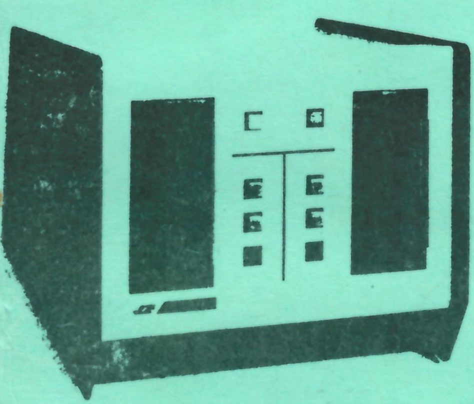
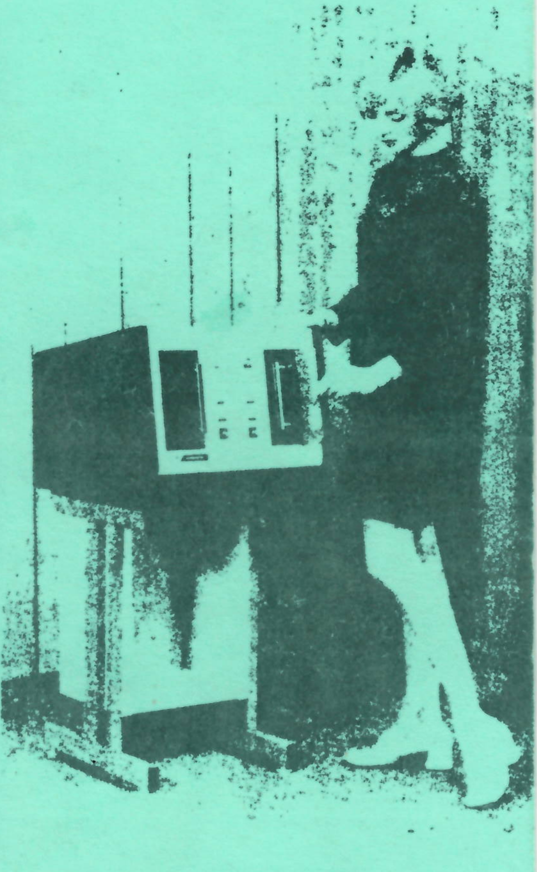
INCOTERM®

SPD/DOS DISKETTE OPERATING SYSTEM PROGRAMMER'S REFERENCE MANUAL





SPD/DOS PROGRAMMER'S REFERENCE MANUAL

ORDER NUMBER: MS-7178.0

DATE: AUGUST, 1975

PREFACE

SPD/DOS is a diskette resident operating system. This manual provides the necessary information for preparation of assembly language programs which are intended to run under SPD/DOS. Knowledge of the general structure of SPD/DOS as documented in the SPD/DOS Operator's Reference Manual is assumed.

This publication specifically documents SPD/DOS Version 6, and should not be construed to precisely describe earlier versions. It is the only SPD/DOS manual required for the SPD 10/20, SPD 10/24, SPD 10/25, and SPD 20/20.

It should be noted that the SPD 10/24 was especially designed for overseas customers. This system is built and sold solely by a licensee, TRANSAC. The SPD 10/24 is not available within the U. S. A.

Table of Contents

Table of Contents
(cont'd)

Table of Contents
(Cont'd)

Table of Contents
(cont'd)

Table of Contents
(cont'd)

List of Illustrations
(cont'd)

List of Tables

SECTION I

PROGRAM FORM AND EXECUTION

INTRODUCTION

Programs to be run under control of SPD/DOS are written in assembly language. See the "SPD Symbolic Assembly Language Reference Manual", MS-7215 for general details of the syntax and semantics of this language. See also, the list of other useful, related publications at Appendix A. Normally the source program will be prepared using the SPD/DOS EDIT or UPDATE utilities and then assembled to produce an executable object program using the SPD/DOS ASSEMBLE and RASSEMBL utilities.

PROGRAM RESTRICTIONS

Programs to be loaded by SPD/DOS must obey the following restrictions:

(a) BOOT mode programs cannot be loaded.

(b) No external memory can be specified (SIZE third parameter must not be given).

(c) Segment zero must not assemble data into the area X'0000' to X'00FF'.

(d) CNFG 10 programs will load only if SPD/DOS is running on a 10/20.

(e) CNFG 20 programs will load only if SPD/DOS is running on a 20/20 with at least the memory specified by the first parameter of SIZE.

(f) CNFG 24 programs will load only if SPD/DOS is running on a 10/24 with at least the memory specified by the first parameter of SIZE.

(g) CNFG 25 programs will load only if SPD/DOS is running on a 10/25.

(h) CNFG 0 programs will load if SPD/DOS is running on any machine with at least the memory specified by the first parameter of SIZE.

Note that it is possible to load data into the auto-exec areas (but not into the index register or cursor) but this practice is unadvisable if there is any possibility that the program may have to be loaded using some other device than the diskette.

ENTRY CONDITIONS

On completion of loading, control is passed to the entry point specified on the END card (or the first assembled location if no entry point is specified). The conditions at the point of entry are as follows:

(a) Interrupts are enabled but all devices are masked.

(b) The index register (cursor) and accumulator contents are undefined.

(c) Memory locations X'0000' - X'004D' contain the parameter field, starting with the initial non-blank character, as copied from the SPD/DOS command which loaded the program and right filled with blanks. If no parameter field was given, this area contains 78 blanks.

(d) The contents of all other memory locations not containing assembled data (e.g., BSS areas) are undefined.

(e) The screen and cursor are disabled (10/20 or 10/24) or all screens blanked and the cursor disabled (20/20 or 10/25).

(f) Positions X'83' - X'FF' of the diskette controller buffer are initialized with operating system control information as described in a separate section.

(g) The diskette unit itself is not busy.

(h) (10/25 and 20/20 only) Screen zero selected and display format set to 30 x 64.

(i) (10/24 only) Display setup as by IOR.

(j) The keyboard lights are all off.

(k) (20/20 only) If a printer is configured on a multi-printer controller, then the appropriate unit is selected and setup with the correct parameters.

PROGRAM EXECUTION

There are three possible modes for a program running under SPD/DOS.

(a) The program never returns to the DOS nucleus. A manual boot is required to return.

(b) The program will make an abnormal return to the nucleus which will cause complete reinitialization of the diskette buffer control informatic

(c) The program will make a normal return to the nucleus signalling successful or error completion. In this case the nucleus will assume that the diskette buffer control information is valid.

In the first case, there are no restrictions whatsoever on the executing program although it is desirable that any diskette input/output be performed in a manner which is compatible with SPD/DOS formats and in particular the nucleus bootstrap records on track zero of SPD/DOS diskettes should not be destroyed. Nearly all existing programs for the 10/20 and 20/20 should execute under SPD/DOS in this mode.

To execute an abnormal return to the nucleus (in effect, simulating a manual boot), the program issues the following instruction:

```
            CIO 7,7                       bootstrap from diskette
```

The only restriction which applies to such a program is that location X'94' (DBF&RTN) in the diskette buffer be unchanged from its entry value of X'FF' when the bootstrap is issued.

A program which intends to make a normal return to the nucleus must maintain the validity of the control information in positions X'83' - X'FF' of the diskette buffer. This may be done by following the rule of never changing any of these locations except indirectly through use of the SPD/DOS system subroutines. Control is returned to the nucleus using the D&EXIT system subroutine.

POWER RESTART

If the TPU is powered down, the information in the diskette controller buffer, including the system control information, is lost, making return to the nucleus impossible. Thus one possible convention is to make the power restart location point to a JMP $ loop. This will cause the terminal to come up "dead" on a power restart and the operator can restart with a manual boot.

It is also possible to keep a copy of the diskette buffer system control information in core memory and restore it on a power restart, thus allowing automatic restart.

A third possibility is to set DBF&RTN to X'FF' and issue a CIO bootstrap to the disk. A standard routine (D&POWR) is provided to achieve this action.

Programs which never return to the nucleus may use the power restart facility in any convenient manner.

PROGRAM DEBUGGING

Programs operating under SPD/DOS may use any of the standard interactive DEBUG packages available in the usual manner. In addition, the ZAP utility provides an external debugging capability. A disk bootstrap instruction (X'C977') issued from a program running under SPD/DOS without setting DBF&RTN causes an abnormal return to the nucleus which saves the core image for ZAP. Thus this instruction can be used to provide a breakpoint capability as follows: Load the the program using ZAP, Modify the breakpoint location to contain C977 and Jump to the program entry point.

If sufficient room is available, the D&POWR routine can be included in a program. In this case, the B command in ZAP provides a more elegant breakpoint facility which saves the contents of the (ACR) and ($X) before issuing the bootstrap instruction.

SECTION II

SPD/DOS CONTROL INFORMATION

The SPD/DOS control information is stored in the diskette controller buffer as described in this chapter. The DBF&XXX symbols are defined to have the appropriate buffer position value by the assemblers, without having to define them explicitly.

The parameter values which are described as "set by CNFG" can be modified by the CNFG utility. They are set by the nucleus from the label record (track zero, sector four) of unit zero on a manual or abnormal boot.

Printer Type

Name:	DBF&PTP
Address:	X'83'
Length:	1
Contents:	The printer type as set by CNFG.

X'01' = LP200, LP400

X'02' = P-100, P-165

X'03' = LP125, LP250, LP300

X'04' = P-15

X'05' = Termiprinter

X'FF' = No Printer

Display Size

Name:	DBF&DNL
Address:	X'84'
Length:	1
Contents:	Display number of lines set to 15 or 30 (X'0F' or X'1E') by CNFG. Always 30 if not 10/20.

Current Track Positions

Name:	DBF&TKP
Address:	X'85'
Length:	3
Contents:	Current track position for disk units 0,1,2 respectively. A value of zero forces recalibration. (Third byte for unit 2 currently unused, always zero.)

Currently Selected Unit

Name:	DBF&CSU
Address:	X'88'
Length:	1
Contents:	0,1 for currently selected diskette unit zero or one (2 reserved as possible future use value).

