IBM Application Program

Y20-0345-0

System/360 Problem Language Analyzer (PLAN) (DOS/OS)

(360A-CX-26X, 360A-CX-27X)

Volume I – Flowchart Narratives

System Manual

This manual contains detailed information in the form of flowchart narratives for the DOS/360 PLAN and OS/360 PLAN systems. With this, the user should gain a better understanding of the logic of the system.

First Edition (July 1969)

Significant changes or additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

This edition applies to Version 1, Modification Level 0, of System/360 Problem Language Analyzer (PLAN) (DOS/OS) (360A-CX-26X, 360A-CX-27X) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the specifications herein. Therefore, before using this publication, consult the latest System/360 SRL Newsletter (N20-0360) for the editions that are applicable and current.

Address comments concerning the contents of this publication to: IBM Corporation, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601.

CONTENTS

INTRODUCTION

The Problem Language Analyzer (PLAN) is designed to allow implementation of desirable user-oriented (problem-oriented) languages by providing a common language processor. Previously, problem-oriented languages have required independent language processors that were in themselves major implementation tasks. Even though highly desirable, problem-oriented languages were implemented only for major applications. Reimplementation on new equipment has made long-term costs even higher.

The PLAN system through the common processor allows input to a job to be composed of several dissimilar problem-oriented language jobs, all operating in a homogeneous environment. It also allows easy modification and expansion of existing applications. The PLAN concept of implementation of logic modules makes complete machine independence of logic modules more easily attainable.

Logic module loading is accomplished dynamically at execution time as defined by the current job description. This means that logic modules are loaded only as required and that existing logic modules do not require modification to incorporate new processing capabilities. Multiple versions of the PLAN system for the IBM 1130 System, the IBM System/360 using the Disk Operating System, and the IBM System/360 using the Operating System allow logic modules written in machine independent ASA FORTRAN IV to be executed on either computer system. The job is described in problem-oriented terms on the most accessible system in a language compatible to all systems.

In general, implementation of a problem-solving system operating within a PLAN environment involves the several tasks as defined below:

1. Definition of the problem-oriented language. This definition is processed by PLAN to create the language dictionary.
2. Programming of logic modules (if existent logic modules do not suffice) to support the problem solution functions (note that this does not encompass problems of language processing; these are handled by PLAN).
3. Generation of problem-oriented language statements to describe the particular unique problem to be solved.

Many utility routines are provided with PLAN to make writing of logic modules easier and faster, and to provide a logic module that provides a more powerful and more efficient problem solution.

SYSTEM OVERVIEW

The following diagram shows the overall logic of the PLAN Monitor.

The system is driven by the program pop-up list. This list and each entry is eight EBCDIC characters naming a program module that can be found in the PLAN program library. There are two modules in the PLAN system that are loaded without using the pop-up list. These are:

DFJPSCAN - The command processor and language interpreter which is loaded whenever th pop-up list is empty.

DFJPERRS - The system error processor which is loaded if the monitor obtains control and an error has occurred.

All other modules loaded by PLAN are loaded because their names were encountered in the pop-up list.

The three entries shown represent the loader interface subroutines LEX, LOCAL and LRET.

LEX causes a transfer of control to the next module in the pop-up list. The calling module may be overlayed.

LOCAL causes a transfer of control to the next module in the pop-up list to be executed as a subprogram of the caller. The calling program may not be overlayed.

LRET causes control to be returned to the caller. If the module was called as a LOCAL, control is returned to the calling module. Otherwise, control is given to the PLAN loader.

Refering to the diagram, initial entry to PLAN causes DFJPSCAN to be loaded to process a command. DFJPSCAN initializes BLANK COMMON and places a list of names in the pop-up list and then calls LRET.

On LRET, the loader checks for a return from a LOCAL module. If so, it returns control to the calling module.

The loader control then checks for any errors and loads DFJPERRS to process them.

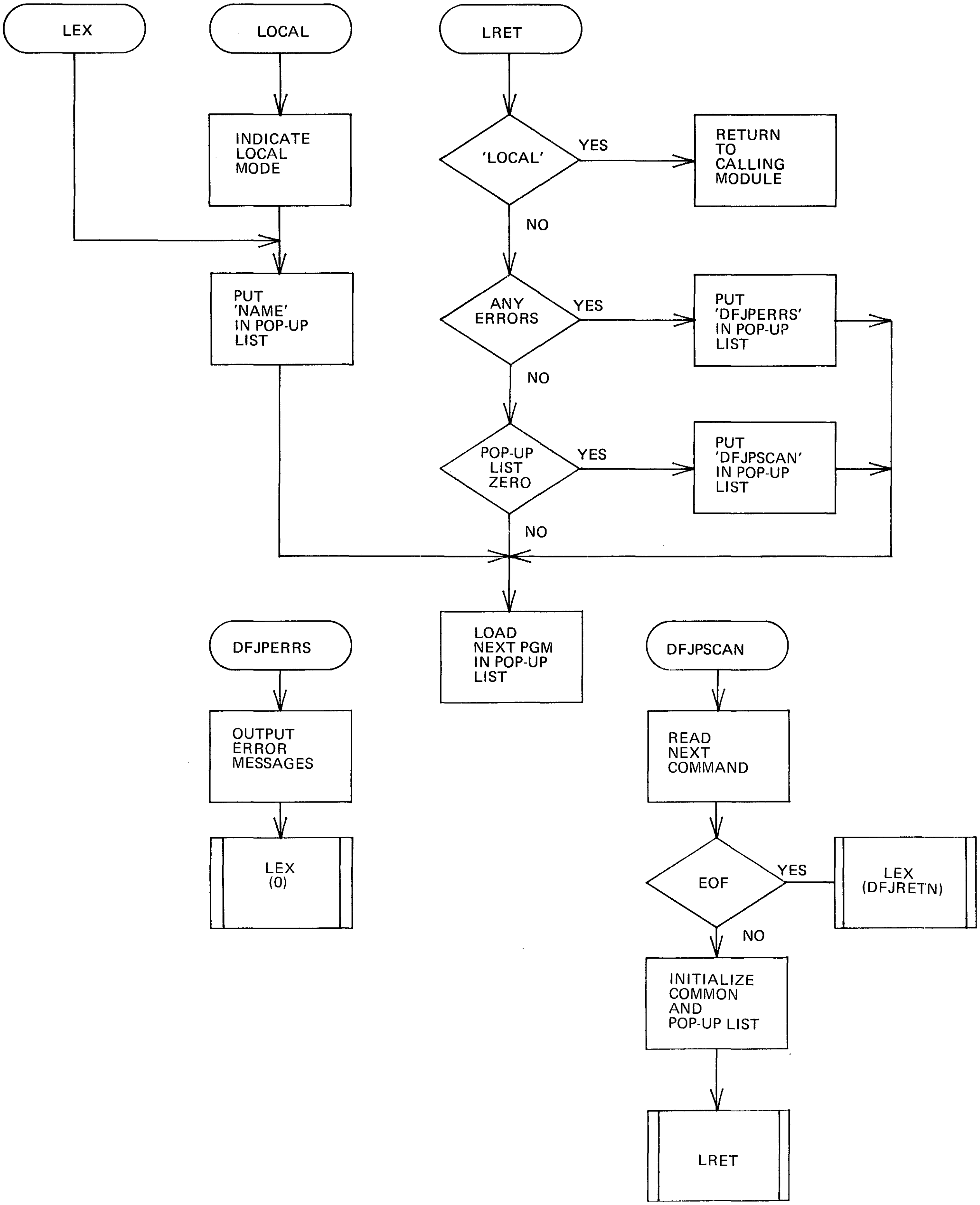
The next name is extracted from the pop-up list and if nonzero, that program is loaded and entered for execution. This processing continues until the list is zero and then DFJPSCAN is reloaded and the cycle repeats itself.

DFJPSCAN will terminate PLAN processing when an end-of-file occurs in the command input stream. It does this by LEX to the PLAN module DFJRETN which returns control to the OS or DOS supervisor.

On a LOCAL entry to the loader, the test to call DFJPERRS is not executed because the calling module may not be overlayed.

On LRET from a LOCAL control is transferred directly to the calling module.

PLAN SYSTEM GENERAL LOGIC



PLAN SYSTEM INITIALIZATION

OS/360 PLAN INITIALIZATION

The general function of the OS PLAN initialization is to a) process PARMS from the EXEC control card, b) allocate the PLAN program COMMON area, c) load resident PLAN modules, d) validate JCL and open data sets.

The following items detail the sequence of events that occur during PLAN initialization for OS:

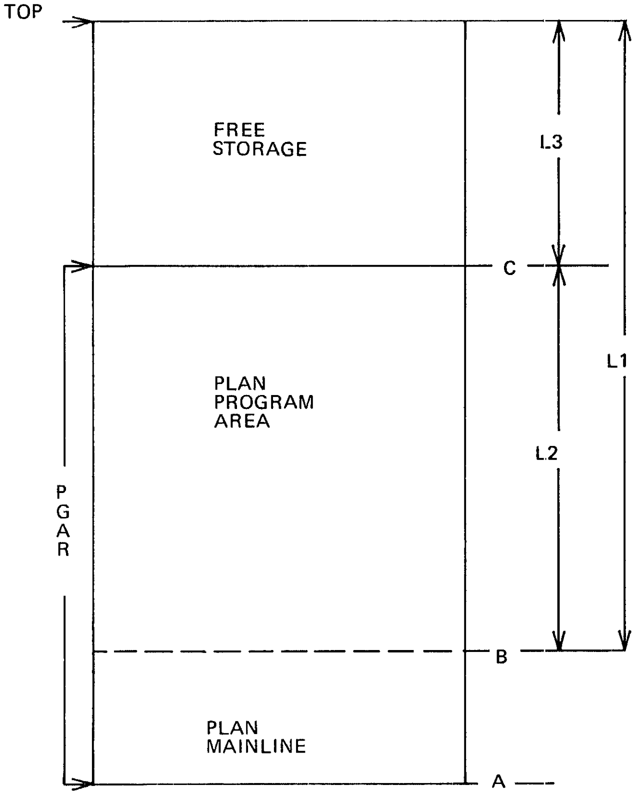
1. The registers are saved and a save area is established.

2. The system type MVT or PCP/MFT is determined. This is done by inspecting the CVT. If the system is MVT, an indicator is set for use by the core management routine.

3. A GETMAIN for 408 bytes is issued to allocate the program pop-up list. The list is cleared and a pointer to the beginning and the end of the list is saved in the PLAN COMMON area.

4. The EXEC control card PARMs are processed.

5. The PLAN program COMMON area is allocated. This area must be contiguous and begin at the start of the partition or region. The PLAN mainline 'DFJPLAN' is exactly 12,288 bytes in length to insure that it is loaded at the beginning of the region on MVT systems. Refer to the following diagram showing the program COMMON area allocation. A VC GETMAIN is issued and an address B and a length L1 is returned. The default value for L2 is calculated as (12K+L1) *2/3. If the PGAR PARM was specified the value for it is used. The length L3 is found from L1+12K-L2. The address C or the top of the PLAN program area is equal to A+L2. A FREEMAIN is then issued using the address C and a length L3. This releases the unused core.



6. The PLAN modules DFJLODER and DFJTRACE are loaded into the partition. This causes the first significant difference in the region layout between PCP/MFT and MVT systsms. On MVT, loading the modules causes creation of a block of core in subpool 252. On PCP/MFT systems, the modules are loaded as high as possible in the partition.

7. Data sets defined by the PLINP, PLOUT and PLSEQ DD cards are opened. If a PLINP or PLOUT DD card are missing, an ABEND user code 100 occurs. The subroutines 'TSRCHA' and 'TSRCHB' are used to search the TIOT and read the JFCB if an equivalent DD name is found. The subroutine 'OPENSEQ' builds a DCB from a skeleton, merges fields from JFCB to the DCB, validates the device type, allocates the buffers, blanks the first buffer, and opens the data set. A TCLOSE macro is issued for all PLSEQ data sets so that the first access may be either READ or WRITE.

8. The PLANLIB PDS is opened.

9. The PLSYSTAB data set (the phrase dictionary) is opened. If the file is new, it is formatted and then initialized. The key thing in this is that the ADD PHRASE phrase is written onto the PFINPUTA record of the file and a switch is set so that the module DFJPHRAS is the first program called in order to actually add this phrase to the dictionary. The phrase dictionary file is assigned a permanent drive number of zero and a logical file number of 255 so that the phrase table dump routine can use the subroutine GDATA and RDATA to access this file.

10. The managed area save file PLMANFIL and the checkpoint file PLCHKPT are opened if present.

11. PLAN PERMANENT drive data sets (PLFSnyyy) are opened if present.

12. PLAN DYNAMIC drive data sets (PLANDRVn) are opened and formatted if new. If the file is old it is validated.

Note that the subroutine DSCHK does a physical open on all direct access file. If disposition is old the DSCB is read from VTOC and validated. If the file is new, it is formatted with the value of FALSE. '7FFFFFFF'

13. A BLDL macro is issued to ensure that the modules DFJPSCAN, DFJPERRS, and DFJRETN are in the PLANLIB PDS. These modules must be present or the PLAN system will loop.

14. The program area is cleared and either DFJPSCAN or DFJPHRAS is loaded to begin PLAN execution.

DOS/360 PLAN INITIALIZATION

The following items detail the sequence of events for DOS PLAN initialization.

1. The address of the top of the partition and saved.

2. PLAN run control cards are read and processed.

3. The FORTRAN I/O area is allocated. This is done by using the PLAN subroutine DFJGMAIN which will allocate core from the top of the partition down.

4. The PLAN transient routine $$BDFJI is called to set the address of the FORTRAN I/O area into the DOS COMREG area. This is necessary because the FORTRAN I/O package uses the address of the end of the phase for a work area for its buffers.

5. If a user work area was specified it is allocated.

6. The resident PLAN subroutines DFJDIOCS, DFJSIOCS, and DFJCNTRL are moved to the top of the partition.

7. The PLAN system CCB control blocks are created and moved to the top of the partition.

8. The pop-up program list is allocated and cleared.

9. The core-image library control block is opened.

10. If an alternate library is specified, its control block is created and opened.

12. The PLAN module DFJIOCBS is loaded into the partition.

13. If any *ASGN cards were processed, their specifications are merged with the existing control blocks in the module DFJIOCBS.

14. If a TRACE is required the subroutine DFJTRACE is moved to the top of the partition.

15. A check is made to ensure that the modules DFJPSCAN, DJFPERRS, and DFJPHRAS are in the core-image library.

16. The PLAN system files DFJPCHK, DFJPDTA, and DFJPFILE are opened using the PLAN transient routine $$BDFJDO.

17. DFJPFILE (the phrase dictionary) is validated. If it is new, the program DFJPHRAS is put into the pop-up list.

18. Transfer is to the loader to begin execution.

OS PROGRAM LOADER

The PLAN system must allow modules not linkedited together to communicate with each other through BLANK COMMON. For this reason the LOAD, LINK, XCTL, or ATTACH macros cannot be used to load modules to be executed under the PLAN system.

Figure 1 shows the layout of a program segment in the PLAN program area.

```
     ┌─────────┐
     │         │
1    │  ENTAB  │
     │         │
     ├─────────┤
     │         │
     │  PGM    │
2    │  AREA   │
     │         │
     ├─────────┤
     │         │
3    │  PCB    │
     │         │
     ├─────────┤
4    │  UAB    │
     ├─────────┤
     │         │
5    │  LSA    │
     │         │
     ├─────────┤
6    │  LCB    │
     └─────────┘
```

Figure 1. PLAN program area layout

1. The entry table contains the name and entrypoint for every CSECT in the program area.

2. Program area contains the module itself with the BLANK COMMON CSECT extracted.

3. The program control block contains the name, entrypoint, and other information about the program.

4. The unresolved adcon block is only created when an unresolved external reference is found in the program area.

5. The LOCAL save area is used to save the program status if CALL LOCAL is used.

6. The level control block describes the length of the entire segment.

The OS program loader replaces the OS fetch facility. It is divided into two sections 1) Loader Control which is located in the PLAN mainline, 2) Module Loader which is a section of the module 'DFJLODER'.

The module loader performs the following:

1. Relocation of the BLANK COMMON CSECT
2. Builds the ENTAB
3. Loads TXT records
4. Relocation of adcons
5. Creates a PCB

The Loader control performs the following:

1. Program area maintenance
2. Free storage management
3. In-core program search
4. Bank loading control
5. Final linkedit and processing of unresolved adcons
6. Local processing control
7. Creation of LSA and LCB

The first step in loading a module is to determine if it is already in the program area. This is done by searching the PCB chain. If the module is in core, it is entered without any further processing. When the module is not found, program area maintenance is performed by adjusting the LCB and PCB chains to release inactive segments. If required, free storage management is performed at this time. The loader is then called.

The first step for the loader is to locate the module in the PLAN library data set. If an in-core directory is available it is searched for the module name. If the name is not found, a BLDL macro is issued to locate the module. The ESD (external-entry symbol table) records are read and processed. From these the entrypoint table (ENTAB) is built, the location and length of BLANK COMMON is determined, and the names of all external references are recorded in a table (ERTAB1).

The TXT records which contain the relocatable code for the module are then read. It is at this point that the BLANK COMMON CSECT is deleted from the module. All CSECTs originating above BLANK COMMON are relocated downward by the length of BLANK COMMON. The RLD records (Relocatable Adcon Dictionary) are read and the adcons in the program area are relocated. If an adcon refers to BLANK COMMON it is relocated to point to PLAN BLANK COMMON. If an unresolved external reference (V-TYPE adcon) is found, its name is determined from the external reference name table (ERTAB1) and then entrypoint tables (ENTAB) for modules already in core and the JOBPAC entrypoint table are searched. If an equivalent name is found, the external reference is resolved. If the adcon cannot be resolved an entry in the unresolved adcon table (ERTAB2) is made. After the module has been loaded the PCB is completed and control is returned to the mainline where the unresolved adcons are processed.

For each external reference still unresolved, a 24-byte control block (URABLK) is built and the adcon is resolved to this block which has the format:

```
0   L     15,16(0,15)
4   BAL   15,URAENT-PLAN(0,15)
8   DC    C18'NAME'
16  DC    V(PLAN)
```

This code causes execution-time reference to the adcon to branch to the PLAN mainline where it is processed as a 'LOCAL'.

The final action in the loading process is to create a LOCAL SAVE AREA and a LEVEL CONTROL block.

SEQUENTIAL FILE PROCESSING

The PLAN subroutine DFJSIOCS processes read/write requests for all sequential files for OS. It is located in the module DFJLODER. For DOS it is loaded at the top of the partition by the initialization routine.

For OS the files are opened at initialization time using the standard OPEN macro and the BSAM access method is used for processing.

For DOS the files are opened when the first read/write request is executed by the transient module $$BDFJSO and the EXCP access method is used for processing. Basically the opening of the file consists of constructing a control block that DFJSIOCS uses to process physical records. The conversion routines (PAIN, PAOUT, etc) locate the buffer address and length from the control block and cause only transmission to/from the buffer.

PLAN SORT FACILITY

The PLAN subroutines PSORT, PMERG, GSORT, and GMERG provide the program interface to the PLAN SORT/MERGE modules. These are:

1. DFJPSRTA and DFJGSRTA which are block sorting routines for DYNAMIC and PERMANENT files respectively.
2. DFJPSRTB and DFJGSRTB which are in-place merge routines which may be used if a SORT cannot be completed by the block sort modules.
3. DFJPMERG and DFJGMERG are the PLAN DYNAMIC and PERMANENT file merge.

The only differences between DFJPXXXX and DFJGXXXX modules is that DFJPSRTA, DFJPSRTB, and DFJPMERG use the READ/WRITE subroutines and DFJGSRTA, DFJGSRTB, and DFJGMERG use the RDATA/WDATA subroutines.

PLAN SORT

The PLAN SORT is invoked by the subroutines PSORT/GSORT issuing a CALL LCHEX (6, 'DFJXSRTA, DFJXSRTB,*'). NOTE that the X in both DFJXSRTA, DFJXSRTB correspond to either a P or G.

DFJXSTRA is the first load of the SORT, validates the sort control field, and does the block sort. It creates a work area in the mainline which contains the following:

WORD 1-2	ID block of file to be sorted
WORD 3	Record length in bytes
WORD 4	Number of words in the Record Sort Area
WORD 5	Number of records in the Record Sort Area
WORD 6	Address of list of RSA pointers
WORD 7	Address of a save area for 1 logical record
WORD 8	Address of the last record in the Record Sort Area
WORD 9	Address of the Record Sort Area
WORD 10	Address of the Sort Control Fields
Word 11	Sequence break KDIS

The block Sort module allocates available core for the sort to two areas.

1. The Sort record area where the actual record will be read/written.
2. A list of pointers to each logical record in the Sort record area.

After reading the block of records into the RSA, the LIST is initialized as shown in Figure 2. Then the SORT routine uses a binary chop search to order the list of pointers rather than the records. In the example, the list of pointers would change as shown at completion of the internal sort. The records themselves are rearranged into the order shown in the list. The process is repeated until each block in the file has been sorted. A check for a sequence break across blocks is made and if none occurs, the sort is complete at the end of the block sort. In this case, the pop-up list pointer is updated to bypass the loading of the merge module.

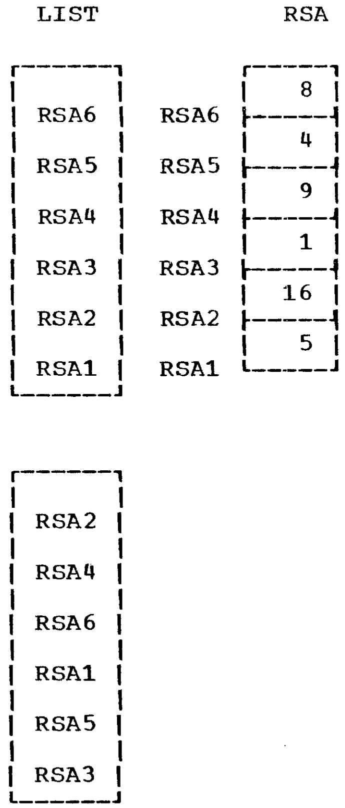


Figure 2.

The in-place merge is performed by the module DFJXSRTB. The managed array save file is used as a work file. The general technique is to locate two sequence breaks and peform a descending merge. When a sequence break is found, the out-of-sequence block is copied to the work file to create space for the output of the merge. Then the work file and the in-sequence block are read backwards and a descending merge performed. The output of the merge is written backwards starting at the end of the out-of-sequence block.

OS CONTROL BLOCKS

ENTAB ENTRY

```
    ┌─────────────┐
 0  │             │
 4  │    ENAME    │
    ├─────────────┤
 8  │     EP ADDR │
    ├──────┬──────┤
12  │ LEIND │      │
    └──────┴──────┘
```

0-7	ENAME	Entrypoint name
8-11	EPADDR	Entrypoint address
12	LEIND	Last entry indicator X'80' Last entry X'00' Not last entry

The ENTAB is created by the program loader and appended to the end of every module loaded. Each entry contains the name and entrypoint for a CSECT in the module. This table is used when processing unresolved adcons.

NAME/PCB

```
    ┌─────────────┐
 0  │             │
 4  │ MNAME       │
    ├──────┬──────┤
 8  │ LEVEL│ ENTAB│
    ├──────┴──────┤
12  │    LCOM     │
    ├─────────────┤
16  │    EPADDR   │
    ├─────────────┤
20  │    PCBCHN   │
    └─────────────┘
```

0-7	MNAME	Program name
8	LEVEL	Segment level assigned to module
9-11	ENTAB	Address of the entry table
12-15	LCOM	Length of BLANK COMMON CSECT for this module
16-19	EPADDR	Module entry point
20-23	PCBCHN	Pointer to next active PCB, zeroes if last PCB

The NAME control block is resident in the PLAN mainline. It is used as a work area by the program loader and is the skeleton used to construct the PCB which is appended to the beginning of every module loaded.

ERTAB1

```
    ┌─────────────┐
 0  │             │
 4  │    CNAME    │
    ├───────┬─────┤
 8  │ ESDNUM│     │
    ├───────┴─────┤
12  │    CHAIN    │
    └─────────────┘
```

0-7	CNAME	Name of external reference
8-9	ESDNUM	ESD number assigned by linkage editor to this external reference
12-15	CHAIN	Pointer to next ERTAB1, zero if last

An ERTAB1 control block is created by the program loader when a Type 2 ESD entry is processed. This table is used to identify the name of an external reference when an unresolved adcon is found during RLD processing.

ERTAB2

```
    ┌───────────────┐
 0  │               │
 4  │     CNAME     │
    ├──────┬────────┤
 8  │ LADC │ ADCADDR│
    ├──────┴────────┤
12  │     CHAIN     │
    └───────────────┘
```

0-7	CNAME	Name of external reference
8	LADC	Length in bytes of adcon
9-11	ADCADDR	Core address of adcon
12-15	CHAIN	Pointer to the next ERTAB2, zero if last

An ERTAB2 control block is created by the program loader during RLD processing when an adcon is found that cannot be resolved. These control blocks are processed by the final cleanup linkedit in the PLAN mainline.

LOCAL SAVE AREA

```
     ┌─────────┐
0-63 │ LOCREGS │
     ├─────────┤
  64 │ LPARMAD │
     ├─────────┤
  68 │ LPICADR │
     ├─────────┤
  72 │ LCURPCB │
     ├─────────┤
  76 │ LCURTOP │
     ├─────────┤
  80 │ LCURBTM │
     ├─────────┤
  84 │ LOCCHNS │
     ├─────────┤
  88 │ LCURLEV │
     └─────────┘
```

0	LOCREGS	Register save area
64	LPARMAD	Callers argument list address
68	LPICADR	Address of caller's pica element
72	LCURPCB	Address of the current PCB
76	LCURTOP	Top of managed free storage
80	LCURBTM	Bottom of managed free storage
84	LOCCHNS	Address of next LOCAL save

		area
88	LCURLEV	Current execution level

A LOCAL save area is part of a level control block but is only used when the LOCAL facility is invoked.

LCB

```
      ┌───────────────┐
0-3   │  CHAINADR     │
      ├──────┬────────┤
4-7   │ LEVEL│ UNUSED │
      ├──────┴────────┤
8-99  │  LSA          │
      └───────────────┘
```

0	CHAINADR	Address of next LCB zero if last
4	LEVEL	Level number assigned to this segment
8	LSA	LOCAL SAVE area

An LCB (Level Control Block) is created whenever a new segment is loaded or a CALL LOCAL occurs and a LSA is not available.

DAFB

```
         ┌──────────────────────────────┐
0-103    │              DCB             │
         ├──────────────────────────────┤
104-107  │    CHAINADR                  │
         ├──────┬────────┬────────┬─────┤
108-111  │STATUS│        │LGDRVNUM│LGFILNUM│
         ├──────┴────────┼────────┴─────┤
112-115  │           DECBCHN            │
         ├────────────┬───────┬─────────┤
116-119  │   NDECBS   │       │NRECTRK  │
         ├────────────┴───────┴─────────┤
120      │           FILWLGTH           │
         ├──────────────────────────────┤
124      │           FILRLGTH           │
         └──────────────────────────────┘
```

0-103	DCB	A standard BDAM DCB
104	CHAINADR	A pointer to the next control block in the chain, zero if last.
108	DTATUS BYTE	X'80' Look ahead file type X'40' Record used indicator X'04' Write operation X'02' Locate mode
110	LGDRVNUM	Logical drive code
111	LGFILNUM	Logical file number
112	DECBCHN	A pointer to the chain of DECB's attached to this file.
116	NDECBS	Number of DEC FILE.
116	NDECBS	Number of DECB's for this file.
119	NRECTRK	Number of record/track for this file.
120	FILWLGTH	Number of words in file
124	FILRLGTH	Number of records in file

A DAFB is created at initialization time for all direct access files defined by PLSYSTAB, PLMANFIL, PLCHKPT, PLANDRVx, PLFSYnnn DD cards.

SFB

```
         ┌─────────────────────────┐
0-95     │          DCB            │
         ├─────────────────────────┤
96-99    │         CURREC          │
         ├──────┬────────┬─────────┤
100-103  │PCCHAR│CALLTYPE│RECSIZE  │
         ├──────┴────────┴─────────┤
104-107  │        CHAINADR         │
         ├──────┬────────┬─────────┤
108-111  │STATUS│STATUSA │FILENUM  │
         ├──────┴────────┼─────────┤
112-115  │LINECNT        │PAGELGTH │
         ├───────────────┴─────────┤
116-119  │        NEXTREC          │
         ├─────────────────────────┤
120-123  │        EOBADDR          │
         ├─────────────────────────┤
124-127  │        CURRDECB         │
         ├─────────────────────────┤
128-131  │        NEXTDECB         │
         └─────────────────────────┘
```

0-95	DCB	Standard BSAM DCB
96	CURREC	A pointer to the current record
100	PCCHAR	ASA carriage control character for next output record
101	CALLTYPE	X'80' PLOUT CALL X'20' PLINP CALL
102	RECSIZE	Logical record size
104	CHAINADR	Pointer to next SFB, zero if last
108	STATUS	Status byte X'80' PLOUT occurred X'20' PLINP occurred X'10' Carriage control allowed X'08' Physical EOF occurred X'04' Logical EOF occurred
109	STATUSA	Status byte X'80' PLOUT device X'20' PLINP device X'02 Double buffering used X'01 Prime required
110	FILENUM	File ID number 1-255
112	LINECNT	Current line counter for files with RECFM FA, FBA
114	PAGELGTH	Number of lines per page
116	NEXTREC	Address of next available record area
120	EOBADDR	Address of end of the current block
124	CURRDECB	Address of the current DECB
128	NEXTDECB	Address of the next available DECB

A SFB is created at initialization time for all data sets defined by PLINP, PLOUT and PLSEQ DD cards.

DOS CONTROL BLOCKS

DAFB DOS

```
         ┌──────────┬──────────┐
 0-3     │DBLKSI    │DLUNIT    │
         ├────┬─────┼────┬─────┤
 4-7     │DCYL│DHEAD│DNET│DNTC │
         ├────┴─────┴────┴─────┤
 8-11    │      FILRLGTH       │
         ├─────────────────────┤
12-15    │      FILWLGTH       │
         ├───────┬─────────────┤
16-19    │DFILNUM│             │
         ├───────┴─────────────┤
20-23    │       DCHAIN        │
         └─────────────────────┘
```

0	DBLKSI	Physical record length
2	DLUNIT	DOS logical unit assignment
4	DCYL	Starting cylinder number of the extent
5	DHEAD	Starting head number of the extent
6	DNRT	Number of records per track
7	DNTC	Number of tracks per cylinder
8	FILRLGTH	Number of records in file
12	FILWLGTH	Number of words in file
16	DFILNUM	Drive code and file number
20	DCHAIN	Address of next DAFB

The DAFB is built by the direct access open routine $$BDFJDO and is used by the subroutine DFJDIOCS to process all direct access files.

DOS SFB

```
         ┌──────┬────────┬──────────────────┐
 0-3     │CTRLOP│        │SLUNIT            │
         ├──────┼────────┼───────┬──────────┤
 4-7     │      │CALLSAV │MODESET│PCCHAR    │
         ├──────┴────────┴───────┴──────────┤
 8-15    │            DATA CCW              │
         ├──────┬────────┬──────────────────┤
16-19    │STATUS│STATUSA │RECSIZE           │
         ├──────┴────────┴──────────────────┤
20-23    │            CURREC                │
         ├───────────────┬──────────────────┤
24-27    │BLKSIZE        │RECLGTH           │
         ├───────────────┼──────────────────┤
28-31    │PAGELGTH       │LINECNT           │
         ├───────────────┴──────────────────┤
32-35    │            NEXTREC               │
         ├──────────────────────────────────┤
36-39    │            EOBADDR               │
         ├──────────────────────────────────┤
40-43    │            CURRBRF               │
         ├──────────────────────────────────┤
44-47    │            NEXTBUF               │
         └──────────────────────────────────┘
```

0	CTRLOP	A control CCW used for carriage control
2	SLUNIT	DOS logical unit assignment
5	CALLSAV	Call type indicator X'01' PLOUT call X'02' PLINP call
8	DATACCW	R/W CCW
16	STATUS	Status byte X'80' Device is a printer X'40' Device is a card reader X'20' Device is a tape X'10' Device is a 1052 X'08' Physical EOF occurred X'04' Logical EOF occurred X'02' PLINP call occurred X"01' PLOUT call occurred
17	STATUSA	Status bytes X'80' WAIT required on device X'40' Prime required on buffers X"20' Stacker select required X'10' Carriage control suppressed X"08' Carriage control allowed X"04' Double buffering used X'02' PLINP device X"01' PLOUT device
18	RECSIZE	Logical record length
20	CURREC	Address of current record area
24	BLKSIZE	Physical block size
26	RECLGTH	Physical record length
28	PAGELGTH	Number of lines per page
30	LINECNT	Current line counter
32	NEXTREC	Address of next available record area
36	EOBADDR	Address of end of current block
40	CURRBUF	Address of current block
44	NEXTBUF	Address of next available block

This SFB is created by the sequential file open routine $$BDFJSO and is used by the subroutine DFJSIOCS to process sequential files.

IOCB

```
        ┌──────┬────┬───────┐
 0-3    │MLUNIT│MNOD│MSTATUS│
        ├──────┼────┴───────┤
 4-7    │MBLKSI│MLRECL      │
        └──────┴────────────┘
```

0	MLUNIT	DOS logical unit assignment
2	MNOD	PLAN device code of rile number
3	MSTATUS	Status byte X"40' DYNAMIC drive file X"20' PERMANENT drive file X'10' Sequential file X'08' ASA carriage control required X"04' Double buffering required X'02' Open indicator X"01' End of type indicator
4	MBLKSI	Physical record length
6	MLRECL	Logical record length

The IOCB control blocks are assembled into the module DFJIOCBS and may be altered by PLAN run control cards at initialization time.

DIRECT ACCESS FILES

On OS PLAN the BDAM access method is used to process direct access files. Files are opened by the initialization routine using the standard OS OPEN macro. The subroutine DFJDIOCS is a section of the PLAN module DFJLODER.

On DOS PLAN the EXCP macro is used to process direct access files. Files are opened by the PLAN transient module $$BDFJD0. The subroutine DFJDIOCS is placed in core by the initialization routine.

The subroutine DFJDIOCS processes read/write requests. It will handle both 'locate' and 'move' mode operations. A file is treated as a byte-addressable string of characters. This facilitates processing for the word addressing used by the PLAN file support subroutines. 'Move' mode requests may address a file on a byte boundary. 'Locate' mode may only use record addressing.

On OS and DOS the file layout and processing of the phrase dictionary and the managed area save file are identical. The checkpoint files have different formats.

CHECKPOINT PROCESSING

An OS checkpoint record is a 256-byte header followed by the active program area. The header is in the following format.

0-63	Register save area
64-67	Length of the program area
68-91	Name control block
92-95	Word displacement of previous checkpoint
96-99	Word displacement of current checkpoint
100-103	Address of the program area
104-107	Address of the current PCB
108-115	Managed free storage limits
116-119	Address of the chain of LOCAL save areas
120	Current execution level
128-256	Bootstrap program
256--	Program area

The LCHEX subroutine writes the checkpoint. On a checkpoint reload, the header is read into a work area and then control is transferred to the bootstrap to reload the program area and restore the system status.

On DOS each module is written out separately and contains a 20-byte header as follows:

0-7	Module name
8-11	Module origin address
12-15	Module end address
16-19	Address of LCHEX restore E.P.

The LCHEX subroutine contains the register save area and other work areas to save the system status. The LCHEX subroutine will only write out the module it is linkedited with. On a checkpoint reload, the header is read and the module reloaded and then control is returned to the LCHEX subroutine to restore the system status.

DYNAMIC FILE PROCESSING

A PLAN DYNAMIC drive contains formatted records of 600 bytes. The formatting must be done by the initialization routine for OS and the program DFJINIT for DOS. The record layout for both systems is identical.

A DYNAMIC drive is logically split into segments of 10 records each. Three control records are kept in the file to control allocation of space within the drive. Two of these records (0,1) are VTOC records which contain pointers to all existing LOGICAL files within the drive. The third record (2) is the availability record which contains pointers to the free segments within the drive. When a file is opened (FIND) the required number of segments are extracted from the availability record and a FDR (File Description Record) is created for the file. This is always the first record in the file and contains pointers to the segments allocated to the file. A pointer to the FDR is kept in the VTOC records. On a READ or WRITE the FDR is used to locate the physical records to be read. When a file is released the space recorded in the FDR is returned to the availability record and the FDR and its pointer in the VTOC record are destroyed.

FILE LAYOUT DYNAMIC DRIVES

RECORD 0 VTOC RECORD

```
| |  |  |  |  |   |  |  |  | | |    |  |  |
|S|P1|P2|P3|P4|   |F1|F2|F3|•|•|F127|  |ID|
| |  |  |  |  |   |  |  |  | | |    |  |  |
└─┴──┴──┴──┴──┴───┴──┴──┴──┴─┴─┴────┴──┴──┘
 0 4  8  12 16 20  24 28 32     528    540
```

RECORD 1

```
|    |    |    |         |    |
|    |F128|F129|• • • •  |F255|
└────┴────┴────┴─────────┴────┘
 0   20                  528
```

S This field contains the count of the total number of available segments in the drive.

P1-P4 Contains the number of segments allocated at priority 1 to priority 4 respectively.

F1-F255 Contains a pointer to the first record of LOGICAL files 1-255 respectively. This field contains zero if the file does not exist. The pointer is in the following format.

Byte 0 File priority

Byte 1-3 Relative record number of the FDR for the file.

ID Is a four-byte ID field used to validate the file.

RECORD 2 AVAILABILITY RECORD

The availability record contains pointers to all the free space in the file. These are a string of pointers to the free extents within the drive.

```
|  |  |  |  |   |   |   |   |  |  |
|S1|E1|S2|E2| • | • | • | • |SN|EN|
└──┴──┴──┴──┴───┴───┴───┴───┴──┴──┘
0  2  4  6  8   10  12
```

S1-SN Is the starting segment number of a free extent

E1-EN Is the ending segment number of a free extent

A high-order bit in the S1 field indicates the end of the string.

FDR File Description Record

The FDR record is the first record of every logical file

```
| | | |  |  |  |  |  |  | | |  |  |
|P|R|W|S1|E1|S2|E2|S3|E3|•|•|SN|EN|
└─┴─┴─┴──┴──┴──┴──┴──┴──┴─┴─┴──┴──┘
 0 1 4 8  10 12 14 16 18 20
```

P Is the file priority

R Is the relative record number of this record

W Is the number of words in the file

S1-SN The starting segment number of an extent allocated to this file

E1-EN The ending segment number of an extent allocated to this file.

FILE RECORD LAYOUT

PFILE LAYOUT

The PLAN language definition file (PFILE) is generated and maintained by the DFJPHRAS logic module and is utilized by PLAN (loader) and DFJPSCAN for temporary system save areas. PFILE is required to be present before a PLAN execution is permitted.

