----------------------------------------*/
  /* ALLOCATE SDFPKG FILE CONTROL BLOCK AREA  */
  /*-----------------------------------------*/

   TMP = MAX_UNITS;
   IF TMP < 8 THEN DO;
      TMP = 8;
   END;

   NBYTES = -TMP;
   CALL STORAGE_MGT (ADDR (PRO), SHL (TMP, 7), 1, 0);
   AFCBAREA = COREWORD (ADDR (PRO));

  /*-------------------------------------------------*/
  /* ALLOCATE AN AUXILIARY SDFPKG PAGING AREA DIRECTORY  */
  /*-------------------------------------------------*/

   CALL STORAGE_MGT (ADDR (PRO), ALT_PAD_SIZE, 1, 0);
   ADDRESS = COREWORD (ADDR (PRO));
```

```
 APGAREA = COREWORD (ADDR (PRO));
 MISC    = MISC_VAL;

/*-------------------*/
/* INITIALIZE SDFPKG */
/*-------------------*/

 CALL MONITOR (22, 0, COMMTABL_ADDR);

 IF CRETURN ^= 0 THEN DO;
    SPACE_1;

   OUTPUT = '*** OPEN ERROR DETECTED FOR SDF PDS ' ||
            '-- CORRECT JCL AND RESUBMIT ***';

    GO TO BAIL_OUT;
 END;

 COREWORD (ADDR (DATABUF_BYTE))     = ADDRESS;
 COREWORD (ADDR (DATABUF_HALFWORD)) = ADDRESS;
 COREWORD (ADDR (DATABUF_FULLWORD)) = ADDRESS;

 CALL MOVE (8, BASE_NAME, SDFNAM);

/*-----------------------------*/
/* GET THE DIRECTORY ROOT CELL */
/*-----------------------------*/

 CALL MONITOR (22, "80000007");
 SELECTED_UNIT = 0;

 IF CRETURN ^= 0 THEN DO;
    CSECT_TYPE = 0;

    OUTPUT = '*** SDF ' || BASE_NAME || ' WAS NOT FOUND -- CSECT ' ||
             CSECT_NAME || ' WILL BE ASSUMED NONHAL';

    GO TO ADD_CSECT;
  END;

  GOT_BLK  = 0;
  GOT_VAR  = 0;
  TMP      = LENGTH (WORK_STRING);
  BLKNLEN  = TMP;
  CALL MOVE (TMP, WORK_STRING, BLKNAM);

 /*-----------------------------------------------*/
```

```
TS = 'BLOCK NOT FOUND: ' || WORK_STRING;
CALL TEXT_ERROR (TS, 0);
  END;

  GOT_BLK = KEY_BLOCK;
  BLKNO   = KEY_BLOCK;

 /*-----------------------------------------------*/
 /* LOCATE BLOCK DATA CELL GIVEN THE BLOCK NUMBER */
 /*-----------------------------------------------*/

  CALL MONITOR (22, 8);

/*-------------------------------------------------------*/
/* SAVE KEY SDFPKG PARAMETERS FOR USE BY PRINTSUMMARY */
/*-------------------------------------------------------*/

 SDFPKG_LOCATES  = LOCCNT;
 SDFPKG_READS    = READS;
 SDFPKG_SLECTCNT = SLECTCNT;
 SDFPKG_FCBAREA  = TOTFCBLN;
 SDFPKG_PGAREA   = NUMOFPGS;
 SDFPKG_NUMGETM  = NUMGETM;

/*-------------------------------------------------------------*/
/* ALLOW SDFPKG TO TERMINATE ITSELF.  THEN IT WILL BE DELETED */
/*-------------------------------------------------------------*/

  CALL MONITOR (22, 1);
```

5.0 Calling SDFPKG From a PL/I Program via an Assembler Interface

The following Assembly Language Interface Program (Section 5.2) is used to invoke SDFPKG from a PL/I Program. A PL/I example that illustrates SDFPKG Initialize and Terminate mode calls is included immediately following the Assembler Interface Program (Section 5.3).

5.1 SDFPKG Assembler Interface Arguments

The PL/I Parameters passed to the SDFPKG Assembler Interface Program are somewhat different than either XPL or Assembler. The parameters are in the following format and are described below:

```
CALL WMHSDFI (MODE, OPTIONS, COMM_TABLE, DEBUG, TITLE)
```

MODE - This parameter is the same 0-17 number used by both other SDFPKG interfaces; however, it does not contain the Disposition (DISP) options that are placed in the upper 4 bits (Nibble) as with the other two interfaces.

OPTIONS - This parameter contains the SDFPKG Disposition options that are specified in the upper 4 bits of the mode parameter for the normal Assembler/SDFPKG (Section 3.0) and the XPL/SDFPKG (Section 4.0) interfaces. In this parameter, however, the options are passed in the low order 4 bits (Nibble) of the fullword.

COMM_TABLE - This parameter serves the same purpose as COMMTABL in the Assembler/SDFPKG (Section 3.0) and the XPL/SDFPKG (Section 4.0) interfaces. In an effort to make the interface more "user-friendly," the table was modified to take advantage of PL/I variable length character strings; thereby automating the process of setting the length for the Block name length and Symbol name length fields. In addition, the interface call to WMHSDFI was changed to pass the COMM_TABLE parameter each time in order to make it more consistent across the different SDFPKG modes.

DEBUG_FLAG - This parameter does not exist in any other interface; its purpose is to "Snap" (Dump) data areas from the WMHSDFI interface so the user can verify the correct data is being imported to his/her program.

TITLE - This parameter contains a title that will be printed at the beginning of the debug data produced by the WMHSDFI

5.2 PL/I SDFPKG Assembler Interface

```
*************************************************************************
*                                                                       *
*  MEMBER NAME: WAHSDFI                                                 *
*  CSECT NAME:  WAHSDFI                                                 *
*  TITLE:       PL/I PROCESSOR -- SDFPKG INTERFACE PROGRAM               *
*  FUNC. AREA:  HALSTAT                                                 *
*  PROGRAMMER:  PETER KOESTER, JEFF DAY, AND GERALD CETRONE, III        *
*  OVERVIEW:    MODULE WHICH IS CALLED BY A PL/I PROGRAM TO             *
*               INITIALIZE, EXECUTE, AND TERMINATE THE SDFPKG           *
*  ENTRY POINT: WMHSDFI                                                 *
*  INVOCATION:  CALL WMHSDFI (PLIMODE, PLIOPTS, PLICOMM,                *
*                             DEBUGFLAG, DEBUGTITLE)                    *
*  RETURN CODE: AT RETURN, REG 15 WILL CONTAIN THE SAME RETURN CODE     *
*               SET BY THE SDFPKG                                       *
*                                                                       *
*  HISTORY==> INITIAL DEVELOPMENT:                                      *
*               DEVELOPED IN 3/91 AS A PART OF AN EFFORT TO PROTO-      *
*               TYPE HALSTAT OPTIONS BY PETER KOESTER.  AT THE TIME     *
*               OF DEVELOPMENT, THERE WERE NO PLANS FOR THIS MODULE     *
*               TO BE PLACED IN PRODUCTION USE.                         *
*                                                                       *
*             SUBSEQUENT CHANGES:                                       *
*                                                                       *
*                                                                       *
*  COMMENTS:                                                            *
*                                                                       *
*    REGISTER ALLOCATIONS:                                              *
*                                                                       *
*  REGISTER          PURPOSE                                            *
*  --------    -----------------------------------                      *
*    $0     -- DSA LENGTH (SETUP)                                       *
*           -- ADDR ASSEMBLER COMMUNICATION TABLE                       *
*                                                                       *
*    $1     -- FIRST BYTE OF DSA (SETUP)                                *
*           -- POINTER TO PL/I PARAMETER ADDRESSES                      *
*           -- COMBINED SDFPKG MODE # AND OPTIONS                       *
*                                                                       *
*    $2     -- FREE                                                     *
*                                                                       *
*    $3     -- FREE                                                     *
*                                                                       *
*    $4     -- POINTER TO PL/I PARAMETER ADDRESSES                      *
*           -- SCRATCH REGISTER                                         *
*                                                                       *
*    $5     -- ADDRESS OF PL/I MODE # PARAMETER                         *
```

```
*   $10     -- BASE REGISTER 1                                          *
*                                                                       *
*   $11     -- BASE REGISTER 2                                          *
*                                                                       *
*   $12     -- USED IN DSA SETUP                                        *
*           -- SCRATCH REGISTER                                         *
*                                                                       *
*   $13     -- ADDRESS OF NEXT DSA AREA                                 *
*                                                                       *
*   $14     -- BALR RETURN ADDRESS                                      *
*                                                                       *
*   $15     -- BALR BRANCH ADDRESS                                      *
*           -- MODULE RETURN CODE                                       *
*                                                                       *
************************************************************************
          EJECT
************************************************************************
*                                                                       *
*  MACRO TO ESTABLISH A PL/I NAME IN AN APPROPRIATE FORMAT              *
*                                                                       *
************************************************************************
          MACRO
          PLINAME &ID
          LCLA  &IFODD,&STRLEN
          DS    0H
&STRLEN   SETA  K'&ID
&IFODD    SETA  &STRLEN-&STRLEN/2*2
          AIF   (&IFODD EQ 1).NODD
          DC    C' '
          DC    C'&ID'
          DC    AL1(&STRLEN)
          MEND
          EJECT
************************************************************************
*                                                                       *
*  GENERAL NON-EXECUTABLE SETUP OPERATIONS:                             *
*     1)  ESTABLISH CSECT NAME                                          *
*     2)  SPECIFY PL/I NAME                                             *
*     3)  ESTABLISH ENTRY POINT                                         *
*     4)  GENERATE REGISTER EQUATE STATEMENTS                           *
*                                                                       *
************************************************************************
WAHSDFI   CSECT
          PLINAME SDF_PACKAGE_CALLER
          ENTRY WMHSDFI
          EQUATE
          EJECT
```