System Type

Name:	DBF&SYS
Address:	X'89'
Length:	1
Contents:	10 (X'0A) for SPD 10/20
	20 (X'14') for SPD 20/20
	24 (X'18') for SPD 10/24
	25 (X'19') for SPD 10/25

Memory Size

Name:	DBF&MEM
Address:	X'8A'
Length:	1
Contents:	X'0F' = 4K memory (SPD 10/20 or SPD 10/25)
	X'1F' = 8K memory (SPD 20/20 or 10/24)
	X'3F' = 16K memory (SPD 20/20 or 10/24)
	X'7F' = 32K memory (SPD 20/20)

Command Mode

Name:	DBF&MOD
Address:	X'8B'
Length:	1
Contents:	'C' (X'43') = command input from cards
	'F' (X'46') = command input from diskette file
	'K' (X'4B') = command input from keyboard
	'T' (X'54') = command input from paper tape

Command Input Unit

Name:	DBF&FUN
Address:	X'8C'
Length:	1
Contents:	(valid only if DBF&MOD = 'F')
	0, 1 for command input file on unit zero, one
	(2 reserved as possible future use value)

Command Input Track

Name:	DBF&FTR
Address:	X'8D'
Length:	1
Contents:	(valid only if DBF&MOD = 'F')
	Absolute track number of next record of command input file, updated appropriately by system routine to read command line.

Command Input Sector

Name:	DBF&FSC
Address:	X'8E'
Length:	1
Contents:	(valid only if DBF&MOD = 'F')
	Sector number of next record of command input file, updated appropriately by system routine to read command line.

Command Input Offset

Name:	DBF&FOF
Address:	X'8F'
Length:	1
Contents:	(valid only if DBF&MOD = 'F') Offset within sector (zero origin) of next record of command input file, updated appropriately by system routine to read command line.

Option Bits

Name:	DBF&OPT
Address:	X'90'
Length:	4
Contents:	26 bits for the 26 possible letter options on the SPD/DOS program load command line, on if on, else off. These bits are arranged from left to right as shown by the following table:

Table 2-1. Letter Options on the SPD/DOS Program Load Command Line

Letter	Option Byte	Option Bit
A	DBF&OPT	X'80'
B	DBF&OPT	X'40'
C	DBF&OPT	X'20'
D	DBF&OPT	X'10'
E	DBF&OPT	X'08'
F	DBF&OPT	X'04'
G	DBF&OPT	X'02'
H	DBF&OPT	X'01'
I	DBF&OPT+1	X'80'
J	DBF&OPT+1	X'40'
K	DBF&OPT+1	X'20'
L	DBF&OPT+1	X'10'
M	DBF&OPT+1	X'08'
N	DBF&OPT+1	X'04'
O	DBF&OPT+1	X'02'
P	DBF&OPT+1	X'01'
Q	DBF&OPT+2	X'80'
R	DBF&OPT+2	X'40'
S	DBF&OPT+2	X'20'
T	DBF&OPT+2	X'10'
U	DBF&OPT+2	X'08'
V	DBF&OPT+2	X'04'

Table 2-1. (Cont'd)

Letter	Option Byte	Option Bit
W	DBF&OPT+2	X'02'
X	DBF&OPT+2	X'01'
Y	DBF&OPT+3	X'80'
Z	DBF&OPT+3	X'40'

The unused six bits in DBF&OPT+3 are always zero.

Return Code

Name:	DBF&RTN
Address:	X'94'
Length:	1
Contents:	X'FF' - initial value set by nucleus; signals abnormal return if still set on exit.
	X'FE' - signal normal return with command in DBF&CMN as set by D&EXCM.
	X'00' - normal return as set by system exit subroutine (D&EXIT)
	01-99 - error code as set by system exit subroutine (D&EXIT)

Program Name

Name:	DBF&PNM
Address:	X'95'
Length:	2
Contents:	First two characters of program name in ASCII code. Used by nucleus to construct error code if DBF&RTN = 1-99 on exit.

Program Unit

Name:	DBF&PUN
Address:	X'97'
Length:	1
Contents:	0, 1 for current program loaded from unit zero, one. (2 reserved for possible future use value.)

Program First Track

Name:	DBF&PFT
Address:	X'98'
Length:	1
Contents:	Absolute track number of start of object file for currently loaded program.

Display End Refresh Character

Name:	DBF&DER
Address:	X'99'
Length:	1
Contents:	End refresh character, set to X'04' or X'74' by CNFG on SPD 10 or 10/24. Always X'01' otherwise (for SPD 20/20 or 10/25).

Printer Lines/Page

Name:	DBF&PLP
Address:	X'9A'
Length:	1
Contents:	Number of lines per printer page as set by CNFG. Only valid if printer present.

Printer Characters/Line

Name:	DBF&PCL
Address:	X'9B'
Length:	1
Contents:	Number of characters per line as set by CNFG. Only valid if printer present.

Card Reader Channel

Name:	DBF&CCH
Address:	X'9C'
Length:	1
Contents:	Channel number of system and reader as set by CNFG. Set to X'FF' if no card reader present.

Printer Channel

Name:	DBF&PCH
Address:	X'9D'
Length:	1
Contents:	Channel number of system printer as set by CNFG. For a multi-printer controller this is the unit number on the controller + (X'80'). Set to X'FF' if no printer configured.

Tape Reader Channel

Name:	DBF&TCH
Address:	X'9E'
Length:	1
Contents:	Channel number of system paper tape reader as set by CNFG. Se to X'FF' if no paper tape reader present.

Log Mode

Name:	DBF&LOG
Address:	X'9F'
Length:	1
Contents:	'L' (X'4C') = log mode set by nucleus .L command 'N' (X'4E') = no-log mode set by nucleus .N command Ø (X'00') = no-log mode set by manual boot

Multi-Printer Parameter Byte

Name:	DBF&MPP
Address:	X'A0'
Length:	1
Contents:	Multi-printer controller setup byte as set by CNFG. Only valid if multi-printer is configured.

Keyboard Type

Name:	DBF&KBT
Address:	X'A1'
Length:	1
Contents:	Keyboard type as set by CNFG. X'01' = upper case X'02' = upper/lower case X'03' = upper/lower, with lower case to be converted to upper case

Number of Units

Name:	DBF&NUN
Address:	X'A2'
Length:	1
Contents:	Number of disk units as set by CNFG (X'01' or X'02'); (X'03' reserved for possible future use value.)

Eject Fill Count

Name:	DBF&ENF
Address:	X'A3'
Length:	1
Use:	Relevant only if DBF&PTP = 5 (termiprinter). Null fill count used after page eject. Set by CNFG.

Line Feed Fill Count

Name:	DBF&LNF
Address:	X'A4'
Length:	1
Use:	Relevant only if DBF&PTP = 5 (termiprinter). Null fill count used after line feed. Set by CNFG.

Release Number

Name:	DBF&RNO
Address:	X'A5'
Length:	1
Use:	Release number of nucleus (X'06' = release 6).

Command Input Interlace Factor

Name:	DBF&FSF
Address:	X'A6'
Length:	1
Contents:	SIF for current command file. Valid only if DBF&MOD = 'F'

Reserved Area

Name:	None
Address:	X'A7'
Length:	9
Contents:	Always zero, reserved for future expansion of SPD/DOS

Interprogram Communication Area

Name:	DBF&ICA
Address:	X'B0'
Length:	16
Contents:	Set to zero by the nucleus on a manual boot or abnormal return, otherwise, unchanged by the nuclus or any DOS utilities. May be used to communicate between applications programs.

Overlay Segment Track Numbers

Name:	DBF&STR
Address:	X'C0'
Length:	32
Contents:	Absolute track numbers of the starting sector locations of object text records for the first 32 overlay segments. Entries for segments not present are undefined.

Overlay Segment Sector Numbers

Name:	DBF&SSC
Address:	X'E0'
Length:	32
Contents:	Sector numbers of the starting sector locations of object text records for the first 32 overlay segments. Entries for segments not present are undefined.

Returned Command

Name:	DBF&CMN
Address:	X'C0'
Length:	64
Contents:	Command to be executed by nucleus on return (DBF&RTN) set to X'FE'), as set by the D&EXCM routine. Note that this area overlays the DBF&STR and DBF&SSC areas, since it is only set when their use is no longer necessary.

SECTION III

SYSTEM SUBROUTINES -- GENERAL DESCRIPTION

An important component of SPD/DOS is a comprehensive set of system subroutines. These subroutines are provided in relocatable form for inclusion in programs using the IN pseudo-operation.

Every routine has a name of the form D&XXXX, where XXXX are four letters. Internal labels in each routine are of the form D&XXXXNN where NN is one or two digits, thus minimizing the possibility of accidental label duplication. All routines are intended to be used in background mode (interrupts enabled). This means that foreground (interrupt driven) tasks may execute at the same time without being delayed. In addition, all routines except D&ENTF, D&RLCS, D&RKBD and D&PRNT may be used in foreground mode (interrupts disabled), but in this case interrupts will be locked out for the duration of the call.

PHYSICAL LEVEL DISKETTE INPUT/OUTPUT

A set of routines is provided to perform diskette input/output on a sector by sector (physical) level. This access method can be used with any type of file, but its direct use in application programs is normally limited to data files since object files are not usually read or written directly and source files are normally accessed using the logical access (record by record) method.

Actual input/output is controlled by means of a six byte control block called a File Control Block (FCB) which has the format shown in Figure 3-1.

FCB&TRK	FCB&SEC
FCB&SIF	FCB&UNI
FCB&LTR	FCB&DIR

Figure 3-1. File Control Block (FCB)

The field names FCB&XXX are symbols whose value is equal to the offset of the field (e.g., FCB&UNI = 3). These symbols are defined automatically by the ASSEMBLE and RASSEMBL utilities.

The usage of the fields is as follows:

FCB&TRK Absolute track number (0-63) of the next sector to be accessed.

FCB&SEC Sector number (0-31) of the next sector to be accessed. Values > 31 are truncated modulo 32.