PFILE is defined as a logical file containing a minimum of 14 (17 on the 1130) and a maximum of 268 (271 on the 1130) records. Records in PFILE are fixed in length at 512 bytes on System/360. On the 1130 each record is 320 (16-bit) words in length. The following table lists the contents of PFILE.

ITEM NAME	SIZE IN RECORDS	RECORD DISPLACEMENT	DESCRIPTION
PFLDRSV	5	0	Error stack area
PFSYMT4	1	5	Level 4 symbol table save area (128 words)
PFINPUTB	1	6	Card image residual save area (20 words)
PFSYMT3	1	7	Level 3 symbol table save area (128 words)
PFPWVTAB	1	8	Phrase-verb validity table (512 bytes)
PFSYMT2	1	9	Level 2 symbol table save area (128 words)
PFINPUTA	1	10	Current statement image save area (114 words)
PFSYMT1	1	11	Level 1 symbol table save area (128 words)
PFPAVTB	1	12	Phrase entry availability table (512 bytes)
PFPETAB	1-255	13	Phrase entry table

The following section describes the functions of each of the areas listed in the above table of contents:

PLFDRSV On OS and DOS PLAN systems the area is used as a temporary stack area for diagnostics awaiting processing by the system error module when a stacked mode of operation is indicated.

PFSYMT4 This area is used to store the level 4 symbol table. The symbol table must be saved for use in initializing the symbol table of a blank-level command following a level 4 command.

PFINPUTB The image of the card, to the right of the semicolon terminating a command, is saved in this area for processing as the start of the following command. (Hexadecimal 00 indicates the end of the image.)

PFSYMT3 This area is used to store the level 3 symbol table. The symbol table must be saved for use in initializing the symbol table of a blank-level command following a level 3 command or the symbol table for a level 4 command following this level 3 command without intervening commands of level 3 or higher.

PFPWVTAB This table is used as an expedient to determining phrase validity. There are 256 entries corresponding to the 256 possible phrase check sums. A zero entry indicates no valid phrase has the check sum; a nonzero entry is a pointer to the phrase entry table.

PFSYMT2 This area is used to store the level 2 symbol table for use in initializing the symbol table of a blank-level command following a level 2 command or the symbol table of a level 3 command following this level 2 command without an intervening command of level 2 or level 1.

PFINPUTA This area is used to store the length and the EBCDIC image of the current phrase. DFJPSCAN places the command in this area for access by DFJPHRAS. The subroutine INPUT reads the statement image from this area and places it in memory.

PFSYMT1 This area is used to store the level 1 symbol table for use in initializing the symbol table for a blank-level command following this level 1 command or the symbol table for a level 2 command following this level 1 command without an intervening level 1 command.

PFPAVTB There is one entry in this table for each record in the phrase entry table. The entry provides information as to the available room within each record for the addition of new phrase definitions.

PFPETAB This portion of the PFILE contains the language description elements. Each command is entered with header information followed by up to seven tables of phrase definition data. The length of this section is variable up to a maximum of 255 records, a function of the number of commands that must be added into the language dictionary.

The following section describes the detail layout of the variable (maintained) portions of PFILE. Those portions that are merely temporary storage areas are not described.

PFPWVTAB (PHRASE-VERB VALIDITY TABLE)

This section has 256 entries corresponding to the 256 possible phrase check sums. The word check sum of each word in the phrase is calculated as:

```
KSUM = L1*4 + L2*2 + L3
  L1 = First letter in EBCDIC in low-
       order eight bits
  L2 = Second letter in EBCDIC in low-
       order eight bits
  L3 = Third letter in EBCDIC in low-
       order eight bits
```

Only the low-order eight bits of the word check sum are saved. The phrase check sum is formed by the "exclusive or" of succeeding word check sums. The following example illustrates the calculation of the phrase check sum for the phrase "DUMP PLAN":

Word Check Sum Calculations

```
D      11 0001 0000     310
U      01 1100 1000     1C8
M      00 1101 0100     0D4
      101 1010 1100     5AC      AC

P      11 0101 1100     35C
L      01 1010 0110     1A6
A      00 1100 0001     0C1
      101 1100 0011     5C3      C3

DUM       1010 1100              AC
PLA       1100 0011              C3
          0110 1111              6F
```

The 256 entries accessed by the phrase check sum have the following format. Each entry contains 16 bits. The term "record/64" in the following discussions means 64 bits on System/360 and 80 bits on the 1130 System. This grouping is one sixty-fourth of a disk record.

```
         | |          |         |
Contents |V|    A     |    B    |
         └─┴──────────┴─────────┘
Bit       0 1 2      7 8      15
```

V Verb Control
0 if no verb phrase has this check sum
1 if a verb phrase has this check sum

A The number of records/64 from the beginning of the sector indicated by B to the first phrase entry in the chain.

B Those bits contain the relative sector address (1-255) of the first phrase entry in the chain of phrases with equal check sums. The field is zero if no valid phrase has this check sum.

PSYMT 1,2,3,4 (SYMBOL TABLES)

This section is made up of 255 bytes of information, including 126 (16-bit) words containing the symbol table entries. The format of the table is shown in the following chart:

```
         | | | | |          |
Contents |0|R|L|0|    E     |
         └─┴─┴─┴─┴──────────┘
Byte      0 1 2 3 4        255
```

R The relative byte (8-bit) address of the first table entry. The tables are built from left to right. The rightmost entry wraps around to the left end. The last (rightmost) value entered is preceded to the right by a zero entry.

L The level of the symbol table is indicated as the level minus one. Thus, the indicator occupies the second and third bits and ranges from 0-3.

E Each symbol is entered in compressed form from the phrase. The table is initialized from the symbol table of the next higher level. The format of the compressed symbol is shown in the chart below. The symbol allows expeditious detection of undefined symbols. Note that the symbol table entry is the same as 1 and 2 of Table 3.

Contents	Letter 1	Letter 2	Letter 3	0
Bit	0 4	5 9	10 14	15

The letters are compressed into five bits through the following code compression:

LETTER	COMPRESSED CODE
A-I	1-9
J-R	11-19
S-Z	22-29
blank	0

PFPAVTB (PHRASE AVAILABILITY TABLE)

This section of PFILE contains a maximum of 256 entries corresponding to the number of records in PFPETAB. Each entry is a halfword (16 bits). The entry format is shown in the following table:

Entry	B	L
Bit	0 7	8 15

B The number of records/64 to the beginning of the first phrase entry or available space entry in the sector. The value of 7FFF (hexadecimal) indicates that the entire sector is available; 8000 (hexadecimal) indicates the end of the table.

L The number of records/64 in the largest contiguous, available block that begins in this sector. This entry is used as a test for the possible addition of the current phrase into this sector.

PFPETAB (PHRASE ENTRY TABLE)

The available space entries and the phrase entries in the phrase entry table are packed across sector boundaries. The first records/64 of the phrase entry table must be initialized when PLAN is invoked. If it is not, the ADD PHRASE command is set and PHRAS is loaded to add it to PFILE. The format of the PFILE header is shown below in hexadecimal.

	P	F	I	L	E	•
0001	D7	C6	C9	D3	C5	4B
0-15	16-23	24-31	32-39	40-47	48-55	56-63

Note that bits 16 to 63 contain the EBCDIC representation of PFILE. On the 1130 System, bits 64-79 are included but unused.

The first word (32 bits) of each phrase (or available space) entry provides data as to the size of the entry and pointers to the next item in the chain. The format of this portion of the entry is provided below:

T	L	X	000	S	V	Z	SA
0	1 3	4	5 7	8 15	16	17 23	24 31

T This bit determines whether this is a phrase entry or an available-space entry.
0 = Phrase entry
1 = Available space (The following fields, except S, are meaningless if this is an available-space entry.)

L These bits (in a phrase entry) define the level of the phrase according to the following table:
000 Level 1
001 Level 2
010 Level 3
011 Level 4
100 Blank level

X The presence of this bit indicates a level zero phrase.

S These eight bits define the number (<128) of records/64 in this entry. No phrase may result in an entry of greater than 128 records/64. The appropriate diagnostic is issued if such an attempt is made.

V This bit (in a phrase entry) defines whether the phrase is a verb or an object phrase.
0 = Object phrase
1 = Verb phrase

Z This six-bit (<64) field defines the number of records/64 (within the sector) that precede the first word of the chained-to (phrase with equal check sums) entry. This entry and the following entry allow direct access of the chained phrase.

SA This eight-bit field (<256) defines the sector address, relative to the first record of the phrase entry table minus one word, of the first word of the next chained-to phrase. This field is zero if this phrase is the last of a chain.

Note that all phrases of equal check sum (as defined under phrase-verb validity table) make up the links of the phrase chain.

Following the phrase entry header, as defined above, are up to eight tables. Each table is ended with 80xx (hexadecimal), where xx is the number of 16-bit half-words in the following table. The last table is terminated with 7FFF (hexadecimal). Trailing tables of zero length are not required, nor is the table length indication (8000) entered.

TABLE 1 (PHRASE NAME)

One word (32 bits) is required for each word in the phrase name. There is a maximum of five double-words used. Letters are coded in EBCDIC code.

```
|        |        |        |          |
|Letter 1|Letter 2|Letter 3|(Not Used)|
└────────┴────────┴────────┴──────────┘
 0      7 8     15 16    23 24       31
```

Note that the next table (80xx) or last table (7FFF) indicator is placed in the next half-word.

TABLE 2 (CONSTANT INITIALIZATION DATA VALUES)

This table contains all constant (default or initialization) values. There are four formats for this entry that depend upon the format of the phrase definition. In the following table definitions, the example phrase entry is given, followed in order by the general form of tle table entry, the description of the table, and the table entry representing the example phrase entry. Note that there is one entry required for each literal character count plus one for each succeeding group of four literal characters.

1. Constant Value: I(35)10,

```
| | |     |     |
|0|0| S   |  V  |
└─┴─┴─────┴─────┘
 0 1 2 15 16 47
```

S This 14-bit (<16,384) field defines the subscript relative to the beginning of the switch area.

V This 32-bit field defines the initialization value as defined in the phrase entry.

```
| | | |  |  |  |  |  |  |  |  |  |
|0|0|2|D |0 |0 |0 |0 |0 |0 |0 |A |
└─┴─┴─┴──┴──┴──┴──┴──┴──┴──┴──┴──┘
 0 4 8 12 16 20 24 28 32 36 40 44
```

2. Symbolic Subscript: I(M)DATA3,

```
| |       | |        |          |
|1|  C    | 0|   S    |    V     |
└─┴───────┴─┴────────┴──────────┘
 0 1    15 16 17   31 32       63
```

C This 15-bit field contains the compressed data name in symbol table code that is to be initialized. The symbol is stored in the same compressed code as defined for the symbol table entries.

S This 15-bit field contains the subscript relative to the data name into which the initialization value is stored.

V This 32-bit field defines the initialization value as defined in the phrase entry.

```
| | | |  |          |            |
|9|0|3| 7|  0001    |  00000003  |
└─┴─┴─┴──┴──────────┴────────────┘
 0 4 8 12 16     28 32          63
```

3. Implied DO: I(30,36,2)15,....

```
| | |       |        |        |     |
|0|1|  S    |   D    |   I    |  V  |
└─┴─┴───────┴────────┴────────┴─────┘
 0 1 2   15 16    31 32    47 48 79
```

S This 14-bit (<16,384) field contains the subscript associated with the data value relative to the beginning of the switch area.

D This 16-bit field contains the displacement (range) for the implied DO. The value must be a multiple of field I. This value is computed from the first two specified implied DO parameters.

I This 16-bit field contains the increment for the implied DO.

V This 32-bit field contains the initialization value as defined in the phrase entry.

```
| | | | |            |      |         |
|4|0|2|8 |   0006    |0002  |0000000F|
└─┴─┴─┴──┴───────────┴──────┴────────┘
 0 4 8 12 16       31 32 47 48    79
```

4. Symbolic Subscript and Implied DO: (M+2,10,2)NAME1,...

```
| |      | |       |     |      |
|1| CS   | 1|  D   |  I  |  V   |
└─┴──────┴─┴───────┴─────┴──────┘
 0 1  15 16 17  31 32 47 48 79
```

CS This field contains the compressed data name of the starting position to be initialized. The symbol is stored in the same compressed code as defined for symbol table entries.

V This 32-bit field contains the initialization value defined in the phrase entry.

D This 16-bit (<65,536) field contains the displacement from the first position to be initialized to the final position to be initialized.

I This 16-bit field contains the increment between succeeding values to be initialized.

```
| |    |     |         |          |
|B| C2E| 800A|  0002   | 00000001 |
└─┴────┴─────┴─────────┴──────────┘
 0     16    32        48        79
```

TABLE 3 (SYMBOL TABLE)

1. Symbol with Constant Subscript and Scale Value: P+2(15)ABC...

```
|    | |      | |      | |         |
| S  |0|  E   |I|  P   |G|   SUB   |
└────┴─┴──────┴─┴──────┴─┴─────────┘
 0 14 15 16-17 18 19-21 22 23     31
```

S This 15-bit field contains the compressed data name to be defined. The format is as defined above for symbol tables.

E This field defines the user-exit number to be associated with this symbol.
00 = No exit
01 = User exit 1
10 = User exit 2
11 = User exit 3

I This field defines the mode for the variable.
0 = Real (floating-point)
1 = Integer (fixed-point)

P This three-bit (<8) field contains the scale factor to be associated with this symbol.

G This one-bit field determines the sign of the scale factor.
0 = Positive
1 = Negative

SUB This nine-bit (<512) field contains the subscript of the value to be entered in the symbol table relative to the first position of the communication array.

```
| | | |  |  |  |  |     |
|0|8|8| 6| 0| 8| 0|  F  |
└─┴─┴─┴──┴──┴──┴──┴─────┘
 0 4 8 12 16 20 24 28 31
```

2. Symbol with Constant Subscript and No P-value: IU2 (25)VALUE...

```
|      |    |       |    |        |
|  S   | 1  |   E   | I  |  SUB   |
└──────┴────┴───────┴────┴────────┘
 0    14   15  16   17  18  19    31
```

S This 15-bit field contains the compressed data name in the mode indicated for symbol table entries.

E This two-bit field defines the user-exit number to be associated with this data name.

I This one-bit field determines the mode of storage.
0 = Real (floating-point)
1 = Integer (fixed-point)

SUB This 13-bit (<8192) field contains the subscript associated with the data name relative to the switch area.

```
| | | |  |  |  |  |       |
|C|8|5| B| A| 0| 1|   9   |
└─┴─┴─┴──┴──┴──┴──┴───────┘
 0 5 8 12 16 20 24 28   31
```

3. Symbols with Symbolic Subscript: (M+2-N)ABC...

The symbolic subscript is indicated by setting SUB to zero. The subscript defining expression is then appended to the symbol table entry in EBCDIC code with a prefixed left parenthesis and a terminating comma.

```
|          |                  |
| 08860000 | 4DD44EF260D56B   |
└──────────┴──────────────────┘
 0         32                87
```

TABLE 4 (PROGRAM LIST)

The program list table is made up of one entry per program in the list.

1. Program Name: M0798,...

```
|                              |
| 8-CHARACTER EBCDIC NAME      |
| (RIGHT-PADDED WITH BLANKS)   |
└──────────────────────────────┘
0                            63

| | | | | | | | | | | | | | | | |
|D|4|F|0|F|7|F|9|F|8|4|0|4|0|4|0|
└─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴─┘
0       16 17 31 32           60
```

2. Checkpoint Return (asterisk)

```
|                   |
| 5C40404040404040  |
└───────────────────┘
 0                64
```

3. Left Parenthesis (EBCDIC)

```
|                   |
| 4D40404040404040  |
└───────────────────┘
 0                64
```

4. Right Parenthesis (EBCDIC)

```
|                   |
| 5D40404040404040  |
└───────────────────┘
 0                64
```

TABLE 5 (DATA CHECK ENTRIES)

1. Test, Abort, Generate PLAN Literal: (5)*,...

```
| |   |            |          |
|0| * |    SUB     |   CTL    |
└─┴───┴────────────┴──────────┘
 0 1 2 3       15 16       31
```

* This two-bit field contains the condition code.
00 = *
01 = *R
10 = *T
11 = *F

SUB This 13-bit (<8,192) field contains the subscript relative to the switch area of the PLAN word to be tested.

CTL If this field is nonzero, there is a suffix section, as defined under 4 and 5, starting at field "F".

```
|    |    |    |       |
|  0 |  0 |  0 |   5   |
└────┴────┴────┴───────┘
0    4    8   12   15
```

2. Test, Abort, Generate PLAN Literal; Symbolic Subscript: (M)NAME*R,...

```
| |   |         | |           |          |
|0| * |  -0-    | 0|   SYM    |   CTL    |
└─┴───┴─────────┴─┴───────────┴──────────┘
 0 1  2 3     15 16 17      31 32       47
```

* (See above)

SYM This 15-bit field contains the compressed data name in the format as defined for symbol tables.

CTL (See above)

```
|    |    |    |     |    |    |    |    |
|  2 |  0 |  0 |   0 |  3 |  C |  2 |  E |
└────┴────┴────┴─────┴────┴────┴────┴────┘
0    4    8   12    16   20   24   28
```

3. Same conditions as above:
Same as previous example, plus: ,*F

```
| |   |       | |        |        |        |
|0| * | -0-   | 1| SYM   |  SUB   |  CTL   |
└─┴───┴───────┴─┴────────┴────────┴────────┘
 0 1 2 3   15 16 17   31 32    47 48    63
```

* (See above)

SYM (See above)

CTL (See above)

SUB This 15-bit (<32,768) field contains the subscript relntive to the data name that is to be checked.

```
| | | |  |  |  |  |  |  |  |  |  |
|6|0|0| 0| B| C| 2| E| 0| 0| 0| 2|
└─┴─┴─┴──┴──┴──┴──┴──┴──┴──┴──┴──┘
 0 4 8 12 16 20 24 28 32 36 40 44
```

Note: In the following examples the formats defined in 1, 2, or 3 above remain the same as a function of conditions except for bit 0 and the last 15-bit field. Bit 0 will indicate whether the literal to be processed is implicit (1) or explicit (0). The last 15-bit field will contain function information for the literal processing.

4. Process Implicit Literal: ( )*TZ(9)

Note: Z in the above example is a user-given function code and will be reflected in the F field below according to the following table.

If Z = A (Abort) then F = 00
  = C (Continue)   = 01

```
     = P (Phrase)        = 11
     = b (List)          = 10
```

```
| |                   |   |     |
|1| SAME AS 1, 2 OR 3 | F | SUB |
L-L-------------------L---L-----J
 0 1
```

F See above table.

SUB This 14-bit (<16,384) field contains the subscript relative to the start of the communication array that contains the literal to be processed.

5. Process Explicit Literal: ( )*TZ'LITERAL'

```
| |                   |   |      |     |
|0| SAME AS 1, 2 OR 3 | F | L    |  Q  |
L-L-------------------L---L------L-----J
 0 1                 n 0 1   15 16   n
```

F Same as example 4.

L This 14-bit field contains the length of the literal in 16-bit words.

Q This variable-length field contains the literal in EBCDIC packed format.

TABLE 6 (PHRASE-DEFINED EXPRESSIONS)

This table is made up of two sections. The following three examples define the format of the possible first-section entries:

1. Value with Scale Factor: P+3(7) A=A*.017453...

```
|     |    |      | |        |        |
| 1 0 | I  |  P   |G|   S    |   T    |
L-----L----L------L-L--------L--------J
0    1 2   3     5 6 7     15 16      n
```

I This field designates the storage mode of the data value.
0 = Real (floating-point)
1 = Integer (fixed-point)

P This three-bit (<8) field designates the scale factor to be applied to the result of the expression before storage.

G This bit designates the sign of the scale factor.
0 = Positive
1 = Negative

S This nine-bit (<512) field contains the subscript associated with the data value relative to the first position of the communication array.

T This variable-length field contains the text of the phrase-defined expression terminated with a comma. The text is compressed to eliminate meaningless blanks and characters.

```
|  |  |   |   |   |   |   |   |         |
|8C|07| C1| 7E| C1| 5C| 4B| F0| F1...   |
L--L--L---L---L---L---L---L---L---------J
 0  8  16  24  32  40  48  56  64
```

2. Values without Scale Factors: I(12) I=I*12...

```
|    | |         |         |
| 11 |I|    S    |    T    |
L----L-L---------L---------J
 0  1 2 3      15 16       n
```

I See above.

S This 13-bit (<8,192) field contains the subscript of the data value relative to the start of the systems switch area.

T See above.

```
|   |   |   |   |   |   |   |   |   |
| E0| 16| 7E| C9| 7E| C9| 5C| F1| F2|
L---L---L---L---L---L---L---L---L---J
 0   8   16  24  32  40  48  56  64
```

3. Value with Symbolic Subscript: (m+5) A,:(B>0)

```
|    |        | |         |        |
| 00 |   S    | 0|    C    |   T    |
L----L--------L-L---------L--------J
 0  1 2    15 16 17     31 32      n
```

S This 14-bit (<16,384) field contains the subscript relative to the data name into which the result of the expression evaluation is stored.

C This 15-bit field contains the compressed data name in the symbol table code.

T See above.

```
|  |  |   |   |   |   |   |   |   |   |
|00|02| 04| 00| 7A| 4D| C2| 6E| F0| 5D|
L--L--L---L---L---L---L---L---L---L---J
 0  8  16  24  32  40  48  56  64  72
```

The second portion of this table contains the expression area in compact literal form (excess blanks and characters eliminated). This portion of the table is introduced with a dollar sign ($).

TABLE 7 (USER-EXIT LIST)

This table is in a format identical to Table 4 and contains the program list defined following the keyword EXIT. The table, when present, always contains three entries.

TABLE 8 (VERB PROGRAM LIST)

This table is in a format identical to Table 4 and contains the program list defined following the term VERB at phrase definition time.

PLAN DYNAMIC FILE CONTROL BLOCK

The following charts provide the content of the PLAN DYNAMIC file control blocks. Note that because of the integer word size differences (16-bit versus 32-bit), the 1130 PLAN system has a different format from that of the System/360 OS or DOS PLAN. The table given below provides the format for the System/360 OS-DOS PLAN.

```
ID(1)                                 ID(2)
|   |          |        |        |          |
| 0 |   TTR    |   D    |   N    |    S     |
L___|__________|________|________|__________J
 0   1        20     23 24     31 32       63
```

TTR This 19-bit field contains the TTR of the FDR for this file.

D This three-bit (<8) field contains the logical drive code for this file.

N This 8-bit (<256) field contains the file identification number. This field is originally set by the user before issuance of the CALL FIND. All other fields within ID(1) are set as a result of CALL FIND or CALL WRITE operations.

S This 32-bit field contains the current size of the file in words.

PLAN PERMANENT FILE CONTROL BLOCKS

This section defines the format of the PERMANENT (GDATA, RDATA, WDATA) file control blocks. The file ID number is set by the user before issuing the CALL GDATA. All other fields are defined as a result of the CALL GDATA and are modified by CALL RDATA. Note that because of the integer word size differences (16-bit versus 32-bit), the 1130 PLAN system has a different format from that of the System/360 OS or DOS PLAN. The table given below provides the format for the System/360 OS/DOS PLAN.

System/360 OS-DOS PLAN

```
ID(1)                                       ID(2)
|           |       |        |        |        |
|    00     |  7F   |   D    |   N    |   S    |
L___________|_______|________|________|________J
 0         7 8    15 16    23 24    31 32     63
```

D This eight-bit (<8) field indicates the logical drive code as 0-7.

N This eight-bit (<64) field contains the number of the file.

S This 32-bit field contains the size of the file in 32-bit words.

SYSTEM/360 PLAN FLOWCHART NARRATIVES

The following flowchart narratives are intended to provide additional detailed information about the logic of the components of the PROBLEM Language Analyzer System. The labels used in the narratives are the same labels as are displayed above the upper left corner of the blocks on the component flowcharts and represent the identification field of the program source statements containing the represented logic. Additional useful information is given at the beginning of each identifiable program item.

BREAK

The BREAK subroutine may be called to separate four bytes of any FORTRAN word into the right-justified byte of a four-word integer FORTRAN array.

BRE250 The registers are saved and the argument list is accessed.

BRE270 The FROM word address is accessed.

BRE280 The word is divided into four bytes which are placed in the user-specified array.

BRE340 The registers are restored and processing is terminated by return to the calling program to the next executable statement.

$$BDFJD (DOS)

This is a DOS transient area routine which is invoked by the PLAN loader on a phrase abort error if the DUMP option is selected at initialization time.

$BA330 The base register is set and the argument for the dump are accessed.

$BA510-$BA810 The dump header line is constructed and printed.

$BA890-$BA1250 Thirty-two characters are formatted into hexadecimal and printed.

$BA1270-$BA1310 A test is made to check if the last line printed was at the end of the partition. If yes, transfer is to $BA1430. Otherwise, a test is made to suppress like lines and transfer is to $BA1250 to suppress and to $BA890 to print the next line.

$BA1430-$BA1470 The dump trailer is printed and control returned to the caller.

$BDFJDO (DOS)

This is a DOS transient area routine which performs initialization functions for all direct access files processed by PLAN.

$BD390 The callers argument list is accessed.

$BD450 Adcons within this routine are relocated.

$BD690 The volume label on the disk unit is read to locate the VTOC.

$BD770 The VTOC is searched for the specified file name.

$BD850 The count portion of the first track and record in the file is read to determine the block size.

$BD910 The subroutine DFJGMAIN is called to obtain core for a file control block.

$BD1070-$BD2050 The control block is completed and contains the DOS logical unit, the starting cylinder number, the number of records per track, the number of records in the file, and the number of words in the file.

$$BDFJI (DOS)

This is a DOS transient area routine which alter the COMREG area phase address and program address. These addresses are used by the FORTRAN I/O package to locate buffer areas.

$$BDFJSO (DOS)

This is a DOS transient area routine which performs initialization functions for all sequential files processed by PLAN.

$BS450 Adcons in this module are relocated.

$BS670-
$BS1330 The subroutine DFJGMAIN is called to get core for the control block and the buffers. If core is available, the control block is initialized and the first buffer area is set to blanks.

$BS1570-
$BS2910 The device type is validated. It must be a reader, printer, punch, or magnetic tape. The control block is completed and control is returned to the caller.

DFJCGET

This subroutine controls transmission from and to the buffer, character-by-character for the PLAN sequential conversion routines.

CGE390 A test is made to see if the field has been exhausted. If not, transfer is to CGE470; otherwise a pointer to a NULL character is set and control is returned to the caller.

CGE470 The field width counter is updated.

CGE550 A test is made to see if the current character is outside the record area. If yes, control is returned to the caller.

CGE650 The character position indicator is incremented.

CGE710 A test is made to check if the character position pointer is outside the range of the field or the record. If valid, transfer is to CGE750, otherwise, control is returned to the caller.

CGE750 A pointer to the actual buffer character is set and then control is returned to the caller.

DFJCNTRL (DOS)

This module controls the issuing of STXIT macros by the DOS PLAN system, and also provides the linkage to the TRACE routine.

CNT470 A standard STXIT macro is issued.

CNT490 If any other type of STXIT is requested, it is issued.

CNT370 A test is made to see if the TRACE facility has been invoked. If yes, this module exits to the DFJTRACE module; otherwise, control is returned to the caller.

DFJCRDIR (OS)

The DFJCRDIR subroutine provides for establishment of an in-core program directory to be utilized by PLAN to provide more efficient program loading.

DCD150 Registers are saved according to standard OS conventions. A base register is set.

DCD250 If there is not a current directory transfer is to DCD400.

DCD280 The current program directory is freed.

DCD400 The name list is initiated.

DCD420 The CREATE CORE DIRECTORY phrase is read.

DCD560 The list of program names is built.

DCD630 The program name list is sorted.

DCD850 A BLDL macro is issued for the program list.

DCD920 If the BLDL macro is executed properly, transfer is to DCD1040.

DCD940 A count is made of the BLDL's which are in valid form.

DCD1040 A GETMAIN is issued for the required core to contain the program directory.

DCD1150 The BLDL entries that are valid are moved to the directory.

DCD1280 A pointer is set to the top of free storage. The subroutine is terminated by return to the caller.

DFJCSET

This subroutine is a conversion interface routine for the PLAN subroutines PFOUT, PFIN, PIOUT, PIIN, and PEOUT.

CSE1200 The base registers are set including the base register for the conversion routine and the conversion buffer control routine DFJCGET.

CSE1700 The NOD argument is validated. If the NOD argument is valid, transfer is to CSE2400; otherwise, control is returned to the caller.

CSE2400 The buffer arguments are set into the DFJCGET routine for use by the conversion routines. These include the buffer address, the record length, and the address of the caller's save area.

CSE2700 Exit from this routine is directly to the called conversion routine.

DFJDIOCS (DOS)

This module handles all the direct access I/O requests for the DOS PLAN system. It handles requests in both the 'locate' and 'move' mode. All direct access files are processed as a byte-addressable string of characters. 'Move' mode requests may address any byte in the file. 'Locate' mode requests may only use record addressing.

DIO650 If this is not a write locate call, transfer is ti DIO810.

DIO690 The write request switch is set on for the last buffer read and control is returned to the caller.

DIO810 The caller's registers are saved and the user's argument registers including the displacement count and array address registers are updated.

DIO1010 If this is a call to quiesce all I/O, transfer is to DIO3550.

DIO1130 If this is not an overlay wait call, transfer is to DIO1290. Otherwise, the subroutine WAIT is called to issued a check on the last I/O operation and transfer is to DIO3390 to return control to the caller.

DIO1290 The I/O arguments are calculated from the user displacement count. This is done to see if a record read had to be done in order to mask in if the user displacement is not on the record boundary.

DIO1490 A request is made to see if the requested record is already in a buffer. If not, transfer is to DIO1770.

DIO1770 If this is a 'locate' mode call, transfer is to DIO2270.

DIO1870 This is a generalized routine that moves data to or from the user array. At completion of the move, transfer is to DIO3270.

DIO2270 The address of the current buffer is set in the user save area so that it will be returned to him in register 1 and transfer is to DIO3390 to return to the caller.

DIO2390 A test is made to see if the displacement argument is zero. If not, transfer is to DIO2990.

DIO2430 If this is a write operation, transfer is to DIO2850.

DIO2470 A test is made to see if the count argument is equal to the record length. If yes, transfer is to DIO2590.

DIO2510 A test is made to see if a physical record will fit in the buffer. If not, transfer is to DIO2590.

DIO2550 If this is not an overlap call, transfer is to DIO2990.

DIO2590 If this is not a 'move' mode call, transfer is to DIO2990; otherwise a read operation is initiated and the subroutine DOIO is called to start a read on the requested record.

DIO2710 If this is an overlap call, transfer is to DIO3270, otherwise; the subroutine WAIT is called to issue a check on the last I/O operation and transfer is to DIO3270.

DIO2850 A write operation indicator is set. If the count argument equals the record length, transfer is to DIO2650.

DIO3090 The subroutine SBUFREAD is called to read the record into the buffer.

DIO3190 If this is an overlay and locate call, transfer is DIO2270.

DIO3230 A wait is issued on the last I/O operation and transfer is to DIO1770.

DIO3270 A test is made to see if the count argument has been satisfied. If not, transfer is to DIO1290 to continue processing;

otherwise, control is returned to the caller.

DIO3550-DIO3830 This routine quiesces all I/O for the buffers. It checks if there has been a write request on either one of the buffers and if it has, the buffer is forced out.

DIO3870 This is the SBUF READ subroutine. A read operation is set and the physical I/O arguments are calculated, that is, the record number.

DIO3950 This is the DOIO subroutine.

DIO3950 A test is made to see if the requested record is in the file. If it is not, control is returned to the caller.

DIO3990 A wait is issued on the last I/O operation.

DIO4050 The CCW string for this I/O request is constructed.

DIO4690 The subroutine CCBSTART in the PLAN mainline is called to execute the I/O operation and control is returned to the caller.

DIO4810 This is the WAIT subroutine. A test is made to see if the last I/O operation was on the requested record. If not, control is returned to the caller. Otherwise, the subroutine CCWWAIT in the PLAN mainline is called to issue a wait on the I/O operation associated with the user record. Control is then returned to the caller.

DFJDLOAD, DFJSLOAD (DOS)

This subroutine is used to fetch a relocatable module to the highest address available in the PLAN partition. The subroutine DFJGMAIN is ised to allocate memory for the module.

SLO290 If this is a call to DFJDLOAD, transfer is to SLO590.

SLO350 A search is made of the PSCB table to check if the program is already in core. If not, transfer is to SLO590.

SLO510 The entrypoint of the program is placed in GPR1 and control is returned to the caller.

SLO610 The subroutine DIRLOOKU in the loader is called to search the DOS core image library directory for the named program.

SLO630 If the name was not found in the directory, transfer is to SLO1450.

SLO810 The subroutine DFJGMAIN is called to allocate core for the named module.

SLO890 If core is not available, transfer is to SLO1450.

SLO910 The PCB chain in updated to include the PCB for the program to be loaded.

SLO1370 The subroutine DLOADP in the loader is called to load the program.

SLO1410 The users registers are restored and control is returned to the caller. The entrypoint of the loaded program is placed in GPR1.

SLO1450 GPR1 is reset to indicate a failure to load and transfer is to SLO1410.

DFJDSLL

This routine manipulates a pseudo accumulator used by the floating-point conversion routines, DFJPFOUT, DFJPEOUT, and DFJPFIN.

DSL630 The entry counter for the shift left routine is stepped.

DSL650 A test is made to see if the accumulator is full, and if yes, control is returned to the caller.

DSL690 The accumulator is shifted left one position.

DSL950 A test is made again to see if the accumulator is full, if not, control is returned to the caller. Otherwise, the entry counter for the shift left routine is saved and then control is returned to the caller.

DSL1250 This is the entry to shift the accumulator right. Pointers to the significant portion of the accumulators and the shift counts are set.

DSL1330 A search is made to locate the first significant digit in the accumulator.

DSL1530 The accumulator is shifted right one position.

DSL1750 A test is made to see if the shift is complete. This test is on the shift count register. If nonzero, transfer is to DSL1330 to continue the shift, otherwise, control is returned to the caller.

DFJDUMP (OS)

This module is 'linked to' by the OS PLAN execution routine when a phrase abort occurs.

DUM1900 The callers registers are saved and a base register set.

DUM3000 The dump heading line is printed.

DUM3600 The failure ID line consisting of the error address, the current PCB, and execution level is printed.

DUM5900 The user's registers at abort time are printed.

DUM7100 The COMMON array is printed.

DUM9000 The active program area is printed.

DUM11000 The dump trailer is printed.

DUM11500 Control is returned to the caller.

DFJFMAIN (DOS)

This subroutine returns core to the queue which describes free core in the partition.

FMA270 The length of the request is rounded to double-word length.

FMA350 The free queue element chain is searched to locate the element to be used to receive the core to be released.

FMA590 The free queue element chain is updated to reflect the addition of the core area.

FMA830 A test is made to see if the core released was above and adjacent to the PSCB table. If it was exit from this routine is to the subroutine DFJMPSCB to move the PSCB table. Otherwise, control is returned to the caller.

DFJGMAIN

This subroutine allocates free core to the caller. The subroutine DFJMPSCB is called to move the PSCB table if required.

GMN270-
GMN570 A search is made of the free area chain to see if an area large enough is available. If not, transfer is to GMN750.

GMN630 The free area chain is updated.

GMN710 A return code and the address of the free area is set in GPR1 and control is returned to the caller.

GMN750 A test is made to see if space is available in the program area. If not, transfer is to GMN710 indicating no core found.

GMN950 The subroutine DFJMPSCB is called to move the PSCB table.

GMN970-
GMN1050 A test is made to see if any inactive programs in the program area were overlaid. If yes, the PSCB for these programs are marked as such and transfer is to GMN710.

DFJISET

This is the conversion interface routine for the core-to-core conversion routines, PCAI, PCAF, PCIA and PCFA.

ISE1100 The field width is calculated based on the mode of the subroutine called and the user's width arguments.

ISE1600 The buffer arguments are set into the DFJCGET routines for the conversion routines. These include the address of the buffer and the length of the buffer, plus the address of the caller's save area.

ISE2200 Exit from this routine is directly to the conversion routine.

DFJLLIST (OS)

The DFJLLIST module provides for processing of the program load list.

DLL140 Registers are saved according to standard OS convention. Base register is set.

DLL230 If a list is not found to be present transfer is to DLL470.

DLL280 Current modules are deleted from the list.

DLL350 A FREEMAIN macro is issued for that core occupied by the current list.

DLL470 The CREATE LOAD LIST phrase is read.

DLL530 SCAN is initiated to search out a left parenthesis.

DLL560 If the current end of the phrase has not been processed transfer is to DLL670.

DLL580 The load list pointer is reset.

DLL600 A pointer is set to the top of free storage. Subroutine is terminated by return to the user.

DLL670 The current list entry is zero and transfer is to DLL580.

DLL710 A list of names from the phrase is built.

DLL780 The list of names is sorted.

DLL990 A BLDL macro is issued to process the required names. If the BLDL macro is executed properly transfer is to DLL1200.

DLL1110 The list of names is optimized to exclude items which were not processed properly by the BLDL macro.

DLL1200 The named modules are loaded.

DLL1300 The load list table is set to indicate the modules that have been loaded into memory. Transfer is to DLL600.

DFJLODER (OS)

The DFJLOADER module contains the CORECLER, DIOCS, SIOCS, and LOADER subroutines. CORECLR is the core management routine and controls the managed free storage area. DIOCS is the direct access IOCS routine for the PLAN system and processes READ/WRITE requests of all system files, DYNAMIC drives, and PERMANENT files. SIOCS is the sequential IOCS for the PLAN system and processes READ/WRITE requests for all sequentially organized files. LOADER performs part of the DYNAMIC linkedit and loads modules into the PLAN program area.

CORECLER The CORECLER routine manages the free storage area by closing open data sets, deleting loaded programs, and issuing FREEMAINs on storage obtained by problem programs. Upon entry to this routine, register 12 must contain the address of PLAN and the two buckets in the mainline CURTOP, CURBTM must contain the limits of core to be managed.

DLO790 If the system is currently using the managed free storage array, transfer is to DLO890, otherwise, control is returned to the caller.

DLO890 OPEN data sets are closed. The DADD chain is located.

DLO930 If the last DADD has been processed, transfer is to DLO1450.

DLO1030 If DEB is in the release area, transfer is to DLO1270.

DLO1110 If the DCB is in the release area, transfer is to DLO1270.

DLO1190 If the DCB is in the program area that may be released, transfer is to DLO1270; otherwise, transfer is to DLO930.

DLO1270 The DCB is located and closed and transfer is to DLO930.

DLO1450 Loaded programs are deleted and the load list is located.

DLO1490 If the end of the load list has been reached, transfer is to DLO1830.

DLO1630 If the module is in the release area, transfer is to DLO1710; otherwise, transfer is to DLO1490.

DLO1710 The module is deleted and transfer is to DLO1490.

DLO1830 Free storage is released. If this is an MVT system, transfer is to DLO2110.

DLO1930 All of core is obtained. The area to be released is freed and control is returned to the caller.