```
WMHSDFI  DS    0H
         STM   $14,$12,12($13)                  STORE CALLER'S REGISTERS
         BALR  $10,0                            ESTABLISH BASE REGISTER
         USING *,$10,$11
         LR    $11,$10
         A     $11,=F'4096'
         LR    $4,$1                            SAVE POINTER TO INPUT
*                                               * PARAMETERS
*********************************************************************
*                                                                   *
*  ALLOCATE AND INITIALIZE A SAVE AREA FROM THE PL/I LIFO STACK SO   *
*  THAT IT IS IDENTIFIABLE AS A PL/I-LIKE DSA                       *
*                                                                   *
*********************************************************************
         LA    $0,MAINEND-MAINSA                LENGTH OF DSA
         L     $1,76(,$13)                      ADDR OF FIRST BYTE
*                                               * BEYOND CURRENT DSA
         ALR   $0,$1                            IF NEW DSA WILL NOT FIT
*                                                  IN STACK
         CL    $0,12(,$12)
         BNH   SDF001                              THEN
         L     $15,116(,$12)                       CALL THE PL/I
*                                                  * STORAGE OVERFLOW
         BALR  $14,$15                             * TO TRY FOR MORE
*                                                  * STACK SPACE
SDF001   EQU   *                                ENDIF
         ST    $0,76(,$1)                       POINT TO NEXT AVAILABLE
*                                               * STACK ADDR
         ST    $13,4(,$1)                       SET DSA CHAIN-BACK ADDR
         MVC   72(4,$1),72($13)                 COPY ADDRESS OF LIBRARY
*                                               * WORKSPACE
         LR    $13,$1                           ESTABLISH BASE REG OF
*                                               * NEW DSA
         USING MAINDSA,$13
         MVI   MAINDSA,X'80'                    SET FLAGS IN NEW DSA TO
*                                               * PRESERVE
         MVI   MAINDSA+1,X'00'                  * PL/I ERROR HANDLING IN
         MVI   MAINDSA+86,X'91'                 * THIS ASSEMBLER ROUTINE
         MVI   MAINDSA+87,X'C0'
         EJECT
*********************************************************************
*                                                                   *
*  LOAD ADDRESSES OF INPUT PARMS                                    *
*  IF THE MODE_# (INPUT PARM #1) IS ZERO, THEN                      *
*     LOAD THE SDFPKG PROGRAM                                       *
*  ENDIF                                                            *
*                                                                   *
```

```
*
         IF    F,($8),EQ,=F'1',THEN              IF DEBUG FLAG THEN
         SNAPL ID='MODE #'
         ENDIF                                   ENDIF
*
         IF    F,($1),IS,ZERO,THEN               IF MODE 0 THEN
         LOAD  EP=SDFPKG
         ST    $0,SDFPKG                            SAVE ADDRESS OF
*                                                   * SDFPKG ENTRY POINT
         SLR   $1,$1                                RESET MODE_# = 0
         XC    MODCALLS(72),MODCALLS                ZERO MODCALLS TABLE
         ENDIF                                   ENDIF
*********************************************************************
*                                                                   *
*  COUNT NUMBER OF EACH MODE CALL                                   *
*                                                                   *
*********************************************************************
         LR    $2,$1                             GET MODE CALL
         SLL   $2,2                              GET OFFSET INTO MODCALLS
*                                                * TABLE
         L     $3,MODCALLS($2)                   GET #CALLS FOR MODE
         A     $3,=F'1'                          ADD 1 MORE CALL TO ENTRY
         ST    $3,MODCALLS($2)                   PUT # CALLS BACK IN TABLE
*********************************************************************
*                                                                   *
*  MOVE DATA FROM PL/I VERSION OF THE COMMUNICATION TABLE TO THE    *
*  TRUE SDFPKG COMMUNICATION TABLE                                  *
*                                                                   *
*********************************************************************
         USING PCOMMTAB,$7
*
         IF    F,($8),EQ,=F'1',THEN              IF DEBUG FLAG THEN
         SNAPL ID='PCOMTAB1',START=($7),LENGTH=132
*
         IF F,($1),IS,ZERO,THEN                     IF MODE 0 THEN
         L     $2,PADDR
         SNAPL ID='PAD AREA',START=($2),LENGTH=4000
         ELSE                                       ELSE
         L     $2,DBUFADDR
         SNAPL ID='DATABUFF1',START=($2),LENGTH=124
         ENDIF                                      ENDIF
         ENDIF                                   ENDIF
*
         IF    F,PAPGAREA,EQ,=X'FF000000',THEN   IF PL/I NULL PTR, THEN
         MVC   APGAREA(4),=F'0'                     ZERO FIELD
         ELSE                                    ELSE
         MVC   APGAREA,PAPGAREA                     MOVE DATA DIRECTLY
```

```
         LA    $0,COMMTABL                        COMM TABLE ADDR
*
         IF    F,($1),IS,ZERO,THEN                IF MODE_# IS 0, THEN
         IF T,MISC+1,X'04',OFF,THEN                  IF DEFAULT SDF NAME
*                                                       TO BE USED THEN
         MVC   SDFDDNAM(8),HALSDF                    MOVE DEFAULT DD
*                                                    * NAME
         MVC   PDDNAM(8),HALSDF                      MOVE DEFAULT DD
*                                                    * NAME
         ELSE                                     ELSE
*                                                    ALTERNATE DDNAME
*                                                    * IS TO BE USED
         MVC   SDFDDNAM(8),PDDNAM                    MOVE ALT. DD NAME
         OI    MISC+1,X'04'                          * AND TELL SDFPKG
*                                                    * TO USE IT
         ENDIF                                    ENDIF
*
         ELSE                                     ELSE
         MVC   SDFNAM,PSDFNAM                        MOVE SDF NAME
         ENDIF                                    ENDIF
         MVC   BLKNAM,PBLKNAM                     MOVE VARY-LEN FIELDS
         MVC   SYMBNAM,PSYMBNAM                          AND
         MVC   BLKNLEN(1),PBLKNLEN+1              THEIR LENGTHS
         MVC   SYMBNLEN(1),PSYMNLEN+1
*********************************************************************
*                                                                   *
*  ESTABLISH INPUT REGISTER CONTENTS AND CALL SDFPKG PROGRAM        *
*                                                                   *
*********************************************************************
         L     $15,SDFPKG                         ADDRESS OF SDFPKG
         SLR   $3,$3                              GET THE SDFPKG OPTION
         ICM   $3,B'1000',3($6)                   * BITS AND 'OR' THEM
         SLL   $3,4                               * TO REG 1 -- THE
         OR    $1,$3                              * MODE IS IN REG 1
*
         IF    F,($8),EQ,=F'1',THEN               IF DEBUG FLAG THEN
         SNAPL ID='ACOMTAB1',START=COMMTABL,LENGTH=120
         ENDIF                                    ENDIF
*
         BALR  $14,$15                            CALL SDFPKG
         ST    $15,RETCODE                        SAVE RETURN CODE
*
         IF    F,($1),IS,ZERO,THEN                IF MODE 0 THEN
         MVC   DBUFADDR,ADDR
         ENDIF                                    ENDIF
*
         IF    F,($8),EQ,=F'1',THEN               IF DEBUG FLAG THEN
```

```
*********************************************************************
         IF    F,APGAREA,EQ,=X'00000000',THEN    IF ADDR IS 0, THEN
         MVC   PAPGAREA,=X'FF000000'                MOVE NULL POINTER
         ELSE                                    ELSE
         MVC   PAPGAREA,APGAREA                     MOVE DATA DIRECTLY
         ENDIF                                   ENDIF
         IF    F,AFCBAREA,EQ,=X'00000000',THEN   IF ADDR IS 0, THEN
         MVC   PFCBAREA,=X'FF000000'                MOVE NULL POINTER
         ELSE                                    ELSE
         MVC   PFCBAREA,AFCBAREA                    MOVE DATA DIRECTLY
         ENDIF                                   ENDIF
         MVC   PMOVE1(MOVE1END-MOVE1),MOVE1      BLOCK MOVES
         MVC   PMOVE2(MOVE2END-MOVE2),MOVE2
         MVC   PMOVE3(MOVE3END-MOVE3),MOVE3
         MVC   PBLKNAM,BLKNAM                    MOVE VARY-LEN DATA
         MVC   PSYMBNAM,SYMBNAM                         AND
         MVC   PBLKNLEN,=H'0'                    FIELD LENGTHS
         MVC   PBLKNLEN+1(1),BLKNLEN
         MVC   PSYMNLEN,=H'0'
         MVC   PSYMNLEN+1(1),SYMBNLEN
*
         IF    F,($8),EQ,=F'1',THEN              IF DEBUG FLAG THEN
         SNAPL ID='PCOMTAB2',START=($7),LENGTH=132
         L     $2,DBUFADDR
         SNAPL ID='DATABUFF2',START=($2),LENGTH=124
         L     $2,ADDR
         S     $2,=F'32'
         SNAPL ID='32 BEFORE',START=($2),LENGTH=32
         L     $2,ADDR
         SNAPL ID='DATA @ ADDR',START=($2),LENGTH=1680
         ENDIF                                   ENDIF
*
         EJECT
*********************************************************************
*                                                                   *
*  IF THE MODE_# (INPUT PARM #1) IS ONE, THEN                       *
*     DELETE THE SDFPKG PROGRAM                                     *
*  ENDIF                                                            *
*                                                                   *
*********************************************************************
         IF    F,0(,$5),EQ,=F'1',THEN            IF MODE 1 THEN
         DELETE EP=SDFPKG
         L     $15,RETCODE                          RESTORE RETURN CODE
         ENDIF                                   ENDIF
*********************************************************************
*                                                                   *
*  PROGRAM TERMINATION                                              *
```