FCB&SIF Sector interlace factor. If a file is read sequentially, the question arises of the order in which sectors should be read. This is controlled by the sector interlace factor value. A value of one leads to the sequence:

```
TRACK N        SECTOR 0
TRACK N        SECTOR 1
TRACK N        SECTOR 2
  .    .          .    .
  .    .          .    .
TRACK N        SECTOR 31
TRACK N+1      SECTOR 0
TRACK N+1      SECTOR 1
```

etc.

This sequence requires an entire revolution for each sector to be read.

An interlace factor of 5 gives rise to the following sequence:

TRACK N	SECTOR 0
TRACK N	SECTOR 5
TRACK N	SECTOR 10
TRACK N	SECTOR 15
TRACK N	SECTOR 20
TRACK N	SECTOR 25
TRACK N	SECTOR 30
TRACK N	SECTOR 3
. .	. .
. .	. .
TRACK N	SECTOR 22
TRACK N	SECTOR 27
TRACK N+1	SECTOR 0
TRACK N+1	SECTOR 5

etc.

Providing that processing for each sector can be completed in less than 11 milliseconds, then six sectors can be processed on each revolution.

In general, the interlace factor may be any odd value from 1 to 31. The corresponding available processing time for each sector (to avoid extra revolutions) is 1+5*SIF-15 msec.

If a file is to be accessed in a purely random manner, the interlace factor is set to zero.

FCB&UNI	The diskette unit number. This is currently restricted to 0 or 1, but a value of 2 is reserved for possible future use.
FCB&LTR	Last track. The input/output routines will abort with an end of file indication if the value in FCB&TRK exceeds that in FCB&LTR.
FCB&DIR	This byte is used to remember the location of a directory entry when creating a file. It need not be examined, and must not be modified directly by an application program, except as described in D&CLOS. Its values as used by current system routines are:

X'80' = File opened by D&OPEN or D&OPSR

X'81' = File opened by D&OPSW

(Sector number of "created" directory entry)

+(X'40' if second entry in sector)

+(X'20' if "OK" on close)

The four basic routines for input/output are:

D&READ	read sector
D&VRFY	verify sector
D&WRIN	write initial sector
D&WRIT	write sector

Of these routines, D&WRIN is used only when reformatting a track and thus would not, in general, be used directly in an applications program.

All actual input/output is performed using the data area (positions 3-130) of the diskette buffer as a data transfer area. See the section on diskette buffer access for a description of routines to read and write this buffer.

In order to initialize an FCB for use with these routines to read or write a specific file, it is necessary to locate the file by accessing the file directory. The D&OPEN routine is used to perform this function and uses the file directory to initialize an FCB to point to the first sector of a file.

To create a new file, the D&CREA routine is used. D&CREA will establish a new file at the end of the file area in use. Initially, the whole remaining area of the disk is made available. Following writing of the file, a call to D&CLOS completes the directory entry including the actual length of the new file.

Both D&OPEN and D&CREA make use of a File Description Block (FDB), which has the format shown in Figure 3-2, and which describes the file to be opened or created:

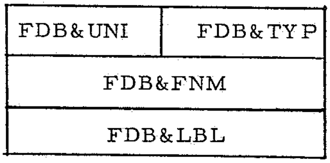


Figure 3-2. File Description Block (FCB)

FDB&UNI	The diskette unit number. This is currently restricted to 0 or 1, but a value of 2 is reserved for possible future use.
FDB&TYP	The file type: 'D' Data File 'O' Object File 'R' Relocatable File 'S' Source File
FDB&FNM	The 8 character file name, left justified, right blank filled. For open calls, this name may be abbreviated with a terminating X'FF'.
FDB&LBL	The 40 character file label. For a D&OPEN call, this is set from the directory entry on return. For a D&CREA call, it is set by the caller and then written to the directory.

These routines return with no action (error status set) if the diskette in questioı shows inoperable status, as will always occur following a diskette reload. If the user's application permits diskette reloads, two routines are provided to handle this condition. D&DKRS resets the status without testing it. D&CHEK resets the status and informs the caller how it was set on the call.

One further routine, D&LOCF provides the capability of finding the name, type and label of a file given its starting track and unit number.

LOGICAL LEVEL DISKETTE INPUT/OUTPUT

The logical level diskette input/output routines provide for sequential reading and writing of source files by logical records. They do not allow for in-place updating, or random access, or access to other types of files.

Input/output is controlled by means of fourteen byte block called a Source Control Block (SCB) which has the format shown in Figure 3-3.

SCB&TRK	SCB&SEC
SCB&SIF	SCB&UNI
SCB&LTR	SCB&DIR
SCB&CTR	SCB&LEN
SCB&BUF	
SCB&BFP	
SCB&REC	

This is a model File Control Block.

Figure 3-3. Source Control Block (SCB)

The field names SCB&XXX are symbols whose value is equal to the offset of the field (e.g., SCB&BUF = 8). These symbols are defined by the ASSEMBLE utility.

The usage of the fields is as follows:

SCB&TRK	Absolute track number of next sector to be accessed
SCB&SEC	Sector number of the next sector to be accessed
SCB&SIF	Sector interlace factor
SCB&UNI	Unit number
SCB&LTR	Last track of file
SCB&DIR	Directory flag byte
SCB&CTR	Bytes remaining in buffer
SCB&LEN	Logical record length
SCB&BUF	Pointer to 128 byte work buffer
SCB&BFP	Current location in work buffer
SCB&REC	Pointer to logical record area

SCB&BUF must be set before the open or create call and never changed under any circumstances.

The SCB&TRK, SCB&SEC, SCB&UNI, SCB&LTR, SCB&CTR and SCB&BFP fields are set at open time and maintained automatically. They should not be modified by the calling program.

The SCB&SIF field is set to the system default (currently 11) by the create routines. This value may be changed to any odd (non-zero) value following the create call.

SCB&LEN, SCB&REC must be set before each call to read or write a record, although they are not modified and can thus be left unchanged after setting them once before the first call. The permitted range for the record length is 1-255.

Files are opened for reading using the D&OPSR routine, following which, calls to D&RDSR obtain successive records. D&CRES creates a new source file to which records are written using D&WRSR. It is also possible to rewrite an existing file by opening it with D&OPSW followed by D&WRSR calls, subject to the restraint that the new file must occupy no more space than the original one (taking data compression into account). The close routine D&CLSS must be called after writing the last record to empty buffers, write an end of file and, in the case of a created file, complete the directory entry.

The create and open calls use a file description block (FDB) in a manner analogous to that previously described for the physical level routines. D&DKRS and D&CHEK may also be used on conjunction with the logical level routines to provide control over inoperable status and diskette reloads.

Note that source files are normally written with a standard interlace factor of 11, but this should not be assumed on reading a source file. Instead, the proper SIF should be read and set from the directory (e.g. by use of D&OPSR).

DISKETTE BUFFER ACCESS

A collection of routines is provided for accessing the data and control information in the diskette buffer. They handle details of delays and dummy reads required by the hardware.

D&MRTB, D&MRFB allow transfer of the data area of the diskette buffer (positions 3-130 or DBF&DTA-(DBF&DTA+127)) to and from a 128 byte area in main memory.

D&SBPR sets the buffer pointer to a specified location and reads the character at that location. Subsequent calls to D&RDBF provide the following characters in sequence.

D&SBPW sets the buffer pointer so that subsequent calls to D&WRBF write data to successive locations in the buffer.

All system routines which reference the diskette buffer only return after delaying long enough so that the buffer is not busy on exit. D&SBPR, D&SBPW, D&RDBF, D&WRBF are the only routines which leave the buffer pointer at a defined position. All other routines are considered to destroy the buffer pointer value.

OVERLAY SEGMENT MANAGEMENT

A program running under SPD/DOS may contain up to 250 overlay segments assembled using the SEG and ESEG operations. The D&LOAD routine loads a specified segment into memory. D&LINK loads a specified segment and transfers control to its entry point as specified on the ESEG line, or to its first assembled location if no entry point is provided on the ESEG line.

The assembled data in a segment may load into any addresses including the region X'0000'-X'00FF' normally forbidden to segment zero. Data must not load into the cursor location ($X) or on top of the load or link routine and routines used by it or on top of the load or link call itself.

D&LNKS and D&LODS are small versions of D&LINK and D&LOAD for use where the number of overlay segments is 32 or fewer. Segment dictionary entries for the first 32 segments are kept in random access memory (the disk buffer) and thus the first 32 segments can be accessed faster than the remaining ones when using the full size routines.

ASYNCHRONOUS DISK INPUT/OUTPUT

Normally all disk input/output subroutines complete the requested operation before returning control. This is adequate in many applications, but more sophisticated programs may require that disk operations be overlapped with other computation. A routine D&WAIT is supplied which enables a program to regain control during disk operations. It works in conjunction with both physical and logical procedures as well as segment load calls.

PARAMETER FIELD SCANNING

A set of routines is provided for scanning out parameter fields or other similar information. D&SCNI is the initialization routine which defines the location and length of the field to be scanned. D&SCNC scans one character from the field. D&SCNF and D&SCNL scan file name specifications (without and with a label given respectively) and build corresponding FDB's.

PRINTER OUTPUT

SPD/DOS supports several different types of printers. The D&PRNT routine is provided to allow generation of printed output without knowing what type of printer is attached. It may be used in application programs to generate printed output, although it should be noted that its size is much larger than would be required to drive a single known printer type.

DISPLAY HANDLING

Routines are provided for display handling to provide 10/20, 10/24, 10/25, and 20/20 compatibility. These routines should normally only be used by programs requiring this compatibility because otherwise they take up more space than is necessary.

The display compatibility is obtained by always formatting low core in 30 x 64 10/20 style with end refresh characters (the end refresh character being copied from DBF&DER). The call to D&DSPL copies this data to screen memory in the case of the 10/25 and 20/20. No point plot mode is available and the eighth bit should never be set. D&DSIN is provided to allow the screen to be cleared (refresh disabled). On the 20/20 and 10/25, it is always screen zero which is referenced.