DLO2110 The system is set in the supervisor via the subroutine STATESW.

DLO2170 The SPQE chain is located.

DLO2210 The next DQE is accessed.

DLO2310 If this is the last DQE on this chain, transfer is to DLO3030.

DLO2370 A test is made to see if the block described by this DQE should be released. If not, transfer is to DLO2220.

DLO2760 If this is a system subpool, transfer is to DLO2770.

DLO2750 The system is set into the problem state by the subroutine RESETN.

DLO2770 The block of core described by DQU is freed.

DLO2790 If this block was a system subpool, transfer is to DLO2170; otherwise, transfer is to DLO2110.

DLO3030 If this is not the last SPQE on the chain, transfer is to DLO2170.

DLO3070 The system is reset to the problem state by the subroutine RESETN and control is returned to the caller.

DIOCS The DIOCS subroutine is a direct access IOCS for PLAN. It uses BTAM to process all read/write requests. On entry to this routine, register 3 must contain the address of the DCB control block. Register 4 must contain a relative record displacement in bytes. Register 5 must contain the relative physical record number in the file. Register 6 must contain the caller's count in bytes. Register 7 contains the caller's array. Register 12 must point to PLAN.

DLO3690 The registers are saved and the base is set.

DLO3850 If the buffers have been primed, transfer is to DLO4230.

DLO3890 Buffer areas are primed.

DLO4230 If this is not a 'locate' mode call, transfer is to DLO4910.

DLO4330 The subroutine RINCS is called to search the in-core buffers to determine if the requested record is already in core.

DLO4350 If the requested record is not in core, transfer is to DLO4710.

DLO4370 The subroutine WAITCHK is called to issue a WAIT, if required, on the record area.

DLO4430 If this is not a WRITE request, transfer is to DLO4550.

DLO4470 The subroutine WRITE is called to force the buffer to be written out.

DLO4550 The registers are restored and control is returned to the caller.

DLO4710 If this is a WRITE operation, transfer is to DLO4550.

DLO4750 The subroutine GETBUF is called to obtain a buffer area.

DLO4790 The subroutine READ is called to read the requested record into the buffer and transfer is DLO4370.

DLO4910 If the user count is zero, transfer is to DLO4550.

DLO5030 A test is made to see if the users array can be used, if not, transfer is to DLO5150.

DLO5130 The record-used indicator is set for the buffer.

DLO5150 The subroutine RINCS is called to see if the next requested record is in core.

DLO5170 If the record is not in core, transfer is to DLO6490.

DLO5270 Data is moved either to, or from, the user array.

DLO6030 If this is not a WRITE operation, transfer is to DLO6110.

DLO6070 The WRITE request indicator is set on for this buffer.

DLO6110 If the record-used indicator is not on, transfer is to DLO4550.

DLO6250 If look-ahead is not required for this file, transfer is to DLO6930.

DLO6310 If the WRITE request indicator for this buffer is not on, transfer is to DLO6370.

DLO6330 The record is forced out and transfer is to DLO6930.

DLO6370 The subroutine READ is called to read the look-ahead record and transfer is to DLO5270.

DLO6490 The subroutine GETBUF is called to locate a buffer area.

DLO6510 If the record-used indicator is on, transfer is to DLO6690.

DLO6570 The subroutine READ is called to read the record into the buffer.

DLO6590 The subroutine WAITCHK is called to issue a WAIT for the I/O on that buffer and transfer is to DLO5270.

DLO6690 A register is set so that the return from the I/O will be to DLO6790. If this is a WRITE operation, transfer is to DLO7590; otherwise, transfer is to DLO7470.

DLO6790 The subroutine WAITCHK is called to issue a WAIT on the buffer.

DLO6810 The status of the buffer is updated.

DLO6890 The user counts and array address are updated.

DLO6930 The record number is set and transfer is to DLO4910.

GETBUF The subroutine GETBUF locates an available buffer from the buffer chain.

DLO7050 The buffer chain is located.

DLO7090 If the buffer is not busy, transfer is to DLO7910.

DLO7150 If this is not the last buffer in the chain, transfer is to DLO7090.

DLO7190 If there is not a WRITE request on the buffer, transfer is to DLO7910.

DLO7250 If this is not the last buffer in the chain, transfer is to DLO7190.

DLO7270 A WRITE on the buffer is forced and transfer is to DLO7910.

DLO7470 This is the entrypoint for the READ subroutine. READ and BUSY status are set for the buffer and transfer is to DLO7630.

DLO7590 This is the entrypoint for the WRITE subroutine. WRITE status and NOT BUSY are set for the buffer.

DLO7630 If the requested record number is in the extent, transfer is to DLO7710; otherwise, control is returned to the caller.

DLO7710 The I/O operation is executed and control is returned to the caller.

DLO7910 This is the entrypoint for the WAITCHK subroutine. A check macro is issued and control is returned to the caller.

DLO8090 This is the entrypoint for the RINCS subroutine. The call buffers are searched for the requested record number.

DLO8150 If an equal record number is found in core, a zero condition code is set and returned to the caller; otherwise, a nonzero condition code is returned to the caller.

SIOCS The SIOCS subroutine processes PLINP and PLOUT calls to the system. On entry to this routine, register 1 must contain the address of the callers argument list, register 12 must contain the address of PLAN and register 3 must contain the address of the PCB control block.

DLO8950 This is the end-of-file exit for SIOCS. The TRUE end-of-file indicator is set in the control block and exit is through the COMRET entry in CIOEN.

DLO9070 This is the normal entry to SIOCS. The NOD argument is validated by calling the subroutine SRCHIOC and CIOEN. If the NOD is valid, transfer is to DLO9190; otherwise, control is returned to the caller through COMRET.

DLO9190 The logical end-of-file indicator is reset.

DLO9230 If the file has previously been accessed, transfer is to DLO9730.

DLO9330 The internal open on the file is performed. This includes initializing the record area pointers and the prime indicator.

DLO9730 A test is made to see if the status of the file is the same. If it is not, transfer is to DLO8950.

DLO9790 A test on the device is made to see if it is capable of performing the requested function. If it is not, transfer is to DLO8950.

DLO9890 If the true end-of-file indicator has been previously set 'on', transfer is to DLO8950.

DLO9930 If this is a PLINP call, transfer is to DLO10970.

DLO9970 If RECFM equals FA or FBA, transfer is to DLO10090; otherwise, transfer is to DLO10970.

DLO10090 If the carrige control character is not equal to 1 transfer is to DLO10370.

DLO10130 The line counter is reset and transfer is to DLO10770.

DLO10370 The line counter is stepped.

DLO10650 If the line counter is not equal to zero, transfer is to DLO10770.

DLO10690 The logical end-of-file indicator is turned 'on'.

DLO10770 The carriage control character is set as the first character of the record.

DLO10970 If a buffer is not available, transfer is to DLO11810.

DLO11030 The buffer pointers are updated.

DLO11090 If this is not a PLINP call, transfer is to DLO11330.

DLO11190 If this is not a UREND record, transfer is to DLO11670.

DLO11230 The logical end-of-file indicator is set and transfer is to DLO11670.

DLO11330 The buffer area is blank.

DLO11670 The current buffer address is set in the control block and exit is to the COMRET entry in CIOEN.

DLO11810 The next record is read or written. If an end-of-file occurs, transfer is to DLO8950.

DLO12210 The current record area pointers are set and transfer is to DLO10970.

LOADER The LOADER subroutine is the PLAN program loader. On entry to this program, register 12 must contain the address of PLAN. The name of the program to be loaded must be stored in the first eight bytes of the name control block. This subroutine performs the following:

1. The BLANK COMMON control section is eliminated from a module by relocating all CSECTs which originate above BLANK COMMON downward by the length of BLANK COMMON.
2. All adcons within the module are relocated. Those that reference BLANK COMMON are relocated to PLAN COMMON.
3. If a load is successful, program control block (PCB) is completed and control is returned to the caller. An error during loading causes a phrase abort and exit into the ERRABORT entry in the DFJPLAN mainline. The BSAM access method is used to load the program.

DLO14440 The control blocks including the name control block and the COMMON control block are initialized.

DLO14570 The subroutine LLSRCH is called to check the load list which contains names of modules loaded into the partition to see if this program name is in that list. If it is, transfer is to DLO18210.

DLO14470 A search is made of the in-core directory. If the name is found in the directory, transfer is to DLO15290.

DLO15110 A BLDL is issued on the PLANLIB PDS for the module name.

DLO15290 The module attributes are checked.

DLO15450 A FIND is issued on the first record of the module.

DLO15610 A read is issued for a ESTRLD or CTL record. Exit from this routine is on the register PROCESS which points to either the EST processing routine or the RLD processing routine.

DLO15750 This is the ESD processing routine. If the record is not an ESD record, transfer is to DLO17230.

DLO15910 If this CSECT refers to BLANK COMMON, transfer is to DLO16970.

DLO16090 If the NOLINK option was specified, transfer is to DLO16590.

DLO16130 If the ESD type is not an SD or LD transfer is to DLO16730.

DLO16330 An entry is made in the ENTAB table for this module.

DLO16590 If this is not the last ESD entry for this record transfer is to DLO15910; otherwise, transfer is to DLO15610.

DLO16730 An ERTAB1 entry is created for the external reference and transfer is to DLO16590.

DLO16970 The origin and the length of the COMMON CSECT was saved and transfer is to DLO16590.

DLO17230 The length of PLAN COMMON is set. This will be the longer of the values set in Switch Word 9 or the length of the longest COMMON CSECT required by any program in the PLAN program area.

DLO18210 Core is obtained for the program itself.

DLO18490 If the program module was named as an entry point in the loaded program list, transfer is to DLO19170.

DLO18570 If the NOLINK PARM was specified transfer is to DLO18890.

DLO18670 The entrypoint in the ENTAB entries for this module are relocated.

DLO18890 A FIND is issued for the first text record of the module.

DLO18990 The first text record is read into the program area.

DLO19010 If more than one text record is in this module transfer is to DLO15610.

DLO19110 A check is issued on the last text record read.

DLO19170 The module entrypoint is set in the name control block.

DLO19250 The program control block (PCB) is completed from the main control block and control is returned to the caller.

DLO19490 A check is issued on the last text record read.

DLO19510 If this is not a CTL record transfer is to DLO19670.

DLO19550 A read is issued for the next text record.

DLO19570 If this is not an RLD record, transfer is to DLO22670.

DLO19670 Pointers to the RLD information in the record are set.

DLO19810 The adcon is located and moved to a work area.

DLO20510 If the adcon was resolved by the linkage editor, transfer is to DLO22110.

DLO20610 If the NOLINK PARM was specified, transfer is to DLO22490.

DLO20750 If the adcon can be resolved, transfer is to DLO22110.

DLO21810 An ERTAB2 entry is built and transfer is to DLO22490.

DLO22110 If the adcon reference is COMMON, transfer is to DLO22790.

DLO22310 The adcon is relocated.

DLO22490 If this is not the last RLD entry in this record, transfer is to DLO19810.

DLO22670 If this is an EOS or EOM record, transfer is to DLO19110; otherwise, transfer is to DLO15610.

DLO22790 The adcon is relocated to point to PLAN COMMON and transfer is to DLO22490.

DFJMPSCB (DOS)

This subroutine is called by DFJGMAIN and DFJFMAIN. It moves the PSCB table if it exists.

MPS230 The system pointer to the top and bottom of the free core in the partition are updated.

MPS410 The PSCB table is moved to its new location and control is returned to the caller.

DFJPCDMP

The DFJPCDMP module is entered as a result of the DUMP, DUMP MANAGED and DUMP NON-MANAGED command. The module requires the use of ERASABLE COMMON.

PCD260 The size of the managed array is picked up from Switch Word 10.

PCD270 The total size of the communication array, that is the managed array and the nonmanaged array, is calculated as the size of common as contained in Switch Word 9 minus the 640-word size of the combined PLAN loader and PLAN switch words.

PCD290 The device to be used for the printing of the dump is picked up from the 12th word of ERASABLE COMMON.

PCD310 Double buffer set B is assigned to the output device.

PCD330 The printer is skipped to a new page.

PCD350 The heading for the switch word listing is set to the print area.

PCD360 The printer is spaced twice.

PCD380 The contents of the switch words are set to the print area in hexadecimal form. Eight words are set on the first line and seven words on the second line. The contents of the switch words are printed.

PCD470 The printer is spaced two lines.

PCD500 The first position of ERASABLE COMMON is tested to see if the managed array is to be dumped. A negative value indicates that the managed array only is to be dumped. A zero value indicates that both managed and nonmanaged arrays are to be dumped, whereas a positive value indicates that only the nonmanaged array should be dumped.

PCD520 The heading for the managed array is set to print.

PCD540 The number of words contained in the managed array is set to print.

PCD560 The heading is printed.

PCD580 The printer is double spaced.

PCD600 A check for a no managed array is made. If there is no managed array transfer is to PCD880.

PCD620 A DO loop is initialized to dump eight words of the nonmanaged array per line.

PCD640 If this the first line of the managed array dump, transfer is to PCD680.

PCD660 A check is made to determine if this line is equal to the previously printed line. If it is, transfer is to PCD840.

PCD680 A pointer is set to the current managed array position that is to be dumped.

PCD690 The print position control is set to print position 1.

PCD700 The PHTOE subroutine is called in to convert the hexadecimal representation of the word to EBCDIC.

PCD710 The PAOUT subroutine is called in to set the first four characters to print.

PCD720 The PAOUT subroutine is called a second time to set the second four characters to print with a blank space between the previously printed four characters.

PCD740 If we are at the managed array transfer is to PCD810.

PCD760 The print position indicator and the managed array indicator are incremented to the next group to be processed.

PCD790 If the entire line has not been set to print transfer is to PCD700.

PCD810 The PIOUT subroutine is called to set the decimal representation of the managed array subscript to print.

PCD820 The line is printed.

PCD830 A test is made to determine if the entire managed array has been printed. If it is not, transfer is to PCD640.

PCD850 The printer is spaced five lines.

PCD880 A test of the first postion of ERASABLE COMMON is made to determine if the nonmanaged array is to be dumped. If it is, transfer is to PCD920.

PCD900 The PCDMP module is terminated by a CALL LRET.

PCD920 The nonmanaged array header is set to the printer.

PCD950 The size in FORTRAN words of the nonmanaged array is set to print area.

PCD960 The nonmanaged array header is printed.

PCD990 The printer is double spaced.

PCD1010 A test is made to determine if there is a NULL nonmanaged array. If there is transfer is to PCD900.

PCD1030 The start of the nonmanaged array is calculated as 640 words plus the length of the managed array.

PCD1040 A DO loop is initialized with a limit equal to the number of words in the nonmanaged array and an index equal to eight words are to be dumped on one line.

PCD1060 A test is made to determine if this is the first line of the nonmanaged array to be printed. If it is, transfer is to PCD1100.

PCD1080 A test is made to determine if this line is equal to the previously printed line. If it is, transfer is to PCD1270.

PCD1100 The print position indicator is initialized to the beginning of the print line and the array indicator is set to the current position to be printed.

PCD1120 The PHTOE subroutine is called to convert the current nonmanaged array position from hexadecimal to EBCDIC.

PCD1140 The PAOUT subroutins is called to set the two groups of four characters to the print area.

PCD1170 A test is made to determine if the last position of the nonmanaged array has been set to the print area. If it has transfer is to PCD1240.

PCD1200 The print position indicated in the nonmanaged array indicator are incremented to the next position to be processed.

PCD1220 If eight array words are not currently in the print buffer transfer is to PCD1120.

PCD1240 The PIOUT subroutine is called in to convert the nonmanaged array subscript in decimal form to the print area.

PCD1250 The line is printed.

PCD1270 A test is made to determine if the entire array has been printed. If it is not the loop is incremented and transfer is to PCD1060.

PCD1290 The printer is skipped two lines. Transfer is to PCD900.

DFJPDIAG

The DFJPDIAG module processes the information put into erasable COMMON by the SET LITERAL command. The information contained in the erasable COMMON at the time the module is entered is the file number, the file name, the drive code, the literal number, the number of characters in the literal message, and the literal text.

PDI210 The file number is picked up from the erasable COMMON and put into the permanent file control block.

PDI230 The index to the drive code in the erasable COMMON is set with a GDATA call.

PDI240 The GDATA subroutine is called to open the permanent file that contains the literal text.

PDI260 The PHOUT subroutine is called to write the literal information to the listeral file.

PDI280 The PDIAG module is terminated by a call to LRET.

DFJPEDMP

This utility module is invoked by the standard command DUMP ERRORS.

PED55 The subroutine ERLST is called to cause the error queue file to be dumped.

DFJPERRS

This module is the error processing module of the OS PLAN system.

PER160 The registers are saved according to standard OS conventions.

PER290 If the error indicator is on transfer is to PER3180.

PER310 The error indicator is turned on.

PER320 The error stack pointers within PFILE are reset.

PER340 If this entry into PERRS is for the function of error listing transfer is to PER380.

PER360 If the error processing is to be performed by a user-defined error processing module transfer is to PER580.

PER380 If the specified diagnostic device is valid transfer is to PER480.

PER450 The diagnostic device is set equal to the standard PLAN output device.

PER480 If any errors have been encountered transfer is to PER920.

PER500 If this entry into PERRS was not for the function of doing error listing transfer is to PER530.

PER520 The FLUSHQUE subroutine is called to clean out the PLAN file containing the diagnostic messages.

PER530 The internal counters and pointers are reset.

PER550 Registers are restored according to standard OS conventions. The module is terminated by return to the caller.

PER580 If switch word 8, that is, the pointer to erasable COMMON is not valid transfer is to PER380.

PER690 If a pointer to erasable COMMON is not available transfer is to PER380.

PER760 The READSTAK subroutine is called to read the error stack into memory.

PER770 If there are any errors still to be processe, processin

PER770 If there are any errors still to be processed, processing continues, otherwise transfer is to PER530.

PER790 The error array is built in erasable COMMON.

PER880 The user error module is brought into memory as a PLAN local and transfer is to PER760.

PER920 If the number of error messages to be queued is positive processing continues; otherwise transfer is to PER1050.

PER950 If the logical drive 0 does not exist transfer is to PER1060.

PER970 The GETSIZ subroutine is called to determine the count of queued error messages.

PER980 If this entry into PERRS is for the purpose of listing errors transfer is to PER1050.

PER1000 If the error message count is less than the number of error messages to be queued transfer is to PER1050.

PER1030 The output is set to drive 0 and transfer is to PER1070.

PER1050 The FLUSHQUE subroutine is called to process the messages from logical drive 0.

PER1060 The output device is set equal to the diagnostic device and transfer is to PER1070.

PER1070 The READSTKA subroutine is called to read the PLAN error stack.

PER1080 If there are no error messages remaining to be processed transfer is to PER530.

PER1090 The PRTERR subroutine is called to print the error messages and transfer is to PER1070.

FLUSHQUE This is the entrypoint for the logic that processes the PLAN error queue.

PER1110 The GETSIZ subroutine is called to determine the count of the number of PLAN error messages to be processed.

PER1130 If the count of messages to be processed is equal to zero transfer is to PER1240.

PER1160 The error message is read.

PER1190 The output parameters are set.

PER1220 The OUTM subroutine is called to generate the appropriate output for the PLAN diagnostic.

PER1230 If there are more messages to be processed transfer is to PER1160.

PER1240 The RELES subroutine is called to release logical file 255 on logical drive 0.

PER1270 The error message file indicator is reset. The PERRS module is terminated.

GETSIZ This is the entry point for the GETSIZ subroutine that determines the number of error messages within the PLAN file.

PER1320 If logical drive 0 exists transfer is to PER1540. Otherwise, processing is terminated and a return to the next executable statement is initiated.

PER1340 The FIND subroutine is called to do an open of logical file 255 on logical drive 0.

PER1380 The number of error messages contained in the file is calculated from the file size.

PER1410 The GETSIZ subroutine is terminated with a return to the next callable statement.

READSTKA This subroutine reads the PLAN error stack.

PER1440 If there are error messages remaining to be processed from the error stack transfer is to PER1490. Otherwise, the subroutine is terminated by return to the next executable statement.

PER1490 The next error message is read from the PLAN error stack.

PER1670 The short-form portion of the user error array is built.

PER1790 If the long-form error array is required, processing continues, otherwise, transfer is to PER1870.

PER1810 The phrase is read into memory.

PER1870 If a literal has been supplied with the error message transfer is to PER2060.

PER1890 If the error message being processed is not a PLAN system error transfer is to PER2000.

PER1910 The required literal is accessed from the liter table. If the literal is found in the table transfer is to PER2050. Otherwise, the space normally occupied by the literal of the diagnostic is replaced with an asterisk.

PER2030 The error module subroutine READSTKA is terminated.

PER2050 The correct literal is moved to the user array.

PER2060 The literal indicator is set and transfer is to PER2030.

PRTERR This subroutine prints the error messages.

PER2110 If the sequence number associated with the current phrase is the same as the sequence number associated with the phrase with the last diagnostic transfer is to PER2430.

PER2140 The new sequence number is saved.

PER2150 If the user array is to be built in short form transfer is to PER2430.

PER2220 The current PLAN phrase is printed.

PER2430 The error message line is set to the print area.

PER2650 The OUTM subroutine is called to output the print line and the subroutine is terminated.

OUTM This subroutine is called to generate the output of the error message.

PER2730 If the output is not to be placed on logical drive 0, file 255 transfer is to PERI800.

PER2750 The message is written to logical file 255 on logical drive

PER2800 If an end-of-file has not been processed on the diagnosticdevice transfer is to PER280.

PER2850 The skip count is set to one.

PER2860 If a carriage control character is not required transfer is to PER2920.

PER2880 The PCCTL subroutine is called to effect the necessary carriage control.

PER2910 The carriage control character is reset.

PER2920 The PAOUT subroutine is called to transmit the user error array in the print buffer.

PER2950 The PLOUT subroutine is called to print the diagnostic line. The subroutine is terminated by return to the caller.

ERRINPER This subroutine is called when an error is found or processing within the error module.

PER3180 A diagnostic message is generated to indicate that error processing cannot continue.

PER3240 The necessary internal indicators are reset.

PER3260 The phrase abort indicator is set. Ther error processing is terminated by transfer to DFJPLAN.

DFJPFDMP

This module is used by the standard PLAN commands DUMP PERMANENT and DUMP DYNAMIC to dump GDATA and FIND type files to the PLAN output device. Information about the file to be dumped is placed in ERASABLE COMMON by the appropriate command as follows:

Word 1	= File number
Word 2	= Second word of file control block
Word 3	= File dump start address
Word 4	= File dump end address
Word 5	= Logical drive of file
Word 6-12	= Header 'Length Drive File'
Word 13-15	= File name (PLAN Literal Form)
Word 16	= File type switch 0 - PERMANENT file 1 - DYNAMIC file

Halts: None.
Error Conditions: None.
Subroutines:
Monitor: FLDX, FSTOX, and SUBSC
PLAN: PDBFA, PLOUT, PAOUT, PIOUT, GDATA, FIND, RDATA, READ, PCOMP, PHTOE, PEOF, PCCTL, and LRET
Switches: Word 16 of ERASABLE COMMON is 0 for a PERMANENT file dump and 1 for a DYNAMIC file dump.

PFDMP-PFD250 The output device number requested by the command is picked up from the 16th word of ERASABLE COMMON.

PFD270 The record size is set to 0, the first line switch tested at PFD820 is turned on and the number of equal lines counter used at PFD1230 is set to 0.

PFD320-PFD330 The printer is spaced five lines.

PFD370 The number of characters in the file name is picked up from the 13th word of ERASABLE COMMON to be used in the PAOUT call at PFD420.

PFD400-PFD460 The header for the dump is created and placed in the printer output buffer.

PFD475-PFD480 If this is a DUMP PERMANENT command, GDATA is called to open the file. Otherwise, FIND is called.

PFD500-PFD520 If the command did not specify the last word of the file to be dumped, then the length of the dump is set to the total file size found in ID(2).

PFD540-PFD570 The length of the file to be dumped is placed in the printer output buffer, the header record is printed, and a blank line is printed to effect a double space.

PFD590 If the file did not exist or the length specified to dump is 0, transfer is to PFD1140.

PFD610-PFD690 The number of words left to be dumped is divided by 160 to determine if 160-word record can be read. If not, the partial record size is calculated. If there is nothing to be read, transfer is to PFD1140.

PFD711-PFD720 RDATA is called for a PERMANENT file or READ is called for a DYNAMIC file.

PFD730 The loop initiated here will process the number of lines (8-word records) just read in.

PFD750-PFD790 If this is the last record to be read from the file, then the number of words to be printed in the last line is calculated.

PFD820-PFD840 If this is the first line to be printed from this record, line count is set to 0 and transfer is to PFD890. Otherwise, PCOMP is called to compare this line with the line saved at PFD890, and if they are equal, transfer is to PFD1230.

PFD890 This print line is saved for the compare in PFD870.

PFD930 The print line is converted to hexadecimal printout form and placed in the printer output buffer.

PFD1030 The file location of the first word in the print line is moved to the print buffer in the left hand column position.

PFD1050-PFD1080 The line is printed and if this was the last line on the page, PCCTL is called so that the next call to PLOUT will skip to the beginning of the new page.

PFD1100 If all the lines in this record have not been processed, transfer is to PFD750 to continue the loop.

PFD1120-PFD1180 If the dump request has been fulfilled, the printer is spaced five lines and control is returned to the resident loader.

PFD1230-PFD1240 The number of equal lines is incremented and if this is the first equal line found, transfer is to PFD1050.

DFJPFIN

This subroutine converts an A4 format field into a floating-point FORTRAN word.

FIA1150 The user arguments are accessed, the field width is calculated, and the buffer pointer and pseudo accumulator is initialized.

FIA1650 Leading blanks are eliminated and the sign is collected. This is done by the subroutine PSCAN.

FIA1690 Digits to the left of the decimal point are collected and placed in the pseudo accumulator.

FIA1890 When a nonnumeric digit is found, a test is made to see if it is a decimal point. If not, transfer is to FIA2090. If it is, a test is made to see if this is the second decimal point. If it is not, the count of the digits above the decimal point are saved and transfer is to FIA1690 to continue collection of digits below the decimal point. If this is a second decimal point, transfer is to FIA2090.

FIA2090 A test is made to see if the symbol E is present in the input stream. If it is not, transfer is to FIA2170. Otherwise, the pointer is stepped past the E and the E-value sign is collected.

FIA2170 The E-value itself is collected.

FIA2490 The collected integers are positioned in the pseudo accumulator based on the E-value and the actual decimal point if one was present. If a decimal point was not present, the input arguments are used to determine the position of the decimal point.

FIA2910 The exponent is calculated from the position of the mantissa in the pseudo accumulator.

FIA3210 The mantissa is normalized in the pseudo accumulator.

FIA3750 The results are stored in the user array and exit is made to the caller.

DFJPFOUT, DFJPEOUT

This subroutine converts a floating-point word into A4 format in either E or F mode.

FOA1230 This is the entrypoint for DFJPEOUT. The E-format switch is set.

FOA1290 The user arguments are accessed. A test is made to see if any characters are to be outputted. If not, return is given to the caller. The field width and decimal width of the output field are calculated and a pseudo accumulator is cleared and initialized.

FOA1690 The mantissa is placed into the pseudo accumulator. The characteristic is used to determine the accumulator positions used.

FOA2210 The mantissa is shifted right in the pseudo accumulator until the digit appears above the decimal point. The decimal point is located at position 11 in the 23-position pseudo accumulator.

FOA2410 If this is a call to PEOUT or the E-format switch was set by the

PFOUT routine because of insufficient space, transfer is to FOA4730.

FOA2510 The rounding factor is computed. This is based on the number of digits to the right of the decimal point.

FOA2870 The rounding factor which has been calculated is added to the mantissa.

FOA3480 A test is made to see if there is enough room in the output field for the requested format. If not, transfer is to FOA1230 to set the E-format switch and try and output the number in E format. Otherwise, transfer is to FOA3570.

FOA3570 A test is made to check if leading blanks are required. If not, transfer is to FOA3710.

FOA3650 Leading blanks are placed in the output field.

FOA3710 If the input floating-point number was negative, a minus sign is outputted.

FOA3810 A leading zero is outputted if required. This can be caused if the rounded value is less than 1.

FOA3850 Characters to the left of the decimal point are outputted.

FOA4070 Characters to the right of the decimal point are outputted.

FOA4170 If this is not E format, transfer is to FOA4630.

FOA4210 An EBCDIC E is outputted.

FOA4250 The E-value is calculated.

FOA4370 The sign of the E-value is outputted.

FOA4490 The E-value itself is outputted and control is returned to the caller.

FOA4730 A test to see if there is enough room in the output field for E format. If yes, transfer is to FOA5290; otherwise, a test is made to check if there is enough room for normal format. If yes, transfer is to FOA5290. Otherwise, the output field is filled with asterisks and control is returned to the caller.

DFJPHRAS

This module provides maintenance capability for a language dictionary (PFILE). It will add, alter or delete a phrase from the dictionary. Logical and syntax verification is performed before each phrase is stored. The system error module (DFJPERRS) is called to log any required diagnostics. This program is specified in the program list for the standard PLAN commands ADD PHRASE, ALTER PHRASE and DELETE PHRASE.

PHR38 The base registers are set and various constants, switches, and table areas are initialized.

PHR70 The operation ADD, ALTER, or DELETE is determined from the value found in ERASABLE COMMON (1). The appropriate indicators are set in the switch 'PHRASW'.

PHR82 The subroutine COMPRESS is called to read the phrase image from the PFINPUTA record of the phrase dictionary file.

PHR84 A CAP subscript pointer register is initialized to point to the beginning of the managed array.

PHR86 The subscript register is incremented to the next CAP position.

PHR88-PHR104 A test is made for a valid end to the phrase name. This must be a comma or a semicolon. If an invalid character is found, an error message is issued and the scan continues until a comma or a semicolon is found. If a comma is found, a transfer is to PHR110. If a semicolon, transfer is to PHR148.

PHR110 The subroutine ADVSUP is called to slide over the comma in the input stream.

PHR112 A test is made to check if a dollar sign has been encountered in the input stream. If the current character is a dollar sign, transfer is to PHR548 which is the formula collect subroutine.

PHR116 The subroutine ALPHAC is called to collect the symbols and CAP pointer for the phrase entry.

PHR118 The subroutine CONSTANT is called to collect any default values.

PHR120-PHR128 A test is made to see if an error occurred in collecting an Implied Do subscript. If an error occurred, an error message is issued.

PHR130 The subroutine LITERALT is called to collect any literals that may be present in the phrase entry.

PHR132 The subroutine CHKENTRY is called to collect check entries, if present.

PHR134 The subroutine EXPRESSC is called to collect phrase-defined expressions.

PHR138-PHR144 A test is made to see if a literal was found in the phrase entry. If a literal was found, the subscript pointer is set to the end of the literal and the transfer is to PHR88.

PHR148 The subroutine STRING is called to collect the table together and build the phrase entry.

PHR150 A test is made to see if any errors occurred in the phrase scan. If no errors occurred, transfer is to the routine TUPDATE to enter the phrase in the phrase dictionary. Otherwise, exit is to the PLAN loader.

LITERALT The LITERALT subroutine tests for and collects phrase-defined literals.

PHR168 The next character in the input stream is checked to see if it is a literal delimiter. These are ', @, and ". If the character is not a delimiter, return is to the caller.

PHR180 The literal delimiter is saved in order to locate the end of the literal.

PHR180-PHR208 The length of the literal is determined by scanning the input stream for a delimiter that is the same as the saved delimiter. A test is also made to see if there is a semicolon in the literal or the length of the literal is zero. In either case, an error message is issued and return is made to the caller.

PHR212 The word count of the literal is collected.

PHR222 An indicator is set so that on return to the caller, the subscript register may be incremented to the end of the literal.

PHR224 If the literal delimiter was a double quote indicating that the count is not required in this literal, transfer is to PHR232.

PHR228 The subroutine WSYML is called to write a full word of the literal into Table 2.

PHR230-PHR240 The CAP subscript register is incremented and the next four bytes of the literal are collected. If this is not the last four bytes of the literal, transfer is to PHR228 to write the word into the table.

PHR244 A test is made to see if the last word of the literal contains any residual characters.

PHR248 The last word is padded with blanks if required.

PHR260 The end subscript of the literal is stored and the current subscript pointer is restored and return is made to the caller.

CONSTANT A CONSTANT subroutine collects single logical values and numeric values both integer and REAL.

PHR276-PHR294 A test is made to see if a sign is present in the input stream. If it is not, transfer is to PHR298; otherwise, an indicator is set and the subroutine ADVSUP is called to increment the input pointer past the sign.

PHR298 The subroutine COLNUMT is called to collect any numeric constant if present.

PHR300 If a valid constant was collected, exit is from this subroutine to the WSYM subroutine which will store the word in the phrase entry in Table 2 and then return to the caller.

PHR302 A test for a uniary sign or a single logical value, plus or minus, is made. If a sign was present, transfer is to PHR306. Otherwise, return is made to the caller.

PHR306 The logical value TRUE or FALSE depending on the sign, is set and exit is from this routine to the WSYML subroutine to store the value into Table 2.

EXPRESSC The EXPRESSC subroutine scans and collects all phrase-defined expressions.

PHR336 If the input character is an equal or a pound sign, transfer is to PHR348.

PHR340 If the input character is a colon indicating a LOGICAL expression, transfer is to PHR452; otherwise, return is made to the caller.

PHR348 The subroutine DARITHX is called to scan the arithmetic expression.

PHR350 A test is made to check if this is a conditional expression and if yes, transfer is to PHR488.

PHR354 A pointer is set to the expression entry in Table 6.

PHR356 A test is made to check if this subroutine is called from the EXPCNTRL subroutine which collects and scans the formula area. If yes, transfer is to PHR414.

PHR360 The subscript is validated and if good transfer is to PHR378; otherwise an error is issued.

PHR378 The symbol table entry is collected.

PHR402 The subscript and the compressed symbol are placed in Table 6.

PHR414 The length of the expression is calculated.

PHR420 The expression is moved to Table 6.

PHR444 Return is made to the caller.

PHR452-PHR454 If the next character in the input stream is a dollar sign indicating a formula number, transfer is to PHR492.

PHR458-PHR460 The subroutine DLOGICAL is called to scan the logical expression.

PHR462 A test is made to see if this expression is in the TRUE leg and if yes, transfer is to PHR488.

PHR468 If the next character in the input stream is not a question mark indicating a TRUE leg of an expression, transfer is to PHR354.

PHR470 The conditional switch is inverted indicating that we are processing a TRUE leg.

PHR472-PHR474 If the next character in the input stream is a dollar sign indicating a formula number, transfer is to PHR490.

PHR478 If the next character is an equal or pound sign indicating an arithmetic expression, transfer is to PHR348. If the character is a colon, transfer is PHR452. If none of these, an error message is issued and transfer is to PHR444 to return to the caller.

PHR488 If the next character is not an exclamation mark which denotes a FALSE leg, transfer is to PHR354. Otherwise, transfer is to PHR470 to process the FALSE leg.

PHR492-PHR494 The subroutine INTEGRI is called to collect an expression number.

PHR496 A test is made to see if this subroutine was called from the EXPCNTRL subroutine which processes the formula area. If it was not, an error message was issued and transfer is to PHR350.

PHR496-PHR514 The expression number is validated, and if incorrect, an error message is issued and transfer is to PHR350.

PHR524-PHR534 This expression number is placed in the expression number table and an indicator is set to show that this number was referenced. Transfer is to PHR350.

EXPCNTRL The EXPCNTRL subroutine is the formula expression area collect routine.

PHR552 If the next character in the input stream is not a dollar sign indicating the formula number, transfer is to PHR598.

PHR556-PHR558 The subroutine INTEGER is called to collect the expression number.

PHR560 If the expression number is zero, transfer is to PHR552 and this number is ignored.

PHR564 If the expression number is greater than 1024, transfer is to PHR552 and this number is ignored.

PHR576 The formula number table is accessed and if this is a multiply-defined number, an error message is issued and transfer is to PHR552.

PHR598 The subroutine DALPHA is called to slide over the left hand symbol of the expression.

PHR600 If a valid alphabetic symbol was not found, transfer is to PHR620.

PHR602 If the next character in the input stream is not a comma indicating the end of the expression, transfer is to PHR644. Otherwise, the subroutine ADVPUL is called to slide over the comma and transfer is to PHR552.

PHR606-PHR616 If the next character in the input stream is a comma indicating the end of the expression, transfer is to PHR546. If the next character is a semicolon indicating the end of the phrase, transfer is to PHR656. If neither, an error message is issued and the input pointer is incremented to the next character. This routine continues to slide until a comma or a semicolon is found.

PHR620-PHR622 If the next character in the input stream is not a left parenthesis indicating a subscript expression, transfer is to PHR644.

PHR630 Subroutine DARITH is called to collect the subscripted expression.

PHR634 If the expression ends with a right parenthesis, transfer is to PHR642. Otherwise, an error message is issued and transfer is to PHR608 to slide to the end of the expression.

PHR642 The subroutine ADVPUL is called to slide over the right parenthesis.

PHR644 Subroutine EXPRESSC is called to collect the expression and put it in Table 6.

PHR648 If the next character in the input stream is a comma, indicating the end of the expression, but not the end of the phrase, transfer is to PHR546.

PHR652 If the next character in the input stream is not a semicolon, transfer is to PHR606 to issue an error message and continue the processing.

PHR656-PHR696 The formula number table is scanned and error messages are issued for all referenced and undefined formula numbers, and defined and unreferenced numbers. Exit from this routine is to PHR148.

COMPRESS The COMPRESS routine reads the phrase image from PFINPUTA record of the phrase dictionary, compresses the phrase name, and collects the checksum.

PHR710 The subroutine READ is called to read the input statement from disk.

PHR722 Subroutine ADVSUP is called to slide over the command.

PHR732 A test is made to see if the command ends with a colon. If not, transfer is to PHR810 to issue an error message.

PHR738 The phrase name is collected and the checksum is computed.

PHR798 A test is made to see that the phrase name is terminated properly with either a comma or a semicolon. If not, transfer is to PHR810 to issue an error message.

PHR806 A test is made to see if any name at all was collected and if yes, exit is to the caller.

PHR810 An error message is issued and transfer is to ABORTEND to terminate the processing of this phrase.

DLOGICAL The subroutine DLOGICAL performs a diagnostic scan on logical expressions.

PHR832 The ADVPUL subroutine is called to slide over any NOT symbol.

PHR838 The next character in the input stream is not an EBCDIC left parenthesis indicating an expression, transfer is to PHR910.

PHR848 The subroutine ADVSUP is called to slide over the left parenthesis.

PHR850 - PHR856 If the next character in the input stream is a BCD left parenthes, an error message is generated and transfer is to PHR904.

PHR860 The next character in the input stream is not a left paren transfer is to PHR868. Otherwise, the arithmetic parenthesis count is incremented and transfer is to PHR848.

PHR868 If a semicolon is found in the input stream indicating the end of the phrase, transfer is to PHR904 to try a logical scan.