```
*  DATA DECLARATIONS                                                  *
*                                                                     *
***********************************************************************
         SPACE 2
SDFPKG   DS    A                   ADDRESS OF SDFPKG ENTRY POINT
         SPACE 2
DBUFADDR DS    A                   DATA BUFF ADDRESS
         SPACE 2
HALSDF   DC    CL8'HALSDF  '       DEFAULT SDF DD NAME
         SPACE 2
MODCALLS DS    50F                 TABLE SHOWING # TIMES A MODE WAS
*                                  USED
         SPACE 2
RETCODE  DS    F                   RETURN CODE FROM BALR TO SDFPKG
         SPACE 2
WORKTITL DS    CL133               TITLE WORKING AREA
         EJECT
***********************************************************************
*                                                                     *
*  DATA DECLARATIONS FOR ASSEMBLER SDFPKG COMMUNICATIONS TABLE        *
*                                                                     *
***********************************************************************
COMMTABL DS    0D                 SDFPKG COMMUNICATION TABLE
APGAREA  DS    A                  * PAGING AREA ADDRESS POINTER
AFCBAREA DS    A                  * FILE CNTL BLK AREA ADDRESS POINTER
MOVE1    EQU   *                  * START OF BLOCK MOVE 1
NPAGES   DS    H                    * NO. OF PAGES IN PAGING AREA
NBYTES   DS    H                    * NO. OF BYTES IN FCB AREA
MISC     DS    H                    * MISCELLANEOUS PURPOSE FIELD
CRETURN  DS    H                    * SDFPKG RETURN CODE
BLKNO    DS    H                    * BLOCK NUMBER
SYMBNO   DS    H                    * SYMBOL NUMBER
STMTNO   DS    H                    * STATEMENT NUMBER
MOVE1END EQU   *                  * END OF BLOCK MOVE 1
BLKNLEN  DS    CL1                * NO. OF CHAR IN BLOCK NAME (BLKNAM)
SYMBNLEN DS    CL1                * NO. OF CHAR IN SYMB NAME (SYMBNAM)
MOVE2    EQU   *                  * START OF BLOCK MOVE 2
PNTR     DS    F                    * SDF POINTER TO DATA
ADDR     DS    A                    * MEMORY ADDRESS OF DATA IN PG. AREA
MOVE2END EQU   *                  * END OF BLOCK MOVE 2
SDFDDNAM EQU   *                  * ALTERNATE DD NAME FOR SDF DATASET
SDFNAM   DS    CL8                * NAME OF SDF TO BE SELECTED
MOVE3    EQU   *                  * START OF BLOCK MOVE 3
CSECTNAM DS    CL8                  * NAME OF CODE CSECT FOR BLOCK
SREFNO   DS    CL6                  * STATEMENT REFERENCE NUMBER (SRN)
INCLONT  DS    H                    * INCLUDE COUNT (FOR SRN)
MOVE3END EQU   *                  * END OF BLOCK MOVE 3
```

```
*    2)  PL/I VERSION OF SDFPKG COMMUNICATION TABLE                  *
*    3)  PL/I DEBUG TITLE                                            *
*                                                                    *
**********************************************************************
MAINDSA  DSECT                  SAVE AREA IN PL/I DSA FORMAT
MAINSA   DS    22F
MAINEND  DS    0D
PCOMMTAB DSECT                  PL/I VERSION OF SDFPKG COMMUN. TABLE
PAPGAREA DS    A                * PAGING AREA ADDRESS POINTER
PFCBAREA DS    A                * FILE CNTL BLK AREA ADDRESS POINTER
PMOVE1   DS    0H
PNPAGES  DS    H                * NO. OF PAGES IN PAGING AREA
PNBYTES  DS    H                * NO. OF BYTES IN FCB AREA
PMISC    DS    H                * MISCELLANEOUS PURPOSE FIELD
PCRETURN DS    H                * SDFPKG RETURN CODE
PBLKNO   DS    H                * BLOCK NUMBER
PSYMBNO  DS    H                * SYMBOL NUMBER
PSTMTNO  DS    H                * STATEMENT NUMBER
PPAD     DS    H           ********** ALIGNMENT PAD ************
PMOVE2   DS    0H
PPNTR    DS    F                * SDF POINTER TO DATA
PADDR    DS    A                * MEMORY ADDRESS OF DATA IN PG. AREA
PDDNAM   DS    CL8              * ALTERNATE DD NAME FOR SDF DATASET
PSDFNAM  DS    CL8              * NAME OF SDF TO BE SELECTED
PMOVE3   DS    0H
PCSECTNM DS    CL8              * NAME OF CODE CSECT FOR BLOCK
PSREFNO  DS    CL6              * STATEMENT REFERENCE NUMBER (SRN)
PINCLONT DS    H                * INCLUDE COUNT (FOR SRN)
PBLKNLEN DS    H                * NO. OF CHAR IN BLOCK NAME (BLKNAM)
PBLKNAM  DS    CL32             * BLOCK NAME
PSYMNLEN DS    H                * NO. OF CHAR IN SYMB NAME (SYMBNAM)
PSYMBNAM DS    CL32             * SYMBOL NAME
PCOMMEND DS    0F               END OF PL/I VERSION OF COMMUN. TABLE
*
DTITLE   DSECT                  PL/I DEBUG TITLE
TITLELEN DS    H                * TITLE LENGTH
TITLE    DS    CL133            * PL/I TITLEBLK AREA ADDRESS POINTER
DTITLEND DS    0F               END OF PL/I TITLE AREA
*
WAHSDFI   CSECT
         END   WMHSDFI
```

5.3 PL/I Procedures Which Call the SDFPKG Assembler Interface

```
/*-----------------------------------------------------------------*/
/*    FUNCTIONAL TITLE OF CSECT:  INCLUDE MEMBER CONTAINING SDFPKG- */
/*                                PL/I INTERFACE ROUTINES          */
/*                                                                 */
/*    LANGUAGE: PL/I                                               */
/*                                                                 */
/*    PROGRAMMER: JEFF DAY, AND GERALD CETRONE III                 */
/*                                                                 */
/*    COMMENTS:                                                    */
/*                                                                 */
/*    THE FOLLOWING DECLARES ARE MAPPINGS OF DATA AREAS WITHIN SDFS */
/*    THAT ARE RETURNED BY SDFPKG.POINTER (SDFADDR).  FOR FURTHER  */
/*    INFORMATION CONCERNING THESE DATA AREAS AND THEIR MEANING    */
/*    SEE :                                                        */
/*                                                                 */
/*      "SPACE SHUTTLE ORBITER AVIONICS SOFTWARE"                  */
/*      "HAL/SDL INTERFACE CONTROL DOCUMENT"                       */
/*      "SDFPKG MEMO"                                              */
/*-----------------------------------------------------------------*/

%PAGE;
/*---------------------------------------------------------*/
/*  DECLARE MISCELLANEOUS VARIABLES FOR EXAMPLE PROGRAM  */
/*---------------------------------------------------------*/
DECLARE
   TOGGLE_BIT_STR(100)  BIT,              /* TRACE FLAG BITS    */

%PAGE;
/*-----------------------------------------------------------------*/
/*  DECLARE COVER VARIABLES FOR SDFPKG MONITOR CALLS               */
/*-----------------------------------------------------------------*/

DECLARE
  SDF_INIT                    FIXED BIN(31)     /*SDFPKG INITIALIZE*/
     INIT (0),                                  /* MODE            */
  SDF_TERM                    FIXED BIN(31)     /*SDFPKG TERMINATE */
     INIT (1),                                  /* MODE            */
  SDF_AUG_PG                  FIXED BIN(31)     /*SDFPKG AUGMENT   */
     INIT (2),                                  /* PAGING AREA     */
  SDF_RESC_AUG                FIXED BIN(31)     /*SDFPKG RESCIND   */
     INIT (3),                                  /* ALL AUGMENTS    */
  SDF_SEL_EXPL                FIXED BIN(31)     /*SDFPKG SELECT SDF*/
     INIT (4),                                  /* EXPLICITLY      */
  SDF_LOC_PTR                 FIXED BIN(31)     /*SDFPKG LOCATE SDF*/
     INIT (5),                                  /* POINTER         */
```

```
     INIT (10),                                /* USING STMT NO.  */
  SDF_BLK_NAME                FIXED BIN(31)    /*GET BLOCK DATA   */
     INIT (11),                                /* USING BLOCK NAME*/
  SDF_SYMB_BLK_SYMB           FIXED BIN(31)    /*GET SYMBOL DATA  */
     INIT (12),                                /* USING BLK &SYMB */
  SDF_SYMB_NAME               FIXED BIN(31)    /*GET SYMBOL DATA  */
     INIT (13),                                /* USING SYMB NAME */
  SDF_STMT_SRN                FIXED BIN(31)    /*GET STATEMNT DATA*/
     INIT (14),                                /* USING SRN       */
  SDF_BINDEX_BLK#             FIXED BIN(31)    /*GET BLOCK INDEX  */
     INIT (15),                                /* TABLE DATA      */
  SDF_SYINDEX_SYMB#            FIXED BIN(31)   /*GET SYMBOL TABLE */
     INIT (16),                                /* DATA            */
  SDF_STINDEX_STMT#            FIXED BIN(31)   /*GET STATEMENT    */
     INIT (17);                                /* INDEX TABLE DATA*/