KEYBOARD HANDLING

The D&RKBD routine is provided to allow reading the keyboard in a manner compatible with the 10/20, 10/24, 10/25 or 20/20. It should normally only be used if such compatibility is required since it is much larger than specific routines for one keyboard type. D&RKBR operates in conjunction with this routine to provide automatic repeat capability.

D&KALM sounds the alarm on the keyboard if available. D&KLIT sets the keyboard lights.

COMMAND SOURCE INPUT

The D&RLCS routine reads the next line of command input from the current command source (keyboard, paper tape, cards, command disk file). Since it is quite large, it should only be used if this generality is required.

END OF EXECUTION

The D&EXIT routine provides for normal return to the nucleus signalling either successful completion or returning an error code.

D&POWR provides a power restart routine which performs an abnormal disk boot (i.e., one which reinitializes the DBF). D&POWR also provides a breakpoint facility for use with the ZAP B and T commands.

D&EXCM provides an alternate method for returning to the nucleus signalling successful completion. It has the ability to provide a command for execution by the nucleus on return. This facility can be used for program to program linkage.

UTILITY ROUTINES

Utility routines are provided which are used by other D&XXXX routines for various purposes but which can also be used directly by an applications program. D&BLMV is used to move a contiguous block of data from one region of memory to another. D&ENTF provides for synchronous execution of foreground code.

SECTION IV

SYSTEM SUBROUTINES -- CALLING SEQUENCES

This section contains calling sequence details for all system subroutines. The subroutines are arranged in alphabetical order for easy reference. See Section III for general details by function.

D&BLMV -- Block Move

D&BLMV moves a contiguous series of bytes from one area in main memory to another.

Call:	JSR D&BLMV	call to move block
	DAC from	starting location of source
	DAC to	starting location of destination
Entry	(ACR)	number of bytes to move (1-255)
	($X)	irrelevant
Exit:	(ACR)	last byte moved
	($X)	points past last byte stored

NOTES: The move is one character at a time left to right and thus moving a field to a location one character higher propogates the first character throughout the field.

Routines Used: None

D&CHEK -- Check Diskette Status

D&CHEK tests and resets inoperable status on a specified diskette unit.

Call:	JSR D&CHEK	call to check status
Entry:	(ACR)	diskette unit number
	($X)	irrelevant
Exit:	(ACR)	status
	($X)	unchanged
Status:	0	inoperable status not set on call
	1	inoperable status set on call and has been reset
	2	inoperable status set on call and cannot be reset

NOTES: Status 1 indicates a probable reload. The inoperable status is raised as soon as the door is opened and is reset by this call. There is no way of telling when the reload is complete except to perform a read and see if it completes successfully. The application will dictate whether files must be re-opened or not.

Status 2 indicates a hardware fault has occurred (i.e., FAULT light on) or that the disk unit is powered down.

Routines Used: None

D&CLOS - Close File

D&CLOS is used to close a file which was created with a prior call to D&CREA. It is typically called following a series of D&WRIT calls which write the data to the new file. D&CLOS completes the directory entry which was partially made by D&CREA. D&CLOS may also be used with previously existing files opened with D&OPEN, but in this case it has no effect.

Call:	JSR D&CLOS	call to close file
	DAC fcb	pointer to FCB
Entry:	(ACR)	irrelevant
	($X)	irrelevant
	FCB FCB&TRK	irrelevant
	FCB&SEC	irrelevant
	FCB&SIF	SIF (written to directory entry if created file)
	FCB&UNI	unit for file
	FCB&LTR	last track (written to directory entry if created file)
	FCB&DIR	value set by open or create call (bit 7 set if open call, bit 5 = "OK").
Exit:	(ACR)	status
	($X)	destroyed
	FCB All Fields	unchanged

Status:	0	close successful
	1	search check writing directory
	2	write check writing directory
	3	unit inoperable
	4	write protected

NOTES: FCB&LTR is set to 63 by the D&CREA call. It should be reset to its proper value just before the D&CLOS call. This arrangement allows creating a file without knowing its size in advance. If the V option is set in DBF&OPT, then the directory write is performed with a verify check.

If the DIR&EPT field of the directory entry being completed indicates that there is a duplicate file to be erased, the erase occurs at this time.
If created file data is questionable due to write errors or inconsistencies detected by the application, the directory status displayed may be set to "?" if bit 5 is turned off in FCB&DIR prior to calling D&CLOS. This is the only bit in FCB&DIR which may be modified.

Routines Used: D&RDBF, D&READ, D&SBPR, D&SBPW, S&WRBF, D&WRIT

D&CLSS -- Close Source File

D&CLSS is used to close a source output file after writing all the records with calls to D&WRSR. It empties the last buffer and writes a logical end of file. In the case of a file which was created using D&CRES, it also completes the directory entry.

D&CLSS may be used with input files opened with D&OPSR and read with D&RDSR but has no effect in this case.

Call:	JSR D&CLSS	call to close source file
	DAC scb	pointer to SCB
Entry:	(ACR)	irrelevant
	($X)	irrelevant
	SCB SCB&TRK	track number of next sector to be written
	SCB&SEC	next sector to be written
	SCB&SIF	interlace factor
	SCB&UNI	unit number
	SCB&LTR	last track of file
	SCB&DIR	as set by D&OPSR, D&OPSW or D&CRES
	SCB&CTR	bytes left in work buffer
	SCB&LEN	max length of record to be written
	SCB&BUF	pointer to 128 byte work buffer
	SCB&BFP	pointer to next location in work buffer
	SCB&REC	pointer to record to be written

Exit:	(ACR)	status
	($X)	destroyed
	SCB All Fields	destroyed
Status:	0	close successful
	1	search check
	2	write check
	3	unit inoperable
	4	attempted write past end of file
	5	write protected

If the DIR&EPT field of the directory entry being completed indicates that there is a duplicate file to be erased, the erase occurs at this time.

If created file data is questionable due to write errors or inconsistencies detected by the application, the directory status displayed may be set to " " if bit 5 is turned off in FDB&DIR prior to calling D&CLOS. This is the only bit in FCB&DIR which may be modified.

Routines Used: D&WRSR, D&CLOS

D&CREA - Create New File

D&CREA is used to create a new file on an existing diskette. An FCB is initialized for subsequent writes (D&WRIT) to the new file. A new directory entry containing the file name and label is added to the end of the specified directory but not completed. The completion of this entry occurs when the file is closed (D&CLOS).

Call:	JSR D&CREA	call to create file
	DAC fdb	pointer to FDB
	DAC fcb	pointer to FCB
Entry:	(ACR)	irrelevant
	($X)	irrelevant
	FDB FDB&UNI	unit on which file is to be created
	FDB&TYP	file type
	FDB&FNM	file name
	FDB&LBL	file label
	FCB All fields	irrelevant
Exit:	(ACR)	status
	($X)	destroyed
	FDB All fields	unchanged
	FCB FCB&TRK*	first track of new file
	FCB&SEC*	zero
	FCB&SIF*	zero

FCB&UNI*	unit (from FDB&UNI)
FCB&LTR*	63 (end of diskette)
FCB&DIR*	(sector number of new directory entry) + (X'40' if second entry in sector) + X'20'

Status:	0	create successful
	1	search check reading or writing directory
	2	read or write check on directory
	3	unit inoperable
	4	duplicate file name
	5	unit write protected
	6	file area full. No room for new file

NOTES: If the E option is set in DBF&OPT, then status 4 is not reported if a duplicate file name is encountered. Instead, an entry is made in DIR&EPT of the new directory entry which causes the corresponding D&CLOS call to erase the previous file.

If the file is never closed then the directory does not contain the new entry.

Only one file may be created and written on one unit at a time. Thus, a D&CLOS call must intervene between two calls to D&CREA for the same unit.

The FCB&SIF field should be set immediately following the D&CREA call (before the first D&WRIT call) if a value other than zero is required.

If the V option is set in DBF&OPT, then the directory write operations are performed with a verify check.

Fields marked with an asterisk (*) are set only if the create call is successful. Otherwise they are unchanged.

Following the call to D&CREA, the FDB is no longer required and may be overwritten.

Routines used: D&RDBF, D&READ, D&SBPR, D&SBPW, D&WRBF, D&WRIT

D&CRES -- Create Source File

D&CRES is used to create a new source file to which records are to be written sequentially with subsequent calls to D&WRSR. The entire remaining unused data area on the specified disk is available for output.

Call:	JSR D&CRES	call to create source file
	DAC fdb	pointer to FDB
	DAC scb	pointer to SCB
Entry:	(ACR)	irrelevant
	($X)	irrelevant
	FDB FDB&UNI	unit on which file is to be created
	FDB&TYP	set to 'S' (X'53')
	FDB&FNM	file name
	FDB&LBL	file label
	SCB SCB&BUF	pointer to 128 byte work buffer
	Other fields	irrelevant
Exit:	(ACR)	status
	($X)	destroyed
	FDB All Fields	unchanged

SCB	SCB&TRK*	first track of file
	SCB&SEC*	zero
	SCB&SIF*	standard system default value (11)
	SCB&UNI*	unit number
	SCB&LTR*	last track of file (=63)
	SCB&DIR*	directory byte X'20' + directory entry sector number + X'40' if in 2nd half of sector
	SCB&CTR*	128 (X'80')
	SCB&BFP*	pointer to start of work buffer
	Other fields	unchanged

Status:	0	create successful
	1	search check reading or writing directory
	2	read or write check on directory
	3	unit inoperable
	4	duplicate file name
	5	unit write protected
	6	file area full. No room for new file.

NOTES: If the E option is set in DBF&OPT, then status 4 is not reported if a duplicate file name is encountered. Instead, an entry is made in DIR&EPT of the new directory entry which causes the corresponding D&CLSS call to erase the previous file.

If the file is never closed then the directory does not contain the new entry.

Only one file may be created and written on one unit at a time. Thus, a D&CLSS call must intervene between a call to D&CRES and another create call (D&CREA or D&CRES) for the same unit.

The FCB&SIF field should be set immediately following the D&CRES call (before the first D&WRSR call) if a value other than the system default is required.

If the V option is set in DBF&OPT, then the directory write operations are performed with a verify check.