PHR872 If the input stream character is a relational operator, <, >, =, or # sign, transfer is to PHR900.

PHR888 If the input stream character is not an EBCDIC right parenthesis, transfer is to PHR848 to continue the scan.

PHR892 If the arithmetic parenthesis count is zero indicating that an equivalent left parenthesis has not been found, transfer is to PHR904.

PHR896 The arithmetic parenthesis counter is decremented and transfer is to PHR848.

PHR900 If the arithmetic parenthesis counter is zero transfer is to PHR996 to process a relational expression.

PHR904 The input pointer is restored to the start of the expression and the logical parenthesis counter is incremented and transfer is to PHR832 to begin processing another expression.

PHR910 The subroutine DALPHA is called to test for and slide over an alphabetic symbol if present. If an alphabetic symbol is not present, transfer is to PHR952.

PHR916 A test is made for left parenthesis in the input stream indicating a subscript expression. If not, transfer is to PHR932.

PHR920 The subroutine DARITH is called to slide over the subscript expression.

PHR922- PHR930 The next character in the input stream is checked to ensure that it is a right parenthesis and a proper end to the subscript expression. If it is, transfer is to PHR930 and the subroutine ADVPUL is called to slide over the right parenthesis. Otherwise, an error message is issued and transfer is to PHR958.

PHR932 If the next character in the input stream is the logical operator OR/AND, transfer is to PHR832.

PHR940 If the logical parenthesis counter is zero, which indicates that we are not within an inner set of parentheses, transfer is to PHR958.

PHR944- PHR958 If the next character in the input stream is an EBCDIC right parenthesis, the logical parenthesis counter is incremented and transfer is to PHR930. Otherwise, an error message is issued. The input pointer is set to the end of the expression and control is returned to the caller.

DRELATS This is the relational expression evaluation routine.

PHR996- PHR998 The current input stream character is saved and the subroutine ADVPUL is called to slide over the operator.

PHR1000 If the input stream character is a literal delimiter ("), transfer is to PHR1004.

PHR10022 A check is made to see if a sign is present. If it is not, transfer is to PHR1058.

PHR10034 The subroutine ADVPUL is called to slide over the sign.

PHR10036 The next input character is checked to see if it is a right parenthesis. If it is not, transfer is to PHR1058.

PHR1004 At this point we have determined that we have a single logical value as the right side of a relational expression. The input pointer is restored to the beginning of the expression.

PHR1006 The subroutine ADVPUL is called to slide over the input operator.

PHR1008 The subroutine DALPHA is called to test for and slide over the alphabetic symbol. If the alphabetic symbol is not present, transfer is to PHR1072.

PHR1014 The next character is checked to see if it is a left parenthesis and if not, transfer is to PHR1026.

PHR1018 The subroutine DARITH is called to slide over the subscript expression.

PHR1020 A check is made for a proper end to the expression. The next character in the input stream must be an EBCDIC right parenthesis. If not, transfer is to PHR1072.

PHR1024 The subroutine ADVPUL is called to slide over the right parenthesis.

PHR1026 A check is made to see that the input stream character is the same as the operator that was saved on entry to this routine. This ensures that the scan has returned to the correct position in the input stream. If these characters are not equal, transfer is to PHR1072.

PHR1030 A check is made to see if the input character is an equal or a pound sign which are the only valid operators for a single logical value relational. If not, transfer is to PHR1072 to issue an error.

PHR1038 Subroutine ADVPUL is called to slide over the operator.

PHR1040 The input stream character is checked to see if it is a double quote. If it is not, transfer is to PHR1054.

PHR1044 The literal is scanned and checked to see if it contains a semicolon. If it does, transfer is to PHR1072.

PHR1054 The subroutine ADVPUL is called to slide over the literal delimiter and transfer is to PHR1068.

PHR1058-
PHR1066 The subroutine DARITH is called to evaluate both sides of the expression. A test is also made to see that the input operator is the same as the one saved on entry to this routine in order to ensure that the input stream pointer is back in the correct place.

PHR1068 A check for a closing parenthesis is made. If yes, transfer is to PHR1090.

PHR1072 An error message is issued.

PHR1076 If this is not a literal compare transfer is to PHR1090. This check is made by comparing the input character with a double quote.

PHR1080 The literal is scanned for a semicolon. If the semicolon is found, transfer is to PHR932.

PHR1090 The input stream is stepped to the end of the expression and transfer is to PHR932.

DARITH The subroutine DARITH performs a diagnostic scan on arithmetic expressions.

PHR1130 The subroutine ADVPUL is called to slide over the arithmetic operator.

PHR1134 A test is made to see if the next character in the input stream is a sign. If it is not, transfer is to PHR1148.

PHR1146 The subroutine ADVPUL is called to slide over the sign.

PHR1148 A check is made to see if a left parenthesis is present in the input stream indicating a subscript expression. If the parenthesis is present, transfer is to PHR1178.

PHR1156 The subroutine DALPHA is called to test for and slide over a symbol if present. If the symbol is present, transfer is to PHR1170.

PHR1160 The subroutine COLNUMT is called to test for and collect a numeric constant, if present. If it is present transfer is to PHR1194;

otherwise, an error message is issued and control is returned to the caller.

PHR1170 A test is made on the input character to see if it is a left parenthesis. If it is not, transfer is to PHR1194.

PHR1178 The parenthesis counter is incremented and transfer is to PHR1130.

PHR1182 If we are processing an inner nest, a test is made for a right parenthesis. If a right parenthesis is not present in the input stream, transfer is to PHR1164 where an error message is issued and control is returned to the caller. If the parenthesis is present, the parentheses counter is decremented and the subroutine ADVSUP is called to slide over the parenthesis.

PHR1194 If an arithmetic operator is present in the input stream, transfer is to PHR1146.

PHR1214 If the parentheses counter is not zero indicating that we are in an inner nest, transfer is to PHR1182, otherwise control is returned to the caller.

DARITHX The subroutine DARITHX performs the diagnostic scan on logical expressions.

PHR1222 The subroutine ADVPUL is called to slide over the equal sign.

PHR1224 A test is made to see if a sign is present in the input stream. If it is not, transfer is to PHR1256.

PHR1236 The subroutine ADVPUL is called to slide over the sign.

PHR1238 The input stream is checked for either a left parenthesis, an alpha symbol, or a numeric constant. If any of these are present transfer is to PHR1130 to process an arithmetic expression. If none of these are present, return is made to the caller.

PHR1256 If the input stream character is not a literal delimiter, that is, a ', @, or ", transfer is to PHR1130 to process the arithmetic expression.

PHR1270 The literal is scanned for a semicolon. If a semicolon is found, transfer is to PHR1288.

PHR1276-
PHR1280 The length of the literal is checked. If it is not zero, transfer is to PHR1292 where return is made to the caller.

PHR1288 An error message is issued.

PHR1292 Return is made to the caller.

DALPHA The subroutine DALPHA tests for and slides over alphabetic symbols.

PHR1308 The subroutine ALPHAT is called to see if the first character is alphabetic. If yes, transfer is to PHR1316; otherwise, return is made to the caller.

PHR1316 The first character of the symbol is saved so that it can later be checked to see if this symbol is a single E.

PHR1320 If this is not a subscript operand, that is, an S' operand, transfer is to PHR1338.

PHR1328 The input stream pointer stepped over the subscript operand.

PHR1338 The count of the characters in the symbol is collected.

PHR1340 A test is made to see if the symbol is over three characters, and if not, transfer is to PHR1352.

PHR1350 The subroutine PULADV is used to pull down and suppress extra characters in the symbol.

PHR1352 A test is made to see if the symbol is a single E. If not, transfer is to PHR1314 where a return is made to the caller.

PHR1364 An error message is issued and transfer is to PHR1438, the PULADV subroutine to suppress any blanks following the symbol.

ALPHAT The ALPHAT tests a character for alphabetic.

PHR1372 A test is made to see if a character is alpha or nonalpha. If it is alpha, transfer is to PHR1396; otherwise, return is made to the caller.

PHR1396 The alpha-found exit is set and return is made to the caller.

PHR1412-PHR1460 The subroutines ADVSUP, SUPADV, ADVPUL, and PULADV are used to control the scanning of the input stream so that blanks may be suppressed. The subroutine PULADV steps the input pointer to the next significant character in the input stream and then moves the entire remainder of the input stream down over the blanks.

VERBY This subroutine processes the 'VERB' keyword.

PHR1486 An indicator is set in the phrase entry to show that this is a verb phrase.

PHR1488 A test for a program list delimiter is made. If the next character in the input stream is not a comma or a semicolon, transfer is to PHR1522.

PHR1500 A test for a valid end to the keyword entry is made. If the keyword ends with a comma, transfer is to the ADVSUP routine to slide over the comma.

PHR1504 If the keyword is ended with a semicolon, transfer is to PHR112.

PHR1508 An error message is issued and transfer is to PHR112.

PROGRAM This routine processes the 'PROGRAM' keyword.

PHR1524 If the character in the input stream is a program list delimiter, that is, a quote, a comma, or a double quote, transfer is to the subroutine COLPLIST to collect the program list. If not an error message is issued and transfer is to PHR112.

PLEVEL This routine processes the 'LEVEL' keyword.

PHR1548 The subroutine INTEGERI is called to collect the level number.

PHR1550 The level number is tested for validity, and if proper, transfer is to PHR1568.

PHR1562 An error message is issued and a phrase level is set to blank.

PHR1568 The level of the previous phrase is cleared.

PHR1572 The new level is placed into the phrase entry and transfer is to PHR1500.

EXIT This routine processes the EXIT keyword entry.

PHR1578 The user-exit program list is collected and placed in Table 7. This is done by linking N to the program collect routine.

PHR1592 A test is made on the number of names in the user exit list. If not higher than three, transfer is to PHR1612.

PHR1596 An error message is issued and transfer is to PHR112.

PHR1600 A test is made to see if user-exit names were omitted, and if not, transfer is to PHR1606.

PHR1604 The standard PSCAN user-exit names are used. These are EXIT1, EXIT2, and EXIT3.

PHR1606 A test is made to see that all of the user-exit names in Table 7 are alphabetic. If any name is found not to be alphabetic, transfer is to PHR1596 to issue an error message.

PHR1618 If this is not the last entry in Table 7, transfer is to PHR1600; otherwise, transfer is to PHR1500.

CHKENTRY This routine tests for and collects all phrase check entries.

PHR1634 If the next character in the input stream is not an asterisk, return is made to the caller.

PHR1638 The subroutine ADVSUP is called to slide over the asterisk.

PHR1640 The current subscript or pointer to the CAP is placed in Table 5.

PHR1646 The subscript is validated. If valid, transfer is to PHR1556, otherwise, an error message is issued.

PHR1656 A test is made to see if this is an execution-defined symbol. If it is not, transfer is to PHR1662.

PHR1660 The subscript in Table 5 is set to zero.

PHR1662 The check entry type R, T, or F, and the function code A, C, and P are placed into Table 7.

PHR1712 A test is made to see if a literal is present in the check entry, and if yes, transfer is to PHR1736.

PHR1724 A test is made to see if the next character in the input stream is a left parenthesis indicating a COMMON subscript. If it is not, transfer is to PHR1738.

PHR1732-
PHR1736 An indicator is set to show that a COMMON subscript is present.

PHR1738 A test is made to see if an execution-defined symbol is being used. If not, transfer is to PHR1762.

PHR1744 The compress symbol and the subscript are placed in Table 5.

PHR1762 A test is made to see if either a literal or a subscript was present. If not, transfer is PHR1792.

PHR1768 If a literal is not present, transfer is to PHR1802.

PHR1772 A test is made on the function code to see if a program list is present. If not, transfer is to PHR1824.

PHR1782 The subroutine COLPLIST is called to collect the program list.

PHR1784 If the length of the program list was zero, transfer is to PHR1872.

PHR1792 The function code was tested to see if it is a pushed phrase. If it is not, transfer is to PHR1872, otherwise an error message is issued and transfer is to PHR1634 to process another check entry if present.

PHR1802 The subroutine INTEGER is called to collect the COMMON subscript.

PHR1804 A test is made on the next character in the input stream to see that it is a right parenthesis. If not, transfer is to PHR1796 to issue an error message.

PHR1812 The subroutine ADVSUP is called to slide over the right parenthesis.

PHR1814 The COMMON subscript is validated. If the value is zero, or too large, transfer is to PHR1796 to issue an error message; otherwise, transfer is to PHR1872.

PHR1826 The literal is scanned to see if it contains a semicolon. If it does, transfer is to PHR1796.

PHR1840 The subroutine ADVSUP is called to slide over the ending quote.

PHR1824 The length of the literal is tested for zero, and if it is, transfer is to PHR1792 to issue an error message.

PHR1850 The literal is moved to Table 5.

PHR1872 The length of the literal is placed into Table 5.

PHR1876 The function code is placed into Table 5.

PHR1878 The end of the check entry in Table 5 is located and its address is saved. Transfer is to PHR1634 to process the next check entry.

COLPLIST The COLPLIST subroutine collects program lists for the phrase check entries and user exit.

PHR1904 The subroutine ADVSUP is called to slide over the program list delimiter.

PHR1910 The subroutine ALPHAT is called to see if the first character of the name is alphabetic. If yes, transfer is to PHR1932 to collect the program name.

PHR1914 A test is made to see if this is an empty name in the middle of a bank load, that is, two successive commas after a left parenthesis. If this is the case, transfer is to PHR2012 to issue an error message.

PHR1922 If this is not a bank load, a zero entry is created and transfer is to PHR1948 to put the entry into the table.

PHR1932 The program name is collected.

PHR1940 If the name is not eight characters or less, an error message is issued and transfer is to PHR1954.

PHR1948 The subroutine PGMNEXT is called to put the program name in the appropriate table.

PHR1954 The next character in the input stream is checked to see if it is a comma indicating that another program name is in the list. If this is so, transfer is to PHR1904.

PHR1958 A check is made to see if the next character in the input stream is the program list delimiter. If yes, transfer is to PHR2004.

PHR1962 If the program name is an asterisk indicating a checkpoint return, transfer is to PHR1976.

PHR1966 If the program name is left parenthesis indicating the start of a bank of names processing continues, otherwise, transfer is to PHR1990.

PHR1970 If a left parenthesis, transfer is to PHR2014.

PHR1974 The left parenthesis found switch is inverted.

PHR1976 The character, either an asterisk or a left or right parenthesis, is placed in the program name entry.

PHR1978 The subroutine PGMNEXT is called to place the entry in the appropriate table.

PHR1980 The subroutine ADVSUP is called to slide over the character.

PHR1982 If the next character is alphabetic or numeric indicating a program name, transfer is to PHR1904 to collect the next name. Otherwise, transfer is to PHR1954 to test for the end of list.

PHR1990-
PHR1994 If the next character in the input stream is a right parenthesis, the left parenthesis found indicator is reset and transfer is to PHR1976, otherwise an error message is issued.

PHR1998 A test is made to see if a semicolon is in the program list. If not, transfer is to PHR1904 to process the next name on the list.

PHR2004 The input pointer is restored to the semicolon.

PHR2007 The left paren found switch is tested to see if it is on indicating unbalanced parentheses. If it is off, return is made to the caller; otherwise, transfer is to PHR2012.

PHR2012 The left parenthesis found switch is reset and the error message is issued. Transfer is to PHR1980 to test the next name in the list.

PGMNEXT This subroutine moves a program name to the appropriate table and updates the table pointer.

PHR2034 A test is made to see if the table will overrun by making this entry. If it will, transfer is to PHR2042 to return to the caller.

PHR2038 The program name is moved to the table and the table pointer is updated.

PHR2042 Return is made to the caller.

WSYM The WSYM subroutine formats a fixed or REAL value so that it may be placed into Table 2.

PHR2056 A constant is scaled by the P-value factor.

PHR2074 If a test is made to check if the constant is a fixed-point number. If not, transfer is to WSYML.

PHR2078 A fixed-point value is half adjusted.

WSYML This subroutine creates a Table 2 entry which contains the symbol, if present, the subscript, and the default value.

PHR2102 The subscript is adjusted. If this is a long-form subscript indicating it does not reference the switch words, a constant of 15 is added to the switch words so that it is a true reference of the managed array.

PRH2114 A test is made to check if an execution-defined symbol is being processed. If not, transfer is to PHR2138.

PHR2118 The compressed symbol is placed into Table 2.

PHR2132 A test is made to see if the subscript is an Implied Do, if yes, transfer is to PHR2140.

PHR2138 The subscript is placed in Table 2.

PHRI140 The constant value is placed into the table.

PHR2144 A test is made to see if the subscript is valid. If it is, transfer is to PHR2152, otherwise, an error message is generated and exit is made to the caller.

PHR2152 A test is made to see if the subscript was an Implied Do. If not, return is made to the caller.

PHR2156 The Implied Do parameters are placed in a table and exit is made to the caller.

ALPHAC the ALPHAC subroutine collects keywords, symbols, and symbol table entries.

PHR2232 If the input contains a subscript indicated by a left parenthesis, transfer is to PHR2304.

PHR2248 The subroutine ALPHAT is called to check the next character in the input for alphabetic.

PHR2254 If not alphabetic, return is made to the caller. The next character in the input stream is tested to see if it is the delimiter. If it is not, transfer is to PHR2264.

PHR2258 The subroutine INTEGERI is called to collect the user-exit number.

PHR2264 The mode of the constant is determined. If the next character in the input stream is an I, the integer mode switch is set and the subroutine ADVSUP is called to slide over the 'I'.

PHR2272 A test is made to see if the P-value is present. If not, transfer is to PHR2296.

PHR2276 The P-value is collected.

PHR2296 The next character in the input stream is tested to see if it is a left parenthesis indicating a subscript or an Implied Do. If it is not, transfer is to PHR2570.

PHR2304 The P-value is validated and if in range, transfer is to PHR2316. Otherwise an error message is generated.

PHR2316 The phrase entry indicators are set based on the mode, the subscript, and the P-value if present.

PHR2342 The user-exit number is tested, if present, to see if it is valid. If it is less than four, transfer is to PHR2350; otherwise, an error message is generated.

PHR2315-
PHR2424 The input stream is scanned to determine if an integer subscript is present. If it is, a subroutine INTEGERI is called to collect the subscript. The value of the subscript is then validated and if within range, transfer is to PHR2464. Otherwise, an error message is generated and transfer is to PHR2464.

PHR2430 The subroutine DARITH is called to diagnose the subscript expression.

PHR2438 A test is made to see if there is a valid end to the expression, that is, a right parenthesis in in the input stream. If yes, transfer is to PHR2460, otherwise an error message is generated and a check is made to see if a semicolon has been found in the input stream. If not, transfer is to PHR2430 to continue the diagnostic scan of the arithmetic expression. Otherwise, transfer is to ABORTEND to cease phrase processing.

PHR2460 The limits of the expression in the input stream are saved. These will be used later to move the expression to the appropriate table.

PHR2464 A test is made to see if an Implied Do is being processed. If not, transfer is to PHR2542.

PHR2474 The subroutine INTEGER is called to get the ending and the increment subscripts.

PHR2476 Both subscripts are validated and if either is invalid, transfer is to PHR2544 to issue an error message.

PHR2496 The Implied Do parameters are placed into the symbol entry and transfer is to PHR2542.

PHR2544 An error message is issued and a test for semicolon is made. If a semicolon is found, transfer is to ABORTEND to cease phrase processing.

PHR2542 A test is made to see that there is a valid end to the expression. This is a right parenthesis. If not, transfer is to PHR2544 to issue an error message.

PHR2552 The subroutine ALPHAT is called to see if a symbol is present in the input stream. If it is, transfer is to PHR2596.

PHR2558 A test is made to see if a symbol is required. If not, return is made to the caller.

PHR2562 An error message is issued and return is made to the caller.

PHR2570 The symbol that has just been located is checked against the keyword table. If a match is found, exit is through the subroutine DALPHA which will slide over the symbol and transfer to the correct processing routine.

PHR2588 A test is made to see if a symbol is allowed. If yes, processing continues at PHR2596; otherwise, an error message is generated.

PHR2596 A symbol is collected.

PHR2602 A test is made to see if the symbol is a single character E. If it is, an error message is generated.

PHR2628 The subroutine SUPADV is called to slide past the end of the symbol.

PHR2634 The first three characters of the symbol are compressed.

PHR2666 The compressed symbol is placed in the symbol entry.

PHR2670 A symbol entry is placed in Table 3.

PHR2680 A test for valid symbol subscript is made. If the subscript is valid, processing continues at PHR2698, otherwise, an error message is issued.

PHR2698 A test is made to see if any expressions were collected. If not, transfer is to PHR2566 to return to the caller.

PHR2702 The expression that was collected is moved to Table 3 and transfer is to PHR2566 to return to the caller.

STRING The STRING subroutine pulls all of the tables that were created together and creates the phrase entry.

PHR2746 Pointers are set to the tables and the phrase entry.

PHR2762 The length of the current table is computed.

PHR2774 The table is moved to the phrase entry.

PHR2782 If the table was over 255 half-words an error message is issued.

PHR2804 If this is not the last table, transfer is to PHR2762 to continue processing.

PHR2816 The size of the phrase entry is computed and if less than 1024 half-words, return is made to the caller. Otherwise, an error message is issued and then return is made to the caller.

COLNUMT This subroutine collects REAL values.

PHR2864 The interger value of the numeric is collected.

PHR2920 The integer value is floated or converted to floating-point form.

PHR2962 A test is made to see if any integers were collected. If not, return is made to the caller.

PHR2972 A check is made to see if E-value is present. If not, transfer is to PHR3056 to return to the caller.

PHR3028 The E-value is collected and validated. If the E-value is valid, transfer is to PHR3038, otherwise, an error is generated and transfer is to PHR3056 to return to the caller.

PHR3038 The constant is scaled by the P-value. Return is made to the caller.

INTEGER This subroutine collects integer values.

PHR3082 The subroutine ADVSUP is called to slide to the delimiter.

PHR3084 The integer value is collected.

PHR3106 Return is made to the caller.

WRITE, READ The READ and WRITE subroutines provide proper interface to the PLAN DISK IOCS subroutine to process the phrase dictionary file. The registers that are used in phrase are saved and then the proper I/O parameters are set for the disk I/O routines. The proper routine READ or WRITE is called. On return from the DISK I/O routine, the phrase registers are restored and return is made to the caller.

ERROR The ERROR subroutine provides standard interface for processing phrase errors. An error indicator is turned on and may be tested just before the phrase update is made. If the ECODE has not been provided, it is computed from the cursor. The error number and the ECODE are set and the error is logged by calling the WRITERR subroutine in the PLAN loader.

TUPDATE This routine performs the maintenance of adding or deleting phrases from PFILE.

PHR3272 PFILE is searched for a phrase of the same name as the one to be added or deleted. Note that for two phrases to be equal, their names must be equal and they must both be verbs or object phrases.

PHR3346-PHR3350 If the phrase already exists and this is an ADD PHRASE operation, an error is given to indicate the phrase already exists.

PHR3358 If the phrase was found and this is a DELETE operation, the phrase is marked as available space and transfer is to PHR3574.

PHR3368-PHR3374 If this is a DELETE PHRASE, an error is given to indicate that the phrase to delete cannot be found. If it is the delete part of an ALTER PHRASE, transfer is to PHRASOUT without giving the error. If this is an ADD PHRASE or the add section of an ALTER PHRASE, transfer is to PHR3384.

PHR3384-PHR3422 This code searches the availability table for a space large enough to hold the phrase to be added. If no space large enough is found, an error is given.

PHR3442-PHR3530 This code will read five records from PFILE and search those records for the best fit for the current phrase to be added.

PHR3574-PHR3732 This code will combine and chain together free spaces in the five records that are currently in core. It also updates the availability table to indicate the largest space available in each record.

PHR3738-PHR3792 If this is an ADD PHRASE, the last phrase in the chain, or the PWV table is now queued to indicate that the new phrase exists. If this is a DELETE PHRASE, then the chain pointer in the previous phrase or the PWV table is queued to indicate that this phrase is now deleted.

DFJPIDMP

The DFJPIDMP module is entered only as a result of its name being placed on the pop-up list. The module provides a dump of the last command processed by the PSCAN module. The command is currently in EBCDIC image in PFINPUTA section of PFILE, that is, the PLAN file dictionary. The message is printed on the devices indicated in the first position of erasable COMMON. Therefore, switch word 8 must be set to point to erasable COMMON.

PID190 The device on which the command is to be listed is picked up from the first position of erasable COMMON.

PID210 The single buffer set A is assigned to the device.

PID230 The INPUT subroutine is called to read the image of the last phrase into memory. Erasable COMMON is not used as an input area.

PID250 The number of characters in the phrase image is set to the print are by a call to PIOUT.

PID270 The number of characters in the phrase image is used to calculate an account of the number of words that are to be set to the print area.

PID290 The position at which the phrase image will be placed in the print area is set to position 8.

PID310 The pointer is initialized at the second word of the input area. The first word contains the character count of the phrase image.

PID330 The PEOUT subroutine is called to convert four characters of the phrase image to the print area.

PID350 The pointer is incremented to the next word of the input area.

PID370 A test is made to see if all characters of the phrase image have been set to the output area. If they have transfer is to PID430.

PID390 The print position indicator is incremented by four.

PID410 A test is made to see if the print line is currently full. If it is not transfer is to PID330.

PID430 The line is printed.

PID450 The print position indicator is reset to print position 8.

PID470 A test is made to see if the entire command has been printed. If it is not, transfer is to PID330.

PID490 The PIDMP routine is terminated by a CALL LRET.

DFJPIIN

This subroutine converts an A4 format field into a FORTRAN integer word.

IIA1050 The user arguments are accessed.

IIA1310 Leading blanks in the field are skipped.

IIA1430 If a sign is present, it is collected. If a negative sign was present, an indicator was set that causes the resulting number to be set negative for the user.

IIA1670 The buffer pointer is slid past blanks after the sign if any are present.

IIA1790 The integer field is collected and accumulated.

IIA2410 The result is stored in the user array and control is returned to the caller.

DFJPIOCS

This utility routine is invoked by the standard PLAN commands INPUT and OUTPUT. It uses the PLAN subroutine IOCS to switch the PLAN input/output devices.

The1subroutine IOCS is called to switch the PLAN devices. The subroutine LRET is called to return control to the loader.

DFJPIOUT

IOA1090 The user arguments are accessed.

IOA1750 The user word is converted to EBCDIC notation.

IOA2050 The EBCDIC characters are moved to the output field. Control is returned to the caller.

DFJPLAN (OS)

DFJPLAN is the mainline executive for the OS PLAN system. It resides in the first 640 words of blank COMMON. It is always located at the beginning of the partition or region.

PLA1930 Return is to the caller via an LPSW instruction.

APPROUT This is the start I/O appendix subroutine which switches the PLAN system into the supervisor state. It is entered from the IOS supervisor.

PLA2250 The PLAN RB is located.

PLA2350 The WAIT gate set by the STATESW is opened.

PLA2370 The PSW is saved from the RB.

PLA2390 The PSW is altered so that on return from IOS the system will be in the supervisor state.

PLA2410 The old SVC PSW is checked to see if it is the same as the PLAN RB PSW. If yes, transfer is to PLA2450; if not, control is returned to the I/O supervisor to abort the I/O operation.

PLA2450 The old SVC PSW is saved.

PLA2490 The SVC PSW is altered so that on return from IOS the system will be in the supervisor state. Control is returned to the I/O supervisor to abort the I/O operation.

STATESW This subroutine executes the EXCP that causes the start I/O appendage to be entered.

PLA2650 A WAIT gate is set. This gate is necessary in case the EXCP request is queued on the channel.

PLA2670 The EXCP is issued when the WAIT gate is turned off by the start I/O appendage control is returned to the caller in the supervisor mode.

RESETM This subroutine switches the system back to the problem state.

PLA2890 The caller's address is stored in the saved PSW.

RETURN This entrypoint is the normal return for all PLAN modules. Entry occurs here from execution of a CALL LRET of a FORTRAN RETURN statement.

PLA6470 The PLAN base is restored.

PLA6570 If this is not a local return transfer is to PLA9830.

PLA6630 The execution level, the caller's regs, and the caller pica element are restored and control is returned to the calling module.

CLEANUP This subroutine manages the PLAN program area.

PLA8210 The last level control block above the current execution level is located.

PLA8310 The new top of the program area is set.

PLA8450 The PCB change is truncated if necessary.

PLA8690 DYNAMIC file FD records are purged if any are present in the program area. Exit from this subroutine is to the CORCLEAN subroutine in the module DFJLODER.

URENT This point is entered on execution-time reference to an unresolved external reference.

PLA8870 The caller's registers are saved.

PLA8910 If this is a CALL LRET, transfer is to PLA6470.

PLA8950 The name of the external reference is placed in the pop-up program list.

LOCAL This entrypoint is entered from the LOCAL subroutine.

PLA9270 The current execution level is incremented.

PLA9330 A local control block is located. The LCB is located adjacent to and above a level control block.

PLA9430 If any LCB's are left, the transfer is to PLA9490.

PLA9470 An indicator is set to force a new segment level.

PLA9490 The caller's registers system status, etc. are saved in the local control block.

NEXTLOAD PLAN is entered here to load the next program.

PLA9830 A SPIE macro is issued.

PLA9970 If this is a local call, transfer is PLA10190.

PLA10070 If any errors have occurred, that is, if there are any errors on the stack in the phrase dictionary, transfer is to PLA11550.

PLA10190 If there is not an asterisk in the pop-up list indicating a checkpoint recall, transfer is PLA10370.

PLA10250 The current execution level is reset and transfer is to PLA8590.

PLA10370 The pop-up list is updated.

PLA10470 If there is a right parenthesis in the pop-up list transfer is to PLA10190 and this entry in the list is ignored.

PLA10570 If there is not a left parenthesis in the list, the transfer is to PLA10730.

PLA10610 A left parenthesis has been found in the pop-up list to indicate the start of a bank loading operation so the BANKA indicator is set on to indicate this. Transfer is to PLA10190.

PLA10730 If this is a checkpoint recall transfer is to PLA14750.

PLA10910 If the pop-up list contains a zero, transfer is to PLA16030.

PLA10950 A temporary pointer to the pop-up list is set and saved and will be used as a bank load list pointer.

PLA11130 If the program is not in core transfer is to PLA11550. This is determined by searching the PCB chain.

PLA11350 If bank loading is not in progress transfer is to PLA11730.

PLA11450 If the program is located below the segment which is equal to the current execution level, transfer is to PLA11730.

PLA11550 If the clean switch is not on, transfer is to PLA11690. The clean switch controls program area cleanup and free storage management. It is only performed once for every call to the loader.

PLA11590 The clean switch is reset, that is, the branch is turned on.

PLA11670 The subroutine CLEANUP is called to perform program area and free storage maintenance.

PLA11690 The subroutine LOADER is called in the module DFJLODER to load the names module.

PLA11730 If bank loading is in progress, transfer is to PLA11970.

PLA11830 The address of the PCB for the program just loaded is saved.

PLA11850 If the bank load start indicator is not on transfer is to PLA12150.

PLA11890 A bank load in progress indicator is turned on.

PLA11970 If the next entry in the pop-up list is a right parenthesis transfer is to PLA12150.

PLA12030 If the next entry in the pop-up list is not an asterisk transfer is to PLA10730.

PLA12150 This is the beginning of the final linkedit of unresolved external references. If we are at the end of the ERTAB2 or it does not exist, transfer is to PLA13710.

PLA12310 If the adcon has already been resolved in a previous pass transfer is to PLA13610.

PLA12350 If this is not a bank load transfer is to PLA13030.

PLA12470 The ENTAB's for programs in this segment are searched for a name equivalent to the external reference. If a hit is not made transfer is to PLA13030.

PLA12930 The entrypoint is extracted from the ENTAB and transfer is PLA13230.

PLA13030 Program area core is obtained for an unresolved adcon control block.

PLA13110 The unresolved adcon block is constructed including the name of the adcon plus a V type adcon pointing to PLAN COMMON.

PLA13230 The adcon is resolved to point to either the unresolved adcon block or the address found in the ENTAB entry.

PLA13350 The ERTAB2 entry for this external reference is flagged to indicate that this adcon has been resolved.

PLA13430 If this is the last ERTAB2 entry, transfer is to PLA13610.

PLA13490 If the name of the external reference is the same transfer is to PLA13230; otherwise, transfer is to PLA13430. This is a pass over the ERTAB2 entries to disolve all external references to the same name to the same unresolved adcon block.

PLA13610 The pointer to the ERTAB2 entries is stepped to the next entry and transfer is to PLA12150.

PLA13710 The subroutine FRERT is called to release the ERTAB2 table.

PLA13830 If a new segment level has not been created transfer is to PLA14130. A new segment is always created when either a program module is loaded or the local execution level goes beyond the number of segment levels currently in core.

PLA13870 A new level control block including a local save area is created.

PLA14130 The TRACE routine is called if TRACE was invoked.

PLA14230 If this is not a LOCAL call transfer is to PLA14470.

PLA14290 The execution level is incremented if necessary. This is done if a CALL LOCAL is executed which references a program that resides in a segment level which is more than greater than the current execution level.

PLA14470 The argument register is set for the called program.

PLA14550 All loader switches are reset.

PLA14610 Exit from PLAN to enter the program for execution.

CHPTIN This subroutine reloads a PLAN checkpoint.

PLA14750 The subroutine PLANLOPF is called to reset the system pointers and status.

PLA14790 If a checkpoint does not exist transfer is to PLA10190.

PLA14850 The checkpoint note pointers CURRNOTE and PREVNOTE are updated.

PLA14910 A read operation is set with a checkpoint bootstrap routine.

SCHPTR This routine reads and writes the checkpoint bootstrap. It is called as a subroutine from the LCHEX subroutine.

PLA15030 Any DYNAMIC file FD records in the program area are purged.

PLA15090 The argument for DIOCS are set.

PLA15250 Exit is to DIOCS to read or write the checkpoint. The return from DIOCS is set to enter the bootstrap itself.

ERRABORT This is the entrypoint to PLAN on a phrase abort.

PLA15730 The subroutine WRITERR is called to log the error onto the phrase dictionary.

PLA15810 The phrase abort indicator is set for PSCAN.

ERLSTENT This is the entry to PLAN from the ERLST subroutine.

PLA15890 DFJPERRS, the error processing module, is selected for execution.

PLA15910 The pop-up list is cleared and transfer is to PLA16250.

PLANLOPZ Enter here when the pop-up list goes to zero.

PLA16030 The pop-up list is cleared.

PLA16070 DFJPSTSV is selected for execution.

PLA16090 If Switch Word 2 indicates that saved statements are being processed transfer is to PLA16250.

PLANLOOP This is the entrypoint to PLAN to invoke the next command.

PLA16170 DFJPSCAN, the interpreter, is selected for execution.

PLA16250 The system status is reset. This includes clearing any checkpoints, resetting the LOCAL chain, and resetting the pointers for managed free storage.

PLA16330 The exit from the PLANLOPF subroutine is set.

PLA16410 If the program is in core transfer is to PLA16570. This is determined by searching the PCB chain.

PLA16550 This is the entrypoint for the PLANLOPF subroutine and the name control block is cleared.

PLA16570 The system status is reset including the address of the save area, the ERTAB2 if it exists, is released, current execution-level is reset to zero, and any loader indicators are reset. Exit from PLANLOPF is either to the checkpoint recall routine or back to the loader.

PLA16790 This is the entrypoint for the WRITERR subroutine. The error message is built in a work area.

PLA16890 The error message is written onto the phrase dictionary.

PLA17130 The error stack pointers are updated and control is returned to the caller.

PLA17230 This is the FRERT subroutine. The ERTAB2 is released by use of the FREEMAIN macro and control is returned to the caller.

SPIENT This is the entry to PLAN on the program interruption which is controlled by the PLANSYP macro.

PLA18070 Error message is built from the PSW.

PLA18530 The SPIE return for OS is set and control is returned to the OS supervisor.

PLA20130 The PLAN program area is cleared.

PLA20390 If the data set defined by the PLSYSTAB DD card is old, transfer is to PLA16170.

PLA20430 DFJPHRAS, the phrase dictionary maintenance routine is selected for execution to add the phrase 'ADD PHRASE' to the dictionary and transfer is to PLA16250.

PLANINIT This is the initial entrypoint for the PLAN system.

PLA21070 The registers are saved, bases are set in a save area, and pointers are set.

PLA21370 A determination is made if this is an MVT system. It is made by inspecting the CVT. If this is not an MVT system transfer is to PLA21470.

PLA21430 A PLAN indicator is set to show that this is a MVT system.

PLA21470 The address of the TCB and TIOT are saved for use during PLAN execution.

PLA21650 The program pop-up list is allocated and cleared and pointers to the end of the list and the current entry are saved in the PLAN COMMON area.

PLA21870 A special I/O save area, used by DIOCS is allocated. This save area eliminates the need for a save area in PLAN subroutine.

PLA22030 The EXEC card PARMs are collected.

PLA22290 If a PARM is not valid, transfer is to PLA23850.

PLA22430 The PARM is processed and transfer is to PLA24770.

PLA23850 An invalid PARM message is typed and transfer is to PLA33630.

PLA24770 The program COMMON area is allocated. This is done by using the GETMAIN and FREEMAIN macros. The length of the COMMON area is either the specification in the PGAR PARM field or 66 per cent of the partition or region size.

PLA25650 The module DFJLODER is loaded into the partition.

PLA25790 The module DFJTRACE is loaded into the partition if TRACE was invoked.

PLA26130 The subroutine TSRCHA is called to search the TIOT for the PLINP DD card.

PLA26310 If there was not a hit in the TIOT, transfer is to PLAN26910.

PLA26330 The subroutine OPENSEQ is called to open the PLINP data set.

PLA26350 If the PLINP data set is not open correctly transfer is to PLA26910.

PLA26450 The subroutine TSRCHA is called to search the TIOT for the PLOUT DD card.

PLA26510 If there is not a hit in the TIOT transfer is to PLA26910.

PLA26530 The subroutine OPENSEQ is called to open the PLOUT data set.

PLA26550 If the PLOUT data set did not open correctly transfer is to PLA26910.

PLA26650 The subroutine TSRCHA is called to search the TIOT for PLSEQ DD cards.

PLA26730 If a hit is not found in the TIOT transfer is to PLA27050.

PLA26750 The subroutine OPENSEQ is called to open the PLSEQ data set.

PLA26770 If the PLSEQ data set did not open correctly transfer is to PLA26650.

PLA26810 A TCLOSE macro is issued on the data set so that the first reference may be either READ or WRITE, then transfer to PLA26730.

PLA26910 An exit from PLAN is made via an ABEND 100.

PLA27050 The subroutine TSRCHA is called to search the TIOT fo the PLANLIB DD card.

PLA27110 If a hit was made in the TIOT search transfer is to PLA27210.

PLA27130 The subroutine DDERR is called to log an error message and transfer is to PLA27710.

PLA27210 The PLANLIB DCB is opened.

PLA27710 The subroutine TSRCHA is called to search the TIOT for the PLSYSTAB DD card.

PLA27770 If a hit is made in a TIOT search transfer is to PLA27870.