%PAGE;
/*-------------------------------------------------------------------*/
/* DECLARE COVER VARIABLES FOR SDFPKG MODE OPTIONS                   */
/*-------------------------------------------------------------------*/

DECLARE
  SDF_AUTO_SELECT             FIXED BIN(31)     /*SDFPKG PERFORMS  */
     INIT (8),                                 /* AUTO-SELECT     */
  SDF_MODIFY                  FIXED BIN(31)     /*WRITE MODIFIED   */
     INIT (4),                                 /* SDF PAGE TO SDF */
  SDF_RELEASE                 FIXED BIN(31)     /*RELEASE RESERVED */
     INIT (2),                                 /* SDF PAGE        */
  SDF_RESERVE                 FIXED BIN(31)     /*RESERVE AN SDF   */
     INIT (1),                                 /* PAGE            */
  SDF_NO_OPT                  FIXED BIN(31)     /*NO SDFPKG OPTION */
     INIT (0);                                 /* SPECIFIED       */

%PAGE;
/*-------------------------------------------------------------------*/
/* DECLARE COVER VARIABLES SDFPKG INITIALIZATION OPTIONS.            */
/* (MISC VALUES)                                                     */
/*-------------------------------------------------------------------*/

DECLARE
  SDF_AUTO_FCB                FIXED BIN(31)     /*SDFPKG PERFORMS  */
     INIT (1),                                 /* GETMAIN FOR FCB */
  SDF_UPDATE_MODE             FIXED BIN(31)     /*UPDATE MODE IS   */
     INIT (2),                                 /* ALLOWED FOR SDF */
  SDF_ALT_DD                  FIXED BIN(31)     /*USE DD NAME OTHER*/
     INIT (4),                                 /* THAN HALSDF     */
  SDF_ONE_FCB                 FIXED BIN(31)     /*USE ONLY 1 FCB   */
```

```
DECLARE
                                   /*OPTIONS IN EFFECT ARE A     */
                                   /* SUMMATION OF THE INITIAL-  */
                                   /* IZATION OPTIONS DECLARED   */
                                   /* ABOVE.                     */
   SDF_MISC_VAL              FIXED BIN(31);      /*SDFPKG INIT OPTS*/

%PAGE;
/*-------------------------------------------------------------------*/
/* DECLARE SDFPKG PAGING AREA IN PL/I                                */
/*-------------------------------------------------------------------*/

DECLARE
   SDF_DBUFF_AREA_PTR          POINTER,           /*DATA BUFFER PTR */
   SDF_NO_PAGES                FIXED BIN(31)      /*NO SDFPKG PAGING*/
      INITIAL (250),                              /* AREAS          */
   SDF_ALT_PAD_SIZE            FIXED BIN(31)      /*ADDED PAGING    */
      INITIAL (4000),                             /* AREA DIR. SIZE */
   SDF_PAGING_AREA (250, 420)  FIXED BIN(31),     /*SDF PAGING AREA */
   SDF_FCB_AREA (2000, 32)     FIXED BIN(31),     /*FILE CNTL BLK   */
   SDF_FCB_SIZE                FIXED BIN(15)      /*FILE CNTL BLK   */
       INITIAL (-2000),                           /* AREA SIZE      */
   SDF_PAD_AREA (250, 4)       FIXED BIN(31);     /*PAGING DIRECTORY*/

%PAGE;
/*-------------------------------------------------------------------*/
/* WMHSDFI IS AN ASSEMBLY LANGUAGE INTERFACE ROUTINE BETWEEN A       */
/* PL1 PROCESSOR AND SDFPKG.  IT MUST BE LINK-EDITED INTO THE        */
/* CALLING PL/I MODULE.                                              */
/*-------------------------------------------------------------------*/

DECLARE
   WMHSDFI                     ENTRY OPTIONS (ASSEMBLER, INTER);

%PAGE;
/*-------------------------------------------------------------------*/
/* THE FOLLOWING IS A MAPPING OF THE SDFPKG/USER COMMUNICATION       */
/* AREA. ITS ADDRESS IS RETURNED IN SDFCOMAREA AS A RESULT OF THE    */
/* MODE 0 CALL TO SDFPKG                                             */
/*-------------------------------------------------------------------*/

DECLARE
   1 SDF_COMM_TABLE,
    2 ADDR_PAGING_AREA         POINTER,           /*ADDR PAGING AREA */
    2 ADDR_FCB_AREA            POINTER,           /*ADDR FCB AREA    */
    2 #PAGES                   FIXED BIN(15),     /*NO PAGES IN AREA */
```

```
    2 ADDRESS                 POINTER,           /*ADDR RETURNED    */
                                                 /* CELL            */
    2 DD_NAME                 CHAR(8),           /*NAME SDF DATASET */
                                                 /* DD CARD         */
    2 SDF_NAME                CHAR(8),           /*NAME SDF TO      */
                                                 /* SELECT          */
    2 CSECT_NAME              CHAR(8),           /*NAME CODE CSECT  */
    2 STMT_REF_NO             CHAR(6),           /*STATEMENT REF NO */
    2 INCLUDE_COUNT           FIXED BIN(15),     /*INCLUDE COUNT    */
                                                 /* (SRN)           */
    2 BLOCK_NAME              CHAR(32) VARYING,  /*BLOCK NAME       */
    2 SYMBOL_NAME             CHAR(32) VARYING;  /*SYMBOL NAME      */

%PAGE;
DECLARE
   TEMP_ADDR   FIXED BIN(31);
/*-------------------------------------------------------------------*/
/* DECLARES FOR SDFPKG INTERNAL DATA BUFFER (DATABUF)                */
/*-------------------------------------------------------------------*/

DECLARE
  1 SDF_DATABUFF              BASED(SDF_DBUFF_AREA_PTR),
    2 #LOCATES                FIXED BIN(31),     /*CURRENT NO.      */
                                                 /* SDFPKG LOCATES  */
    2 ADDR_VULN_ENTRY         POINTER,           /*ADDR VULNERABLE  */
                                                 /* PAD ENTRY       */
    2 ADDR_CURRENT_FCB        POINTER,           /*ADDR CURRENT FCB */
    2 ADDR_PAD                POINTER,           /*ADDR PAGING AREA */
                                                 /* DIRECTORY       */
    2 ADDR_COMMON_TABLE       POINTER,           /*ADDR SDFCOMTAB   */
    2 ADDR_CURRENT_ENTRY      POINTER,           /*ADDR CURRENT PAD */
                                                 /* ENTRY           */
    2 ADDR_ROOT_FCB           POINTER,           /*ADDR ROOT FCB OF */
                                                 /* FCB TREE        */
    2 SYM_NODE_EXTENT_PTR     FIXED BIN(31),     /*PTR TO SYMB NODE */
                                                 /* EXTENT CELL     */
    2 FIRST_SYM_IN_BLK        FIXED BIN(15),     /*FIRST SYM OF BLK */
    2 LAST_SYM_IN_BLK         FIXED BIN(15),     /*LAST SYM OF BLK  */
    2 #GETMAINS               FIXED BIN(15),     /*NO. SDFPKG       */
                                                 /* GETMAINS        */
    2 #PAGES                  FIXED BIN(15),     /*NO. SDFPKG PAGES */
    2 INITIAL_#PAGES          FIXED BIN(15),     /*INITIAL NO. PGS  */
    2 #FCB_STACK_ENTRIES      FIXED BIN(15),     /*NO. ENTRIES IN   */
                                                 /* FCB STACKS      */
    2 IO_FLAG                 BIT(8),            /*I/O IN PROGRESS  */
                                                 /* FLAG            */
    2 GETM_FLAG               BIT(8),            /*>0 -- DO AUTO    */
```

```
    2 UNUSED                  FIXED BIN(15),     /*SPARE FIELD      */
    2 TOTAL_FCB_LEN           FIXED BIN(31),     /*TOTAL FCB SPACE  */
                                                 /* IN USE          */
    2 #RESERVES               FIXED BIN(31),     /*GLOBAL COUNT OF  */
                                                 /* RESERVES        */
    2 #READS                  FIXED BIN(31),     /*TOTAL NO. READS  */
    2 #WRITES                 FIXED BIN(31),     /*TOTAL NO. WRITES */
    2 #SELECTS                FIXED BIN(31),     /*TOTAL NO. SELECT */
    2 #FCBS                   FIXED BIN(31),     /*TOTAL NO. FCBS   */
    2 GET_MAIN_STK_ADDR       POINTER,           /*ADDR GETMAIN ADDR*/
                                                 /* STACK           */
    2 GET_MAIN_STK_LEN        POINTER,           /*ADDR GETMAIN     */
                                                 /* LENGTH STACK    */
    2 FCB_STK_ADDR            POINTER,           /*ADDR FCB AREA    */
                                                 /* ADDRESS STACK   */
    2 FCB_STK_LEN             POINTER,           /*ADDR FCB AREA    */
                                                 /* LENGTH STACK    */
    2 MAX_STK_ENTRIES         FIXED BIN(15),     /*MAX NO. STACK    */
                                                 /* ENTRIES         */
    2 VERSION                 FIXED BIN(15),     /*SDF VERSION NO.  */
    2 PAGE_BUFF_ADDR          POINTER,           /*ADDR PAGE BUFFER */
    2 DECB_ADDR               POINTER,           /*ADDR DECB        */
    2 ECB                     FIXED BIN(31),     /*EVENT CONTROL BLK*/
    2 IO_TYPE                 FIXED BIN(15),     /*I/O TYPE (DECB)  */
    2 IO_LENGTH               FIXED BIN(15),     /*NO. BYTES TO     */
                                                 /* TRANSFER (DECB) */
    2 DCB_ADDR                POINTER,           /*ADDR SDF DCB     */
    2 BUFF_ADDR               POINTER,           /*ADDR BUFFER AREA */
    2 IO_BLK_ADDR             POINTER;           /*ADDR I/O BLOCK   */