Following the call to D&CRES, the FDB is no longer required and may be overwritten.

Routines Used: D&CREA

D&DKRS -- Diskette Reset

D&DKRS issues a reset to a specified diskette unit and clears inoperable (reloaded) status if it was set.

Call:	JSR D&DKRS	call to reset diskette
Entry:	(ACR)	diskette unit number
	($X)	irrelevent
Exit:	(ACR)	unchanged
	($X)	unchanged

Routines Used: None

D&DSIN -- Display Initialize

D&DSIN is used to blank the display and disable the cursor.

Call:	JSR D&DSIN	call to blank display
Entry:	(ACR)	irrelevant
	($X)	irrelevant
Exit:	(ACR)	destroyed
	($X)	unchanged

NOTES: On the 10/25 and 20/20, all screens are blanked by this call. Data in the TPU memory is unaffected. On the 10/25 and 20/20, D&DSIN exits with 30 x 64 mode set and screen zero selected.

Routines Used: D&SBPR

D&DSPL -- Display

D&DSPL is used to display memory data on the display screen.

Call:	JSR D&DSPL	call to display
Entry:	(ACR)	irrelevant
	($X)	cursor, see notes
Exit:	(ACR)	destroyed
	($X)	unchanged

NOTES: Prior to the call, low core must be formatted with the display data in 10/20 style with the last line terminated by two copies of the end refresh character (DBF&DER).

The entry value of the cursor is in 10/20 style format:

Bits 15-12	zero (on 20/20 or 10/24)
Bits 11-6	line
Bits 5-0	character

The special cursor value of X'FFF' causes the cursor to be disabled on exit.

In the case of the 10/25 and 20/20, the proper format and screen must have been preselected. This is normally achieved by a prior call to D&DSIN in which case the screen used is screen zero.

Routines Used: None

D&ENTF -- Enter Foreground

D&ENTF allows for execution of foreground (interrupts disabled) code. It establishes a temporary background hang loop for the auto-exec so that I/O instructions can be issued in a straight-forward fashion. Other active foreground tasks are held up but not otherwise affected.

```
Call:       JSR D&ENTF           enter foreground
              .
              .
            (foreground code)
              .
              .
            ENB                  re-enable to continue background

Entry:      (ACR)                irrelevant
            ($X)                 irrelevant

Exit:       (ACR)                unchanged      on entry to foreground code
            ($X)                 unchanged
```

NOTES: The ENB should be issued as quickly as possible to avoid holding up other foreground tasks. If the foreground code NAK's to background to wait for its device to accept or provide data, other foreground tasks are released to process interrupts; the device's ready interrupt returns the processor to this foreground code.

Routines Used: None

D&EXCM -- Exit with Nucleus Command

D&EXCM returns control to the nucleus signalling successful completion and supplies a nucleus command to be executed immediately on return. Command input resumes from the current command file following completion of execution of the supplied command. Any valid nucleus command may be submitted. Thus D&EXCM may be used to load another program, switch command files, issue operator messages etc.

Call:	JSR D&EXCM	Exit issuing command
	DAC line	Pointer to command
Entry:	(ACR)	irrelevant
	($X)	irrelevant
Exit:		exit is directly to the nucleus via a normal disk boot.

NOTES: The command line may be up to 64 ASCII characters. Lines shorter than 64 characters must be terminated by X'0D' (carriage return).

D&EXCM works by storing the command in the DBF (at DBF&EXCM, overlapping the overlay segment dictionary which is no longer required) and setting the special value X'FE' in DBF&RTN.

Routines Used: D&SBPW, D&WRBF, D&EXIT

D&EXIT -- Exit to Nucleus

D&EXIT is used to execute a return to the nucleus.

Call:	JMP D&EXIT	jump to exit to nucleus
Entry:	(ACR)	return code
	($X)	irrelevant
Exit:		Exit is directly to the nucleus via a normal disk boot

NOTES: The return code is zero for a normal completion or 1-99 for an error condition. In the latter case, NUCLEUS will post error code XYDD when XY is the first two characters of the program name (from DBF&PNM) and DD is the (decimal) error code.

Routines Used: D&SBPW, D&WRBF

D&KALM -- Sound Keyboard Alarm

D&KALM sounds the alarm on the operator keyboard (if available).

Call:	JSR D&KALM	
Entry:	(ACR)	irrelevant
	($X)	irrelevant
Exit:	(ACR)	destroyed
	($X)	unchanged

Routines Used: D&SBPR

D&KLIT -- Set Keyboard Lights

D&KLIT sets the lights on the operator keyboard.

Call:	JSR D&KLIT	
Entry:	(ACR)	value to be set in lights
	($X)	irrelevant
Exit:	(ACR)	destroyed
	($X)	unchanged
Routines Used:	D&SBPR	

D&LINK -- Link to Segment

D&LINK loads a specified overlay segment and links to its entry point.

Call:	JSR D&LINK	call to link to segment
Entry:	(ACR)	segment number (1-250)
	($X)	irrelevant
Exit:	(ACR)	error status
	($X)	destroyed
Status:	1	search check reading object file
	2	read check reading object file
	3	object file unit inoperable
	4	erroneous end of file indication (object file format error)

NOTES: The segment number is the value of the symbol in the label field of the corresponding SEG statement.

D&LINK returns only if the link is unsuccessful.

Routines Used: D&LOAD

D&LNKS -- Link to Segment (small)

D&LNKS loads a specified overlay segment and links to its entry point.

Call:	JSR D&LNKS	call to link to segment
Entry:	(ACR)	segment number (1-32)
	($X)	irrelevant
Exit:	(ACR)	error status
	($X)	destroyed
Status:	1	search check reading object file
	2	read check reading object file
	3	object file unit inoperable
	4	erroneous end of file indication (object file format error)

NOTES: The segment number is the value of the symbol in the label field of the corresponding SEG statement.

D&LNKS returns only if the link is unsuccessful.

Routines Used: D&LODS

D&LOAD --Load Segment

D&LOAD loads a specified overlay segment.

Call:	JSR D&LOAD	call to load segment
	(ACR)	segment number (1-250)
	($X)	irrelevant
Exit:	(ACR)	status
	($X)	assembled entry point if load successful
Status:	0	load successful
	1	search check reading object file
	2	read check reading object file
	3	object file unit inoperable
	4	erroneous end of file indication (object file format error)

NOTES: The segment number is the value of the symbol in the label field of the corresponding SEG statement.

Routines Used: D&READ, D&SBPR, D&RDBF

D&LOCF -- Locate File

D&LOCF is used to get the name and label of a file given its starting track and unit number.

Call:	JSR	D&LOCF	call to locate file
	DAC	fcb	pointer to FDB
	DAC	fcb	pointer to FCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FDB all fields		irrelevant
	FCB	FCB&TRK	starting track for file
		FCB&UNI	unit number
		Other Fields	irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FDB	FDB&UNI*	unit from FCB
		FDB&TYP*	file type
		FDB&FNM*	file name
		FDB&LBL*	file label
	FCB All Fields		unchanged
Status:	0		locate successful
	1		search check reading directory
	2		read check reading directory
	3		unit inoperable reading directory
	4		file not found

NOTES: The fields marked with an asterisk are only set if the locate is successful.

Routines Used: D&SBPR, D&RDBF, D&READ

D&LODS -- Load Segment (small)

D&LODS loads a specified overlay segment.

Call:	JSR D&LODS	call to load segment
Entry:	(ACR)	segment number (1-32)
	($X)	irrelevant
Exit:	(ACR)	status
	($X)	assembled entry point if load successful
Status:	0	load successful
	1	search check reading object file
	2	read check reading object file
	3	object file unit inoperable
	4	erroneous end of file indication (object file format error)

NOTES: The segment number is the value of the symbol in the label field of the corresponding SEG statement.

Routines Used: D&READ, D&SBPR, D&RDBF

D&MRFB -- More Record from Buffer

D&MRFB moves the 128 bytes of the data area of the DBF (positions 3 to 130 of DBF&DTA to DBF&DTA+127) to a specified location in main memory.

Call:	JSR D&MRFB	call to move record from buffer
Entry:	(ACR)	irrelevant
	($X)	pointer to 128 byte area in main memory
Exit:	(ACR)	destroyed
	($X)	points past 128 byte area in memory

NOTES: All necessary delays are issued so that the buffer is not busy on exit.

Routines Used: D&SBPR

D&MRTB -- Move Record to Buffer

D&MRTB moves 128 bytes from a specified location in main memory to the data area of the DBF (positions 3 to 130 or DBF&DTA to DBF&DTA+127).

Call:	JSR D&MRTB	call to move record to buffer
Entry:	(ACR)	irrelevant
	($X)	pointer to 128 byte area in main memory
Exit:	(ACR)	destroyed
	($X)	points past 128 byte area in memory

NOTES: All necessary delays are issued so that the buffer is not busy on exit.

Routines Used: D&SBPW

D&OPEN -- Open Existing File

D&OPEN is used to search a directory for an existing file and initialize a file control block for subsequent reads (D&READ) or writes (D&WRIN, D&WRIT).

Call:	JSR	D&OPEN	call to open file
	DAC	fdb	pointer to FDB
	DAC	fcb	pointer to FCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FDB	FDB&UNI	unit on which file is to be opened
		FDB&TYP	file type ('S', 'O', 'R', or 'D')
		FDB&FNM	file name or an abbreviation terminated by X'FF'
		FDB&LBL	irrelevant
	FCB All Fields		irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FDB	FDB&UNI	unchanged
		FDB&TYP	unchanged
		FDB&FNM*	file name
		FDB&LBL*	file label

FCB	FCB&TRK*	first track of file
	FCB&SEC*	zero
	FCB&SIF*	SIF from directory
	FCB&UNI*	unit (from FDB&UNI)
	FCB&LTR*	last track of file
	FCB&DIR*	X'80' (non-created file)

Status:	0	open successful
	1	search check reading directory
	2	read check reading directory
	3	unit inoperable reading directory
	4	file not found

NOTES: If an abbreviation is sued for the file name, then the first file whose name matches up to but not including, the X'FF' character is opened and the exit value of FDB&FNM gives the actual name of the file which was opened.