PLA27790 The subroutine DDERR is called to log an error message and transfer is to PLA29050.

PLA27870 The subroutine DSCHK is called to check the data set specifications and open the PLSYSTAB data set.

PLA27910 If the data set did not open correctly transfer is to PLA29050.

PLA28190 The data set defined in PLSYSTAB DD card is old transfer is to PLA29050.

PLA28390 The phrase dictionary is initialized.

PLA29050 The DCB for the SIO appendage routine is opened.

PLA29310 The PLMANFIL data set is opened if present.

PLA29650 The PLCHKPT data set is opened if present.

PLA29950 Any PERMANENT file data sets 'PLFSYnnn' are opened if present.

PLA30630 Any DYNAMIC drive 'PLANDRVn' are opened if present.

PLA32210 A BLDL is issued for the module names DFJPSCAN, DFJPERRS and DFJRETN to ensure that these modules are locatable in the PLAN library PDS. If they are transfer is to PLA32790.

PLA32510 The subroutine DDERR is called to log an error message and transfer is to PLA33570.

PLA32790 If the NFS PARM is not present transfer is to PLA33390.

PLA32870 The nonmanaged free storage array is allocated by using GETMAIN and FREEMAIN. A pointer to the nonmanaged free storage internal free queue chain is maintained in the PLAN COMMON area.

PLA33390 An initial SPIE macro is issued.

PLA33530 If no errors have occurred during initialization transfer is to PLA18470.

PLA33390 Control is returned to the OS supervisor.

PLA33770 This is the entrypoint for the OPENSEQ subroutine. A GETMAIN is issued and a DCB is created from a skeleton.

PLA34090 The JFCB to DCB merge is performed.

PLA34890 If the unit for the data set is a disk or a tape transfer is to PLA35690.

PLA35070 If the unit is not a card reader control is returned to the caller and an error indication is given.

PLA35690 The open parameter field is set for the device type.

PLA36310 The file is opened.

PLA36450 The control block, for the DCB is completed.

PLA36990 The first buffer area is cleared and control is returned to the caller.

PLA37870 This is the entrypoint for the TSRCHA subroutine. The TIOT is searched for the DD name in the argument.

PLA37910 If a hit is made transfer is to PLA38050; otherwise, control is returned to the caller, with an indication that no hit was found in TIOT.

PLA38050 The JFCB for the DD name specified is read into core and control is returned to the caller.

PLA39110 This is the entrypoint for the DSCHK subroutine. If the disposition on file is new transfer is to PLA41170. This is determined from the JFCB.

PLA39410 The DSCB is read from the VTOC using the obtain macro.

PLA39550 The DSCB is validated.

PLA39930 The format switch is reset and transfer is to PLA41810.

PLA41170 The allocation of the data set is validated. The DSORG specifications must be physically sequential.

PLA41710 The format switch is set.

PLA41810 The buffers for the data set are allocated by using the GETMAIN macro.

PLA41870 The control block associated with the data set is completed.

PLA42270 If the format switch is not on transfer is to PLA43330.

PLA42810 The data set is formatted using the QSAM access method. The values used are the FORTRAN word FALSE or X'7FFFFFFF'.

PLA43330 Control is returned to the caller.

PLA43610 This is the entrypoint for the DDERR subroutine. The error message requested is printed on PLOUT data set.

PLA43790 An abort indicator is set and control is returned to the caller.

DFJPLAN (DOS)

DFJPLAN is the resident loader and mainline control routine. It resides in the first 2560 bytes of COMMON.

ERRABORT This is the phrase abort entry to the loader. It is entered by all subroutines when an error is detected.

PLA3370 The error number and the ECODE are set.

PLA3450 The phrase abort indicator is set for DFJPSCAN.

PLA3470 If the DUMP option was selected via a PLAN run control card. The transient DUMP routine $$BDFJD is invoked to take a partition dump.

PLA6530 The system status is reset including the pop-up list, any checkpoints that are in effect, and if any module that had been previously loaded contained FIND/READ/WRITE, the FIND/READ/WRITE buffers are purged.

PLA3630 The module DFJPERRS, the error processor, is selected for execution and transfer is to PLA8930.

DUMPLIN This entry is used by various PLAN modules to print a line on the current PLAN output device.

PLA3870 The callers registers are saved.

PLA3890 The address of the current output device buffer is located.

PLA3970 The record area is moved to the output buffer.

PLA4110 The SIOCS routine is called to write the line on the output device and control is returned to the caller.

CCBSTART This routine is entered by DIOCS and SIOCS to cause execution of an I/O operation.

PLA3430 A switch indicating a start I/O is set and transfer is to PLA4510.

CCBWAIT This entry is used by SIOCS and DIOCS to force a wait on the last I/O operation.

PLA4510 A scan of the system CCB's is made for a CCB that was associated with the file control block which is the same as the callers.

PLA4590 If a CCB is found that is associated with the caller's FCB, transfer is to PLA4890. Otherwise, a test is made to see if the start I/O switch is on and if not, control is returned to the caller.

PLA4690 A scan is made for a free CCB, and if one is found, transfer is to PLA4890. Otherwise, a wait is issued for the last CCB in the string.

PLA4890 A test to check if an error occurred on the last I/O operation for this CCB. If no error occurred, transfer is to PLA5310. Otherwise, the error status is set in the caller's file control block. A test is made to see if a dump is in progress and if yes, transfer is to PLA5310 to ignore the error.

PLA5150 A test is made to see if errors are allowed by the caller's file control block. If yes, the transfer is to PLA5310; otherwise, transfer is to PLA3450 to cause a phrase abort.

PLA5310 A test is made to see if this is a start I/O operation and if not, control is returned to the caller.

PLA5410 The start I/O switch is reset and the EXCP is issued and then control is returned to the caller.

SRCHIOC This entry in the loader is used by all of the conversion control routines to validate the NOD argument for sequential files.

PLA5670-PLA6130 This routine searches the sequential file chain of control blocks to see of a NOD equivalent to the caller's is available. If yes, the address of the control block is returned in register 3. A test is also made to see if a file is available and not open. If the file is not open, the PLAN transient open routine $$BDFJSO is called to do an open on the file and then control is returned to the caller.

DLOADP This entry is used to load a program into core. The PCB must previously have been loaded with the disk address of the program plus its origin and end point.

PLA6830 The I/O argument registers are set and the READ routine is called to load the program into core. Control is returned to the caller from the DIOCS subroutine.

DIRLOOKU This entry is used to search the DOS core image directory for a program name.

PLA6690 The directory records are read and searched for the program name. If a hit is not found, control is returned to the caller. If the name is found in the directory, the CCHR or disk address of the program is calculated, and control is returned to the caller.

CLOCAL This entry is used by the LRET subroutine to clear the LOCAL chain.

PLA8150 The LOCAL chain is cleared and control is returned to the caller.

ISEARCH This entry is used by various routines in the PLAN system to locate the next entry in the PSCB table or the LOCAL chain.

PLA8390 A test is made to see if the PSCB table is present. If not, transfer is to PLA8590.

PLA8450 The next PSCB table is located and control is returned to the caller.

PLA8590 The next entry in the LOCAL chain is located and control is returned to the caller.

PLANLOPZ This entry is used whenever the pop-up list has gone to zero or by subroutines that want to clear the pop-up list and continue to cause execution of DFJPSCAN.

PLA8810 The pop-up list is cleared.

PLA8830 If a checkpoint is in effect, the NOD pointing to the current checkpoint is reset.

PLA8850 The module DFJPSTSV is selected for execution in case we are processing the statement SAVE.

PLA8870 If we are processing SAVE statements transfer is to PLA8930.

PLA8910 The PLAN interpreter, DFJPSCAN, is selected for loading.

PLA8930 The subroutine CLOCAL is called to reset any local processing in progress and transfer is to PLA9590.

PLA9030 The right parenthesis is floated up in the pop-up list and transfer is to PLA9470. This routine is used whenever a failure to load a program that is in the middle of a bank list or an asterisk in the bank list is encountered.

PLA9310 If the next name in the pop-up list is not a right parenthesis, transfer is to PLA9550.

PLA9470 The bank load switches are reset indicating the end of a bank load.

PLA9550 The bank load list pointer is saved.

PLA9590 The name of the program to be loaded is moved to the name control block.

PLA9610 A test is made to see if the name is a numeric zero. If yes, a branch is to PLA8810 to clear the pop-up list and load PSCAN.

PLA9650 A test to check if the name is an asterisk, which is the check point recall, is made. If yes, transfer is to PLA13850 to reload the checkpoint.

PLA9770 A search is made of the programs already in core and if the program is not in core, transfer is to PLA10250.

PLA9930 If we are in the process of bank loading, transfer is to PLA9310 to get the next program name from the pop-up list.

PLA9990 The subroutine WCHECK is called to see if any programs in core have to be checkpointed. Transfer is to PLA10370.

PLA10070 A test is made to see if we are bank loading. If yes, transfer is to PLA9030, otherwise, transfer is to PLA3450 to cause a phrase abort.

PLA10250 The subroutine DIRLOOKU is called to search the core-image directory for the program name. If a hit is not made, transfer is to PLA10070.

PLA10370 A test is made to see if the program will overrun the partition. If yes, transfer is to PLA10070.

PLA10530 A test is made to see if the program will overlay the PSCB table. If yes, transfer is to PLA11530.

PLA10590 A test is made to see if the program will overlay the COMMON area. If yes, transfer is to PLA10070.

PLA10790 A search is made of all the programs in core and any program that will be overlayed by the program about to be loaded are marked as such.

PLA11130 A test is made to check if the area required for the program is free. If yes, transfer is to PLA11910.

PLA11450 A test is made to check if there is room to create a new PSCB. If not, transfer is to PLA11530, otherwise, transfer is to PLA11910.

PLA11530 A test is made to check if we are bank loading. If yes, transfer is to PLA9030 to float the parenthesis in the pop-up list and stop the bank loading.

WCHECK This routine is entered any time a program in core may be overlaid. It will check to see if any programs have to be checkpointed.

PLA11610 A search is made of all programs in core to see if they must be checkpointed. If not, transfer PLA11910, otherwise, exit is to the LCHEX subroutine in each module that is to be checkpointed.

PLA11910 A test is made to see if any program in core will be overlaid. If not, transfer is PLA12070.

PLA11990 The subroutine WCHECK is called to checkpoint the program that is about to be overlaid.

PLA12070 The PSCB table is extended from the PCB in the loader.

PLA12230 The subroutine DLOADP is called to load the program.

PLA12250 If we are in a process of bank loading, transfer is to PLA9310.

PLA12290 The PARM's for the program to be called are set.

PLA12470 If the TRACE option was selected at initialization time, the TRACE routine is called.

PLA12550 Exit is from the DFJPLAN mainline to the program to be executed.

RETURN This entrypoint is the normal return from all programs executed under the PLAN monitor.

PLA12650 The PLAN base register is restored.

PLA12730 The subroutine CMCLOPTB is called to check if any loader errors had occurred.

PLA12750 The LOCAL return is traced if required.

PLA12890 A test is made to see if any execution errors have occurred while the last program was in control. If not, transfer is to PLA13070.

PLA12930 If the last module loaded was a LOCAL, transfer is to PLA13070.

PLA12970 The module DFJPERRS, the error processor, is selected for execution and transfer is to PLA8930 to load the module.

PLA13070 The pop-up list is updated.

PLA13150 If the next name in the list is a right parenthesis, transfer is to PLA13070 to ignore it.

PLA13190 If the next name in the list is not a left parenthesis, transfer is to PLA13270.

PLA13230 The bank load indicators are turned on and transfer is to PLA13070 to get the next name in the list.

PLA13270 The bank load list pointer is saved and transfer is to PLA9550 to set the name of the program to be loaded.

CMCLOPTB This subroutine checks to see if any loader error had occurred during execution of the last module.

PLA13390 DYNAMIC file FD records are purged if any are in the core area.

PLA13550 A test is made to see if an invalid overlay occurred of either COMMON or a LOCAL caller in a program area. If yes, transfer is to PLA3450 to cause a phrase abort. Otherwise, control is returned to the caller.

CHKPIN This routine reloads the checkpoint.

PLA13850 A test is made to see if we are bank loading and if yes, transfer is to PLA9030.

PLA13290 The subroutine CLOCAL is called to clear any LOCAL processing in progress.

PLA13930 A test is made to see if any programs have been checkpointed. If not, transfer is to PLA3450 to cause a phrase abort.

PLA14050 The checkpoint return is invoked if TRACE is invoked.

PLA14090 The module is reloaded and transfer is to PLA10370.

INITILP This is the initial entry point to PLAN from the DOS supervisor.

PLA15650 Base registers are set and a save area is established.

PLA16050 Pointers pertaining to the size of the partition are set in the PLAN COMMON area.

PLA16510 A test is made to see if an *ASGN PLAN control card was read. If not, transfer is to PLA17290.

PLA16610 The ASGN table pointers for the appropriate control card type are updated.

PLA17290 The next card is read.

PLA17350 A test is made to see if an end-of-file or an error occurred during the read. If not transfer is to PLA17490.

PLA17450 The card save indicator is reset so that the last card read will not be passed to PSCAN for processing and transfer is to PLA22570.

PLA17490 A test is made to see if the card just read contains an asterisk in column 1. If it does not, it is not a PLAN run control card and transfer is to PLA22570.

PLA17610 The card just read is listed on the output device designated by the SYSLST ASGN card.

PLA17830 A test is made to see if this is a valid run control card. If not, transfer is to PLA16510 to read the next card; otherwise, exit is to the control card processing routines.

PLA18410 The alternate library control card is processed. This includes determining the DOS system LOGICAL unit assignment for the alternate library.

PLA18590 The reserve core card is processed. This includes determining the length of the area required for the FORTRAN I/O area and user work area.

PLA18910 The input control card is processed. This routine determines the NOD of the PLAN input device to be used.

PLA19130 The output control card is processed. This card determines the output NOD to be used for PLAN output.

PLA19370 The max I/O card is processed. This card determines the number of CCB's that will be used for PLAN I/O.

PLA19570 The option control card is processed. This includes the setting of LIST, NOLIST, DUMP, NODUMP, PHRASE, and TRACE options.

PLA20090 The ASGN card is processed. This includes building tables for the designated type of file. These tables are processed later by the PARM processing routine.

PLA22570-PLA22890 The FORTRAN I/O and the user work area, if any, are allocated. This is done by using the subroutine DFJGMAIN which allocates core from the top of the partition. Pointers are saved from both of these areas in the PLAN COMMON area.

PLA23010 PLAN system modules are moved to the top of core. These modules include DFJSIOCS, DFJDIOCS, and DFJCNTRL. The subroutine DFJGMAIN is used to obtain core for these modules.

PLA23590 The system CCB's are created. Core is allocated using the DFJGMAIN subroutine.

PLA23850 The program pop-up list is allocated and cleared and pointers to the current entry and the end of the list are stored in PLAN COMMON.

PLA24050 The core image library is opened.

PLA24150-PLA24190 If an alternate library was specified in a control card, it is open.

PLA24470 The module DFJIOCBS which contains the PLAN I/O assignments is loaded by the module DFJDLOAD.

PLA24710 If any ASGN control cards were read, a merge is performed with the control blocks currently existing in the module DFJIOCBS.

PLA25710 The module DFJTRACE is moved to the top of core if the TRACE option was invoked by the option control card.

PLA26030 A test is made to see if PSCAN, PHRAS, and PERRS are in the core-image library. If not, transfer is to PLA30770.

PLA26390 All PLAN system files including the phrase dictionary, the checkpoint file, and the managed area file are opened.

PLA26770 A check is made to see that DFJPFILE, which is the is the phrase dictionary is formatted correctly. If not, transfer is to PLA30770.

PLA27010 If DFJPFILE is new, the program DFJPHRAS is selected for initial execution to bootstrap the ADD PHRASE phrase into the dictionary.

PLA27190 A test is made to see if a PLAN command was read while attempting to read PLAN control cards. If not, transfer is to PLA13390 to begin execution.

PLA27230 The command card is listed on the current output device and the command is saved in the PFINPUTB record of DFJPFILE and transfer is to PLA13390 to begin execution.

PLA30770 An error message is typed on the console indicating that PLAN execution is inhibited and an end-of-job macro is executed returning control to the DOS supervisor.

DFJPLENG

This utility routine is invoked by the standard PLAN command SET PAGE LENGTH. It uses the PLAN subroutine PLENG to set the number of lines per page for an output device.

PLI110 The subroutine PLENG is called to alter the page length for the device. The subroutine LRET is called to return control to the loader.

PLE130 Control is returned to the caller via the LRET subroutine.

DFJPLITL

This module is used in conjunction with the LIST LITERAL command to provide a listing of all literals stored in a PLAN literal file. The first six positions in the communication array are required for the storage of data from the SET LITERAL command for use by this module. The communication array is also used as an input/output area and a work area so that no data may be carried through execution of the LIST LITERAL command. The LIST LITERAL command is a Level 1 phrase which will automatically cause initialization of the managed communication array.

Halts: None
Errors: None
Subroutines: GDATA, RDATA, PDBFA, PAOUT, PCCTL, PLOUT; NDEF, LRET, PDUMP, PIOUT, PEOF, SUBSC

PLI220 The file number specified in the first position of the communication array is set to the file control block.

PLI240 The logical drive code index is set to the fifth position of the communication array.

PLI260 GDATA is called to open the literal file.

PLI280 The header of the literal file is read into memory at the 20th position of the communication array.

PLI300 The highest literal number currently in the literal file is picked up from the file header.

PLI320 PDBFA is called to initialize PLAN I/O and to establish a double buffer for the listing operation.

PLI340 PAOUT is called to set the dump heading to print.

PLI360 The dump heading is printed following a skip to a new page.

PLI390 An index is set to the next literal to be extracted.

PLI400 If there are no more literals to be extracted, transfer is to PLI500.

PLI410 A pointer is set to the literal index in the literal dictionary.

PLI430 An RDATA call is issued to read the literal index into memory.

PLI450 A check is made to see if the literal exists. The literal exists only if the index is a REAL value. The REAL value represents a displacement in the file at which the literal text must be found. If the literal does exist transfer is to PLI540.

PLI470 The index is incremented to the next literal number.

PLI480 If there are more literals to process transfer is to PLI410.

PLI500 The printer is skipped to a new page.

PLI520 An exit from the module is made via a CALL LRET.

PLI540 The literal header is read into memory by a call to RDATA.

PLI550 A check is made to determine if the just read into memory matches the literal number that is searched for. If the literal number does not match, a program error is indicated. If a match is found, transfer is to PLI590. Otherwise, a dump of the input area is produced and transfer is to PLI500.

PLI590 The number of characters in tbe literal is extracted from the literal header.

PLI610 The literal number and the number of characters in the literal is set to print.

PLI630 The literal text is read from the file.

PLI670 The literal text is set to print.

PLI710 If the logical end-of-file switch for the printer is not on, transfer is to PLI730.

PLI720 The printer is skipped to channel one.

PLI730 The literal number, number of characters, and literal text is printed. Transfer is to PLI470.

DFJPMERG, DFJGMERG

DFJPMERG and DFJGMERG are the routines that merge PLAN DYNAMIC PERMANENT files respectively. They are invoked through the LCHEX subroutine by the subroutines PMERG and GMERG. The merge is a standard two-way merge. Any out-of-sequence condition on either of the input files will cause a phrase abort. ID(2) of the file control block is updated to reflect the size of the merged file. The file control blocks of the input files are unchanged. The only difference between DFJPMERG and DFJGMERG is that the DYNAMIC file merge routine uses the FIND/READ/WRITE subroutines and the

PERMANENT file merge routine uses the RDATA/WDATA subroutine.

MER890 The ID blocks for the two files to be merged are moved from the sort work area into two working ID blocks. The location of ERASABLE COMMON is determined and the address of the merge control field is set in the working storage area. The merge control fields are validated.

MER2290 The amount of available core for merging is calculated. On OS, this will be all of available core outside of the MERGE program area. On BOS, this will be a fixed buffer area within the program area.

MER2430 The addresses of the merge area are set and pointers to the beginning and end of both input areas are set.

MER2750 ID(2) for both input files is rounded to the nearest record length.

MER3010 The input record area for both files are primed. This is done by calling the subroutines GETA and GETB.

MER3050 The address of the current A area record is set as a winner.

MER3070 If flushing A area records, transfer is to MER3250.

MER3110 The address of the B area record is set as a winner.

MER3130 If flushing B area records, transfer is to MER3250.

MER3220 The subroutine SORTZ is called to compare the A area and the B area records.

MER3250 The winning record is moved to the output area.

MER3710 If the output area is not full, transfer is to MER3850.

MER3750 The subroutine FLUSHOAR is called to perform a WRITE on the output area.

MER3850 A link register is set so that the return from either the GETA or GETB routine is to MER3050.

MER3870 If the A area record was the winner, transfer is to MER4310.

MER3990 If the B area is empty, transfer is to MER5290.

MER4050 The pointer to the current B area record is updated and return is made to MER3050.

MER4310 If the A area is empty, transfer is to MER4710.

MER4390 The address of the current A area record is updated and return is made to MER3050.

MER4710 If an end-of-file has occurred on the A file transfer is to MER5150.

MER4930 The address of the current record and the end of the A Area is set. Transfer is to MER4390.

MER5150 If flushing A area records, transfer is to MER5870.

MER5190 The flush switch is set for the B area. Transfer is to MER3050.

MER5290 If an end-of-file has occurred on the B file, transfer is to MER5730.

MER5330 The next B file block is read.

MER5510 Pointers to the current B area record and the end of the B area are set. Transfer is to MER4050.

MER5730 If flushing B area records, transfer is to MER5870.

MER5770 The flush switch is set for the A area records and transfer is to MER3050.

MER5870 The end-of-job switch is set and transfer is to MER5950.

FLUSHOAR The FLUSHOAR subroutine writes the output area.

MER5950 The current output area is written on the output file. If the end-of-job switch is not on, transfer is to MER6190.

MER6150 The merge has been completed and exit is to the next load entry in DFJPLAN mainline.

MER6190 Control is returned to the caller.

SORTZ The SORTZ subroutine is used to compare two records. The results of the sort are set in the register WINNER.

MER6690 The merge control fields are located from the working area bucket in the mainline.

MER6770 The merge field is located.

MER6910 The fields are compared. This is done by branching to an appropriate compare routine for the type of sort field. Return is to MER6930 if the compare is equal.

MER6930 If this is not the last sort control field transfer is to MER6770.

MER6970 Return is to here if an unequal compare is found by the compare routine. The A area record is assumed to be the WINNER.

MER6990 If the records were equal, control is returned to the caller.

MER7010 If the A record was high, transfer was to MER7090.

MER7030 If this is an ascending merge, control is returned to the caller with A as the winning record.

MER7090 If this is a descending merge, control is returned to the caller with A as the winning record.

MER7130 The B area record is set as a WINNER and control is returned to the caller.

DFJPSCAN

DFJPSCAN is the central language processor and interpreter for the PLAN system. It is brought into core and given control by the PLAN loader whenever there are no program names in the pop-up list. It may also be given control by CALL LEX as is the case when the subroutine PUSH is executed within the user's module.

DFJPSCAN's order of execution is:

1. The next command to be interpreted is accessed.

2. A PFILE dictionary lookup is done to access the appropriate object and verb phrases for this command. The object phrase is brought into core and a pointer to each of the verb phrases saved in a table.

3. Level management is performed according to the level of the previous phrase and the level of the current object phrase. That is, the managed array is defined is set to FALSE, saved, restored, or left untouched.

4. The symbols from the object phrase (Table 3) are added to the appropriate level symbol table. Starting with the rightmost verb in the command the symbols from each verb phrase are also added to the symbol table.

5. The initialization values (Table 2) from the object phrase are placed in the communication array. Starting with the rightmost verb, the initialization values from each verb phrase in the command are also placed in the communication array.

6. The input stream from the end of the command to the semicolon is scanned. This scan is done in sequential order and includes any data values given, user exits, and expressions.

7. The phrase-defined expressions (Table 6) program lists (Table 4), and check entries (Table 5) are processed in that order from the object phrase.

8. Starting with the rightmost verb phrase in the command, Step 7 above is repeated for each verb.

Halts: None
Errors: All PLAN errors produced directly by PSCAN are in the range from 200 to 299.

Subroutines: LCHEX, LIST, LRET, ERRET, ERRAT, ERROR, PLINP, PAIN, PEOF, PAOUT, PIOUT, PLOUT

DFJPSCAN This is the entrypoint to the PSCAN module.

PSC040 The current status of all registers is saved and then set up for PSCAN execution.

PSC070-PSC074 The special exit from the CHTEST routine is set so that the first call to CHTEST will result in a branch to INITGCHR to process the first record of the command. The first character in the input stream is initialized to hex 00.

PSC084-PSC096 If the repeat switch has been turned on as a result of a call to LREPT or PUSH, or because of a phrase being pushed from a check entry. The current command to be processed is read from PFINPUTA in PFILE. If the last record of the previous command contained residual characters following the semicolon, that residual record is read is read from PFINPUTB.

PSC098-PSC110 All of PSCAN's internal switches are turned off. The switch in the error routine is set to perform a call to ERRAT on any errors given and the phrase checksum table is read in from PFILE.

PSC112-PSC126 The first character of the command is checked for numeric. If it is, the statement save switch checked at PSC1452 is turned on and the input pointer is advanced over the statement number.

PSC142-PSC154 This code initializes switches and pointers used within BSCAN.

PSC156-PSC180 This code will collect one word (3 characters) from the command, compute its checksum, and place it in the phrase name collect area.

PSC184-PSC198 This code will pad out a phrase word with blanks if required.

PSC202-PSC220 This code uses the checksum for that part of the phrase name collected to this point as a pointer into the checksum table. If the checksum table entry indicates that there are verbs in this phrase chain, the verb encountered switch is turned on and the phrase chain pointer is saved for use at PSC296.

PSC224-PSC228 If the current character is not blank, the last word is padded with blanks if necessary before checking for a double quote mark.

PSC230 If the current character is not a double quote transfer is to PSC338 to terminate the collection of the command name.

PSC234 If the verb encountered switch is on, transfer is to PSC292 to search the current phrase chain for a verb by the same name as that part of the command collected to this point.

PSC238-PSC254 The input pointer is set to the communication array and the special exit from CHTEST is set so that the call to that routine at PSC156 will transfer control to PSC260.

PSC260-PSC286 This code will take three characters from one 32-bit word in the communication array to be used as the next word in the command name.

PSC292-PSC324 This code searches the phrase chain for a verb of an equal name. If the verb is found, the verb program list (Table 8) is moved to the pop-up list.

PSC328-PSC336 If over eight verbs are processed an error is given and the collection of the command name is continued.

PSC338-PSC346 The verb encountered switch is turned off and the checksum of the command name is used to pick up the checksum table entry for the object phrase.

PSC348-PSC360 If the command name began with a comma or a semicolon, the pointer to the previous object phrase saved during the last execution of PSCAN at PHRGOT is used to access the object phrase.

PSC362-PSC370 If the object phrase was found, the no-compare switch is turned on for the GETPHENT routine and transfer is to PHRGOT. Otherwise, an error is given and transfer is to ABORTERR to terminate PSCAN execution.

PHRGOT This routine contains all the logic pertinent to phrase level processing. The pointer to the current object phrase is saved in the loader.

PSC386-PSC390 The symbol table write complete switch is turned on and the symbol table is initialized.

PSC394-PSC414 This code checks to see that the requisite of level 0 and level 1 command have been met.

PSC418 The levels of the current and the previous command are combined and used as indicators in the level management of the managed array and the symbol table.

PSC438-PSC462 This code is the level error recovery logic. In other words, if the error recovery switch is on in the loader, it will be turned off if the level of the current command is equal or higher than the last.

PSC472 The appropriate level symbol table is read from PFILE so that the symbols from this command can be added to it.

PSC476 The level shift bits discussed in the introduction are turned on for the FIND/READ/WRITE subroutine.

PSC496-PSC610 If the current command is Level 2, 3, or 4, the managed array, if defined, is saved or restored.

PSC612-PSC614 The level of where to start saving the symbol table in PFILE is set and the subroutine SYMTBGET is called to add the symbol tables from this command to the current symbol table.

PSC616-PSC618 The symbol table write complete switch is turned off if the current command is a Level 0 or utility level. It is turned on for all other level commands. This means that the symbol table for Level 0 or utility commands is not saved on disk and cannot be referenced by other commands. SYMTBPUT is called to initiate saving of the current symbol table.

PSC620 DATAGET is called to place any initialization values for this command into the communication array.

PSC632-PSC6322 If the current command is utility level the utility level switch is turned on in the loader and transfer is to PSC612.

PSC6324-PSC6462 This code is executed only on Level 0 commands. The error queue file, if defined, is dumped, the Level 0 encountered switch is turned on in the loader, the command sequence number is reset to zero, the 15 PLAN switch words are set to system default values, and a pointer to the maximum size of COMMON is set for this PLAN job.

PSC650-PSC696 This code is executed on Level 1 commands. The error queue file is dumped and the symbol table write level is set so that the current symbol table will be saved four times. If the current command is Level 1 (not Level 0), the Level 1 found switch is turned on. The Level 1 shift switch for FIND/READ/WRITE is turned on and the managed array, if defined, is set to FALSE.

DUMPERRS This subroutine will dump the error queue file if there is any significant information in it. The repeat switch is saved, turned on, and restored so that if the checkpoint to PERRS is not successful, PSCAN will repeat the current command. That is, if there is no checkpoint file.

INPUTRD This subroutine is called whenever a null (hex 00) is found in the input scan area indicating that another record should be read from the current PLAN input device.

PSC724-PSC742 The pointer to the current input character is saved and the input stream is scanned looking for a NULL or semicolon. If a semicolon is found, the input pointer is restored and control is returned to the caller.

PSC744 If this is the first call to this routine during this execution of PSCAN, transfer is to PSC808 to start reading the first record and immediately return control to the caller.

PSC754-PSC782 This code checks for either logical or physical end-of-file on the current PLAN input device. On logical end-of-file transfer is to PSC1499 to produce an error diagnostic and terminate the current execution of PSCAN. On physical end-of-file the error queue file, if any, is dumped and the current execution of PLAN is terminated by scheduling the module DFJRETN which will return control to the OS/DOS supervisor.

PSC788-
PSC7905 The next 80-character record is transferred to the input scan area and if the LIST option is on that 80-character record it is printed on the current PLAN output device.

PSC792-
PSC808 If the record just brought in is all blank, a NULL is placed at the beginning of the record which has the effect of ignoring blank cards. Otherwise, the ID field (columns 76-80) are saved and a NULL is placed at column 76.

PSC812-
PSC822 The current record is searched for either a NULL or semicolon. If a NULL is found, a call to PLINP is issued to start reading the next record.

PSC826-
PSC832 The number of characters in this record is added to the total character count, and if not over 450, control is returned to the caller.

PSC836-
PSC848 This code will initiate a loop which will read records from the input stream until either physical end-of-file or semicolon is read. An error is given and the current execution of PSCAN is aborted.

INITGCHR This routine does special processing on the first record of a command. It is accessed only via the special exit from CHTEST and returns to CHTEST after execution.

PSC854-
PSC862 The input area is scanned to find the first nonblank character. The remainder of the input area is then moved down to the beginning to cover the leading blanks.

PSC866-
PSC878 The input area is scanned until a NULL or semicolon is found and the initial character count of the first record is set.

PSC884-
PSC916 This code will make sure that there is at least one nonblank character in the input area. Leading blanks are again suppressed in the case where the first record is read from the current PLAN input device.

PSC920 The input pointer is set to the beginning of the input area and transfer is to PSC4724 to return control to CHTEST.

SYMTBGET This subroutine is called from the PHRGOT routine to process the symbol table for the current command. The symbol table (Table 6) from the object phrase is first added to the in-core symbol table. A symbol table from each verb phrase in the command is then added starting with the rightmost verb. As each symbol is added a search of the in-core symbol table is performed to check for a duplicate definition. If a duplicate is found, the current definition replaces the old. If the CAP reference for any symbol is found to be symbolic, the expression is evaluated to resolve the CAP pointer before placing the symbol in the symbol table.

PSC936-
PSC942 A call to the DISKWAIT subroutine is issued to make sure the current symbol table is in core. If the level of the current object phrase is not blank, then its level is placed in the symbol table header.

PSC944 A pointer is initialized to the first available space in the symbol table.

PSC946 The execution-defined symbol subscript switch is turned on. This switch is used by the expression evaluation routines under error conditions to determine which error should be given.

PSC954-
PSC958 A disk wait is issued to make sure the current phrase is in core and SRCHCT is called to search for Table 3 in the phrase. If there are no symbols in this phrase, transfer is to PSC1190.

PSC960-
PSC962 The initial symbol not collected switch tested at PSC1086 is turned off. Note that this is done on each phrase of the command so that the eventual implied symbol will be the leftmost symbol in the leftmost verb phrase. If the current symbol does not have a symbolic CAP pointer, transfer is to PSC1082.

PSC980-PSC990 The input pointer is set to point to the beginning of the symbolic subscript expression and is evaluated to ARITHEXP.

PSC994-PSC1000 If a logical value was found in the evaluation of the expression the result is set to zero to force an error at PSC1018.

PSC1002-PSC1012 The input pointers saved at PSC980 is restored and a disk wait is issued if the current symbol table is being read back into core.

PSC1014-PSC1024 The result of the expression is converted to fixed point and if the result is not positive, an error is given and the CAP pointer for the current implied symbol saved at PSC1098 or at entry to PSCAN is used as the cap pointer for this symbol.

PSC1042-PSC1046 If the subscript is not less than 16,384 or 512 with P-value, an error is given.

PSC1062-PSC1082 The subscript is combined with the symbol to make a symbol table entry, and the Table 3 pointer is incremented over the subscript expression. The symbol table entry is then made.

PSC1084-PSC1098 The symbol table pointer is incremented to the next available space and if the initial implied symbol not collected switch is off, and the current symbol is not in reference to the switch words it is saved as the initial implied symbol.

PSC1102-PSC1126 The symbol table is searched for a duplicate definition of the current symbol. If found, the old definition is deleted.

PSC1186-PSC1190 If all the symbols for this phrase have been processed, the symbol subscript switch turned on at PSC946 is turned off.

PSC1194-PSC1200 If the current command contains verb phrases, the next verb to the left is read in and transfer is to PSC944 to process its symbols, otherwise, control is returned to the caller.

PSC1214-PSC1240 This code is used as a common exit from both SYMTBGET and DATAGET. It issues a call to GETPHENT to read the object phrase back into core and returns control to the caller.

DATAGET This subroutine is called from the PHRGOT routine to place initialization values (Table 2) from the current command into the communication array. The order of processing with respect to verb phrases is the same as that for SYMTBGET.

PSC1258-PSC1264 A disk wait is issued to make sure that the current phrase is in core and SRCHT is called to search for Table 2 in the current phrase. If there are no initialization values for this phrase transfer is to PSC1290.

PSC1268 The pointers are initialized for the loop through Table 2 initialization values.

PSC1274-PSC1288 This code will loop until all the initialization values for the current phrase are processed. If an execution-defined CAP pointer is encountered or an Implied Do, an appropriate branch is taken.

PSC1290-PSC1298 If there are no verbs in the current command, control is returned to the caller; otherwise, the next verb to the left is read in and transfer is to PSC1258 to process its initialization values.

PSC1308-PSC1328 This code performs a symbol table lookup to find the CAP pointer for the current initialization value. This is done in the case where the CAP pointer was symbolic at phrase-definition time.

PSC1332-PSC1360 This code processes Implied Do subscripts. It will place the initialization value in the communication array the specified number of times.

BALG This routine is executed after PHRGOT. It completes the scan and evaluation of the current command.

PSC1380 If the first character after the command name is not a colon, DATAIN is called to scan the input stream from the command name to the semicolon.

PSC1382 The EBCDIC image of the current command is saved in PFINPUTA of PFILE.

PSC1384 A disk wait is issued to make sure that the current phrase is in core.

PSC1386-PSC1418 This code processes phrase-defined expressions (Table 6), program lists, (Table 4), and check entries (Table 5) first from the object phrase and then from verb phrases starting with the rightmost verb.

CHOVR This routine is always executed to terminate the execution of PSCAN. It performs final housekeeping and sets indicators necessary for intermodule communication. If the phrase skip switch was turned on at PSC456 transfer is to PSC1466.

PSC1434-PSC1438 If the current command is not blank or utility level, its level is saved in the resident loader.

PSC1450 SYMTBLOP is called to ensure that the current symbol table is saved on disk.

PSC1452-PSC1456 If the statement saved switch is turned on at PSC122, then PSTSV is placed in the pop-up as the first program to be executed to save the current command in the statement save file.

PSC1459 If a phrase was pushed from a check entry, transfer is to PSC040 to reenter PSCAN and evaluate that command.

PSC1460-PSC1463 If the phrase skip or phrase error switch is on, the program pop-up list is cleared before returning control to the resident loader.

PSC1466-PSC1472 If this command is to be skipped due to level error recovery and there are no errors in the current command, then the phrase skipped error is given for transfer to PSC1450.

PSC1499-PSC1501 This error abort processing is used in those cases where continuation of the scan is impossible. Entry at PSC1499 will place a semicolon at the end of the last record to ensure that no more records are read.

PSC1502 The pointer to the current object phrase saved in the resident loader by PHRGOT is cleared to prevent repeating of the current command.

SYMPTLOP This subroutine when executed will ensure that all levels of the current symbol table are saved in PFILE.

PSC1508-PSC1510 SYMTBPUT is called to write the next level symbol table in PFILE. If the symbol table write complete switch is not on, transfer is to PSC1516; otherwise, control is returned to the caller.

PSC1516-PSC1518 A special entry is taken into the GSYM routine to ensure that the current symbol table is in core and transfer is to PSC1508.

SRCHCT This subroutine will search the phrase entry currently in core for a specific table by number. It will return to the caller a pointer to the beginning of the table and its length. If the table does not exist, the length will be zero.

PSC1592-PSC1594 A pointer is incremented over the phrase name to point to the first table.

PSC1604-PSC1618 This code will loop chaining from table to table until the table requested is found or the end of the phrase.

ERROR This subroutine is called to process any errors encountered during PSCAN execution.

PSC1668 If the GO TO search switch is on, the error is ignored. Note that during a GO TO search, the input stream is being scanned but not processed.

PSC1672-PSC1676 Unless this is a continue type message being produced from a check entry, the error found switch is turned on to indicate an error was found in the current command.

PSC1678-PSC1682 200 is added to the error number and the high-order bit is turned on to indicate to DFJPERRS that this is a system error.

PSC1690-PSC1694 The symbol table save operation if not finished is completed and the input image if not already saved to PFILE.

PSC1696-PSC1702 If an ECODE has not been supplied by the caller, an input cursor is computed and used as the ECODE.

PSC1710-PSC1722 A call to ERRAT or ERRET is made to process the error and control is returned to the caller.

EDUMP If errors have been queued to file 255 on LOGICAL drive 0, this subroutine will perform a checkpoint to DFJPERRS to dump file 255 to the current PLAN output device.