%PAGE;
/*-----------------------------------------------------------------*/
/* DECLARE RECORD FOR SDFPKG FINAL STATISTICS                      */
/*-----------------------------------------------------------------*/

DECLARE
  1 SDF_TOTALS,
    2 #LOCATES                FIXED BIN(31),     /*TOTAL NO. SDFPKG */
                                                 /* PAGES LOCATED   */
    2 #READS                  FIXED BIN(31),     /*TOTAL NO. SDFPKG */
                                                 /* RECORDS READ    */
    2 #WRITES                 FIXED BIN(31),     /*TOTAL NO. SDFPKG */
                                                 /* RECORDS WRITTEN */
    2 #SELECTS                FIXED BIN(31),     /*TOTAL NO MEMBERS */
                                                 /* SELECTED        */
    2 #RESERVES               FIXED BIN(31),     /*TOTAL COUNT OF   */
                                                 /* RESERVES        */
```

```
   (TEXT) RETURNS (CHAR(133) VARYING);

   DECLARE
      TEXT     CHAR(*)   VARYING,
      NTEXT    CHAR(133) VARYING,
      I        FIXED BIN(15),
      J        FIXED BIN(15);

   NTEXT = '';
   J = LENGTH (TEXT);
   DO I = 1 TO J;
      IF SUBSTR (TEXT, I, 1) ^= ' ' THEN DO;
         NTEXT = NTEXT || SUBSTR (TEXT, I, 1);
      END;
   END;

   RETURN (NTEXT);

END NO_BLANKS;

%PAGE;
/*-------------------------------------------------------------------*/
/* PROCEDURE THAT TURNS FIXED BINARY(31) INTEGERS INTO CHARACTER    */
/* NUMBERS WITHOUT LEADING OR TRAILING BLANKS.                       */
/*-------------------------------------------------------------------*/

FMT31: PROCEDURE
   (DATUM) RETURNS (CHAR(24) VARYING);

    DECLARE
       DATUM         FIXED BIN(31),
       RESULT        CHAR(24) VARYING;

    RESULT = DATUM;

    RETURN (NO_BLANKS (RESULT));

END FMT31;

%PAGE;
/*-------------------------------------------------------------------*/
/* DECODE SDFPKG RETURN CODES AND OUTPUT MESSAGE FOR THEM.          */
/*-------------------------------------------------------------------*/

SDFPKG_RETURN_CODE: PROCEDURE
   (RETURN_CODE) RETURNS (CHAR(100) VARYING);
```

```
   WHEN (4) DO;
      TEMP_STR = 'RETURN CODE = 4;  SDFPKG COULD NOT OPEN ' ||
                 'DATASET ATTACHED TO HALSDF DD.';
   END;

   WHEN (8) DO;
      TEMP_STR = 'RETURN CODE = 8;  SDF MEMBER WAS NOT FOUND ' ||
                 'IN PDS AND COULD NOT BE SELECTED.';
   END;

   WHEN (16) DO;
      TEMP_STR = 'RETURN CODE = 16;  SDFPKG COULD NOT FIND ' ||
                 'SPECIFIED BLOCK NAME.';
   END;

   WHEN (20) DO;
      TEMP_STR = 'RETURN CODE = 20;  SDFPKG COULD NOT FIND ' ||
                 'SPECIFIED SYMBOL NAME OR SRN.';
   END;

   WHEN (24) DO;
      TEMP_STR = 'RETURN CODE = 24;  STATEMENT SPECIFIED IS ' ||
                 'NOT AN EXECUTABLE STATEMENT AND SO DOES NOT ' ||
                 'HAVE A Executable Statement Data Cell.';
   END;

   WHEN (28) DO;
      TEMP_STR = 'RETURN CODE = 28;  THE SDF SPECIFIED DOES ' ||
                 'NOT CONTAIN SRNS.';
   END;

   WHEN (32) DO;
      TEMP_STR = 'RETURN CODE = 32;  THE SRNS CONTAINED IN ' ||
                 'THE SDF ARE NOT IN INCREASING ORDER.';
   END;

   WHEN (36) DO;
      TEMP_STR = 'RETURN CODE = 36;  THE SPECIFIED STATEMENT ' ||
                 'IS OUTSIDE THE LEGAL RANGE.';
   END;

   OTHERWISE DO;
      TEMP_STR = 'RETURN CODE = ' || RETURN_CODE ||
                 ';  UNKNOWN SDFPKG RETURN CODE.';
   END;
END;
```

```
/*  AREA                                                    SDFIR.1100 */
/*-------------------------------------------------------------------*/

SDFPKG_INITIALIZE:  PROCEDURE
   (SDF_DD_NAME, SDF_TOTS, SUCCESS_FLAG, ABORT_FLAG);
/*-------------------------------------------------------------------*/
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*    PDS_MEMBERS IS USED TO 1) COUNT THE NUMBER OF MEMBERS IN A PDS*/
/*                           2) RETURN AN 8-BYTE STRING OF EACH     */
/*                              MEMBER NAME                         */
/*    THE CALLER MUST DEFINE A 10 FULL WORD WORK AREA FOR EACH DATA */
/*    SET TO BE PROCESSED BY PDS_MEMBERS.  THE CALLER MUST SET THE  */
/*    DDNAME TO BE USED IN WORDS 3 AND 4 OF THE WORK AREA:           */
/*                                                                   */
/*       E.G.  CALL MVC_STR(ADDR(FW(3)),'DDNAME  ');                 */
/*                                                                   */
/*    ALL 8 CHARACTER OF THE DDNAME MUST BE SET.                     */
/*                                                                   */
/*    ON THE FIRST CALL (TYPE 0 OR 3) PDS_MEMBERS WILL SET THE TOTAL*/
/*    NUMBER OF MEMBERS IN WORD 2 OF THE WORK TABLE.                 */
/*                                                                   */
/*    ON THE FOLLOWING NORMAL TYPE CALLS (TYPE 1 OR 4) PDS_MEMBERS  */
/*    WILL RETURN A CHARACTER STRING OF THE NEXT MEMBER NAME.        */
/*                                                                   */
/*    THE FINAL CALL (TYPE 2 OR 5) IS A CLEAN-UP FUNCTION.           */
/*                                                                   */
/*    IF PDS_MEMBERS IS CALLED USING THE 0-1-2 MODE, IT DOES A       */
/*    GETMAIN AND HOLDS ALL OF THE MEMBER NAMES IN CORE.  WHEN USED */
/*    IN THIS MODE, MORE THAN ONE DATASET CAN BE PROCESSED BY USING */
/*    N-10 WORD WORK AREAS.                                          */
/*                                                                   */
/*    WHEN THE 3-4-5 MODE IS USED, A GETMAIN IS NOT PERFORMED WHICH */
/*    ALLOWS EXECUTION IN A SMALL REGION.  ONLY ONE PDS CAN BE       */
/*    PROCESSED AT A TIME IN THE 3-4-5 MODE.                         */
/*                                                                   */
/*    THE PDS_MEMBERS INTERFACE TABLE IS DESCRIBED IN THE CODE       */
/*    BELOW.                                                         */
/*                                                                   */
/*-------------------------------------------------------------------*/


DECLARE
   SMLWJPM   OPTIONS (ASSEMBLER INTER);
```

```
                                          /* OPERATING IN       */
     2 GETMAIN_ADDR        POINTER,       /*ADDR OF GETMAIN AREA*/
     2 GETMAIN_LENGTH      FIXED BIN(31), /*LEN OF GETMAIN AREA */
     2 CURRENT_MEM_NO      FIXED BIN(31), /*# OF CURRENT MEMBER */
                                          /* BEING PROCESSED    */
     2 RETURN_CODE         FIXED BIN(31), /*ERROR RETURN CODE   */
                                          /* 0=OK; 1=OPEN ERROR */
     2 UNUSED              FIXED BIN(31), /*UNUSED FIELD        */
     2 MEMBER_NAME         CHAR(8);       /*DESIRED MEMBER NAME */

  DECLARE
    TYPE_OF_CALL           FIXED BIN(31),
    MEM_PTR                POINTER;

  PDS_WORK_TABLE.CALL_MODE = TYPE_OF_CALL;

  SELECT (TYPE_OF_CALL);

    /*----------------------------*/
    /* CASE 0:  OPEN PDS DIRECTORY */
    /*----------------------------*/
     WHEN (0) DO;
        PDS_WORK_TABLE.CALL_MODE      = 0;
        PDS_WORK_TABLE.RETURN_CODE    = 0;
        PDS_WORK_TABLE.ASSEM_LOAD_MOD = 'SMLWJPM';
        CALL SMLWJPM (PDS_WORK_TABLE);
/*      PDS_WORK_TABLE.CURRENT_MEM_NO = 0;    */
     END;

    /*----------------------------*/
    /* CASE 1:  READ PDS DIRECTORY */
    /*----------------------------*/
     WHEN (1) DO;
        PDS_WORK_TABLE.CALL_MODE = 1;
        CALL SMLWJPM (PDS_WORK_TABLE);
     END;