Fields marked with an asterisk are set only if the open is successful, otherwise they are unchanged.

Following the call to D&OPEN, the FDB is no longer required and may be overwritten.

Routines Used: D&RDBF, D&READ, D&SBPR

D&OPSR -- Open Source File for Reading

D&OPSR opens an existing source file for sequential reading using subsequent calls to D&RDSR.

Call:	JSR	D&OPSR	call to open file
	DAC	fdb	pointer to FDB
	DAC	scb	pointer to SCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FDB	FDB&UNI	unit on which file is to be opened
		FDB&TYP	set to 'S' (X'53')
		FDB&FNM	file name or an abbreviation terminated by X'FF'
		FDB&LBL	irrelevant
	SCB	SCB&BUF	pointer to 128 byte work buffer
		Other Fields	irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FDB	FDB&UNI	unchanged
		FDB&TYP	unchanged
		FDB&FNM*	file name
		FDB&LBL*	file label

SCB	SCB&TRK	first track of file
	SCB&SEC*	zero
	SCB&SIF*	SIF from directory
	SCB&UNI*	unit number
	SCB&LTR*	last track of file
	SCB&DIR*	X'80' (non-created file, via D&OPSR)
	SCB&CTR*	zero
	Other fields	unchanged

Status:	0	open successful
	1	search check reading directory
	2	read check reading directory
	3	unit inoperable reading directory
	4	file not found

NOTES: If an abbreviation is used for the file whose name matches up to, but not including, the X'FF' character is opened and the exit value of FDB&FNM gives the actual name of the file which was opened.

Fields marked with an asterisk are set only if the open is successful, otherwise they are unchanged.

Following the call to D&OPSR, the FDB is no longer required and can be overwritten.

Routines Used: D&OPEN

D&OPSW -- Open Source File for Writing

D&OPSW is used to open an existing file so that new information can be sequentially written using subsequent calls to D&WRSR. The old information is lost and the new data file must not exceed the original file allocation.

Call:	JSR	D&OPSW	call to open file
	DAC	fdb	pointer to FDB
	DAC	scb	pointer to SCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FDB	FDB&UNI	unit on which file is to be opened
		FDB&TYP	set to 'S' (X'53')
		FDB&FNM	file name or an abbreviation terminated by X'FF'
		FDB&LBL	irrelevant
	SCB	SCB&BUF	pointer to 128 byte work buffer
		Other fields	irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FDB	FDB&UNI	unchanged
		FDB&TYP	unchanged
		FDB&FNM*	file name
		FDB&LBL*	file label

SCB	SCB&TRK*	first track of file
	SCB&SEC*	zero
	SCB&SIF*	SIF from directory
	SCB&UNI*	unit number
	SCB&LTR*	last track of file
	SCB&DIR*	X'81' (non-created file, via D&OPSW)
	SCB&CTR*	128 (X'80')
	SCB&BFP*	pointer to start of work buffer
	Other fields	unchanged

Status:	0	open successful
	1	search check reading directory
	2	read check reading directory
	3	unit inoperable reading directory
	4	file not found

NOTES: If an abbreviation is used for the file whose name matches up to, but not including, the X'FF' character is opened and the exit value of FDB&FNM gives the actual name of the file which was opened.

Fields marked with an asterisk are set only if the open is successful, otherwise they are unchanged.

Following the call to D&OPSW, the FDB is no longer required and can be overwritten.

Routines Used: D&OPEN

D&POWR -- Power Restart Routine

The inclusion of the D&POWR routine in a program provides a power restart circuit which causes an automatic reboot from disk if the processing unit is powered down and up again. The boot is an abnormal type boot which saves the core image and reinitializes the diskette buffer. In general, as described previously, such a boot procedure is the only appropriate action on a power up since the disk buffer information is lost in this event.

D&POWR can also be called as a pseudo-subroutine as shown by the calling sequence given below. In this case, D&POWR saves the accumulator and index (cursor) register contents and then boots in the system. Since the boot is abnormal, the core image is saved and the ZAP utility can be used to analyze the "breakpoint" which has occurred (see Operator's Reference Manual).

Call:	JSR*	$X-2	(or X'7FFC')
Entry:	(ACR)		value to be saved
	($X)		value to be saved

Exit: Exits is to nucleus via abnormal boot.

NOTES: The "call" to D&POWR is automatic if the TPU is powered down and powered up again.

Programs including D&POWR must explicitly set the power restart linkage by a sequence of the form:

```
          ORG          $X-2
          DAC          D&POWR
```

The breakpoint call to D&POWR can be set with the B command of the ZAP utility (see Operator's Reference Manual).

Other Routines Used: D&SBPW, D&WRBF

D&PRNT -- Print Line

Call:	JSR	D&PRNT	call to print line
	DAC	line	pointer to line
Entry:	(ACR)		irrelevant
	($X)		irrelevant
Exit:	(ACR)		destroyed
	($X)		destroyed

NOTES: If the printer is not ready, D&PRNT just waits until it is readied. If there is no printer configured, D&PRNT returns immediately with no action. The data line presented to D&PRNT has the following format:

<space>[<shift>]<text> <cr>

<space>	is	X'0C'	for form feed prior to print
		X'0A'	for extra line feed prior to print
		X'01'	for normal single spacing
<shift>	is	X'0E'	to get a heading line (large characters or double spacing) omitted for a normal text line
<text >	is	1-132	ASCII characters in the range X'20' - X'5F'. Printers with less than 132 characters per line will fold if necessary. For correct operation with the Termiprinter, the line length should not be less than 35.

<cr>	is	X'0D'	to end the line

Routines Used: D&SBPR, D&ENTF

D&RDBF -- Read Buffer

D&RDBF reads the byte of data at the current buffer position and advances the buffer pointer.

Call:	JSR	D&RDBF	call to read buffer
Entry:	(ACR)		irrelevant
	($X)		irrelevant
Exit:	(ACR)		byte read
	($X)		unchanged

NOTES: Either D&SBPR or D&RDBF must have been called previously with no other intervening calls to system routines.

All necessary delays are issued so that the buffer is not busy on exit.

The pointer wraps in a circular manner from the end (255) to the start (0) of the buffer.

Routines Used: None

D&RDSR -- Read Source Record

D&RDSR reads a logical record from a source file previously opened with a call to D&OPSR.

Call:	JSR	D&RDSR	call to read source record
	DAC	scb	pointer to SCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	SCB	SCB&TRK	track number of next sector
		SCB&SEC	sector number of next sector
		SCB&SIF	interlace factor
		SCB&UNI	unit number
		SCB&LTR	last track of file
		SCB&DIR	X'80'
		SCB&CTR	bytes left in current sector
		SCB&LEN	maximum record length (1-255)
		SCB&BUF	pointer to 128 byte work buffer
		SCB&BFP	pointer to next data byte in work buffer
		SCB&REC	pointer to area to store record read
Exit:	(ACR)		status
	($X)		pointer just past actual end of record
	SCB	SCB&TRK	Updated as required, ready to read
		SCB&SEC	

SCB&CTR SCB&BFP	next record
Other fields	unchanged

Status:	0	successful completion
	1	search error
	2	read error
	3	unit inoperable
	4	read past physical end of file (source file format error)
	5	logical end of file encountered
	6	record truncated (extra characters ignored)

NOTES: If a status of 1 or 2 is reported, the SCB is left ready to read a subsequent record. One or more records will have been lost.

A status of 5 will be signalled repeatedly once the end of file is reached.

The record is stored in the area pointed to be the SCB&REC field right padded with blanks to the length given by SCB&LEN. The X'0D' which terminates the record on the file in not transmitted. However, $X is left pointing to where the X'0D' would have been placed if it had been stored allowing the caller to truncate the fill blanks if desired.

The only fields of the SCB which are set by the caller are SCB&LEN and SCB&REC, all other fields are maintained automatically.

Routines Used: D&READ, D&MRFB, D&BLMV

D&READ -- Read Sector

D&READ reads a single sector of data into the data area of DBF. The sector is specified by the fields of a supplied FCB. Typically, though not necessarily, this FCB has been initialized by a prior call to D&OPEN and the sector data read is from the file opened by the D&OPEN call. If the SIF value in the FCB is non-zero, then the TRK and SEC fields are updated following the call to point to the next logical sector.

Call:	JSR	D&READ	call to read sector
	DAC	fcb	pointer to FCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FCB	FCB&TRK	track to be read
		FCB&SEC	sector to be read
		FCB&SIF	SIF to be used to update TRK, SEC
		FCB&UNI	unit to be read
		FCB&LTR	last track, used to check for end of file
		FCB&DIR	irrelevant

Exit:	(ACR)		status
	($X)		destroyed
	FCB	FCB&TRK	updated by FCB&SIF value
		FCB&SEC	updated by FCB&SIF value
		Other fields	unchanged

Status:	0	successful completion of read
	1	search check
	2	read check
	3	unit inoperable
	4	attempted read past end of file

NOTES: The read is attempted five times with recalibration before signalling a read or search check.

FCB&TRK, FCB&SEC are updated by the SIF even if an error occurs.
The sync and address bytes in DBF are set by D&READ and need not be set by the caller.

The data read occupies positions 3-130 (DBF&DTA to DBF&DTA+127) of the DBF and may be acquired using D&SBPR, D&RDBF or D&MRFB.

D&READ may be used to read system tracks (0-2) by using a dummy FCB with appropriate values.

For sequential reading of a file, the SIF is non-zero and the TRK and SEC fields are updated automatically. For random access, SIF is zero and the caller sets appropriate TRK and SEC values in the FCB prior to each D&READ call.

The FCB&TRK value is an absolute track number. It is not relative to the beginning of a file.

Routines Used: D&WRIT (No write is performed; D&READ merges into D&WRIT and shares command circuitry).

D&RKBD -- Read Keyboard

D&RKBD reads a character from the keyboard.