PSC1726 If the queue file valid switch has not been turned on by either EWRIT or DFJPERRS, control is returned to the caller.

PSC1730-PSC1732 This code in effect performs a call to ERLST with DFJPERRS returning control PSC1736.

GETPGM This subroutine will transfer program names from a phrase entry to the pop-up list in the resident loader. It gets as a parameter the number of the table within the phrase and the program list to be processed.

PSC1748-PSC1750 SRCHCT is called to search for the appropriate program list table. If the table does not exist, control is returned to the caller.

PSC1754-PSC1766 The table length is converted to 32-bit words as the first parameter in the CALL LIST and LIST is called to move the program list.

GETPHENT This subroutine is called to access a phrase from PFILE. The parameter passed to it is either a direct pointer to the phrase or a pointer to the first phrase in a chain of phrases in an equal checksum. This pointer is of the form xxyy where yy is the relative record in the phrase entry area of PFILE, and xx is the displacement into that record to the beginning of the phrase.

PSC1780 The pointer to the next phrase in the chain is taken from the current phrase header.

PSC1786-PSC1796 The pointer to the next record in PFILE containing the next phrase in the chain is extracted. If this pointer is zero, it indicates the end of the phrase chain and control is returned to the caller.

PSC1810-PSC1816 The record containing at least the beginning of the next phrase in the chain is read into core.

PSC1818-PSC1828 If this is a phrase search operation, the record just read is checked to make sure that it contains at least the phrase name. If not, GETPH4 is called to read in the rest of the phrase entry.

PSC1830-PSC1834 The phrase just read is checked against the compare phrase and if they are not both verbs or object phrases and their names equal, transfer is to PSC1780 to access the next phrase in the chain.

PSC1852-PSC1858 GETPH4 is called to start reading in the rest of the phrase. The phrase found exit is set and control is returned to the caller.

PSC1860-PSC1876 This code will loop reading as many records as necessary from PFILE in order to bring the whole phrase entry into core.

GETPH4 This subroutine is called by GETPH to read in the remainder of the phrase if any. Note that phrases may be up to 512 16-bit words long and therefore may occupy up to three records in PFILE.

PSC1872-PSC1874 If the whole phrase is already in core, control is returned to the caller. Otherwise, the number of words left to read are calculated and transfer is to PSC1868.

PESTCHKE This routine is called to evaluate the check entries associated with the phrase currently in core.

PSC1894-PSC1896 SRCHCT is called to search for Table 5 in the current phrase. If Table 5 does not exist, control is returned to the caller.

PSC1910 If the current entry has an execution-defined symbolic CAP, transfer is to PSC2070.

PSC1920-PSC1921 The CAP pointer is converted to a COMMON location and checked against the current size of COMMON. If the subscript is outside of COMMON, an error is given and transfer is PSC1932 to access the next check entry in the table.

PSC1924-PSC1928 If the value in COMMON is not FALSE, transfer is to PSC1986 for further checking. Otherwise, if this is not an *F entry transfer is to PSC1994 to process the action called for.

PSC1932-PSC1946 This code increments to the next entry in the table and checks for the end of the table. When the end of the table is found, control is returned to the caller.

PSC1952-PSC1974 This code will be executed when a check entry fails and a literal diagnostic is called for.

PSC1982-PSC1990 This code checks the value in COMMON against the value requested by the check entry and takes the appropriate processing branch.

PSC1994 If there is no additional information with this check entry, such as a literal or program list, transfer is to PSC1952 to produce PLAN diagnostics 220 through 223.

PSC1998-PSC2008 The pointer is set to the literal within the check entry or to COMMON whichever was defined at phrase-definition time. If a subscript is indicated, the subscript is checked to see that it is within COMMON. If not, an error is given and transfer is to PSC1932 to get the next table entry.

PSC2010 If this is a literal diagnostic request, transfer is to PSC1952.

PSC2014-PSC2020 If this is a program list entry, the called for program list is moved to the resident loader.

PSC2044-PSC2050 If the statement save switch is on, indicating that the current command is to be saved in the statement saved file, it is turned off. A diagnostic is given indicating that pushed phrases from a check entry and an implicit statement save are incompatible.

PSC2052-PSC2064 The pushed phrase is saved in PFINPUTA of PFILE and the repeat switch is turned on to cause execution of that command.

PSC2070-PSC2100 This code performs a symbol table lookup on those checkentries performed on a symbolic CAP pointer.

SYMTBPUT This subroutine is called to save the current symbol table in PFILE. Note that if the current command is a Level 1, the symbol table is saved four times. If Level 2 it is saved three times etc. The saving of the symbol table is processed so that it goes on concurrently with execution wherever possible. If this routine determines that execution time would be impaired by initiating a disk operation it will act as a no-op.

PSC2114 If the symbol table write complete switch is on, control is returned to the caller.

PSC2118 The special exit from CHTEST is set with the address of PSC2166. This will result in a call to CHTEST possibly initiating another symbol table save if warranted.

PSC2128 If the current symbol table to be saved is not in core, control is returned to the caller.

PSC2136-PSC2148 The symbol table is written to the appropriate save area of PFILE. If this is the Level 4 symbol table, the symbol table write complete switch is turned on and the special exit from CHTEST is turned off.

PSC2166 CHTEST will pass control to here as a result of the special exit set at PSC2118. A call to SYMTB-PUT is executed and control is returned to CHTEST.

RSYM This routine is called to read and unformat values from the communication array.

PSC2268-PSC2270 GSYM is called to lookup the current symbol in the symbol table. The RSYM/WSYM TRUE and FALSE switches are turned off.

PSC2272 If the subscript for the symbol is outside of COMMON, an error is given and control is returned to the caller.

PSC2276-PSC2286 The value is read from COMMON and checked for TRUE or FALSE. If yes, the appropriate RSYM switch is turned on.

PSC2290-PSC2294 If this value was last written as fixed-point, it is converted back to floating-point.

PSC2306-PSC2316 If the value was adjusted by a P-value when written to COMMON, it is now divided by the same P-value.

PSC2320-PSC2324 If the value read was TRUE or FALSE, the appropriate switch is turned on and control is returned to the caller.

WSYM This routine is called to format and write values to the communication array.

PSC2340-PSC2352 If the WSYM TRUE or FALSE switch is on, the appropriate value is placed in the output bucket and the WSYM literal switch is turned on to suppress formatting.

PSC2354 GSYM is called to lookup the current symbol in the symbol table.

PSC2356-PSC2364 If PSCAN is currently in a GO TO search or in the FALSE leg of a TRUE expression, or the TRUE leg of a FALSE expression, control is returned to the caller without writing the value to COMMON.

PSC2368 If the subscript is not in COMMON, an error is given and control is returned to the caller.

PSC2372 If the WSYM literal switch is on, transfer is to PSC2410 to suppress formatting.

PSC2376-PSC2380 If the current symbol has a P-value, the value to be written to COMMON is multiplied by it.

PSC2382-PSC2386 If the current symbol is fixed-point the value is adjusted by a plus or minus 0.5 and converted to fixed-point.

PSC2410-PSC2414 The value is written to COMMON, the RSYM and the WSYM switches are turned off and control is returned to the caller.

USYM This subroutine is called during the scan of the input stream to test for a user-exit associated with the current symbol.

PSC2428 The no user-exit processed switch is set.

PSC2430-PSC2438 GSYM is called to look up the current symbol. The relative subscript is restored by decrementing it by one. The symbol table entry is checked for a user exit. If there is none, control is returned to the caller.

PSC2442-PSC2448 If PSCAN is currently in a GO TO search, the inhibit switch (ISW) is turned on so that the user exit will not store values into COMMON.

PSC2450-PSC2456 The name of the user exit program is accessed for the call at PSC24685 and the current COMMON location is calculated and put in ISUBS for the NUSER subroutine.

PSC2466-PSC2468 A check is made to make sure the symbol table and the current command are saved on disk in case the user exit program does not return successfully.

PSC24685 The user-exit program is called as a LOCAL.

PSC2480-PSC2496 The error parameters returned by the user exit in the call to EUSER, if any, are processed.

PSC2504-PSC2516 PSCAN's pointer to COMMON is updated to reflect the last location used by the user exit program and control is returned to the caller.

LUSYM This subroutine is used by DATAGET and TESTCHKE to lookup execution-defined symbol subscripts.

PSC2526-PSC2542 A special entry is taken into the GSYM routine to look up the compressed symbol. The COMMON location is converted to a subscript and control is returned to the caller.

GSYM This routine performs symbol table look up for the symbol found in the symbol bucket or compressed symbol bucket. It searches the current and higher-level symbol tables if necessary.

PSC2552 The P-value switch set at PSC2662 is turned off.

PSC2554 If the symbol bucket contains blanks, it indicates a second lookup on the same symbol and transfer is to PSC2628.

PSC2560-PSC2566 If the compressed symbol bucket is not zero, it indicates the symbol is already in compressed form. Otherwise, three characters are taken from the symbol bucket and compressed.

PSC2592-psc2598 A disk wait is issued to make sure the current symbol table is in core and the current symbol table in the core switch is turned on.

PSC2602-PSC2624 This code will search the symbol table currently in core until either the symbol is found or the end of the table. If the symbol is found, the complete symbol table entry is saved.

PSC2628-PSC2640 If a higher-level symbol table was read in at PSC2724, this code will bring the current symbol table back into core.

PSC2642-PSC2670 If the symbol table entry has a P-value, the P-value is converted to a factor of ten and the subscript is made relative to the beginning of the managed array instead of the switch words.

PSC2672-PSC2684 The subscript is converted to a COMMON location. The relative subscript is incremented for the next call to GSYM. The symbol bucket is set to blanks and control is returned to the caller.

PSC2686 If the symbol table currently in core is Level 1, transfer is to PSC2730 to give an error.

PSC2692-PSC2710 If this symbol table look up is the result of an execution-defined symbol subscript, the current partially created symbol table is saved on disk before the next-higher table is brought in.

PSC2718-PSC2726 The next-higher level symbol table is read into core and transfer is to PSC2592 to continue the search.

PSC2730-PSC2736 The undefined symbol error is given and the initial implied symbol is given in its place.

MSMOVE This subroutine will move variable-length character records between core locations. The three paramaters passed to it are FROM, TO, and KOUNT.

PSC2758-PSC2778 The number of bytes requested is moved and control is returned to the caller.

SUPADV This routine will slide over blanks in the input stream starting with the current character.

PSC2812 The input pointer is backed up to point to the previous character and transfer is to PSC2818.

ADVSUP This subroutine will suppress blanks in the input stream starting with the next character.

PSC2818-PSC2822 This code will loop until a nonblank character is found.

PSC2826-PSC2830 The condition code which indicates whether the current character is alpha, numeric, or special character is restored and control is returned to the caller.

DATAIN This routine functions as the control section for scanning the input stream data and the end of the command name to the semicolon.

PSC2844-PSC2846 The error code is set to zero and the first character after the command name is accessed.

PSC2848 If the current character is a semicolon, control is returned to the caller.

PSC2852-PSC2860 If the command name did not end with a comma, colon, or semicolon, an error is given and the input pointer is decremented to the previous character.

PSC2862-PSC2866 The number of GO TO loops is set to a maximum of 1000, the relative subscript set to 1, and the input pointer is saved.

PSC2870-PSC2872 The symbol bucket is set to blanks and the initial symbol information set up. This is done so that if the first item found in the input stream is a data value with no symbol, that data value will go in the location assigned in the first symbol in the command.

PSC2874-PSC2876 The next nonblank character in the input stream is accessed and the subroutine CENTEST is called to check for a dollar sign expression number.

PSC2878-PSC2880 Any possible blanks in the input stream are slid over and a check is made for an alphabetic character. If not, transfer is to PSC2994.

PSC2882-PSC2888 The Implied Do valid switch tested by the COLSYM routine is turned on and COLSYM is called to collect the alphabetic symbol. The Implied Do valid switch is then inverted. Note that if COLSYM finds an Implied Do, it turns this switch off. The symbol found switch tested at PSC2986 is turned on.

PSC2892-PSC2910 This code will check for literal data values, an expression to be evaluated, or a user-exit program associated with this symbol. If any of these are found, transfer is to PSC2960.

PSC2912-PSC2935 This code collects and moves normal data values to COMMON. The values may be either logical, signed, or unsigned numeric values. If no data value is found, following the symbol then a logical TRUE is assumed.

PSC2936-PSC2950 This code moves the data values to COMMON. If an Implied Do was found by COLSYM, the value is placed in COMMON the correct number of times.

PSC2956-PSC2966 If an Implied Do subscript was not followed by a single-valued constant, the appropriate error is given. Note that an Implied Do subscript cannot be followed by a literal or expression and it must not have a user-exit program associated with the symbol.

PSC2968-PSC2974 The symbol yes switch is turned off and a check is made for a comma. If not, the input pointer is decremented before transferring to PSC2874.

PSC2976-PSC2978 When a numeric constant is found, the WSYM TRUE and FALSE switches are turned off and the numeric constant is placed in the output bucket.

PSC2982-PSC2990 If no numeric constant is found, and the symbol yes switch is off indicating that no symbol was found after the last comma, an invalid character error is given.

PS2994-PSC3010 A check is made for a comma or a semicolon. If a comma is found, the relative subscript is incremented by one and transfer is to PSC2874. On a semicolon, the GO TO search switch is turned off and control is returned to the caller.

LITERALT This routine tests for and processes literal data values from the current input stream.

PSC3022-PSC3032 If the current character in the input stream is not a double quote, single quote, or commercial at sign, control is returned to the caller through the no literal found exit.

PSC3034 The beginning quote sign is saved for testing at PSC3042. The input pointer is saved for calculation of the literal count PSC3060.

PSC3040-PSC3050 This code will loop accessing successive characters until either a quote sign equal to the beginning quote, or a semicolon is found. If a semicolon is found before the end quote, an error diagnostic is given.

PSC3056-PSC3064 The literal count of the number of characters is calculated. If this count is zero, an error diagnostic is given.

PSC3066-PSC3070 If this is a double quote literal. Transfer is to PSC3080 to bypass placing of the character count in COMMON.

PSC3074-PSC3098 This code will loop moving four characters of the literal at a time into successive positions of COMMON until the literal count is exhausted. If the literal count is not a multiple of four, the last word moved is padded with blanks.

COLSYM This routine will collect symbols from the current input stream. These symbols may be optionally preceded by an S' and/or subscripted. The current character in the input stream is assumed to be alphabetic on entry.

PSC3114-PSC3134 Up to three alphabetic characters are collected and placed in the symbol bucket. If the second character is nonalphabetic, transfer is to PSC3240 to test for a possible S' formation. If the symbol is less than three characters, blanks are supplied. If more than three characters, the remaining characters are ignored.

PSC3136-PSC3138 The S' valid switch is turned off and the relative subscript is set to one.

PSC3140-PSC3158 If the first character after the symbol is not an EBC or BCD left parenthesis, control is returned to the caller. Otherwise, the recursive operator is set to an EBC or BCD right parenthesis respectively. The symbol bucket is saved in the recursive accumulator and the recursive routine ARITHEXP is called to evaluate the subscript expression.

PSC3162-PSC3182 If the expression ended with a comma and the Implied Do valid switch is on, then ARITHEXP is called to evaluate the second expression. The result of the expression if positive, is converted to fixed point and used as the upper limit of the implied do.

PSC3186-PSC3202 If the upper limit expression was ended with a comma, then ARITHEXP is called to evaluate the expression defining the implied do increment. Note, if no comma was found, a default increment of one is supplied.

PSC3206-PSC3214 The Implied Do switch is turned off.

PSC3218-PSC3222 An error is given to indicate invalid format, logical value, or negative value in a subscript expression and the initial subscript is forced to a value of 1.

PSC3224-PSC3236 The initial subscript is converted to fixed-point and checked for positive value. If yes, the input pointer is advanced over the right parenthesis and control is returned to the caller.

PSC3240-PSC3248 In the case where the second character of the symbol is not alphabetic, this code checks for the possibility of an S' formation. If not, transfer is to PSC3130.

PSC3252-PSC3254 The S' formation found switch is turned on and the input pointer is incremented over the quote sign.

PSC3256-PSC3260 The first character after the S' is checked for alphabetic. If yes, transfer is to PSC3114, otherwise, the symbol bucket is set to blanks and the initial implied symbol is supplied.

COLNUMT This routine is called to check for and collect numeric data in floating-point form. It will convert a variable length numeric field to a 32-bit floating-point constant.

PSC3274-PSC3284 Pointers and switches are initialized before starting the collection of numeric data.

PSC3286-PSC3308 This code will loop collecting numeric characters until either nine numerics have been collected, or a non-numeric character is found.

PSC3312-PSC3320 The count of the number of digits collected to this point is saved and the current number of characters is converted to floating-point.

PSC3330-PSC3350 The old result is adjusted and the new number of digits just collected is added in. If there are still digits to be collected, transfer is to PSC3286.

PSC3354-PSC3358 If the numeric field was preceded by a minus sign, the result is complimented.

PSC3360-PSC3366 If no numerics have been collected to this point, the no numeric exit is set and control is returned to the caller.

PSC3370-PSC3372 If the next character after the numeric field is not an E, blanks are suppressed and control is returned to the caller.

PSC3374-PSC3380 If the character after the E is alphabetic, the input pointer is decremented and control is returned to the caller.

PSC3384-PSC3398 This code checks for a plus or a minus sign preceding the exponent field. If a minus sign is found, the negative exponent switch is turned on.

PSC3400-PSC3422 This code collects and validates the exponent field. Note, that if the exponent is more than two digits long, an error is given and control is returned to the caller.

PSC3424-PSC3442 The mantissa is adjusted by the exponent field. Blanks are suppressed and control is returned to the caller.

PEXPEVALT This routine is called to test for and evaluate arithmetic and logical expressions. It functions as a control section during the evaluation.

PSC3454-PSC3464 If the current character is not a colon, pound sign, or equal sign, the expression not found exit is set and control is returned to the caller.

PSC3472 The contents of the symbol bucket and the current relative subscript are saved.

PSC3476-PSC3484 If this routine was called during evaluation of phrase-defined expressions, the ECODE is incremented to indicate a new expression.

PSC3486 EXPEVAL is called to do the actual expression evaluation.

PSC3488-PSC3506 If this was a LOGICAL expression followed by a question mark, this code will evaluate the TRUE leg of the expression.

PSC3508 If this is a LOGICAL expression and also has a FALSE leg, transfer is to PSC3572.

PSC3512-PSC3518 The symbol bucket saved at PSC3472 is restored and if this is a LOGICAL expression, the WSYM switches are set to give a logical TRUE or FALSE as an answer.

PSC3522 If this is a LOGICAL expression, that contained a GO TO and either the TRUE or FALSE leg of the expression, transfer is to CENTLU to initiate the GO TO search.

PSC3526 If the right hand operand to the expression was a literal, transfer is to PSC3536 to bypass writing the results to COMMON.

PSC3530-PSC3534 The result of the expression is placed in the output bucket, the subscript saved at PSC3472 is restored, and the value is written to COMMON.

PSC3536-PSC3540 The conditional expression and conditional expression TRUE switches are turned off and a check is made to see that the expression ended with a comma or a semicolon. If yes, control is returned to the caller.

PSC3550-PSC3566 An error is given indicating an invalid end to an expression and an attempt is made to finish the scan of the expression.

PSC3572-PSC3580 The results of the TRUE leg of the expression are saved and the FALSE leg is evaluated.

PSC3582-PSC3588 The final result to the expression is set from either the TRUE or the FALSE leg according to the result of the base leg expression.

EXPEVAL This routine is called by the EXPEVALT routine to evaluate sections of expressions.

PSC3598-PSC3600 The saved code bucket is cleared and the current input stream character is placed in the type box. The type box is used to indicate the type of expression being evaluated.

PSC3602-PSC3608 If the current character is a colon the next nonblank character is a dollar sign. Transfer is to PSC3598 to change the type box to a dollar sign.

PSC3612-PSC3626 The input pointer is decremented and the recursive routine LOGICAL is called to evaluate the LOGICAL expression.

PSC3630-PSC3636 On a :$ formation, the formula number is collected and placed in the output register as a result of the expression. This number is later used by CENTLU to initiate the GO TO search.

PSC3642 If the current character is not a pound or equal sign transfer is to PSC3690 to give an error.

PSC3650-PSC3656 The symbol bucket and subscript, saved at PSC3472 are restored and a test is made for a literal operand. If a literal was found, transfer is to PSC3696.

PSC3660-PSC3664 The arithmetic expression is evaluated.

PSC3668-PSC3682 If a logical operand was found during the evaluation of the arithmetic expression, the type box is changed to a colon to indicate a LOGICAL expression and a result is set to TRUE or FALSE accordingly.

PSC3684-PSC3688 The result of the expression is saved and control is returned to the caller.

PSC3690 An error is given to indicate invalid format and transfer is to PSC3684.

PSC3696 The type box is set to indicate the literal operand and transfer is to PSC3684.

CENTLU This routine will initiate a GO TO search.

PSC3702 If a GO TO search is already in progress, transfer is to PSC3536.

PSC3706 If this is the TURE leg of a FALSE expression, transfer is to PSC3536.

PSC3712 If the formula number for the GO TO is zero, transfer is PSC3536 to ignore the go to.

PSC3716-PSC3722 The GO TO search switch is turned on. The formula number is saved for test by the CENTEST routine, the input pointer saved at PSC2866 is restored, the symbol bucket is set to blanks, and the initial implied symbol is set up.

PSC3726-PSC3728 If over 1000 GO TO search's have been executed, an error is given and transfer is to PSC3006 to return control to the caller of DATAIN.

CENTEST This routine tests for dollar sign formula numbers.

PSC3744-PSC3760 If the current character is not a dollar sign, control is returned to the caller. Otherwise, the formula number is collected and if it is equal to the formula number being searched for in a GO TO search, the GO TO search switch is turned off.

LOGOPF This routine will test for a logical value found in an arithmetic expression.

PSC3762-PSC3770 If the LOGICAL result switch is on as a result of finding a logical value in an arithmetic expression, the WSYM FALSE switch is turned on and the logical value switch is turned off.

INTEGER This subroutine is called to test for and convert a field of EBCDIC numeric characters to an integer value starting with the next nonblank character in the input stream.

PSC3784 The input pointer is advanced to the next nonblank character and transfer is to PSC3786.

INTEGER1 This entry to the INTEGER routine will start collecting numeric data with the current character.

PSC3786-PSC3804 This loop will collect numeric digits until a nonnumeric character is found.

PSC3806-PSC3808 SUPADV is called to slide over any blanks in the input stream and control is returned to the caller.

EVALUATE This routine is called to evaluate phrase-defined expressions (Table 6). These expressions are in two different forms. The first type are those associated with a CAP pointer at phrase-definition time and are of the form (N)A=B+C. This type of expression is in tabular form and is evaluated by this routine. The second type of expression is the dollar sign formula area. If any of this type of expression is found, control is passed to the DATAIN routine to evaluate these expressions in the same manner as those found in the input stream.

PSC3822-PSC3824 If there are no phrase-defined expressions for the phrase currently in core, control is returned to the caller.

PSC3826-PSC3834 The first byte after Table 6 is saved and a semicolon put in its place. The phrase-defined expression switch is turned on.

PSC3840 The current status of the input pointer is saved and the pointer is set to point to Table 6. Note: this means that routines such as CHTEST and ADVSUP, etc will now fetch characters from Table 6 instead of the normal input stream.

PSC3842 If bits 0 and 1 of the first word are zero, the CAP associated with the expression at definition time was of the form (M+6) and transfer is to PSC3856.

PSC3846 If bit 0 is off and bit 1 is on, transfer is to PSC3950 to start the scan of the formula area.

PSC3852 In the case of the symbol table entry, a relative subscript of 1 is created in the previous word of the table and the input pointer is decremented to point to it.

PSC3856 The next two words from Table 6 are put in a save bucket and the input pointer is incremented to point to the first EBCDIC character in the expression.

PSC3866-PSC3872 If the first character is not a colon, the input pointer is incremented over the equal or pound sign and the literal test switch is turned on.

PSC3874-PSC3898 This code sets up the information needed by the GSYM routine. If the CAP pointer is execution-defined, that is, (M+6), the symbol bucket is set to 0 and the compressed symbol bucket will be set from the second word of the Table 6 entry. GSYM will then obtain the information for the symbol table entry bucket by doing a symbol table lookup. Otherwise, the symbol bucket is set to blank to suppress symbol table lookup and the symbol table entry bucket is created from the information in Table 6.

PSC3900-
PSC3902 A test is made for a literal operand. If found, transfer is to PSC3918.

PSC3904-
PSC3914 The arithmetic expression is evaluated and the result is written to COMMON.

PSC3918 If this is a LOGICAL expression, a test is made for the existence of a FALSE leg. If yes, transfer is to PSC3856.

PSC3922-
PSC3926 The conditional expression and conditional expression TRUE switches are turned off and the input pointer is incremented over the comma. If all Table 6 expressions have not yet been processed, transfer is to PSC3842.

PSC3930 The LOGICAL expression is evaluated.

PSC3934-
PSC3944 If the LOGICAL expression was not followed by a TRUE leg, the proper result is set and transfer is to PSC3914.

PSC3950 The expression number error code is saved, the GO TO count is initialized to 1000, and the exit address from DATAIN routine is set to return control to PSC3972. Control is then passed to the DATAIN routine for evaluation of the dollar sign formula area.

PSC3958-
PSC3964 The conditional expression switch is turned on and the conditional expression TRUE switch is turned on if the base leg was TRUE.

PSC3966-
PSC3970 The input pointer is incremented over the question or exclamation mark. The conditional expression TRUE switch is inverted and transfer is to PSC3866 to evaluate the TRUE or FALSE leg expression.

PSC3972-
PSC3984 DATAIN will return control to here for cleanup processing before control is returned to the caller of EVALUATE. The conditional expression, conditional expression TRUE, and phrase-defined expression switches are turned off. The first byte after Table 6 saved at PSC3826 is restored, the input pointer saved at PSC3840 is restored, and control is returned to the caller.

LOGICALT This routine will perform a prescan of a LOGICAL expression in search of a relational operator.

PSC3998 The current status of the input pointer is saved and the parentheses counter is set to zero.

PSC4002-
PSC4004 The next character in the input stream is accessed and checked for a comma or semicolon. If yes, transfer is to PSC4050.

PSC4012 If the current character is a BCD left paren, an error is given.

PSC4016-
PSC4020 If the character is an EBCDIC left parenthesis, the parentheses counter is incremented and transfer is to PSC4002 to get the next character.

PSC4024-
PSC4036 If a relational operator is found, transfer is to PSC4124.

PSC4040-
PSC4046 If the character is a right parenthesis, the parentheses counter is decremented and if not 0, transfer is to PSC4002. Otherwise, the input pointer saved at PSC3998 is restored and the LOGICAL routine is entered.

LOGICAL This routine controls the evaluation of a logical expression at the first level of hierarchy.

PSC4054-
PSC4056 The result is set to a default of FALSE and saved in the recursive accumulator.

PSC4058-
PSC4060 The next character in the input stream is accessed and the HIER2L is called to evaluate the logical

expression at the second level of hierarchy.

PSC4064-PSC4068 If the result returned by HIER2L is TRUE it is left alone, otherwise, the result is set from the recursive accumulator.

PSC4070 If the current character is an OR sign, transfer is to PSC4056 to evaluate the next part of the expression, otherwise, control is returned to the caller.

PSC4076 An error is given to indicate that BCD characters are not allowed in a logical expression and transfer is to PSC4024.

HIER2L This routine evaluates logical expressions at the second level of hierarchy, that is, logical operands separated by 'and' signs. If the routine is entered at PSC4124 it will evaluate a logical relational expression.

PSC4084 The recursive accumulator is set to TRUE and the input pointer decremented to the previous character.

PSC4088-PSC4092 The next nonblank character in the input stream is accessed and an EBC NOT sign exclusive ORed to the recursive operator. Note that the recursive operator is zero on entry to this routine. If the current operator is a NOT sign, transfer is to PSC4088 to slide over the NOT sign and get the next character.

PSC4096 RETRIEVAL is called to evaluate the logical operand.

PSC4100-PSC4110 If the operand was preceded by an odd number of NOT signs, result is inverted. If the result is FALSE it is left alone, otherwise it is set from the recursive accumulator.

PSC4112-PSC4116 If the current character is an AND sign, the operator is set to zero. The current result is saved in the recursive accumulator and transfer is to PSC4088. Otherwise, control is returned to the caller.

PSC4124 If the paren counter is not zero, transfer is to PSC4050 to evaluate this expression as a normal logical.

PSC4128-PSC4132 The current character is saved as a relational operator and the next nonblank character is accessed. The logical relational switch is turned off.

PSC4134-PSC4150 This code checks for an expression of the form (A=+).

PSC4154 The input pointer saved at PSC3998 is restored.

PSC4160 If this is a logical or literal relational of the form (A=+) or (A="BCE") transfer is to PSC4240.

PSC4162-PSC4172 The arithmetic operands on each side of the relational operator are evaluated.

PSC4176-PSC4196 The result is set to a default of FALSE. If a logical value was found in the evaluation of the arithmetic operand, transfer is to PSC4194. If the operator was a less than or greater than sign, the appropriate branch is taken. Otherwise, the results are compared and if equal, the return result is set to TRUE, otherwise it is set to FALSE.

PSC4200 An error is given to indicate invalid format in a relational expression and transfer is to PSC4654.

PSC4204-PSC4220 An error is given to indicate invalid format in a literal relational expression. The result is set to FALSE and if this is a logical relational of the form (A=+) control is returned to the caller. Otherwise, a loop is initiated to slide over the right hand literal operand.

PSC4228-PSC4236 This code compares the left and right hand operands and sets the result to TRUE or FALSE according to whether the operator was a less than or greater than sign.

PSC4240-PSC4254 This code collects the left-hand operand of a literal or logical relational expression. Note that the left-hand operand must be an alphabetic symbol and the operator must be an equal or pound sign.

PSC4262 The next character after the operator is now saved as the operator.

PSC4264-PSC4268 The value for the left hand operand is read from COMMON and the result is set to a default of TRUE. If this is a logical relational, transfer is to PSC4300.

PSC4272-PSC4296 This code compares character by character between the left and right-hand operands. If a nonequal character is found, the result is set to FALSE.

PSC4300-PSC4320 This code will set the result to TRUE or FALSE according to whether the current operator is a plus or a minus sign respectively.

TRITHEXP This entry into ARITHEXP will allow for the detection of a stand-alone plus or minus sign as an operand.

ARITHEXP This recursive routine evaluates arithmetic expressions at the first level of hierarchy and leaves the floating-point value of the expression in the result register.

PSC4336-PSC4338 The next nonblank character is accessed and the result is set with a default of zero.

PSC4340-PSC4356 If the current character is not a plus or a minus sign and the operator is not zero indicating that this is not the first operand evaluation, control is returned to the caller. Otherwise, the operator is set to a default of a plus sign and transfer is to PSC4394.

PSC4360-PSC4368 The plus or minus sign is saved as the operator and if the next character is alpha, numeric, a left parenthesis, or a period, transfer is to PSC4394.

PSC4380-PSC4390 If the single logical operand switch is on, the WSYM TRUE or FALSE switch is turned on and control is returned to the caller.

PSC4394-PSC4396 The current result is saved in the recursive accumulator, a single logical switch is turned off, and HIER2A is called to evaluate the operand at the second level of hierarchy.

PSC4398-PSC4410 The result returned by HIER2A is added to or subtracted from the recursive accumulator.

HIER2A This routine controls the evaluation of arithmetic expressions at the second level of hierarchy.

PSC4418-PSC4420 Blanks are suppressed and RETRIEVAL is called to retrieve the arithmetic operand.

PSC4424-PSC4426 If the current character is not a multiply or a divide sign, control is returned to the caller.

PSC4432-PSC4450 The next operand is collected and the result is adjusted by it according to whether the operator is a slash or an asterisk.

RETRIVA, RETRIEVL These recursive routines collect single-valued operands. That is, symbols, numeric constants, or parenthesized expressions.

PSC4468-PSC4470 If a numeric field is found, control is returned to the caller.

PSC4472-PSC4492 If the current character is a left parentheses, ARITHEXP is called to evaluate the arithmetic operand.

PSC4500-PSC4506 On entry to collect a logical operand if the current character is a left parenthesis LOGICALT is called to evaluate the logical operand.

PSC4512-PSC4518 If the current character is not alphabetic, an error is given to indicate invalid format in an expression and control is returned to the caller.

PSC4524-PSC4560 This code will collect the alphabetic symbol optionally preceded by an S' and will either read the appropriate value from COMMON at PSC4534 or compute the result from the subscript at PSC4538-PSC4560.

PSC4574-PSC4584 If the operator is an EBCDIC right parenthesis, a logical result is given.

PSC4588-PSC4592 If a logical value was found in the evaluation of an arithmetic operand, the logical value switch is turned on.

RCALL, RRETURN This subroutine performs linkage to and from recursive subroutines. It performs a recursive call to the subroutine indicated by the parameter following the BAL to RCALL. It saves the return address, the current recursive operator, and the recursive accumulator. Entry to the routine at the label RRETURN performs the return linkage. It restores the current operator, the recursive accumulator, and branches to the return address. The recursive call save area has room for 64 entries.

PSC4610-PSC4616 If the save area is full, transfer is to PSC4652 to give an error.

PSC4620-PSC4624 The parameters are saved, the save area pointers are incremented, and control is passed to the requested recursive subroutine.

PSC4638-PSC4650 The save area pointer is decremented, the information saved at PSC4620 is restored, and control is returned to the last caller of RCALL.

PSC4652-PSC4654 An error is given to indicate that the expression was too complicated to evaluate. The save pointer is reset to the beginning of the save area and transfer is to PSC4644.

EXPER This subroutine performs an error number computation and call to the ERROR routine for COMMON errors found in input stream expressions, phrase-defined expressions, and execution-defined symbol subscript expressions. It also performs a scan to the next comma or semicolon in the current input stream if the 'TO COMMA' switch is on.

PSC4668-PSC4678 The error number is incremented by one if this is an execution-defined symbol subscript. It is incremented by two for a phrase-defined expression. For an input stream expression, it is left alone.

PSC4680-PSC4700 The error diagnostic is given and if the 'TO COMMA' switch is on, the input stream pointer is incremented until the next comma or semicolon is found.

CHTEST This is the character fetch routine called to fetch and test the next character in the current input stream.

PSC4718-PSC4722 If the special exit is not on, transfer is to PSC4724. If the switch is nonzero, it contains the address of the routine to be given control. The routine given control must return to PSC4724.

PSC4724-PSC4730 The next character in the input stream is tested for NULL (Hex00). If a NULL is found, a subroutine INEUTRD is called to read the next record from the current PLAN input device.

PSC4736 The condition code is set to indicate whether the current character is alphabetic, numeric, or special character.

DISKWRD Since the disk read and write routines do not return control until the information is in core, this is a dummy wait routine.

DISKWT, DISKRD These two routines will read and write information to the disk.

UGCHAR This entry into PSCAN is used by the GUSER subroutine from within a user-exit program. It will fetch the next character from the input stream and return control to GUSER.

MACHK This routine is used by PHRGOT on a Level 1 phrase to check the size of the managed array before setting it to FALSE. If the size of the managed array is larger than the total size of COMMON, an error is given.

COMCHK This subroutine will check to see whether a subscript is within COMMON. If not, an error is given before returning control to the caller. Note that this subroutine is actually called any time a question block is shown in the flowchart which says 'Is subscript within COMMON'.

SETTOPCM This routine is used by PHRGOT to set a pointer to the top of the currently-defined COMMON.

INPUTSAV This subroutine will save the EBCDIC image of the current command being executed and PFINPUTA of PFILE. It will also save the residual of the last record following the semicolon in PFINPUTB of PFILE.

PSC50614 If the input command is already been saved, control is returned to the caller.

PSC50616-
PSC50618 The input saved switch checked at PSC50614 is turned on and the command sequence number in the resident loader is incremented.

INPUTSVA This entry to this routine is used by TESTCHKE to push a command from a check entry.

PSC50632 If entry to this routine was at INPUTSVA, transfer is to PSC50764.

PSC50636-
PSC50640 If this is a repeated command, the repeat switch in the resident loader is turned off. Otherwise, transfer is to PSC50730 to save the current command.

PSC50648-
PSC50714 If this is not a check entry push and the phrase print option is on, the current command is printed on the current PLAN output device before returning control to the caller.

PSC50730-
PSC50762 This code will check for a residual record following a semicolon in the command. If a residual is found it is written to PFINPUTB of PFILE. Otherwise, the input on disk switch checked at PSC092 is inverted.

PSC50764 The current command is written to PFINPUTA and transfer is to PSC50648.

DFJPSRTA, DFJGSRTA

DJFPSRTA and DFJGSRTA are the the block sort routines for the PLAN DYNAMIC file and PERMANENT file respectively. They are the first of two loads and may be required to accomplish the sort. This modules reads a core load from the file, sorts it and returns it to the file. This process continues until end-of-file. As each is written out, a sequence check is made against the preceding block. If no sequence check has occurred when end-of-file is encountered the name of the in-place merge module is expected from the pop-up list. This module is invoked through the LCHEX subroutine by the subroutines PSORT and GSORT. The technique used for sorting is to create an ordered string of record pointers using a binary chart search on the existing string. After the record pointers have been ordered, the records themselves are rearranged to their proper place in core and the block is written out. The only difference between these modules is that DFJPSRTA uses the READ/WRITE subroutines and DFJGSRTA uses the RDATA/WDATA subroutines.

RTA1130 ERASABLE COMMON is located and the address of the SORT control fields is saved.

RTA1370 The SORT control fields are validated.

RTA2570 The amount of available core is calculated. On OS this will be all of the available program area outside of this module. On DOS, it is a fixed buffer within this module.

RTA2630 The SORT area pointers are calculated and initialized.

RTA2970 ID(2) of the file control block is rounded to the nearest record length.

RTA3030 If there are less than two records in the file, transfer is to RTA3430.

RTA3170 The last record save area is cleared to ensure that the sequence check on the first block is correct.

RTA3370 If an end-of-file has not occurred, transfer is to RTA3830.

RTA3430 If no sequence breaks have occurred during the block sort, transfer is to RTA3670.

RTA3550 The sort record area pointers are reset for the merge.

RTA3650 Exit is to the next load entry in DFJPLAN to call the MERGE module.

RTA3670 Exit is to the next load entry in DFJPLAN but the pop-up list is updated so that the merge is skipped.

RTA3830 A block of records is read from the file.

RTA4030 The number of records in the sort area is calculated.

RTA4210 The SRA list is initialized. This is a list of pointers to each record in the sort area.

RTA4350 If only one record is in the sort area, transfer is to RTA6190.

RTA4470 The initial SRA list string is created. This is done by sorting the first two records in the sort area and possibly exchanging the first two pointers in the list.

RTA4550 The list pointer is set to the end of the string.

RTA4570 If the list pointer is at the end of the SRA list, transfer is to RTA5790.

RTA4610 The subroutine SORT is called to sort the next two records pointed at by the LIST pointer.