    /*----------------------------*/
    /* CASE 2:  CLOSE PDS DIRECTORY */
    /*----------------------------*/
     WHEN (2) DO;
        PDS_WORK_TABLE.CALL_MODE = 2;
        CALL SMLWJPM (PDS_WORK_TABLE);
        PDS_WORK_TABLE.MEMBER_NAME = 'LASTCALL';
     END;

    /*----------------------------*/
```

```
      /*-----------------------------------------*/
      /* CASE 4:  READ PDS DIRECTORY MEMBER NAME */
      /*-----------------------------------------*/
       WHEN (4) DO;
          PDS_WORK_TABLE.CALL_MODE = 4;
          CALL SMLWJPM (PDS_WORK_TABLE);
       END;

      /*-------------------------------*/
      /* CASE 5:  CLOSE PDS DIRECTORY */
      /*-------------------------------*/
       WHEN (5) DO;
          PDS_WORK_TABLE.CALL_MODE = 5;
          CALL SMLWJPM (PDS_WORK_TABLE);
          PDS_WORK_TABLE.MEMBER_NAME = 'LASTCALL';
       END;

   END;

END PDS_MEMBERS;

%PAGE;
/*-------------------------------------------*/
/* PROCEDURE:  SDFPKG_INITIALIZE  (CONTINUED) */
/*-------------------------------------------*/

   DECLARE
     1 PDS_WORK_TABLE,
       2 ASSEM_LOAD_MOD      CHAR(8),         /*LOAD MOD NAME CALLED*/
       2 #MEMBERS            FIXED BIN(31),   /*NO. MEMBERS IN PDS  */
       2 PDS_DD_NAME         CHAR(8),         /*DD NAME OF PDS      */
       2 CALL_MODE           FIXED BIN(31),   /*MODE PDS_MEMBERS IS */
                                              /* OPERATING IN       */
       2 GETMAIN_ADDR        POINTER,         /*ADDR OF GETMAIN AREA*/
       2 GETMAIN_LENGTH      FIXED BIN(31),   /*LEN OF GETMAIN AREA */
       2 CURRENT_MEM_NO      FIXED BIN(31),   /*# OF CURRENT MEMBER */
                                              /* BEING PROCESSED    */
       2 RETURN_CODE         FIXED BIN(31),   /*ERROR RETURN CODE   */
       2 UNUSED              FIXED BIN(31),   /*UNUSED FIELD        */
       2 MEMBER_NAME         CHAR(8);         /*DESIRED MEMBER NAME */

   DECLARE
      SDF_DD_NAME            CHAR(8) VARYING, /*SDFPKG DD NAME      */
      MISC_VAL               FIXED BIN(31),   /*SUCCESS PROCESS FLAG*/
      SDF_TOTS               LIKE             /*SDFPKG STATISTICS   */
         SDF_TOTALS,
```

```
 SUCCESS_FLAG    = TRUE;             /*SUCCESSFUL PROCESSING FLAG*/
 TRACE_FLAG      = FALSE;            /*PROC TRACE FLAG IS OFF    */
 TRACE_FLAG_SDFI = FALSE;            /*SDF INTERFACE TRACE IS OFF*/
 TITLE           = '';               /*SET TITLE TO NULL         */

 IF (TOGGLE_BIT_STR(1) = TRUE) THEN DO;    /*IF TRACE DESIRED THEN*/
    TRACE_FLAG = TRUE;                     /*SET TRACE FLAG ON    */
    TITLE = 'SDFPKG INITIALIZE -- INVOCATION # ' ||
            FMT31 (#INVOCATIONS);          /*SET UP TITLE         */
    #INVOCATIONS = #INVOCATIONS + 1;       /*COUNT # INVOCATIONS  */
    CALL PUT_SYSPRINT_TITLE (TITLE, PAGE_NO);

    IF (TOGGLE_BIT_STR(2) = TRUE) THEN DO; /*IF INTERFACE TRACE   */
       TRACE_FLAG_SDFI = TRUE;             /*SET SDFI TRACE FLAG  */
    END;
 END;

/* INITIALIZE SDF_TOTS VARIABLES TO 0 */
 SDF_TOTS.#LOCATES  = 0;             /*TOTAL SDFPKG PAGES LOCATED*/
 SDF_TOTS.#READS    = 0;             /*TOTAL SDFPKG RECORDS READ */
 SDF_TOTS.#WRITES   = 0;             /*TOTAL SDFPKG RECS WRITTEN */
 SDF_TOTS.#SELECTS  = 0;             /*TOTAL SDFPKG MBRS SELECTED*/
 SDF_TOTS.#RESERVES = 0;             /*TOTAL SDFPKG PAGES RESERVD*/
 SDF_TOTS.#FCBS     = 0;             /*TOTAL SDFPKG FCB AREA     */
 SDF_TOTS.#PAGES    = 0;             /*TOTAL SDFPKG PG AREA SIZE */
 SDF_TOTS.#GETMAINS = 0;             /*TOTAL # OF SDFPKG GETMAINS*/

/* SET SDF PDS FILE DD NAME                                      */
 PDS_WORK_TABLE.PDS_DD_NAME = SDF_DD_NAME;
 SDF_COMM_TABLE.DD_NAME     = SDF_DD_NAME;

/* OPEN HALSDF FILE TO GET # MEMBERS IN PDS                      */
 CALL PDS_MEMBERS (PDS_WORK_TABLE, 0);

/* IF FILE WAS NOT OPENED                                        */
 IF (PDS_WORK_TABLE.RETURN_CODE ^= 0) THEN DO;
    CALL FMT_ERROR ('SDFIR', 1101, SDF_DD_NAME ||
                    ' DATASET COULD NOT BE OPENED.', 4, ABORT_FLAG);
 END;

 ELSE DO;   /* SDF PDS WAS OPENED */

   /*IF NO MEMBERS IN PDS */
    IF (PDS_WORK_TABLE.#MEMBERS = 0) THEN DO;
      /* OUTPUT WARNING MESSAGE ONLY */
       CALL FMT_ERROR ('SDFIR', 1102, 'NO SDF MEMBERS WERE FOUND' ||
                       ' IN THE ' || SDF_DD_NAME || ' DATASET.',
```

```
/*-------------------------------------------------------------*/
/* SDFPKG INITIALIZATION OPTIONS ARE:                          */
/*   1.  AUTOMATIC GETMAIN FOR FILE CONTROL BLOCKS (FCBS)      */
/*   2.  RETURN FIRST MATCHING SYMBOL FOUND IN SEARCH          */
/*   3.  USE ALTERNATE PAGING AREA DIRECTORY SPECIFIED (PAD)   */
/*-------------------------------------------------------------*/

 MISC_VAL = SDF_AUTO_FCB + SDF_FIRST_SYMBOL + SDF_ALT_PAD;
 SDF_COMM_TABLE.MISC = MISC_VAL;

/*-------------------------------------------------------------*/
/* ALLOCATE 512 BYTES FOR SDFPKG FILE CONTROL BLOCK AREA       */
/*-------------------------------------------------------------*/

 SDF_COMM_TABLE.ADDR_FCB_AREA = ADDR (SDF_FCB_AREA);
 SDF_COMM_TABLE.#BYTES_IN_FCB = SDF_FCB_SIZE;

/*-------------------------------------------------------------*/
/* ALLOCATE AN AUXILIARY SDFPKG PAGING AREA DIRECTORY          */
/*-------------------------------------------------------------*/

 SDF_COMM_TABLE.ADDRESS = ADDR (SDF_PAD_AREA);
 SDF_COMM_TABLE.PNTR    = SDF_ALT_PAD_SIZE;  /*PG DIR SIZE*/

/*-------------------------------------------------------------*/
/* INITIALIZE SDFPKG WITH DESIRED PAGING AREA                  */
/*-------------------------------------------------------------*/

 SDF_COMM_TABLE.#PAGES           = SDF_NO_PAGES;
 SDF_COMM_TABLE.ADDR_PAGING_AREA = ADDR (SDF_PAGING_AREA);

/*-------------------------------------------------------------*/
/* INITIALIZE OTHER COMMUNICATION TABLE PARAMETERS             */
/*-------------------------------------------------------------*/

 SDF_COMM_TABLE.CRETURN       = 0;
 SDF_COMM_TABLE.BLOCK_NO      = 0;
 SDF_COMM_TABLE.SYMBOL_NO     = 0;
 SDF_COMM_TABLE.STMT_NO       = 0;
 SDF_COMM_TABLE.SDF_NAME      = '';
 SDF_COMM_TABLE.CSECT_NAME    = '';
 SDF_COMM_TABLE.STMT_REF_NO   = '';
 SDF_COMM_TABLE.INCLUDE_COUNT = 0;
 SDF_COMM_TABLE.BLOCK_NAME    = '';
 SDF_COMM_TABLE.SYMBOL_NAME   = '';
```

```
                        SDFPKG_RETURN_CODE (
                                     SDF_COMM_TABLE.CRETURN),
                        4, ABORT_FLAG);
         END;

         ELSE DO;   /* SDFPKG WAS INITIALIZED */

           /*-----------------------------------------------*/
           /* INITIALIZE SDFPKG DATABUFF COMMUNICATION AREA */
           /*-----------------------------------------------*/

            SDF_DBUFF_AREA_PTR = SDF_COMM_TABLE.ADDRESS;
         END;
      END;
   END;

END SDFPKG_INITIALIZE;