Call:	JSR D&RKBD	call to read keyboard
Entry:	(ACR)	irrelevant
	($X)	irrelevant
Exit:	(ACR)	code for key read
	($X)	unchanged

NOTES: The code returned is the natural keyboard encoding except that, on the 10/25 and 20/20 the space bar returns X'20' instead of X'01'.

Parity errors are intercepted and rereads issued as required.

On the 10/25 and 20/20, keyboard zero is the keyboard which is used.

Routines Used: D&SBPR, D&ENTF

D&RKBR -- Read Keyboard Repeat Enable

D&RKBR enables keyboard repeat for the last key read in a manner compatible with the CIO repeat on the 10/20. It works on all models.

Call:	JSR	D&RKBR	call to repeat keyed character
Entry:	(ACR)		irrelevant
	($X)		irrelevant
Exit:	(ACR)		destroyed
	($X)		unchanged

NOTES: D&RKBR is called following a call to D&RKBD to enable repeat for the character just read. Subsequent calls to D&RKBD will (periodically) return the same character if the key is still held down.

D&RKBR modifies locations in D&RKBD which must therefore be loaded at the time of the D&RKBR call and stay loaded as long as the repeat is active.

Routines Used: D&SBPR, D&RKBD

D&RLCS -- Read Line from Command Source

D&RLCS reads one record from the current command source.

Call:	JSR	D&RLCS	call to read line from command source
	DAC	subr	pointer to keyboard subroutine
	DAC	bufr	pointer to 80 char buffer
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	Subr (Keyboard subroutine)		see below
	Bufr		all 80 chars irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	bufr		contains 80 character command line right filled with blanks
Status:	0		command read successfully
	1		search error reading command file
	2		read error reading command file
	3		unit inoperable reading command file
	4		missing end of file on command file
	5		normal end of file (paper tape or command file)

NOTES: The status of 5 corresponds to an EOT or ETX punch on paper tape or logical end of file on a command file.

Hopper empty on the card reader causes a wait for card loading.

.E is not recognized specially. Normally, the caller should test for and recognize such a record as end of file.

On exit, the display is enabled and the cursor disabled.

The buffer is preblanked on initial entry to D&RLCS.

For proper keyboard mode operation, the buffer should be in the active display area, with end refresh codes already setup (see D&DSPL).

The keyboard mode routine (subr) is called immediately following each byte in keyboard mode. It tests for data in the buffer corresponding to commands for which the end of line is not to be required and operates by simply retaining control in these cases. For example, if pressing the space bar is to have a special effect, as in the nucleus, the following could be used:

```
subr    DAC     **             entry point
        LD      CMLINE         load list character
        CJEQ    ' ', special   retain control if blank
        JMP*    subr           else let D&RLCS continue
```

The specified subroutine should return control when the disk operation is complete (disk not busy). On return, the entry values of the accumulator and index register (cursor) must be in effect. During the "disk busy" period, the use of system subroutines will either cause data transfer errors or re-entrancy problems, so only non-SPD/DOS activity may take place.

A typical example of a wait subroutine is:

```
SUBR    DAC     **
        ST      SUBRA+1
        STX     SUBRX

LOOP    (test for other non-DOS background activity)
        JTACK   D&DB, D&D, LOOP
        LDX     SUBRX                   Not Busy, restore $X

SUBRA   LDI     **                      Restore ACR
        JMP*    SUBR                    Disk completion return

SUBRX   WORD    **                      Temporary
```

Note that even a nucleus exit is forbidden during a disk busy period. That is, jumps to D&EXIT and even break points (through D&POWR) could cause a bad sector to be written to disk. A valid nucleus exit could be coded:

```
ABEND   LDI     error-code
        DSB                             make sure
        JTACK   D&DB, D&D, $            await non-busy
        JMP     D&EXIT                  Back to nucleus
```

This subroutine may destroy the (ACR) but normally leaves ($X) unchanged if a return is made ($X points to the next buffer location). A dummy routine (DAC **, JMP* subr) must be supplied even if this feature is not required.

Routines Used: D&DSPL, D&ENTF, D&READ, D&RKBD, D&RKBR, D&SBPR, D&SBPW, D&WRBF

D&SBPR -- Set Buffer Pointer for Read

D&SBPR sets the buffer pointer to a specified location and reads the byte at this location. The buffer pointer is left positioned so that subsequent calls to D&RDBF return the following data bytes in sequence.

Call:	JSR D&SBPR	call to set buffer pointer for read
Entry:	(ACR)	buffer location (0-255)
	($X)	irrelevant
Exit:	(ACR)	byte read
	($X)	unchanged

NOTES: All necessary delays are issued so that the buffer is not busy on exit.

If D&SBPR is called with the ACR set to one less than a given location and the returned value ignored, it gives the effect of setting the pointer without reading.

The sequence LDI DBF&XXX, JSR D&SBPR is the usual method of accessing system control information in the diskette buffer.

Routines Used: D&SBPW, D&RDBF

D&SBPW -- Set Buffer Pointer for Write

D&SBPW sets the buffer pointer so that subsequent calls to D&WRBF can be used to write data to the DBF.

Call:	JSR D&SBPW	call to set buffer pointer for write
Entry:	(ACR)	buffer location (0-255) for first write
	($X)	irrelevant
Exit:	(ACR)	unchanged
	($X)	unchanged

NOTES: All necessary delays are issued so that the buffer is not busy on exit.

Routines Used: None

D&SCNC -- Scan Character

D&SCNC obtains the next character from the field being scanned and bumps the scan pointer.

Call:	JSR	D&SCNC	call to scan character
Entry:	(ACR)		irrelevant
	($X)		irrelevant
Exit:	(ACR)		character scanned
	($X)		unchanged

NOTES: If the pointer is past the end of the field, a blank is returned. Calls to D&SCNC must be preceded by an earlier call to D&SCNI. The location D&SCNC1 contains the address of the next byte to be scanned. D&SCNC3 contains a count of characters left in the field.

Routines Used: None

D&SCNF -- Scan File Name

D&SCNF is used to scan out a file name in standard SPD/DOS form (unit name, currently selected unit is default if no unit specified). The file name starts at the current scan location and must be terminated by comma or blank. A File Description Block (FDB) is built as the result.

Call:	JSR	D&SCNF	call to scan file name
	DAC	fdb	pointer to FDB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FDB		all fields irrelevant
Exit:	(ACR)		status
	($X)		points to FDB&LBL field in FDB
	FDB	FDB&UNI	unit
		FDB&TYP	unchanged
		FDB&FNM	file name
		FDB&LBL	unchanged
Status:	0		scan successful
	1		syntax error in file name

NOTE: The call to D&SCNF must be preceded by an earlier call to D&SCNI.

The scan pointer is left set (on successful completion) so that the next call to D&SCNC loads the terminating characters (blank or comma).

Routines Used: D&SBPR, D&SCNC

D&SCNI -- Scan Initialize

D&SCNI is used to define the length and location of the field to be scanned.

Call:	JSR	D&SCNI	call to initialize scan
Entry:	(ACR)		field length in bytes
	($X)		address of first character of field
Exit:	(ACR)		unchanged
	($X)		unchanged

NOTES: Following the call to D&SCNI, the D&SCNC routine must remain loaded until the field is completely scanned out.

Routines Used: D&SCNC

D&SCNL -- Scan File Name and Label

D&SCNF is used to scan out a file name followed by a comma followed by a forty character label. A file Description Block (FDB) is built as a result.

Call:	JSR	D&SCNL	call to scan file name
	DAC	fdb	pointer to FDB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FDB		all fields irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FDB	FDB&UNI	unit
		FDB&TYP	unchanged
		FDB&FNM	file name
		FDB&LBL	file label
Status:	0		scan completed successfully
	1		syntax error in file name/label

NOTES: The call to D&SCNL must be preceded by an earlier call to D&SCNI. The scan pointer position is left undefined.

Routines Used: D&SCNC, D&SCNF

D&VRFY -- Verify Sector

D&VRFY verifies a single sector of data from the data area of DBF which is preloaded by the caller. The sector is specified by the fields of a supplied FCB. Typically, though not necessarily, this FCB has been initialized by a prior call to D&OPEN or D&CREA. If the SIF value in the FCB is non-zero, then the TRK and SEC fields are updated following a call to point to the next logical sector.

Call:	JSR	D&WRIT	call to write sector
	DAC	fcb	pointer to FCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FCB	FCB&TRK	track to be verified
		FCB&SEC	sector to be verified
		FCB&SIF	SIF to be used to update TRK, SEC
		FCB&UNI	unit number
		FCB&LTR	last track, used to check for end of file
		FCB&DIR	irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FCB	FCB&TRK	updated by FCB&SIF value
		FCB&SEC	updated by FCB&SIF value
		Other fields	unchanged

Status:	0	successful completion of write
	1	search check
	2	verify check
	3	unit inoperable
	4	attempted access past end of file

NOTES: The verify is attempted five times with recalibration before signalling a verify or search check.

The sync and address bytes in DBF are set by D&VRFY and need not be set by the caller.

DBF&TRK, DBF&SEC are updated by SIF even if an error or verify check occurs.

The data for the verify occupies positions 3-130 (DBF&DTA to DBF&DTA+127) of the DBF and may be set using D&SBPW, D&WRBF or D&MRTB.

For routine verification of all writes, the V option may be set in DBF&OPT in which case explicit D&VRFY calls are not required.

The FCB&TRK value is an absolute track number. It is not relative to the beginning of a file.

Routines Used: D&WRIT (No write is performed; D&VRFY merges into D&WRIT and shares command circuitry

D&WAIT -- Set Disk Wait Routine

D&WAIT specifies the location of a subroutine to be used to test completion of disk operations initiated by other system calls.

Call:	JSR D&WAIT	call to set wait routine
	DAC subr	entry point of subroutine
Entry:	(ACR)	irrelevant
	($X)	irrelevant
Exit:	(ACR)	unchanged
	($X)	destroyed

NOTES: D&WAIT modifies a location in D&WRIT which must stay loaded after the call.