RTA4630 If the records are in sequence transfer is to RTA4550.

RTA4770 The current SRA list string search using a binary search method to locate the insert point for the out-of-sequence record pointer.

RTA5470 The pointer to the out-of-sequence record is inserted into the string and transfer is to RTA4550.

RTA5790 The SRA list is searched and all pointers to records that are out of place are flagged.

RTA5970 All out-of-place records are exchanged to their correct place in core.

RTA6190 The subroutine SORT is called to check the last high record in the previous block against the low record in this block.

RTA6250 If a sequence break has not occurred, transfer is to RTA6410.

RTA6270 If this is not the first sequence break, transfer is to RTA6410.

RTA6330 The sequence break KDIS is saved.

RTA6410 The last record in this block is saved in the last record saved area.

RTA6590 The block is written back onto the file and transfer is to RTA3370.

SORT The SORT subroutine compares two records and returns the results of the compare in the register WINNER.

RTA7350 The address of the A and B records is determined.

RTA7430 The field displacement in the record of the SORT field is determined.

RTA7550 The fields are compared. This is done by calling an appropriate compare routine for the type of sort involved. Return is to RTA7590 if the fields are equal.

RTA7590 If this is not the last control field transfer is to RTA7430.

RTA7630 This is the return point if the compare routines find an unequal. The address of the A area record is set to be the WINNER.

RTA7650 If the records were not equal, transfer is RTA7670; otherwise, control is returned to the caller with the A record as the WINNER.

RTA7670 If the A record was high transfer is to RTA7750.

RTA7690 If this is not an ascending sort transfer is to RTA7790; otherwise, control is returned with the A record as a WINNER.

RTA7750 If this is not a descending sort transfer is to RTA7790; OTHERWISE, THE A area record is returned as the WINNER.

RTA7790 The B area record is set as the WINNER and control is returned to the caller.

DFJPSRTB, DFJGSRTB

DFJPSRTB and DFJGSRTB are the in-place merge routines for the DYNAMIC file and PERMANENT file sort respectively. They are the second of two loads and may be required to accomplish the sort. These modules are only loaded if DFJPSRTA or DFJGSRTA find a sequence break. A merge is done by relocating the out-of-sequence block to the managed area file and then performing a descending merge. The difference between the two modules is that DFJPSRTA uses the READ/WRITE subroutines while DFJGSRTA uses the RDATA/WDATA subroutines.

RTB1010 The size of the working area available on the managed area file is calculated.

RTB1450 The merge area pointers are calculated and initialized.

RTB1850 The next out-of-sequence block is located. This is done by reading two records which bridge two blocks in the file and comparing them for a sequence break.

RTB2270 The merge pointers are initialized.

RTB2530 The count for the READ operation is determined. This may be less than a full block depending on the size of the work area available in the managed area file.

RTB3030 If an end-of-file has not occurred on the file transfer is to RTB3170.

RTB3090 The merge switch is set to force the beginning of the merge at the end of the READ of this block.

RTB3170 If there is room in the managed area file transfer is to RTB3410.

RTB3230 The size of the managed area file work area is used for a count to read the file.

RTB3310 The merge switch is set on to force the merge to the end of this read.

RTB3410 A block is read from the file.

RTB3490 A check is made for an out-of-sequence record. If this occurs, transfer is to RTB4310.

RTB3650 The block is written onto the managed area file.

RTB3950 If the merge switch is on, transfer is to RTB4450. The last record in the block is saved to perform a sequence check on the next block.

RTB4230 A check is made to see if the merge can start because the last record of the block just read is higher than the last record of the blocks that were in sequence in the file. RTB2530 to read the next block from the file.

RTB4310 The KDIS for the sequence break record is saved for the next cycle.

RTB4450 The MERGE switch is reset.

RTB4470 The working area file pointers are initialized.

RTB4670 The merge areas are primed.

RTB4770 The next output buffer is located.

RTB4830 If the output area is not full transfer is to RTB5290.

RTB4950 The output area is rewritten onto the file.

RTB5250 If the end-of-merge switch is on, transfer is to RTB1850.

RTB5290 If the B area is being flushed transfer is to RTB5570.

RTB5450 The subroutine SORTZ is called to compare the A and B records.

RTB5470 If the A record was in sequence, transfer is to RTB5690.

RTB5570 The B area record is set as a WINNER. Transfer is to RTB5730.

RTB5690 The A area record is set as a WINNER.

RTB5730 The winning record is moved to the output area.

RTB5770 If the B record was the WINNER, transfer is to RTB6750.

RTB5910 If the A record area is empty, transfer is to RTB6090.

RTB5950 The A area record pointer is updated. Transfer is to RTB4770.

RTB6090 If an end-of-file has not occurred on the PLAN file, transfer is to RTB6270.

RTB6150 If the B area is being flushed, transfer is to RTB6230.

RTB6190 The flush switch is set for the B area and transfer is to RTB4770.

RTB6230 The end-of-merge switch is set and transfer is to RTB4950.

RTB6270 The KDIS and KOUNT for the next A area block are calculated.

RTB6490 The A area pointers are initialized.

RTB6630 The next A area block is read and transfer is to RTB5910.

RTB6750 If the B area is empty, transfer is to RTB6930.

RTB6790 The B area record pointer is updated and transfer is to RTB4770.

RTB6930 If an end-of-file has occurred on the working area file, transfer is to RTB6230.

RTB6970 The KDIS and KOUNT for the next working area file block are calculated.

RTB7270 The B record area pointers are reset.

RTB7330 The next block is read from the working area file and transfer is to RTB6750.

SORT The SORTZ subroutine is used to compare two records. The result is returned by branching to either caller plus zero or caller plus 4.

RTB8170 The sort control fields are located.

RTB8250 The next sort field is located.

RTB8390 The fields are compared. This is done by calling an appropriate compare routine for the type of sort involved. Return is to RTB8410 if the fields are equal.

RTB8410 If this is not the last field, transfer is to RTB8250.

RTB8450 The A area record is set as a WINNER.

RTB8470 If the records were not equal, transfer is to RTB8490; otherwise, control is returned to the caller with the A record as the winner.

RTB8490 If the A record was high transfer is to RTB8570.

RTB8570 If this is not a descending sort, transfer is to RTB8610; otherwise, control is returned to the caller with the A record as the winner.

RTB8610 The B area record is set as the winner and control is returned to the caller.

DFJPSTSV

This module provides the statement save facility of PLAN. It is loaded for execution under any of the following conditions:

1. The standard PLAN command SAVE has been given to read and save numbered commands in the input stream.
2. Scheduled by PSCAN to save a numbered command found in the input stream.
3. Scheduled by the resident loader when PLAN Switch Word 2 contains the number of a command to be retrieved for execution.

Halts: None

Error Conditions: PLAN diagnostics produced directly by this module are in the range of 170-179.

Subroutines:
Monitor: USBSC
PLAN: GTVAL, PFSPC, FIND, READ, NDEF, TRUE, FALSE, WRITE, PSTS1, PUSH, INPUT, PUNPK, PPACK, LRET, PSBFB, PLINP, PEOF, PAIN

Switches:
PLAN Switch Word 1 - Used for saving ID(1) (file number) of the save statement file.
PLAN Switch Word 2 - Contains the number of the last statement to be executed.
PLAN Switch Word 3 - Contains the number of the last statement to be executed plus drive code, times 2048.

PSTSV-PST520 If Switch Word 2 does not contain a statement number, transfer is to PST1440 to perform the explicit of implicit save.

PST540-PST560 The file number is picked up from PLAN Switch Word 1 and FINDL is called to open the file.

PST590-PST750 If the file is less than 28 words long or the first word of the file is not a logical TRUE, the

28-word header is created and written back to the file.

PST770-PST810 ID(2) of the file control block is set to the file size indicated in the file header. On an execution request transfer is to PST930, otherwise, transfer is to PST1500 or PST2630 for implicit or explicit save respectively.

PST840-PST890 Switch Words 1 and 3 are cleared to suppress further execution of save statements. The file number is used as an error code and the error number is set to 171 to indicate an invalid saved statement file.

PST895-PST910 Switch Word 2 is cleared to suppress SAVE statement execution and PST1 is called to process the error and return control to the resident loader.

PST930-PST1050 This code chains through the 26-word control blocks until the control block containing the searched for statement number is found. If there is no control block for this statement, error 172 is given to indicate the statement does not exist.

PST1070-PST1090 If the statement does exist, the first word is read into core.

PST1110-PST1140 If the statement is not where it should be in the file, error 173 is given to indicate that the file has been destroyed or overwritten.

PST1170 If Switch Word 3 indicates there is another statement to be executed, transfer is to PST1260 to find the next higher statement in the file less than Switch Word 3.

PST1190 Switch Word 2 is set to 0 to indicate the last statement has been executed.

PST1200-PST1250 The full statement is read into core, the current statement save file number is saved in PLAN Switch Word 1 and PUSH is called to pass the command and control to PSCAN.

PST1260-PST1400 This loop searches the file for the next statement to be executed. This is done so that Switch Word 2 will point to the next statement on exit from PSTSV. If no statement is found with a number less than that found in Switch Word 3, then Switch Word 2 is cleared to indicate that all statements have been processed.

PST1440-PST1480 On an implicit or explicit save, this code is used to open the file.

PST1500-PST1610 On an implicit save, this code is used to access the phrase image from PFINPUTA in PFILE. The statement number is collected from the beginning of the command and the numeric characters are replaced with blanks. This is done so that later execution of the command from the file will not cause PSCAN to execute another implicit save.

PST1630-PST1670 This code searches the file to see if a statement exists with the same number.

PST1700-PST2100 This code will delete the current statement from the file and update the file to reflect the deletion. In other words, all of the information in the file beyond the statement deleted is moved down over the deleted statement. The control block chain pointers and individual statement pointers in each control block are updated to reflect the shift.

PST2170-PST2420 This code chains through the control blocks until the control block for the statement to be added is found. If new control blocks are needed, they are created and chained together.

PST2450-PST2480 The pointer to the statement being added is placed in the control block and the block is written to the file.

PST2500 The statement to be added is written to the file.

PST2540-
PST2560 The second word of the file header is updated to indicate the next avalable space in the file and the file number is saved in PLAN Switch Word 1.

PST2590 If this is an implicit save, LRET is called to return control to the resident loader.

PST2630-
PST3035 This code will read a statement from the input stream.

PST2630-
PST2670 PSBFB is called to set up the input buffer for the current PLAN input device. The statement number is initialized to 0 and the pointer is initialized to the first character in the statement.

PST2680 The number collect switch is turned on to indicate collection of a statement number is legal.

PST2700-
PST2710 PLINP is called to read a record from the current PLAN input device. If logical or physical end-of-file was found, control is returned to the resident loader.

PST2730-
PST2740 The next character in the statement is read.

PST2770-
PST2810 If the number collect switch is on and this is a numeric character, it is added to the statement number.

PST2840-
PST2860 The number collect switch is turned off and the number of characters in the statement is incremented by 1.

PST2900 If the semicolon at the end of the statement has been found, transfer is to PST2970.

PST2920 If all 75 characters have been processed, transfer is to PST2700 to read the next record.

PST2950 If the character is not a blank or numeric, transfer is to PST2840 to turn off the number collect switch.

PST2990-
PST3035 If this was a non-numbered statement, PUSH is called to pass the statement and control to PSCAN.

PST3080-
PST3270 This code chains through the control blocks until the control block containing the statement number to be added is found. If new control blocks are needed, they are created and chained together.

DFJPTDMP

The DFJPTDMP module is the utility module to provide a tabular listing of the PLAN phrases that exist in PFILE, that is, the PLAN file dictionary. This module requires use of all of a 840-word communication array. No data may be carried over through use of this module. It must be called by the DUMP PHRASES command which in turn invokes execution of the CONTINUE DUMP PHRASES command. These commands set up extensive literal information, constant data, and control parameters. The DFJPTDMP module is the mainline for the phrase table dump. It calls several subroutine modules. The dump is structured in this manner because of its use on both System/1130 and System/360. On the 1130 system, the entire core image module is too large for an 8K memory. Therefore, the subroutines that dump some of the tables are loaded as monitor system locals when running in an 8K environment. When running in a 16 or 32K environment, the modules are not localed and therefore, throughput is improved. The system, although written in FORTRAN, is written to dump the phrase table on both the 1130 System and the System/360 by a technique that makes the difference in construction of the dictionaries on those two systems invisible. The extent of the dump produced is controlled by the phrase with a parameter called LEVEL. If LEVEL is zero or one, only the header of the phrase is printed. The maximum value for LEVEL is six. Any value of six or greater produces an entire tabulated listing of the phrases. A value in between zero and six produces a dump of the internal tables up to and including the table number equal to the LEVEL number, that is, if the LEVEL level is four, the internal dictionary entry tables 1, 2, 3, and 4 will be dumped. The PBTST routine is used extensively in the dump to do bit extractions which then allows the dump activity to be programmed in the FORTRAN language.

PTD770 The internal PFILE record size is set as either 64 bits or 80 bits. This choice of internal bit size structre makes the difference in the record size on the System/360 and 1130 invisible. Since a record on the 1130 system is one sector, that is 320 words, the record is ivided into 64ths yielding an 80 bit internal record size. System/360 yields a record size of 64 bits. This determination is based upon a value set by the DUMP PHRASES command. The value is the machine type on which the dump is being run, that is, 1130 or 360. If the system type is not in the command a completely garbage dump can be anticipated.

PTD830 The GDATA subroutine is called to open the PFILE. As file number 255, the drive code is picked up as the parameter supplied by the DUMP PHRASES command.

PTD850 The XACES subroutine is called to read the phrase validity table. The XACES subroutine masks the backward construction of the 1130 PFILE by converting the sector number into the proper GDATA displacement for the appropriate system upon which the dump is being produced.

PTD880 A single buffer set A is assigned to the output device.

PTD900 The printer is skipped to a new page.

PTD920 A loop is initialized for the 256 checksums.

PTD1000 The synonym indicators are reset. The synonym indicators are words that contain the displacement and sector number which contain the next phrase of equal checksum in the phrase chain. If the indicators are zero no phrase of equal checksum remains in the chain. The synonym indicators are printed at the right-hand side of the phrase table dump.

PTD1020 A check is made in the validity table to determine if there is a phrase with this check sum. If there is not, transfer is to PTD2850.

PTD1040 The sector number that includes the start of the phrase is determined.

PTD1060 The check sum heading line is set up.

PTD1080 The check sum number is set to the print buffer.

PTD1100 The internal record displacement to the start of the phrase is calculated. Since a phrase must by definition be included within two sectors, two sectors are always read.

PTD1180 The DFJPTDP1 subroutine is called to produce a heading line for the phrase. The heading line includes such things as the phrase type, the phrase name, and level, and also includes the synonym indicator bits. The DFJPTDP1 subroutine is called as a PLAN LOCAL.

PTD1200 If the phrase table dump is running under level 0 or 1 transfer is to PTD2850.

PTD1230 A double space is set up for the printer.

PTD1240 The internal bit index is incremented by 16.

PTD1250 A determination is made to see if there are additional internal tables in this phrase entry. If there are not transfer is to PTD2850.

PTD1320 The XTRAC subroutine is called to extract the 8-bit table length.

PTD1330 If this is a null table transfer is to PTD1370.

PTD1350 The DFJPTDP2 subroutine is called to dump the initialization values The DFJPTDP2 subroutine is called as a monitor system LOCAL on the 8K version of the 1130 phrase table dump.

PTD1370 The XTRAC subroutine is called to extract the 8-bit table control code.

PTD1390 If this is the end of this phrase entry transfer is to PTD2850.

PTD1410 A check is made to determine that this a header for a new internal phrase table entry. If it is not transfer is to PTD2750.

PTD1440 If we are running at level 2 transfer is to PTD28508.

PTD1510 The XTRAC subroutine is called to extract the 8-bit table length.

PTD1540 If this is a null table transfer is to PTD1570.

PTD1550 The DFJPTDP3 subroutine is called to dump the symbol table. The DFJPTDP3 subroutine is called as a PLAN LOCAL.

PTD1570 The XTRAC subroutine is called to extract the 8-bit table control code.

PTD1600 If we are running at level 3 transfer is to PTD2850.

PTD1620 The internal switch is set to indicate that the program list to be dumped are those programs associated with the phrase entry keyword program.

PTD1640 A test is made to determine if there additional internal tables in this phrase entry. If there is not transfer is to PTD2850.

PTD1660 A test is made to determine if this is a valid header for an internal phrase entry table. If it is not transfer is to PTD2750.

PTD1720 The XTRAC subroutine is called to extract the 8-bit table length control indicator.

PTD1750 A test is made to determine if this is a null table. If it is transfer is to PTD1960.

PTD1770 The header line for the internal table 4 is set to print.

PTD1780 The table 4 header is printed.

PTD1790 The XTRAC subroutine is called to extract the first bit of the program entry.

PTD1810 If the bit is on the entry is determined to be alphabetic and transfer is to PTD1890.

PTD1830 A 32-bit binary field is extracted by a call to the XTRAC subroutine. If the 32-bit field is all zeros transfer is to PTD1890.

PTD1850 The numeric zero program number is set to print.

PTD1870 The bit index is incremented by 64. Transfer is to PTD1940.

PTD1890 The EXTRAC subroutine is called to extract a 64-bit program name entry.

PTD1920 The bit index is incremented by 64.

PTD1930 The program name is set to print by a call to PAOUT.

PTD1940 A program name or number is printed.

PTD1950 A check is made to determine if this internal table is entirely processed. If it is not transfer is to PTD1490.

PTD1960 If the program list which we just processed was associated with the phrase keyword PROGRAM, processing will continue at PTD1980. If it is associated with the keyword EXIT transfer is to PTD2650. If it is associated with the keyword VERB transfer is to PTD2680.

PTD1980 The XTRAC subroutine is called to extract the 8-bit table control code for table 5.

PT2000 A test is made to determine if there are additional internal tables in this phrase entry. If there are not transfer is to PTD2850.

PTD2060 A check is made to determine if the indicator is a valid indicator for a new internal table. If it is not transfer is to PTD2750.

PTD2080 The XTRAC subroutine is called to extract the 8-bit table length.

PTD2100 A test is made to determine if we are running at DBUG level 4. If we are transfer is to PTD2850.

PTD2140 A test is made to determine if there are check entries for this phrase. If there are not transfer is to PTD2260.

PTD2160 Table 5 header is set to print.

PTD2200 A test is made to determine if all check entries have been processed. If they have transfer is to PTD2260. Otherwise, the DFJPTDP5 subroutine is called to dump the check entries.

PTD2260 The XTRAC subroutine is called to extract the 8-bit table control code.

PTD2280 A test is made to see if this is the end of the phrase entry, that is, a test for 7FFF. If it is transfer is to PTD2850.

PTD2300 A test is made to determine that this is a valid new table indicator. If it is not transfer is to PTD2750.

PTD2320 The XTRAC subroutine is called to extract the 8-bit table length code.

PTD2340 A test is made to determine if we are running at DBUG level 5. If we are transfer is to PTD2850.

PTD2390 A test is made to determine if there are expressions associated with this phrase. If there are not transfer is to PTD2480.

PTD2410 The DFJPTDP6 subroutine is called to dump the phrase associated expressions. DFJPTDP6 is called as a PLAN LOCAL.

PTD2490 The internal switch is set to indicate that the program list to be dumped is associated with the keyword EXIT.

PTD2470 The XTRAC subroutine is called to extract the 8-bit table control code.

PTD2490 A test is made to determine if the control code is 7FFF in the phrase indicator. If it is transfer is to PTD2850.

PTD2510 A test is made to determine if the control code is a valid new table indicator. If it is not transfer is to PTD2750.

PTD2530 The XTRAC subroutine is called to extract the 8-bit table length code.

PTD2550 A test is made to determine if we are running at DBUG evel 5. If we are transfer is to PTD2850.

PTD2600 A test is made to determine if the program list to be dumped is associated with the phrase keyword EXIT. If it is not transfer is to PTD2630.

PTD2610 The EXIT list header is set to print and transfer is to PTD1790.

PTD2630 A VERB list head is set to print. Transfer is to PTD1790.

PTD2650 The internal switch is set to indicate that the program list to be dumped is associated with VERB and transfer is to PTD2470.

PTD2680 The printer is skipped one line.

PTD2750 The contents of the switch words is dumped in hexadecimal notation.

PTD2810 A DFJPTDMP module is terminated by a CALL LRET.

PTD2820 The synonym indicators are accessed and transfer is to PTD1160.

PTD2850 A test is made of the synonym indicators. If the synonym indicators are zero, there is no additional synonym phrase. If there is a synonym transfer is to PTD2820.

PTD2870 If all 250 check sums have been processed transfer is to PTD2890. Otherwise, the loop is incremented to the next check sum and transfer is to PTD1000.

PTD2890 The end of the phrase table dump message is set to the print area and printed. Transfer is to PTD2810.

DFJPTDP1

The DFJPTDP1 module is a special purpose module that is used only by the phrase table dump. It is of absolutely no function in any other context DFJPTDP1 produces the dump of the phrase header information as controlled in the header information for each phrase entry. Four indicators JSECT, L4, L5, and NBUCK which are used commonly by the mainline DFJPTDMP and this subroutine are passed in the call list for DFJPTDP1. Additional information common to both the mainline and the subroutine are passed through COMMON.

PT1760 A printer double space is set up.

PT1780 The literal mask for the header line is set to the print area.

PT1830 The XTRAC subroutine is called to extract the 1-bit level zero indicator.

PT1840 If this is a level zero phrase transfer is to PT1890.

PT1850 The XTRAC is called to extract the 3-bit level indicator.

PT1870 If this is a blank level phrase transfer is to PT1920. Otherwise, transfer is to PT1900.

PT1890 The level indicator is set to a negative one. The negative 1 value is established to allow entry into the COMMON processing that is utilized for all other level indicator processing.

PT1900 The level code defined in the phrase entry is decremented by one.

PT1920 The XTRAC subroutine is called to extract the 8-bit phrase size indicator.

PT1940 The phrase entry size is set to the output area. The entry size is the number of internal PFILE records, that is the number of 80-bit or 64-bit records in PFILE on the 1130 PLAN system or System/360 PLAN system respectively.

PT1960 The GDATA displacement of the phrase entry is set to output. This value is printed to allow easier finding by the reader and by the user if looking at a straight dump of PFILE.

PT11020 The XTRAC subroutine is called to extract the 1-bit phrase type. Phrase types are either object or verb.

PT11040 A test is made to determine if this is an object phrase. If it is transfer is to PT11090.

PT11060 VERB is set to the print line to override the object designation as established in the phrase mask.

PT11090 The XTRAC subroutine is called to extract the 6-bit displacement to a chained phrase. This displacement is what is called synonym indicators in the flowchart of the phrase table dump routine.

PT11100 A dusplacement to the synonym phrase is set to the print area.

PT11120 XTRAC is called to extract the 8-bit relative sector of the synonym phrase.

PT11130 The relative sector of the synonym phrase is set to the print area.

PT11150 A pointer is set to the beginning of the phrase name in the phrase entry table.

PT11160 The print position indicator is set to print position 17 for output of the phrase name.

PT11180 XTRAC is called to extract a three-character phrase name entry.

PT11220 A three-name indicator is set to the print area.

PT11230 The print position indicator is indicated by four and the internal bit pointer is set to the next phrase entry location.

PT11290 A test is made to determine if the next portion of the phrase entry table contains another word of the phrase name or the start of internal table 2. If there is another word in the phrase name transfer is to PT11180.

PT11340 The phrase entry header line is printed. A return to the mainline is executed.

DFJPTDP2

This module is a single function module used in conjunction with the phrase table dump module. It has no other function in any other context. Data required for this subroutine and by the mainline is passed through COMMON. There are no calling parameters.

PT2750 The table 2 header line is printed.

PT2790 The XTRAC subroutine is called to extract the 1-bit format indicator.

PT2800 If the subscript for this initialization value is an expression transfer is to PT2940.

PT2830 XTRAC is called to extract the 14-bit constant subscript.

PT2850 The constant subscript is set to the print area.

PT2900 XTRAC is called to extract the 1-bit format indicator.

PT2910 If this initialization value is associated with an implied DO transfer is to PT21160. Otherwise, transfer is to PT21280.

PT2940 XTRAC is called to extract the 15-bit name associated with this symbolic subscript.

PT2970 The name is set to the print area.

PT21000 XTRAC is called to extract the 1-bit indicator that indicates whether or not this is an implied DO.

PT21010 If this is an implied DO transfer is to PT21160.

PT21030 XTRAC is called to extract the 15-bit subscript.

PT21050 The subscript is set to print.

PT21110 The bit pointer is incremented by 64 and transfer is to PT21340.

PT21160 XTRAC is called to extract the 15-bit implied DO displacement.

PT21200 XTRAC is called to extract the 16-bit implied DO increment.

PT21220 The increment is set to print.

PT21230 XTRAC is called to extract the 32-bit initialization value.

PT21250 The bit index is incremented by 32.

PT21280 The 32-bit initialization value is extracted.

PT21290 The initialization value is set to print.

PT21320 The bit index is incremented by 48.

PT21340 The table 2 line is printed.

PT21360 A test is made to determine if table 2 is completely processed. If it is not transfer is PT22790; otherwise, a return to the mainline is executed.

DFJPTDP3

This module is a single function module used exclusively with the phrase table dump. It has no use in any other connotation. All parameters required by this subroutine and by the mainline are passed through COMMON.

PT3760 The table 3 header is set to the print area and is printed.

PT3790 XTRAC is called to extract the 15-bit data name.

PT3820 The name is set to print.

PT3850 The bit index is incremented by 15.

PT3870 XTRAC is called to extract the 2-bit user exit number.

PT3880 If there is not a user exit associated with this data name transfer is to PT3920.

PT3900 The number of the user exit is set to the print area.

PT3920 XTRAC is called to extract the format indicator. The format indicator is a 1 bit if the mode of the variable is integer; otherwise, it will be a 0 bit indicating REAL.

PT3940 The format indicator either an R or an I is set to the print area.

PT31030 XTRAC is called to extract the 1-bit scale factor indicator.

PT31040 If there is not a scale factor transfer is to PT31200.

PT31060 XTRAC is called to extract the 3-bit scale factor.

PT31080 XTRAC is called to extract the 1-bit scale factor sign.

PT31140 XTRAC is called to extract the 9-bit subscript.

PT31170 The bit index is incremented by 179.

PT31180 If the subscript is zero transfer is to PT31320; otherwise, transfer is to PT31290.

PT31200 XTRAC is called to extract the 13-bit subscript.

PT31210 The bit index is incremented by 17.

PT31220 If the subscript is zero transfer is to PT31320.

PT31250 The subscript is set to the print area and transfer is to PT31500.

PT31320 The print position pointer is set to position 46.

PT31330 The bit index is incremented by eight.

PT31350 XTRAC is called to extract the 8-bit expression character.

PT31370 If the character is a comma transfer is to PT31460.

PT31390 The expression character is set to the print area.

PT31420 The print position pointer is incremented to the next print position.

PT31430 The bit index is incremented by eight and transfer is to PT31350.

PT31460 The bit index is incremented to round out to the end of a full word.

PT31500 The expression line is printed.

PT31520 If all expressions have been processed for this phrase, an exit is made from this routine; otherwise, transfer is to PT3790.

DFJPTDP5

This module is a special purpose module used only in conjunction with the phrase table dump module. It produces the dump of the internal check entry table.

PT5760 A test is made to determine if the entire table has been processed. If it is transfer is to PT51960.

PT5770 XTRAC is called to extract a 2-bit test type, that is, to determine whether the test is for TRUE, FALSE, REAL or NOT FALSE.

PT5790 The test type is set to print.

PT5970 XTRAC is called to extract a 13-bit subscript.

PT5990 XTRAC is called to to extract the 1-bit suffix indicator. This indicator determines whether there is an additional suffix record in internal table 5.

PT51010 The subscript is set to the print area.

PT51080 The bit index is incremented by 16.

PT51090 XTRAC is called to extract the 13-bit suffix indicator.

PT51110 If there is not a suffix record transfer is to PT51410.

PT51120 The bit index is incremented by 16.

PT51130 This check entry line is printed and transfer is to PT5760.

PT51160 XTRAC is called to extract the 1-bit format indicator.

PT51180 XTRAC is called to extract the 15-bit symbol.

PT51210 The symbol is set to print.

PT51240 The bit index is incremented by 16.

PT51260 A test is made to determine if there is a relative subscript. If there is transfer is to PT51320.

PT51270 The subscript is set equal to 1.

PT51280 The subscript is set to print.

PT51290 The bit index is incremented by 13 and transfer is to PT51090.

PT51320 Plus sign is set to output.

PT51350 XTRAC is called to extract the 16-bit subscript. Transfer is to PT51090.

PT51370 The subscript is set to the print area.

PT51380 The bit index is incremented by 32. DICK: is transfer to PT51090 required?????

PT51410 XTRAC is called to extract the 2-bit suffix type indicator. This determines whether the suffix is a program list, a literal, a subscript, or a push phrase.

PT51430 The action code, that is an A, C, P or a blank, is set to print.

PT51520 XTRAC is called to extract the 14-bit subscript.

PT51530 If there is not a suffix record transfer is to PT51570.

PT51550 The subscript is set to print.

PT51570 The suffix switch is reset.

PT51580 The bit index is incremented by 16.

PT51590 If there is not a suffix record transfer is to PT51160.

PT51610 If the action list is a program list transfer is to PT51750.

PT51620 The number of characters in the action list is determined.

PT51630 The print position indicator is set to 34.

PT51640 XTRAC is called to extract two characters from the action list.

PT51660 The two characters are set to the print area.

PT51670 The bit index is incremented by 16.

PT51680 The print position pointer is incremented by 2.

PT51690 If a full line of print has not been set up transfer is to PT51720.

PT51710 The current line is printed.

PT51750 If the action list is entirely processed transfer is to PT5760.

PT51770 A test is made to determine if the program is a name. If it is transfer is to PT51900.

PT51800 XTRAC is called to extract a 32-bit entry from the program list.

PT51820 If the extracted number is not a zero transfer is to PT51900.

PT51830 The program number 0 is set to print.

PT51850 The action list pointer is incremented.

PT51860 The bit index is incremented by 32.

PT51870 The check entry line is printed.

PT51900 XTRAC is called to extract the 8-character name.

PT51930 The program name is set to the print area.

PT51940 The action list pointer is incremented. Transfer is to PT51870.

PT51960 Exit from DPDP5 is to the next sequence instruction in the phrase table dump program.

DFJPTDP6

This module is a special purpose module used only in conjunction with the phrase table dump module. It has no other function. All data required by the mainline and by this subroutine are passed through COMMON.

PT6750 An internal switch is set to indicate the initial entry into the program.

PT6760 XTRAC is called to extract eight bits.

PT6780 A test is made to determine if the character just extracted is a dollar sign. If it is the beginning of the expression area is indicated and transfer is to PT61710.

PT6800 A test is made to determine if this is the initial entry. If it is not transfer is to PT6900.

PT6850 The initial entry switch is turned off.

PT6870 The table 6 header line is set to the print area and printed.

PT6900 A test is made to determine if this variable has a symbolic subscript. If it does transfer is to PT61510. If this variable does not have a scale factor transfer is to PT61420. If the indicator is invalid transfer is to PT61930.

PT6920 XTRAC is called to extract the 3-bit scale factor.

PT6940 XTRAC is called to extract the 1-bit sign indicator.

PT6960 The signed scale factor is set to print.

PT61010 XTRAC is called to extract the 9-bit subscript.

PT61030 The subscript is set to print.

PT61050 XTRAC is called to extract the 1-bit mode indicator that is set to the print area.

PT61160 The bit index is incremented by 16.

PT61180 The print position pointer is set to postion 32.

PT61200 XTRAC is called to extract an 2-bit EBCDIC character.

PT61220 If the character is a comma transfer is to PT61370.

PT61240 The extracted character is set to print.

PT61270 Print position indicator is incremented by 1.

PT61290 If a full line of print is not set up transfer is to PT61340.

PT61310 The line of expression is printed.

PT61330 The print position indicator is reset to position 34 allowing for an indentation of two spaces of continue lines.

PT61340 The bit index is incremented by 8. Transfer is to PT61200.

PT61370 The bit index is incremented by 8.

PT61380 The expression line is printed.

PT61390 If the entire table has been processed transfer is to PT61930. Otherwise, transfer is to PT6760.

PT61420 XTRAC is called to extract the 13-bit subscript and the subscript is set to print. Transfer is to PT61030.

The following processing is for expressions with symbolic subscripts.

PT61510 The 15-bit associated symbolic name is extracted and set to print.

PT61590 XTRAC is called to extract the 14-bit relative subscript.

PT61610 If it is relative 1 transfer is to PT61670.

PT61630 The relative subscript and a plus sign are set to the print area.

PT61670 The bit index is incremented by 32. Transfer is to PT61180.

The following narrative describes processing of the dollar sign expressionarea.

PT61710 The dollar sign formula area header is set to the print area and printed.

PT61740 XTRAC is called to extract the 2-bit EBCDIC character.

PT61770 The bit index is incremented by 8.

PT61790 A check is made to determine if the extracted character is a comma. If it is transfer is to PT61750.

PT61810 The extracted character is set to the print area.

PT61820 If the current line is not full transfer is to PT61740.

PT61830 A line full indicator is set.

PT61850 The expression line is printed.

PT61870 If a line complete indicator is set transfer is to PT61920.

PT61900 The character index is reset to print position 12.

PT61920 If there are more expressions to be processed transfer is to PT61740; otherwise, this subroutine returns control to the calling phrase table dump module at the call statement +1.

DFJREN (OS)

Tnis module insures that all direct access records that are currently in core buffers are written out on their respective files. It also closes the sequential files.

RTN270 The subroutine FLUSH is called to purge any DYNAMIC file records.

RTN310 The subroutine FLUSH is called to purge any PERMANENT file records.

RTN410-RTN510 The sequential files are closed and control is returned to the OS supervisor.

DFJRETN (DOS)

This module insures that the buffers in DFJDIOCS are flushed.

DRE290 A call is made to DFJDIOCS to insure that all buffers are quiesced.

DRE450 An EOJ macro is issued to return control to the DOS supervisor.

DFJSCHB

This subroutine searches the PERMANENT file control chain for an open file control block. If an equal is found, the address is returned in GPR3.

SCB270 GPR3 is set to point out the file control block chain.

SCB330 A test is made to see if this control block was the last in the chain. If yes, control is returned to the caller indicating an error.

SCB370 A test is made to see if this control block is for the requested file. If yes, the address is returned to the caller in GPR3. Otherwise, the next control block in the chain is located and transfer is to SCB330.

DFJSCHN

This subroutine searches the PERMANENT file drive chain. If an equal drive is found, the address of the control block is returned in GPR3.

SCN290 The search argument in GPR4 is set in a work area for the search and the PERMANENT file drive chain is located.

SCN370 A test is made to see if this is the requested drive. If not, transfer is to SCN490; otherwise, control is returned to the caller +4.

SCN490 A test is made to see if this is the last control block in the chain. If yes, control is returned to the caller +0 to indicate no equal drive. Otherwise, the pointer to the drive table is stepped and transfer is to SCN370.

DFJSIOCS (DOS)

This is the sequential file IOCS routine for DOS PLAN. It handles all the I/O for card readers, card punches, printers and magnetic tapes. This routine uses the subroutines CCBSTART and CCBWAIT along with SRCHIOC and COMRET in the PLAN loader.

SIO570 The TRUE end-of-file indicator is set in the status byte of the control block and control is returned to the caller through the COMRET exit in the PLAN loader.

SIO710 The NOD argument is validated by calling the subroutine SRCHIOC in the PLAN loader. If a NOD is invalid, control is returned to the caller through the COMRET subroutine in the loader.

SIO1010 A test is made to see if the device is capable of satisfying the request. If not, transfer is to SIO570.

SIO1070 A test is made to see if the file is open. If not, transfer is to SIO2930.

SIO1130 A test is made to see if the file is in the same status. That is, if the call type PLINP or PLOUT is the same as previous calls. If not, transfer is to SIO570.

SIO1170 A test is made to see if a physical end-of-file has occurred on the file. If yes, transfer SIO570.

SIO1210 A test is made to see if carriage control is required. If not, transfer is to SIO2130.

SIO1250 If this is a PLINP call, transfer is to SIO1930.

SIO1370 Line count maintenance is performed.

SIO1930 Carriage control character is set in the buffer. This character is set as a result of calling PCCTL.

SIO2130 A check is made to see if the next buffer is available. If not, transfer is to SIO3430.

SIO2190 The buffer address is updated.

SIO2310 If this is a PLOUT call, transfer is to SIO2510.

SIO2350 If the current card in the buffer is a /*, //, or UREND card, transfer is to SIO570. Otherwise, transfer is to SIO2790.

SIO2510 The buffer area is blank.

SIO2790 The new buffer pointer is set in the control block and exit is to the caller through the COMRET subroutine in the loader.

SIO2930 The file is opened. This includes setting the CCW count for the data CCW and initializing the buffer pointers. Transfer is to SIO1210.

SIO3430 If a wait is required on the last I/O operation, it is issued.

SIO3510 If the carriage control CCW operation is set to a no-op.

SIO3630 A test is made to see if the device is a tape. If it is not, transfer is to SIO3770; otherwise, the carriage control CCW operation code is set to a valid mode set character and transfer is to SIO4950.

SIO3770 If the device is not a 1052, transfer is to SIO4390.

SIO3850 If this is a PLOUT call, transfer is to SIO4150.

SIO4070 The buffer area is cleared and the CCW count is set and transfer is to SIO4950.

SIO4150 The blanks are backscanned off the buffer so that the CCW may be reduced accordingly and transfer is to SIO4090.

SIO4390 A test is made to see if carriage control is required. If yes, the data address of the data CCW is stepped past the carriage control character in the buffer.

SIO4510 If this is not a card device transfer is to SIO4730. Otherwise, a check is made to see if a stacker select is required. If yes, the control CCW op code is set for the appropriate stack select command and transfer is to SIO4950.

SIO4730 A test is made to see if carriage control is required. If yes, the carriage control CCW operation code is set accordingly.

SIO4950 The data CCW is completed. The address of the buffer and the CCW count are stored.

SIO4990 The subroutine CCBSTART in the loader is called to execute the I/O operation.

SIO5010 If this file is not double buffered, a wait is issued. If it is double buffered, transfer is to SIO5110.

SIO5110 The buffer pointers for the current record area are swapped and transfer is to SIO2190.

DFJTRACE (DOS)

This routine provides a tracing capability for the DOS PLAN system.

DTR190 The skeleton message is moved to the print area.

DTR210 If this is a LOCAL return, the local trailer is moved to the print area.

DTR290 If this is a checkpoint return, the checkpoint trailer is moved to the print area.

DTR410 The TRACE entries are completed. This includes the origin, the end, and the entrypoint of the programs involved.

DTR790 A GET time macro is issued and the time is placed in the message.

DTR1110 The TRACE line is printed and control is returned to the caller.

DFJTRACE (OS)

This module is loaded at initialization time if the PARM TRACE is specified in the EXEC JCL control card.