%PAGE;
/*-------------------------------------------------------------------*/
/* SAVE SDFPKG FINAL JOB STATISTICS AND TERMINATE SDFPKG.  SDFPKG    */
/* STATISTICS CAN BE USEFUL TO THE MAINTAINER.          SDFIR.1200 */
/*-------------------------------------------------------------------*/

SDFPKG_TERMINATE:  PROCEDURE
   (SDF_TOTS);

   DECLARE
      SDF_TOTS               LIKE            /*SETUP LOCAL SDFPKG  */
         SDF_TOTALS;                         /* TOTALS STRUCTURE   */

   DECLARE
      TITLE                  CHAR(132)       /*DEBUG TITLE         */
                             VARYING,
      TRACE_FLAG             BIT(1),         /*PROC TRACE FLAG     */
      TRACE_FLAG_SDFI        BIT(1),         /*SDF INTERFACE TRACE */
      #INVOCATIONS           FIXED BIN(31)   /*# TIMES PROC CALLED */
                             INITIAL (1);

   TRACE_FLAG      = FALSE;            /*PROC TRACE FLAG IS OFF    */
   TRACE_FLAG_SDFI = FALSE;            /*SDF INTERFACE TRACE IS OFF*/
   TITLE           = '';               /*SET TITLE TO NULL         */

   IF (TOGGLE_BIT_STR(1) = TRUE) THEN DO;    /*IF TRACE DESIRED THEN*/
      TRACE_FLAG = TRUE;                     /*SET TRACE FLAG ON    */
      TITLE = 'SDFPKG TERMINATE -- INVOCATION # ' ||
```

```
  /*-------------------------------------------------------------*/
  /* SAVE KEY SDFPKG PARAMETERS FOR USE BY PRINTSUMMARY          */
  /*-------------------------------------------------------------*/

   SDF_TOTS.#LOCATES  = SDF_DATABUFF.#LOCATES;
   SDF_TOTS.#READS    = SDF_DATABUFF.#READS;
   SDF_TOTS.#WRITES   = SDF_DATABUFF.#WRITES;
   SDF_TOTS.#SELECTS  = SDF_DATABUFF.#SELECTS;
   SDF_TOTS.#RESERVES = SDF_DATABUFF.#RESERVES;
   SDF_TOTS.#FCBS     = SDF_DATABUFF.#FCBS;
   SDF_TOTS.#PAGES    = SDF_DATABUFF.#PAGES;
   SDF_TOTS.#GETMAINS = SDF_DATABUFF.#GETMAINS;

  /*-------------------------------------------------------------*/
  /* ALLOW SDFPKG TO TERMINATE ITSELF.  THEN IT WILL BE DELETED  */
  /*-------------------------------------------------------------*/

   CALL WMHSDFI (SDF_TERM, SDF_NO_OPT, SDF_COMM_TABLE,
                 TRACE_FLAG_SDFI, TITLE);

END SDFPKG_TERMINATE;


%PAGE;
/*-------------------------------------------------------------*/
/* PRINT SUMMARY SDFPKG STATISTICS BEFORE JOB TERMINATES PROCES-  */
/* SING.  THESE STATISTICS CAN BE USEFUL TO THE MAINTAINER.       */
/*                                                    SDFIR.1300 */
/*-------------------------------------------------------------*/

SDFPKG_SUMMARY:  PROCEDURE
  (SDF_TOTS);

   DECLARE
      SDF_TOTS               LIKE            /*SETUP LOCAL SDFPKG  */
         SDF_TOTALS;                         /* TOTALS STRUCTURE   */

   CALL PUT_SYSPRINT_TITLE ('SDFPKG SUMMARY STATISTICS', PAGE_NO);

   PUT SKIP(1) LIST ('NUMBER OF SDFPKG LOCATES  = ' ||
                     SDF_TOTS.#LOCATES);
   PUT SKIP(1) LIST ('NUMBER OF SDFPKG READS    = ' ||
                     SDF_TOTS.#READS);
   PUT SKIP(1) LIST ('NUMBER OF SDFPKG WRITES   = ' ||
                     SDF_TOTS.#WRITES);
   PUT SKIP(1) LIST ('NUMBER OF SDFPKG SELECTS  = ' ||
```

```
END SDFPKG_SUMMARY;
```

6.0 SDFPKG Assembler Data Overlay MACROS

(These macros overlay the various data tables contained within the SDFs and are used by SDFPKG. They should correspond to the diagrams contained in the current Revision of the HAL/SDL ICD.

6.1 SDFPKG Communication Table (COMMTABL)

```
         MACRO
         COMMTABL
COMMTABL DSECT       SDFPKG COMMUNICATION AREA
APGAREA  DS    A     ADDRESS OF EXTERNAL PAGING AREA
AFCBAREA DS    A     ADDRESS OF EXTERNAL FCB AREA
NPAGES   DS    H     # OF PAGES IN PAGING AREA OR AUGMENT
NBYTES   DS    H     # OF BYTES IN FCB AREA OR AUGMENT
MISC     DS    H     MISCELLANEOUS PURPOSES
CRETURN  DS    H     SDFPKG RETURN CODE
BLKNO    DS    H     BLOCK NUMBER (BLOCK INDEX TABLE INDEX)
SYMBNO   DS    H     SYMBOL NUMBER (SYMBOL INDEX TABLE INDEX)
STMTNO   DS    H     STATEMENT NUMBER (STATEMENT INDEX TABLE INDEX)
BLKNLEN  DS    CL1   NUMBER OF CHARACTERS IN BLOCK NAME (BLKNAM)
SYMBNLEN DS    CL1   NUMBER OF CHARACTERS IN SYMBOL NAME (SYMBNAM)
PNTR     DS    F     VIRTUAL MEMORY POINTER LAST LOCATED
ADDR     DS    A     CORE ADDRESS CORRESPONDING TO PNTR
SDFDDNAM EQU   *     NAME OF ALTERNATE DD FOR SDF DATASET
SDFNAM   DS    CL8   NAME OF SDF TO BE SELECTED
CSECTNAM DS    CL8   NAME OF CODE CSECT FOR BLOCK
SREFNO   DS    CL6   STATEMENT REFERENCE NUMBER
INCLCNT  DS    H     INCLUDE COUNT (FOR SRN)
BLKNAM   DS    CL32  BLOCK NAME
SYMBNAM  DS    CL32  SYMBOL NAME
         MEND
```

6.2 SDFPKG Common Data Pool Buffer Table (DATABUF)

```
         MACRO
         DATABUF
DATABUF  DSECT       COMMON DATA BUFFER TEMPLATE
LOCCNT   DS    F     CURRENT LOCATE COUNTER
AVULN    DS    A     ADDRESS OF VULNERABLE PAD ENTRY
CURFCB   DS    A     ADDRESS OF CURRENT FCB
PADADDR  DS    A     STARTING ADDRESS OF PAD
ACOMMTAB DS    A     ADDRESS OF COMMUNICATION AREA
ACURNTRY DS    A     ADDRESS OF CURRENT PAD ENTRY
ROOT     DS    A     ADDRESS OF ROOT FCB OF FCB TREE
SAVEXTPT DS    F     POINTER TO SYMBOL NODE EXTENT CELL
SAVFSYMB DS    H     FIRST SYMBOL OF BLOCK
SAVLSYMB DS    H     LAST SYMBOL OF BLOCK
NUMGETM  DS    H     NUMBER OF ENTRIES IN GETMAIN STACKS
NUMOFPGS DS    H     NUMBER OF PAGES IN CURRENT PAGING AREA
BASNPGS  DS    H     INITIAL NUMBER OF PAGES IN PAGING AREA
FCBSTKLN DS    H     NUMBER OF ENTRIES IN FCB STACKS
IOFLAG   DS    C     I/O IN PROGRESS INDICATOR
GETMFLAG DS    C     > 0 IMPLIES AUTO GETMAINS FOR FCBS
GOFLAG   DS    C     > 0 IMPLIES SUCCESSFUL INITIALIZATION
MODFLAG  DS    C     > 0 IMPLIES UPDAT MODE ACTIVE
ONEFCB   DS    C     > 0 IMPLIES ONLY ONE FCB KEPT
FIRST    DS    C     > 0 IMPLIES TAKE FIRST SYMBOL FOUND
         DS    2C    SPARE
TOTFCBLN DS    F     TOTAL AMOUNT OF FCB SPACE IN USE
RESERVES DS    F     GLOBAL (TOTAL) COUNT OF RESERVES
READS    DS    F     TOTAL NUMBER OF READS
WRITES   DS    F     TOTAL NUMBER OF WRITES
SLECTCNT DS    F     TOTAL NUMBER OF 'REAL' SELECTS
FCBCNT   DS    F     TOTAL NUMBER OF FCBS IN EXISTENCE
GETMSTK1 DS    A     ADDRESS OF GETMAIN ADDRESS STACK
GETMSTK2 DS    A     ADDRESS OF GETMAIN LENGTH STACK
FCBSTK1  DS    A     ADDRESS OF FCB AREA ADDRESS STACK
FCBSTK2  DS    A     ADDRESS OF FCB AREA LENGTH STACK
MAXSTACK DS    H     MAXIMUM NUMBER OF STACK ENTRIES
SDFVERS  DS    H     SDF VERSION NUMBER (OF SELECTED SDF)
APGEBUFF DS    A     ADDRESS OF PAGE BUFFER
ADECB    DS    A     ADDRESS OF DECB
ECB      DS    F     EVENT CONTROL BLOCK (DECB)
IOTYPE   DS    H     I/O TYPE (DECB)
IOLENGTH DS    H     NUMBER OF BYTES TO TRANSFER (DECB)
DCBADDR  DS    A     ADDRESS OF HALSDF DCB (DECB)
BUFLOC   DS    A     ADDRESS OF BUFFER AREA (DECB)
IOBADDR  DS    A     ADDRESS OF IOB (DECB)
```

6.3 SDFPKG File Control Block (FCB)

```
         MACRO
         FCBCELL
FCBCELL  DSECT
TTRK     DS    F
GTTREEPT DS    A
LTTREEPT DS    A
FILENAME DS    CL8
BLKPTR   DS    F
SYMBPTR  DS    F
STMTPTR  DS    F
TREEPTR  DS    F
NODESIZE DS    H
FLAGS    DS    H
NUMBLKS  DS    H
NUMSYMBS DS    H
FSTSTMT  DS    H
LSTSTMT  DS    H
LSTPAGE  DS    H
VERSIONX DS    H
STMTEXPT DS    F     POINTER TO STATEMENT NODE EXTENT CELL
SPARE2   DS    F
FCBLEN   EQU   *-FCBCELL
FCBTTRZ  DS    F
FCBPDADR DS    A
         MEND
```

6.4 SDFPKG Paging Area Directory (PAD)

```
         MACRO
         PDENTRY
PDENTRY  DSECT       PAGING AREA DIRECTORY ENTRY TEMPLATE
PAGEADDR DS    A     ADDRESS OF IN-CORE PAGE
FCBADDR  EQU   *     ADDRESS OF FILE CONTROL BLOCK (FCB)
MODFIND  DS    CL1   '80' > PAGE IS MODIFIED
         DS    CL3   ADDRESS OF FILE CONTROL BLOCK (FCB)
USECOUNT DS    F     USAGE COUNTER
PAGENO   DS    H     PAGE # * 8
RESVCNT  DS    H     RESERVE COUNTER
PDENTLEN EQU   *-PDENTRY
         MEND
```

```
DROOTPTR DS    A     POINTER TO DIRECTORY ROOT CELL
DATFCPTR DS    A     POINTER TO DATA FREE CELL CHAIN
         MEND
```

6.6 Directory Root Cell

```
         MACRO
         DROOTCEL
DROOTCEL DSECT       DIRECTORY ROOT CELL TEMPLATE
SDFFLAGS DS    2C    SDF FLAGS
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 INDEX TABLE ENTRIES
SYMNODES DS    H     # OF SYMBOL INDEX TABLE ENTRIES
FBNPTR   DS    A     POINTER TO FIRST BLOCK INDEX TABLE ENTRY
LBNPTR   DS    A     POINTER TO LAST BLOCK INDEX TABLE ENTRY
INSTRCNT DS    H     NO. OF EMITTED MACHINE INSTRUCTIONS
FREEBYTE DS    H     TOTAL AMT 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 INDEX TABLE ENTRY
LSNPTR   DS    A     POINTER TO LAST SYMBOL INDEX TABLE ENTRY
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 INDEX TABLE ENTRIES
FSTNPTR  DS    A     POINTER TO FIRST STATEMENT INDEX TABLE ENTRY
LSTNPTR  DS    A     POINTER TO LAST STATEMENT INDEX TABLE ENTRY
SNELPTR  DS    A     POINTER TO STATEMENT NODE EXTENT LIST
FIRSTSRN DS    CL8   FIRST SRN IN SDF
LASTSRN  DS    CL8   LAST SRN IN SDF
CUBTCNUM DS    H     BLOCK NUMBER OF UNIT BLOCK
COMPUNIT DS    H     COMPILATION UNIT ID CODE
TITLEPTR DS    F     VIRTUAL MEMORY POINTER TO TITLE INFO
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 STRINGS
XREFCNT  DS    F     ACTUAL NUMBER OF XREF ENTRIES
DRCLEN   EQU    *-DROOTCEL
         MEND
```

6.7 Block Index Table Entry

```
           MACRO
         BLCKNODE
BLCKNODE DSECT       BLOCK INDEX TABLE ENTRY TEMPLATE
CSCTNAME DS    CL8   CSECT NAME OF BLOCK
BLOCKPTR DS    A     POINTER TO BLOCK TREE CELL
         MEND
```

6.8 Block Data Cell

```
          MACRO
         BLKTCELL
BLKTCELL DSECT       BLOCK DATA CELL TEMPLATE
RTREEPTR DS    A     RIGHT TREE POINTER (>)
LTREEPTR DS    A     LEFT TREE POINTER (<)
FNESTPTR DS    F     FIRST NESTED BLOCK
LNESTPTR DS    F     NEXT BLOCK AT SAME LEVEL
EXTPTR   DS    A     SYMBOL EXTENT CELL POINTER
         DS    F     SPARE
BLKFLGS  DS    C     FLAGS APPLICABLE TO THE BLOCK
         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
```

6.9 Block Statement Extent Cell

6.9.1 Invariant part of Block Statement Extent Cell

6.9.2 Variant part of Block Statement Extent Cell

```
         MACRO
         STMTEXTV
STMTEXTV DSECT       DSECT FOR VARIABLE PART OF STMT EXTENT CELL
FSTOFF1  DS    H     OFFSET TO FIRST SRN ON PAGE
LSTOFF1  DS    H     OFFSET TO LAST SRN ON PAGE
FSTSRN   DS    CL8   FIRST SRN ON PAGE
LSTSRN   DS    CL8   LAST SRN ON PAGE
         MEND
```

6.10 Block Symbol Extent Cell

6.10.1 Invariant part of Block Symbol Extent Cell

```
         MACRO
         SYMEXTF
SYMEXTF  DSECT DSECT FOR FIXED PART OF SYMBOL EXTENT CELL
SUCCPTR  DS    F     POINTER TO SUCCESSOR CELL (USUALLY 0)
NEXTNTRY DS    H     NUMBER OF EXTENT ENTRIES
FSTPAGE  DS    H     PAGE # CORRESPONDING TO 1ST ENTRY
         MEND
```

6.10.2 Variant part of Block Symbol Extent Cell

```
         MACRO
         SYMEXTV
SYMEXTV  DSECT       DSECT FOR VARIABLE PART OF SYMBOL EXTENT CELL
FSTOFF   DS    H     OFFSET TO FIRST SYMBOL ON PAGE
LSTOFF   DS    H     OFFSET TO LAST SYMBOL ON PAGE
FSTSYMB  DS    CL8   NAME (8 CHARS) OF FIRST SYMBOL
LSTSYMB  DS    CL8   NAME (8 CHARS) OF LAST SYMBOL
         MEND
```

6.11 Symbol Index Table Entry

```
         MACRO
         SYMBNODE
SYMBNODE DSECT       SYMBOL INDEX TABLE ENTRY TEMMPLATE
SYMBNAME DS    CL8   FIRST EIGHT CHARACTERS OF SYMBOL NAME
```

6.12 Symbol Data Cell

6.12.1 Invariant part of Symbol Data Cell

```
         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    0C    SYMBOL NAME CONTINUATION
SDCLEN   EQU   *-SYMBDC
         MEND
```

6.12.2 Array Dimensions in Symbol Data Cell

```
         MACRO
         ARRADATA
ARRADATA DSECT       ARRAYNESS DATA TEMPLATE
ARRAYNUM DS    H     NUMBER OF DIMENSIONS
RANGE1   DS    H     RANGE OF DIMENSION 1
RANGE2   DS    H     RANGE OF DIMENSION 2
RANGE3   DS    H     RANGE OF DIMENSION 3
```

```
STRCDATA DSECT       STRUCTURE LINKAGES TEMPLATE
LINK1    DS    H     LINK TO UNQUALIFIED STRUCTURE
LINK2    DS    H     LINK TO ELDEST SON
LINK3    DS    H     LINK TO BROTHER
STRCLEN  EQU   *-STRCDATA
         MEND
```

6.13 Statement Index Table Entry

6.13.1 Statement Index Table Entry (Without SRNs)

```
         MACRO
         STMTNOD0
STMTNOD0 DSECT       STATEMENT INDEX TABLE ENTRY TEMPLATE (SRN_FLAG=0)
STDCPTR  DS    A     POINTER TO STATEMENT DATA CELL
         MEND
```

6.13.2 Statement Index Table Entry (With SRNs)

```
         MACRO
         STMTNOD1
STMTNOD1 DSECT       STATEMENT INDEX TABLE ENTRY TEMPLATE (SRN_FLAG=1)
SRN      DS    CL6   STATEMENT REFERENCE NUMBER
INCOUNT  DS    CL2   INCLUDE COUNT
STDCPTR1 DS    A     POINTER TO STATEMENT DATA CELL
         MEND
```

6.14 Executable Statement Data Cell

```
         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
```

7.0 Bibliography

HAL/SDL ICD, NAS 9-14444, Version 8.0

Shulenberg, C., Shuttle Memo #12-75, 07 March 1975.

Day, J., Intermetrics MSD Memo #TX-108-86, 30 January 1986.

Day, J., Care and Feedings of SDFs, Presentation, 29 April 1986.

Day, J., XPL Information Guide.

Harper, B., Simulation Data Files, Presentation, 01 November 1988.

IBM, OS Data Management Services, Part 1.

HALSTAT 1,2,3 XPL Programs

PRE-HALSTAT XPL Program

SDFPKG Source

********* THIS PAGE IS NOT FOR DISTRIBUTION **********

ITEMS IN MEMO THAT MUST BE MANUALLY FIXED:

1. mode 0 item # 3 the NBYTES parameter (page 31) the first one must have an not equal instead of an equal (=)!!!!!