DTR270 The address of the current output buffer is located.

DTR310 The trace time is built depending on the type of call which can be a phrase abort, checkpoint reload, or a module to be entered.

DTR830 The registers are saved and the return from SIOCS is set.

DTR890 Exit is to the SIOCS routine to print the line.

ERLST

The ERLST subroutine is the error list module of the PLAN error processing module.

ERL390 The location of blank COMMON is accessed.

ERL410 The error list indicator is turned on within the loader. The subroutine is terminated by transfer to the DFJPLAN module.

ERRET

The following narratives describe the logic of the ERROR, ERREX, ERRET, and ERRAT subroutines.

ERR1310 The registers are saved according to standard convention.

ERR1410 The error message is assembled within the work area.

ERR2110 The error message is placed in the error stack.

ERR2130 If the call was to the ERROR subroutine the module is terminated with a transfer to DFJPLAN.

ERR2170 If the immediate mode of error processing is in effect transfer is to ERR2430.

ERR2270 If the error stack is found not to be full transfer is to ERR3130.

ERR2450 A save area is set and transfer is to ERR3330. (DOS)

ERR2710 If a user-exit module is to be invoked to process the error message transfer is to ERR3330. (OS)

ERR2810 If there is not room within memory for the PERRS module transfer is to ERR3330.

ERR2950 The local subroutine call is initiated to invoke execution of the PERRS module. Transfer is to ERR900. (OS)

ERR3130 If this call is to the ERREX module processing is terminated by transfer to DFJPLAN.

ERR3170 If this call is to the ERRAT subroutine processing continues, otherwise, transfer is to ERR3230.

ERR3210 The abort indicator is turned on.

ERR3230 The registers are restored and processing is terminated by a return to the caller.

ERR3330 PERRS is called via LCHEX.

ERR3530 The save area is released and transfer is to ERR3130.

EWRIT

The EWRIT subroutine provides a listing of errors contained in the PLAN error file. The PLAN error file is DYNAMIC file 255 within DYNAMIC drive 0.

EWR610 Registers are saved according to standard PLAN conventions.

EWR830 If drive 0 has been defined for this PLAN run processing continues, otherwise, transfer is to EWR1490.

EWR930 The FIND subroutine is called to open logical file 255 on logical drive 0.

EWR950 If the file was not successfully opened transfer is to EWR550.

EWR1050 The appropriate record to be printed is assembled.

EWR1390 The record is written to file 255.

EWR1410 The error file indicator is turned on.

EWR1490 The registers are restored. The subroutine is terminated by return to the caller at the next executable statement.

FIND

This logic describes processing of the FIND/READ/WRITE/RELES subroutines. These subroutines comprise the OS PLAN file processing.

FIN3050 The registers are saved according to standard OS conventions. Base registers are set.

FIN3170 The appropriate call type is set.

FIN3310 If this is not a READ or WRITE entry transfer is to FIN3590.

FIN3410 If the file control block is not open, exit is to the ERRABORT entry in DFJPLAN.

FIN3470 The drive code is extracted from the ID block.

FIN3590 If the drive code is invalid transfer is to FIN19730.

FIN3850 DOS processing to locate the control block for the associated drive.

FIN4450 OS processing to locate control block for the associated drive. If the search fails, transfer is to FIN19730.

FIN4850 Linkage into the processing routine is executed.

RELESR FIN5030, the NSQZ argument is saved.

FIN5050 If the amount of words to remain in the file is zero or negative transfer is to FIN5090.

FIN5070 An indicator is set to indicate that this release operation is for a partial release.

FIN5090 The RELESF subroutine is entered to release this file and transfer is to FIN7370.

FINDR This is the entry point for the FIND subroutine. If this find does not indicate a change in level transfer is to FIN5910.

FIN5510 The priority for the release operation is set.

FIN5830 The PRIREL subroutine is called to release all files up to the level of this file.

FIN5910 The file number is set.

FIN5970 The number of words to be allocated to the file is saved.

FIN6010 If the priority of the files is found not to be valid transfer is to FIN19730.

FIN6090 If the priority of the file is not zero transfer is to FIN1580.

FIN6170 The priority of the file is set to the level of the current phrase.

FIN6370 The VTOC record is read.

FIN6390 If the file is a new file transfer is to FIN6930.

FIN6410 The FD record for this file is read.

FIN6650 the file control block is marked as open and transfer is to FIN7430.

FIN6930 If this is a FINDL call, transfer is to FIN7370.

FIN6970 The NALLO parameter, that is, the number of words to be allocated to this file is converted to the number of segments.

FIN7070 The return linkage is saved.

FIN7090 The table of contents record for this drive is read into memory.

FIN7150 If there is space for this file transfer is to FIN7850.

FIN7250 If there are files that can be released to provide more space for the file transfer is to FIN7650.

FIN7370 The file control block is marked as closed.

FIN7430 ID(2) of the file control block is set.

FIN7510 Return is made to the caller.

FIN7650 The priority for the release operation is set.

FIN7250 The PRIREL subroutine is called to release the files by priority.

FIN7850 The return from the allocation is set. Transfer is to FIN15510.

FIN8230 The necessary arguments are accessed from the call list.

FIN8290 If KDIS and KOUNT are not valid transfer is to FIN19730.

FIN8430 If this is a read operation transfer is to FIN8530.

FIN8410 If KDIS and KOUNT result in a displacement outside the current value specified in the second word of the file control block processing continues, otherwise, transfer is to FIN8530.

FIN8510 The new value representing KDIS plus KOUNT is set into the second word of the file control block.

FIN8530 If KDIS plus KOUNT is greater than the second word of the file control block transfer is to FIN7370 .

FIN8610 KDIS and KOUNT are converted to a value in bytes.

FIN8810 The FDRTTR is retrieved from the coding within the first word within the file control block.

FIN8890 If the TTR is not valid transfer is to FIN19730.

FIN8930 The FD record is read.

FIN9030 If the required data is not within the current file size processing continues; otherwise, transfer is to FIN9810.

FIN9070 If this is a read operation transfer is to FIN7370.

FIN9430 If the required record is not within the current file allocation transfer is to FIN3180.

FIN9470 The new file size is set and transfer is to FIN2710.

FIN9590 The required allocation in words is set.

FIN9630 The number of words to be allocated is converted to segments.

FIN9650 The volume table of contents record is read.

FIN9670 The priority of this allocation is set.

FIN9690 The ALLOC subroutine is called to allocate the required space for this file. Transfer is to FIN8810.

FIN9810 The LOGICAL record number currently required is calculated.

FIN10030 The segment number containing the LOGICAL record is calculated.

FIN10190 The actual segment is located.

FIN10410 The required record is located.

FIN10510 The number of bytes remaining in the allocation is calculated.

FIN10830 If the user count is greater than the number of bytes remaining processing continues, otherwise, transfer is to FIN10930.

FIN10870 The remaining byte count is used.

FIN10930 The values of KDIS and KOUNT are updated.

FIN11010 The necessary read or write is initiated.

FIN3940 If the user specified count is not zero transfer is to FIN7510; otherwise, transfer is to FIN8810.

PRIREL This subroutine performs a priority release operation.

FIN11250 The return linkage is saved.

FIN11270 The return linkage from the RELES subroutine is set.

FIN11410 The SCAN subroutine is called to release any file numbered from 1 to 127.

FIN11450 The SCAN subroutine is called to release any file numbered from 128 to 255.

FIN11490 If the RELES priority is not equal to four transfer is to FIN11530. Otherwise, the PRIREL subroutine is terminated.

FIN11530 The RELES priority is incremented and transfer is to FIN11410.

SCAN The volume table of contents record for the current file is read.

FIN11670 A search is initiated for a file at the RELES priority.

FIN11750 If a file is found transfer is to FIN11950. Otherwise the subroutine is terminated by return to the user.

RELESF This subroutine is entered to release a file.

FIN11950 The volume table of contents record is read.

FIN11970 If the file is found not to exist transfer is to FIN15352.

FIN12070 The FD record is read.

FIN12090 If this is not a partial release transfer is to FIN13190.

FIN12210 If the NSQZ argument is not less than the current file size transfer is to FIN12270.

FIN12250 The new file size is established.

FIN12270 The NSQZ argument is converted to segments.

FIN12390 A pointer is set to the first segment to be released.

FIN12430 If this is not a NSQZ or an allocation function transfer is to FIN12650.

FIN12470 The partial release indicator is reset.

FIN12650 The FD record is updated.

FIN13190 The availability record is read.

FIN13390 The segment which was just released is recorded in the availability record.

FIN14490 The availability record is optimized.

FIN14990 The availability record is rewritten to the file.

FIN15030 The allocation counts are updated.

FIN15050 If this is a partial release transfer is to FIN15170.

FIN15090 The file directory ID is destroyed.

FIN15170 The file directory record is written.

FIN15190 If this is a partial release operation transfer is to FIN15352.

FIN15230 The volume table of contents record is updated.

FIN15352 The subroutine is terminated by return to the caller.

FIN15510 The volume table of contents record is read. record is read.

FIN15530 If the file is found to exist transfer is to FIN16990.

FIN15590 The new file directory skeleton is built.

FIN15730 The availability record for the file is read.

FIN15750 The space for the file is located from the availability record.

FIN15770 The availability record is updated.

FIN15790 The availability count is updated.

FIN16750 If this is a partial allocation transfer is to FIN15170.

FIN16790 The ID for the file directory block is created. Transfer is to FIN15170.

FIN16990 The FD record is read.

FIN17010 The partial allocation indicator is set. Transfer is to FIN15730.

FIN19730 ECODE and ENUMP are set. The FIND subroutine is terminated by transfer to DFJPLAN.

GMERG/PMERG

These subroutines invoke the DYNAMIC file and PERMANENT file merge facility.

XME610 The caller's registers are saved and a base register is set.

XME730-
XME950 The ID blocks of the merge files and the output files are tested. If any of the files are not open, transfer is to XME1170 to avoid the merge. Otherwise, the ID blocks are moved to a save area in the mainline.

XME1030 The LCHEX subroutine is used to invoke the module DFJGMERG or DFJPMERG to perform the merge.

XME1090 The caller's ID block for the output file is updated to reflect the size of the merge files and return is made to the caller.

XME1170 The appropriate error number and ECODE are set and exit is to ERRABORT in the mainline to force a phrase abort.

GSORT/PSORT

These subroutines invoke the PERMANENT file and DYNAMIC file sort facility.

XSO470 The caller's registers are saved and a base register is set.

XSO610 If the ID block of the file to be sorted is not open, transfer is to XSO830.

XSO650 The ID block is moved to a save area in the mainline.

XSO710 The LCHEX subroutine is used to invoke the SORT modules DFJ*SRTA and DFJ*SRTB. On return from the LCHEX routine, control is returned to the caller.

XSO830 The appropriate error number and ECODE is set and exit is to ERRABORT in the mainline to force a phrase abort.

NUSER, IUSER, GUSER, EUSER

These subroutines are the user-exit interface subroutines for DFJPSCAN, the PLAN interpreter.

NUS1550 This is the GUSER call processing. The addresses of the character access routine and the next character bucket in DFJPSCAN are located.

NUS1590 If the next character is a comma or a semicolon, transfer is to NUS1850.

NUS1630 The next character is moved to the user's array word.

NUS1650 The access routine in DFJPSCAN is called to get the next character in the input stream and transfer is to NUS1850.

NUS1730 This is the NUSER processing. If this is not the first call to NUSER, transfer is to NUS1910.

NUS1810 ISUBS and ISW which are located in PSCAN are moved to the user word arrays.

NUS1850 Exit is to the calling program.

NUS1910 The CAP pointer for the user ISUBS argument is incremented.

NUS2050 The user's ISUBS is tested and if invalid, transfer is to NUS1810.

NUS2090 The ISW arguments are set to a positive value and transfer is to NUS1810.

NUS2130 This is the EUSER processing. The N1, N2, and literal are moved to DFJPSCAN for processing on return.

NUS2370 Exit is to RETURN in DFJPLAN which simulates a CALL LRET from the user-exit module. Note that there no IUSER processing, it is a no-op.

INPUT

The INPUT subroutine retrieves the image of the current PLAN statement and places it in the user-specified array in memory.

INP730 Registers are saved according to the standard OS conventions. The base register is set.

INP1070 PFINPUTA record of the PLAN language dictionary is read into memory.

INP1210 The end of the PLAN statement is located.

INP1590 The calculation is made of the number of blanks that must be inserted beyond the end of the PLAN statement.

INP1830 The statement is moved to the user-specified array.

INP2110 If blank characters are not required at the end of the statement transfer is to INP2310.

INP2150 Positions within the user array to the right of the semicolon of the PLAN statement are set to blank.

INP2310 Registers are restored. Subroutine is terminated by return to the caller.

IOCS (DOS)

This subroutine allows reassignment of the standard PLAN input and output devices.

IOC470 The caller's registers are saved and the base registers are set.

IOC590 The INPUT argument is validated using the subroutine SRCH. If the argument is invalid, transfer is to IOC630.

IOC610 The current input device code is set to the caller's argument.

IOC630 The LIST argument is validated using the subroutine SRCH. If the argument is invalid, control is returned to the caller.

IOC670 The current output device code is set to the caller's argument and control is returned to the caller.

IOCS (OS)

The IOCS subroutine allows the PLAN input and output device specifications to be altered during execution.

IOC470 Registers are saved according to standard OS conventions. The base register is set.

IOC590 If the device code specified by the input argument is not valid, transfer is to IOC890.

IOC610 A pointer to the new current input device is set.

IOC890 If the list argument is found to be valid transfer is to IOC950. Otherwise the subroutine is terminated by return to the caller.

IOC950 The output device specification is altered to the user-specified device. Subroutine is terminated by return to the caller.

LCHEX (OS)

This subroutine allows exit to a module that overlays the calling module. The system status and the program area are saved on the checkpoint file.

LCH990 The LIST subroutine is called to manipulate the pop-up list.

LCH1190 The checkpoint control record is built in a work area in the mainline.

LCH1390-LCH1590 A test is made to see if a checkpoint file exists and if there is enough room to write the checkpoint. If yes, transfer is to LCH1710; otherwise an error number of 110 is set and exit is to 'ERRABORT' in the mainline to cause a phrase abort.

LCH1710 The checkpoint control record is written on the file.

LCH1950 The active program area is written on the checkpoint file.

LCH2090 The system status is updated and exit is to PLANLOPF in the mainline to load the next program in the pop-up list.

LCHEX (DOS)

This subroutine causes the modules named in the argument list to be invoked through the PLAN checkpoint facility.

LCH690 The callers registers are saved and a test is made to see if this subroutine is in the calling module. If not, exit is to ERLINK in the loader to force a phrase abort.

LCH870-LCH1190 The PSCB for this module is saved and a test is made to see if there is enough room in the checkpoint file for this module. If not, exit is to ERLINK in the loader to cause a phrase abort.

LCH1430-LCH1650 If a PSCB does not exist for this module, the module is written on the checkpoint file immediately. Otherwise, the PSCB is marked as requiring a checkpoint and exit is to the LEX subroutine to update the pop-up list.

LCH1970-LCH2230 This routine is used to write this module on the checkpoint file when the loader determines that the module will be overlaid.

LCH2530-LCH3430 This routine is entered from the loader when an asterisk is found in the pop-up list. It restores the program area to the condition at the time of the CALL LCHEX and returns control to the calling module.

LEX

The LEX subroutine provides transfer to the resident PLAN loader with manipulation of the program pop-up list.

LEX610 The LIST subroutine is called to do the necessary manipulation of the program pop-up list.

LEX670 A pointer is set to blank COMMON. The subroutine is terminated by transfer to DFJPLAN.

LIST

The LIST subroutine is called to manipulate the pop-up program list.

LIS730 The registers are saved according to conventions.

LIS940 If the user count is zero transfer is to LIS2450.

LIS1035 The user count is rounded to an even integer.

LIS1095 If this is a negative call, transfer is to LIS2532.

LIS1175 Pointers are set to the user array and the pop-up list. If the user entry is zero transfer is to LIS2472.

LIS1890 If the pop-up list has not overflowed transfer is to LIS2020.

LIS1910 Exit is to DFJPLAN for phrase abort.

LIS2020 The entry is moved to the pop-up list. If not, the last entry transfer is to LIS1890.

LIS2430 The pop-up list pointers are updated.

LIS2450 Control is returned to the caller.

LIS2472 The pop-up list is reset and transfer is to LIS2020.

LIS2532 A pop-up list entry is moved to the users array.

LIS2536 If the list entry was zero, transfer is to LIS2430.

LIS2544 If the user count is not zero transfer is to LIS2532, otherwise, transfer is to LIS2430.

LISTB

The LISTB subroutine is entered to add one program to the bottom of the pop-up list.

LIB530 Registers are saved according to standard OS conventions. Base registers are set.

LIB770 A pointer is set to the end of the pop-up list.

LIB890 If addition of one program will not cause pop-up list overflow transfer is to LIB1030.

LIB990 The LISTB subroutine is terminated by transfer to DFJPLAN.

LIB1030 The list pointer is updated to reflect the new entry to be added.

LIB1110 The current pop-up list entries are shifted by one.

LIB1290 The new name is added to the bottom of the list.

LIB1310 Registers are restored and the subroutine is terminated by transfer to DFJPLAN.

LISTZ

This subroutine resets the pop-up list. It performs the same function as a CALL LIST (1,0).

LSZ1800 The pop-up list is reset and control is returned to the caller.

LNCHX

This subroutine resets the status of the checkpoint file.

LNX1800 The note pointer to the checkpoint file is reset to zero and control is returned to the caller.

LNRET (OS)

The LNRET subroutine is called to terminate the return chain of LOCAL list processing.

LNR410 Registers are saved according to standard PLAN convention.

LNR450 A pointer is set to BLANK COMMON.

LNR490 The execution level indicator is reset to zero.

LNR510 The LOCAL chain is cleared.

LNR650 Registers are restored and the subroutine is terminated by return to the calling module at the next executable statement.

LNRET (DOS)

This subroutine cancels any LOCAL processing. The caller of this module becomes the mainline program.

LNR410 The subroutine CLOCAL in the loader is called to clear the LOCAL chain and control is returned to the caller.

LOCAL (OS)

The LOCAL subroutine provides multiple level LOCAL's, that is, the ability to call a LOCAL from a LOCAL for the PLAN system.

LOC710 Registers are saved according to standard PLAN conventions.

LOC750 The LIST subroutine is called to manipulate the pop-up program list.

LOC810 A pointer is set to blank COMMON.

LOC850 A register is set to point to the parameter list. The subroutine is terminated by transfer to the DFJPLAN module.

LOCAL (DOS)

This subroutine causes the program named in the argument list or the next program in the pop-up list to be loaded and executed as a subprogram.

LOC730 The subroutine LIST is called to process the N and L arguments.

LOC890 A test is made to see if the LOCAL subroutine itself is within the calling module. If not, exit is to ERLINK in the loader to force a phrase abort.

LOC1050 The address of the callers argument list is saved in the loader to be passed to the called program.

LOC1250 The PCB for this module is marked as a LOCAL caller. This prevents reuse of this copy of the module until control is returned from the LOCAL module. Exit is to NEXTLOAD in the pop-up list to load the subprogram.

LOC1690 This is a special entry to write the module in the checkpoint file.

LOC2270 Return from the called program is to here. The system status is updated and return is made to the caller of this subroutine.

LREPT

The LREPT subroutine is called to repeat execution of the current phrase.

REP430 A pointer is set to BLANK COMMON.

REP470 The repeat indicator is turned on within the resident loader indicating that the current phrase should be repeated. This indicator is interrogated by PSCAN. The subroutine is terminated by transfer to DFJPLAN.

LRET

The LRET subroutine provides linkage to the resident PLAN loader with no manipulation of the program pop-up list.

LRT450 A pointer is set to BLANK COMMON. The subroutine is terminated by transfer to DFJPLAN.

LSAV, LRLD

The LSAV and LRLD subroutines are provided only for the function of providing compatibility to the 1130 PLAN system.

LSA290 Error codes are set to indicate an invalid call. Subroutine is terminated by transfer to DFJPLAN.

NDEF

The NDEF subroutine provides the user with the ability to test any PLAN word for a content of a logical TRUE logical FALSE or REAL.

NDE790 If the current contents of the user-specified word is not logical TRUE transfer is to NDE270.

NDE810 The subroutine is terminated with a value of logical TRUE.

NDE890 If the content of the user-specified word is not a logical FALSE transfer is to NDE330.

NDE930 The subroutine is terminated with an indicator set to logical FALSE.

NDE1010 The subroutine is terminated with an indicator set to a value of REAL.

PAIN

The PAIN and PAOUT subroutines are the A format input and output subroutines of the OS PLAN I/O package.

PAI1410 The registers are saved according to standard conventions. The base register is initiated.

PAI1930 If the device code specified is valid transfer is to PAI2070; otherwise, the subroutine is terminated by return to the caller.

PAI2070 If the arguments in the call list are valid transfer is to PAI2270; otherwise, the subroutine is terminated by return to the caller.

PAI2270 A pointer is set to the beginning and the end of the buffer.

PAI2390 A pointer is set to the data to be moved.

PAI2692 The data is moved between the system buffer and the user-specified array. The subroutine is terminated by return to the user.

PBFTR

The PBFTR subroutine provides for a transfer for the entire contents of one buffer to a second buffer.

PBF670 If the first specified device is valid transfer is to PBF710; otherwise, the subroutine is terminated by return to the user.

PBF710 A pointer is set to the buffer associated with the first device.

PBF810 If the second specified device is valid transfer is to PBF870; otherwise the subroutine is terminated by return to the caller.

PBF870 A pointer is set to the buffer associated with the second specified device.

PBF1010 The contents of buffer one are transferred to buffer two. Subroutine is terminated by return to the user.

PBTST

The PBTST subroutine is the bit manipulation extract under mask and test under mask logical routine of the OS PLAN I/O package.

TST1130 Registers are saved according to standard conventions.

TST1230 The user-specified NWRD is set to zero.

TST1290 If the OP code specified is zero transfer is to TST2270.

TST1330 The result is initiated to all 1-bits.

TST1350 If the OP code specified is negative transfer is to TST2270.

TST1370 If the OP code specified is not valid transfer is to TST2290.

TST1470 The test mask is assembled based on the bits specified to be tested.

TST1670 If the operation code specified is less than four transfer is to TST2090.

TST1710 A bit test is performed.

TST1890 A result of the bit test is placed in the end skip argument.

TST1910 If the operation code specified is 1,2,3,5,6,7,9,10, or 11 transfer is to TST2090.

TST1950 If the operation code specified is a 4 or an 8 transfer is to TST2290.

TST1990 The bits corresponding to the NBIT argument are accessed. Transfer is to TST2290.

TST2090 The required result to be placed in NWRD are accumulated.

TST2270 The argument is set to the user-specified NWRD.

TST2290 The registers are restored. Subroutine is terminated by return to the caller.

PBUSY, PDBFA, PDBFB, PDBFC, PDBFD, PDBFE, PSBFA, PSBFB, PSBFC, PSBFD, PSBFE

These routines are provided on the DOS and OS PLAN systems only for 1130 compatibility. They are no-ops and consist only of a single instruction which is a return on register 14.

PCAF, PCAI, PCEA, PCFA, PCIA

These are the core-to-core conversion routines They are interface routines into the actual conversion routines.

900 The caller's registers are saved.

1000 The entrypoints for the setup and conversion routines are set.

1100 Register 2 is set to indicate the mode of the conversion routine, INTEGER or REAL.

1200 Exit is to the setup routine DFJISET.

PCCTL

The PCCTL subroutine is the device function control routine of the OS PLAN I/O package. It provides for such functions as carriage control, and stacker select.

PCC810 If the device code specified is valid transfer is to PCC290. Otherwise the subroutine is terminated by return to the caller.

PCC950 If the arguments specified in the call list are valid transfer is to PCC1090.

PCC1030 The arguments are set to default values.

PCC1090 The carriage control character is set for the next operation. Subroutine is terminated by return to the caller.

PCOMP

PCO250 Registers are saved according to the standard conventions.

PCO350 The result is set to indicate that the first array was found to be low.

PCO360 The registers are returned. Subroutine is terminated by return to the caller.

PEOF

This is a function subroutine that tests a device for an occurrence of a logical or physical end-of-file. FP register 0 is used to return the result.

PEO830 FP register 0 is reset to indicate a physical end-of-file condition.

PEO950 The subroutine SRCHIOC is called to validate the NOD argument. If NOD is invalid, control is returned to the caller.

PEO990 The status of the file specified by the NOD argument is tested. If a physical EOF condition has occurred, control is returned to the caller.

PEO1030 FP register 0 is set to positive value to indicate no EOF condition.

PEO1050 The file status is tested for a logical end-of-file status and if not present, control is returned to the caller.

PFO1090 FP register 0 is set to a negative value indicating a logical end-of-file condition and control is returned to the caller.

PENDF (DOS)

This subroutine closes a sequential file. If the unit is a magnetic tape, and end-of-file mark is written and the file is rewound.

PEN490-PEN730 The sequential file control block chain is searched to determine if the file exists and is open. If not, control is returned to the caller.

PEN770 If the device is not a tape unit, transfer is to PEN1150.

PEN910 If the file was in output status, a tape mark is written. The tape is rewound to the load point.

PEN1150 The subroutine DFJFMAIN is called to release the core for the file control block and buffers. Exit is to the caller.

PENDF (OS)

This routine closes a sequential file. The data set is repositioned to the first record.

PEN430 The subroutine SRCHIOC is called to validate the NOD argument. If it is invalid, control is returned to the caller.

PEN510 If the file has not been used, control is returned to the caller.

PEN550-PEN850 If the file is in output status, the buffers are flushed if necessary.

PEN950-PEN1330 The file status and buffer pointers are initialized and the first buffer is set to blanks.

PEN1350 A TCLOSE macro is issued to reposition the data set to the beginning of the file and control is returned to the caller.

PEOUT, PFOUT, PFIN, PIOUT, PIIN

These are the PLAN sequential conversion subroutines. These are actually routines which just interface into the actual conversion routines.

900 The caller's registers are saved.

1600 The conversion routine and the setup routine entrypoints are set.

1700 Exit from this routine is to the setup routine DFJCSECT.

PFSPC

The PFSPC subroutine provides a linkage to allow the user to determine the amount of file space available on a DYNAMIC drive at any priority.

PFS630 The caller argument list is accessed.

PFS690 The return parameter is initiated to zero.

PFS770 If the specified drive is invalid transfer is to PFS1930.

PFS1930 If a priority is specified transfer is to PFS1530.

PFS1450 The level of the current phrase is accessed in a set as the priority for which the search is to be initiated.

PFS1530 The volume table of contents record is read.

PFS1770 If any level change is indicated transfer is to PFS1970.

PFS1870 A determination is made of the space available at the required priority.

PFS1930 The determined amount of space is set to the user argument. The subroutine is terminated by return to the caller.

PFS1970 A determination is made of the available space of the highest of the two priorities encountered at the level change. Transfer is to PFS1870.

PHIN

This subroutine retrieves EBCDIC literals from the literal file as established by the PHOUT subroutine or by the PDIAG module as initiated by the SET LITERAL command. Calling parameters to this subroutine are a pointer to the file control block, the literal number that is to be extracted, and the location in memory at which the literal is to be placed. The literal location in the user's array will be a positive literal count if execution of this subroutine is successful. If the location is a fixed-point zero the file control block was found to be invalid or not opened when the subroutine was called. If the position is a minus one, the header of the indicated file was found not to be valid for a literal file. If the position is a minus two, the requested literal number was higher than the greatest literal number contained in the file. If the value is a minus three, the literal number was not found to be in the file.

PHI190 A test is made to determine if the file is properly opened. If it is transfer is to PHI250.

PHI210 The error return is set to zero.

PHI230 Control is returned to the program at call +1.

PHI250 A test is made to determine if a literal file is properly initialized. If it is transfer is to PHI230. Otherwise, the error return is set to a minus one and transfer is to PHI230.

PHI320 a test is made to determine if the number of the requested literal is larger than the highest-numbered literal contained in the file. If it is not, transfer is to PHI370.

PHI340 The error return is set to a -2 and transferred to PHI230.

PHI370 The literal index record is read into memory.

PHI380 A test is made to determine if the literal is in file. If it is transfer is to PHI430.

PHI410 The error return is set to -3 and transfer is to PHI230.

PHI430 The literal length indicator is read into memory.

PHI460 The length of the literal record is calculated from the number of characters in the literal.

PHI480 The literal is read into memory into the user's array and transfer is to PHI230.

PHOUT

This subroutine is used to store literals in the standard PLAN literal file. The subroutine is required by the PDIAG module. The PDIAG module is initiated as a result of the SET LITERAL command. Calling parameters of the PHOUT subroutine are a pointer to the open file control block, the number of the literal that is to be added to the literal file, and the location in memory that contains the PLAN literal text of the literal to be added to the file. The format of the literal file is as follows:

The first thru word of the file contains a logical FALSE that is, 7FFFFFFF.

The second word of the file contains the number of PLAN words within the file.

The third word of the file indicates the highest number of any literal contained in the file.

The fourth word of the file indicates the number of FORTRAN words that have been used from the end of the file toward the beginning of the file for storage of literal information. Each literal is stored as the literal number, the literal character count followed by the literal text.

PHO210 A test is made to determine if if the file is properly opened. If it is transfer is to PHO270.

PHO230 The literal count in the users array is set to zero.

PHO250 Control is returned to the user at call + 1.

PHO270 A four-word file header is read into file memory.

PHO300 A test is made to determine if the first word is FALSE. If it is transfer is to PHO720.

PHO320 A four-word file header block is initialized and written to the literal file.

PHO390 The remainder of the file is set to logical FALSE.

PHO520 A test is made to determine if this is a delete operation. A literal delete operation may be as a result of entry of an equal number literal or may be indicated by a negative or zero. literal count. If this is a delete operation transfer is to PHO250.

PHO540 A test is made to determine if there already exists in this file a literal with the same number. If there is transfer is to PHO740.

PHO580 The displacement at which this literal will be stored is written in the literal displacement table.

PHO600 The literal and literal number are written to the file.

PHO640 The space used indicator in the file header is updated.

PHO660 If required, the indicator in the header is updated to reflect the new highest literal number.

PHO690 A four-word header is rewritten to the literal file.

PHO720 A test is made to determine if this a literal delete. If it is not transfer is to PHO540.

PHO740 If the literal to be deleted is greater than the highest-numbered literal currently in the file, transfer is to PHO250.

PHO760 The displacement to the literal to be deleted is read and then is set to logical FALSE.

PHO800 The literal count of the literal to be extraced from the file is written into memory.

PHO820 The literal count is converted to record size.

PHO840 The total record size of the literal to be deleted is used to adjust the space used indicator.

PHO860 The displacement table for all literals that are to be pushed down as a result of the delete is updated to reflect the size of the literal that is to be deleted.

PHO980 All literals which were beyond the literal to be deleted are pushed down to fill up the unused space.

PHO1090 The literal file header is updated and is rewritten to the literal file. A test is made to determine if this is an ADD literal. If it is transfer is to PHO580; otherwise, transfer is to PHO250.

PHTOE

The PHTOE subroutine converts hexadecimal notation to EBCDIC representation so that it may be printed by PLAN I/O package.

PHT550 Registers are saved according to standard conventions.

PHT630 The PLAN word is converted to eight bytes .

PHT810 The TO and FROM pointers are incremented.

PHT870 If the last input word has not been processed transfer is to PHT630.

PHT890 Registers are restored. Subroutine is terminated by return to the caller.

PIOC

This is a function subroutine which tests the availability of a device.

PIO690 F. P. register 0 is set to zero to indicate the requested unit is not available.

PIO710 The subroutine SRCHIOC is called to validate the NOD argument. If invalid, control is returned to the caller.

PIO750 FP register 0 is set to a positive value to indicate that the requested unit is free and control is returned to the caller.

PLINP

The PLINP subroutine is the input routine of the OS I/O package.

PLI410 The call type is set to indicate a PLINP call.

PLI430 The DFJSIOCS subroutine is located. The subroutine is terminated by transfer to DFJSIOCS.

PLOUT

The PLOUT subroutine is the output routine of the OS PLAN I/O package.

PLO410 The call type is set to indicate a PLOUT call.

PLO430 Linkage to the DFJSIOCS module is set. The subroutine is terminated by transfer to DFJSIOCS.

PPACK

The PPACK subroutine allows a user to pack a right-adjusted byte of any PLAN word into any byte position of a character array.

PPA610 The registers are saved according to standard conventions. The argument list is accessed.

PPA650 The address of the word into which the byte is to be packed is accessed.

PPA710 The required byte is transmitted to the TO array.

PPA730 The registers are restored. The subroutine is terminated by return to the caller at the next executable statement.

PPAGL

The PPAGL subroutine is used to set the page length for the PLAN I/O package.

PGL490 If the specified device code is valid transfer is to PGL570. Otherwise, the subroutine is terminated by return to the caller.

PGL570 If the user-specified argument is valid transfer is to PGL630; otherwise processing is terminated by return to the caller.

PGL630 The logical page length is set equal to the new specified value. The subroutine is terminated by return to the user.

PRGIO

The PRGIO subroutine is a common subroutine to perform the functions required of the PARGO and PARGI subroutines.

RGO230 Registers are saved according to standard OS conventions. The argument list is accessed.

RGO310 A pointer to the location within the communication array is set.

RGO460 The array is moved to or from common as required by the call.

RGO540 The registers are restored. The PRGIO subroutine is terminated by return to the caller at the next executable statement.

PUNPK

The PUNPK subroutine may be called to extract any byte of a character array and to place it right-justified into any other FORTRAN word.

PUN610 The registers are saved according to standard conventions. The argument list is accessed.

PUN630 The address of the array from which the byte is to be extracted is accessed.

PUN730 The byte is moved from the array to the receiving word.

PUN750 The registers are restored and processing is terminated by return to the caller at the next executable statement.

PUSH

The PUSH subroutine allows the user to force execution of a command that exists in memory in EBCDIC format.

PUS510 Base registers are set according to standard OS standard conventions.

PUS570 The appropriate I/O parameters are set up.

PUS690 The count of the number of characters within the EBCDIC literal are verified.

PUS790 A semicolon is inserted at the end of the literal.

PUS870 The literal text is written to the PFINPUTA record of PFILE, the language dictionary of PLAN.

PUS890 The repeat phrase indicator is set for the interpreter DFJPSCAN.

PUS910 If any DYNAMIC drive FD records are in the program area they are purged. The subroutine is terminated by a transfer to DFJPLAN.

RWDATA

The RWDATA subroutine is the processing logic for the RDATA, WDATA, RDAT1, and WDAT1 subroutines.

RDA930 The registers are saved according to standard conventions.

RDA1210 If any of the call parameters are found to be invalid transfer is to RDA2770.

RDA1730 The displacement within the file is calculated.

RDA1970 If this call is a read call transfer is to RDA2070.

RDA2010 If the indicated displacement is greater than the second word of the file control block processing continues. Otherwise, processing continues to RDA2070.

RDA2050 The second word of the file control block is updated to the new value.

RDA2070 If the indicated displacement is greater than the current file size processing continues; otherwise, transfer is to RDA2770.

RDA2650 The appropriate read/write parameters are calculated.

RDA2670 A read or write operation is initiated to transmit the required data. The subroutine is terminated by return to the user.

RDA2770 The necessary error number is set and the subroutine is is terminated by transfer to the ERRABORT subroutine.

STVAL

The STVAL and GTVAL are array transmission subroutines.

STV670 Registers are saved according to standard PLAN conventions. The argument list is accessed.

STV770 The TO array and FROM array addressed in KOUNT are set.

STV870 The array is transferred.

STV1070 The registers are restored and the subroutine is terminated by return to the caller at the next executable statement.

TRUE

The TRUE and FALSE subroutines set the specified user word to the value assoicated with logical TRUE or logical FALSE.

TRU750 The registers are saved according to standard conventions.

TRU770 The argument list is accessed.

TRU790 The specified user word is set to the value of logical TRUE (8000000) or logical FALSE (7FFFFFF).

TRU810 The registers are restored and return is to the caller at the next executable statement.

XACES

This subroutine is a special purpose PFILE sector read subroutine that is used exclusively with a phrase table dump module. It has not function in any other use. The calling parameters are the relative PFILE sector numbers to be read into memory and the communication array subscript into which the sector is to be read.

XAC160 A test is made to determine if the file control block is proper. If it is transfer is to XAC210.

XAC180 File number 255 is set to the file control block. Transfer is to XAC230.

XAC210 A test is made to determine if the file is open. If it is transfer is to XAC250.

XAC230 GDATA is called to open PFILE.

XAC250 A test is made to determine if this dump is being made on the 1130. If it is transfer is to XAC320.

XAC270 The PFILE displacement is set equal to 128 FORTRAN words multiplied by the relative sector minus 1.

XAC290 The number of words to be read by RDATA is set as 128 FORTRAN words. Transfer is to XAC360.

XAC320 The PFILE read displacement is set equal to the file size in FORTRAN words minus 160 multiplied by the relative sector number.

XAC340 The number of FORTRAN words to be read is set equal to 160.

XAC360 RDATA is called to read the sector into the communication array. This subroutine returns control to the mainline program at the calling statement +1.

XBIT

This subroutine is a special purpose subroutine used only with the phrase table dump. Its function is to adjust the internal bit pointer and the count of the internal phrase entry table size. The only calling parameter is the increment/ decrement that is to be used in the required adjustment.

XBI90 The count of the number of bits still to be processed in this internal table is decremented by the amount of the calling parameter.

XBI100 The internal bit pointer is incremented by the amount of the calling parameter. Control is returned to the calling program at call +1.

XPRNT

This subroutine is a special purpose subroutine used only with the phrase table dump. Its function is to test the end-of-file indicator to skip to a new page if required and to print the existing line.

XPR100 A test is made of the physical end-of-file indicator. If a physical EOF has not been processed transfer is to XPR140.

XPR120 A skip to a new page is initiated.

XPR140 The current print buffer is printed and control is returned to the user at call +1.

XTRAC

This subroutine extracts a bit field from a PFILE entry as it exists in a communication array. The calling parameters are the bit number, the number of bits to be extracted, and the PLAN word that is to receive the extracted field. If the routine is to be executed on the 1130, a field of 16 bits or less is placed into the left two bits of a FORTRAN word. If the field width is greater than 16 when the subroutine is executed on an 1130 system or if the execution is on a System/360 the field extracted is right-justified in the 32-bit FORTRAN word.

XTR140 A pointer is set to the input read area.

XTR160 The FORTRAN word that is to receive the extracted field is cleared by call to PBTST.

XTR180 Calculations are made to determine the bit position, the relative record number for the start of the field.

XTR200 The internal record number is adjusted if the bit position indicated is greater than the size of an internal record.

XTR220 The appropriate FORTRAN word within the communication array is located.

XTR340 The desired field is extracted by calls to PBTST. Consecutive bits within the field are tested one at a time. If the bit is found to be on a subsequent call to PBTST places the bit in the receiving field. Subroutine execution is terminated by a return to the calling program at the calling statement +1.

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)