The specified subroutine should return control when the disk operation is complete (disk not busy). On return, the entry values of the accumulator and index register (cursor) must be in effect. During the "disk busy" period, the use of system subroutines will either cause data transfer errors or re-entrancy problems, so only non-SPD/DOS activity may take place.

A typical example of a wait subroutine is:

```
SUBR    DAC     **
        ST      SUBRA + 1
        STX     SUBR X
LOOP    (test for other non-DOS background activity)
        JTACK   D&DB, D&D, LOOP  loop while busy
        LDX     SUBR X           not busy, restore $X
SUBRA   LDI     **               resume ACR
        JMP*    SUBR             disk completion return
SUBRX   WORD    **               temporary
```

Note that even a nucleus exit is forbidden during a disk-busy period. That is, jumps to D&EXIT and even breakpoints (through D&POWR) could cause a bad sector to be written to disk. A valid nucleus exit could be coded:

```
ABEND   LDI     error-code
        DSB     make sure
        JTACK   D&DB, D&D, $ await non-busy
        JMP     D&EXIT Back to nucleus
```

Routines Used: D&WRIT no write is performed, D&WAIT modifies D&WRIT and the read/write/verify routines exit to the supplied routines.

D&WRBF -- Write Buffer

D&WRBF writes a byte to the diskette buffer and advances the buffer pointer.

Call:	JSR	D&WRBF	call to write buffer
Entry:	(ACR)		byte to be written
	($X)		irrelevant
Exit:	(ACR)		unchanged
	($X)		unchanged

NOTES: Either D&SBPW or D&WRBF must have been called previously with no other intervening calls to system routines.

All necessary delays are issued so that the buffer is not busy on exit.

Routines Used: None

D&WRIN -- Write Initial Sector

D&WRIN writes a single sector of data from the data area of DBF which is preloaded by the caller using a write initial instead of an ordinary write. The sector is specified by the fields of a supplied FCB. Typically, though not necessarily, this FCB has been initialized by a prior call to D&OPEN or D&CREA. If the SIF value in the FCB is non-zero, then the TRK and SEC fields are updated following a call to point to the next logical sector.

Call:	JSR	D&WRIN	call to write sector
	DAC	fcb	pointer to FCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FCB	FCB&TRK	track to be written
		FCB&SEC	sector to be written
		FCB&SIF	SIF to be used to update TRK, SEC
		FCB&UNI	unit to be written
		FCB&LTR	last track, used to check for end of file
		FCB&DIR	irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FCB	FCB&TRK	updated by FCB&SIF value
		FCB&SEC	updated by FCB&SIF value
		Other fields	unchanged

Status:	0	successful completion of write
	1	search check
	2	write check
	3	unit inoperable
	4	attempted write past end of file
	5	write protected

NOTES: The write is attempted five times with recalibration before signalling a write or search check.

The sync and address bytes in DBF are set by D&WRIT and need not be set by the caller.

If the V option is set in DBF&OPT, the write initial is followed by a verify initial to check that the data was written correctly. This takes an entire extra revolution of the disk.
DBF&TRK and DBF&SEC are updated by SIF even if an error occurs.
The data to be written occupies positions 3-130 (DBF&DTA to DBF&DTA+127) of the DBF and may be set using D&SBPW, D&WRBF or D&MRTB.

D&WRIN is usually only used for reformatting a track. Following SPD/DOS conventions, the sector number should normally be zero.

The FCB&TRK value is an absolute track number. It is not relative to the beginning of a file.

Routines Used: D&WRIT (A normal write does not occur; D&WRIN merges into D&WRIT and shares command circuitry).

D&WRIT -- Write Sector

D&WRIT writes a single sector of data from the data area of DBF which is preloaded by the caller. The sector is specified by the fields of a supplied FCB. Typically, though not necessarily, this FCB has been initialized by a prior call to D&OPEN or D&CREA. If the SIF value in the FCB is non-zero, then the TRK and SEC fields are updated following a call to point to the next logical sector.

Call:	JSR	D&WRIT	call to write sector
	DAC	fcb	pointer to FCB
Entry:	(ACR)		irrelevant
	($X)		irrelevant
	FCB	FCB&TRK	track to be written
		FCB&SEC	sector to be written
		FCB&SIF	SIF to be used to update TRK, SEC
		FCB&UNI	unit to be written
		FCB&LTR	last track, used to check for end of file
		FCB&DIR	irrelevant
Exit:	(ACR)		status
	($X)		destroyed
	FCB	FCB&TRK	updated by FCB&SIF value
		FCB&SEC	updated by FCB&SIF value
		Other fields	unchanged

Status:	0	successful completion of write
	1	search check
	2	write check
	3	unit inoperable
	4	attempted write past end of file
	5	write protected

NOTES: The write is attempted five times with recalibration before signalling a write or search check.

The sync and address bytes in DBF are set by D&WRIT and need not be set by the caller.

If the V option is set in DBF&OPT, the write is followed by a verify to check that the data was written correctly. This takes an entire revolution of the disk.

FCB&TRK, FCB&SEC are updated by SIF even if an error occurs.

The data to be written occupies positions 3-130 (DBF&DTA to DBF&DTA+127) of the DBF and may be set using D&SBPW, D&WRBF or D&MRTB.

For sequential writing of a file, the SIF is non-zero and the TRK and SEC fields are updated automatically. For random writing, SIF is zero and the caller sets appropriate TRK and SEC values in the FCB prior to each D&WRIT call.

The FCB&TRK value is an absolute track number. It is not relative to the beginning of a file.

Routines Used: D&SBPR, D&SBPW, D&WRBF

&WRSR -- Write Source Record

&WRSR is used to write a source record to a file previously opened with call to D&OPSW or created with a call to D&CRES.

all:	JSR	D&WRSR	call to write source record
	DAC	scb	pointer to SCB
ntry:	(ACR)		irrelevant
	($X)		irrelevant
	SCB	SCB&TRK	track number of next sector to be written
		SCB&SEC	next sector to be written
		SCB&SIF	interlace factor
		SCB&UNI	unit number
		SCB&LTR	last track of file (D&OPSW) or of disk (D&CRES)
		SCB&DIR	as set by D&OPSW or D&CRES
		SCB&CTR	bytes left in work buffer
		SCB&LEN	max length of record to be written
		SCB&BUF	pointer to 128 byte work buffer
		SCB&BFP	pointer to next location in work buffer
		SCB&REC	pointer to record to be written
xit:	(ACR)		status
	($X)		irrelevant
	SCB	SCB&TRK	Updated ready to write
		SCB&SEC	

SCB&CTR	next record
SCB&BFP	
Other fields	unchanged

Status:	0	successful completion of write
	1	search check
	2	write check
	3	unit inoperable
	4	attempted write past end of file
	5	write protected

NOTES: The record to be written is stored in the SCB&LEN bytes starting at SCB&REC or it can be shorter than the max length given if it is terminated by an X'0D'.

Since the source file format involves data compression, D&WRSR cannot be used to update a source file in place.

No track reformatting will occur, even if a hard write error occurs on sector zero of a track while SIF = 1.

Routines Used: D&WRIT, D&MRTB, D&BLMV, D&WRIN

SECTION V

DISKETTE FORMATS

This chapter contains sections describing the detailed format of files and control information stored on SPD/DOS diskettes.

Format of Nucleus Bootstrap and Code

The nucleus bootstrap occupies sectors 0-3 of track 0. This bootstrap saves the current core image on track two for a manual or abnormal return boot and then loads in the nucleus code. The nucleus code itself is written with a sector interlace factor of five starting with sector 8, skipping over sectors 0-4.

The nucleus bootstrap and code are assembled as part of the FORMAT utility, whose source listing should be consulted for further details of the exact format of track zero.

Format of Label Record

The label record is sector 4 of track 0. It has the following format:

Bytes 0-7	Diskette Serial Number. Eight ASCII characters.
Byte 8	Printer lines/page Number of lines per printed page X'FF' if no printer
Byte 9	Printer characters/line Number of characters per printed line. X'FF' if no printer
Byte 10	Printer type X'01' LP200, LP400 X'02' P-100, P-165 X'03' LP125, LP250, LP300 X'04' P-15 X'05' Termiprinter X'FF' No printer

Byte 11	Printer channel/unit. Either the channel of number of a parallel or async controller or the unit number of the printer on the multi-printer controller + X'80'. X'FF' if no printer.
Byte 12	Multi-printer setup byte. X'FF if no multi-printer controller.
Byte 13	Number of disk units. X'01' or X'02'
Byte 14	Keyboard type X'01' Upper case X'02' U/L X'03' U/L, convert lower to upper case
Byte 15	Display number of lines X'0F' (15) for 10/20 half screen display X'1E' (30) for 10/20 full screen display or 10/24 or 10/25 or 20/20.
Byte 16	End refresh character X'04' for 10/20 with X'04' end refresh code X'74' for 10/20 with X'74' end refresh code or 10/24 X'01' for 10/25 or 20/20
Byte 17	Card reader channel Channel number of card reader X'FF' if no card reader
Byte 18	Tape reader channel Channel number of paper tape reader X'FF' if no paper tape reader

The serial number is set when the disk is formatted. It can be changed only by reformatting (possibly with the N option to avoid destruction of file data). The configuration parameters may be modified by use of the CNFG utility.

Format of File Directory

The file directory is written on track one start ing with sector zero using a sector interlace factor of five. Each sector contains two 64 byte entries:

Track 1	Sector 0	Entries 1, 2
Track 1	Sector 5	Entries 3, 4
Track 1	Sector 10	Entries 5, 6
. .	.	. . .
. .	.	. . .
etc.		

The format of each 64 byte entry is as follows (the DIR&XXX names are offsets defined in the standard symbols provided by ASSEMBLE).

Byte 0 (DIR&STA)	X'20' (' ')	Active entry
	X'2A' ('*')	Deleted entry
	X'3F' ('?')	Active entry, error status
Byte 1 (DIR&TYP)	X'44' ('D')	Data File
	X'4F' ('O')	Object File
	X'52' ('R')	Relocatable File
	X'53' ('S')	Source File
Bytes 2-9 (DIR&FNM)		File name. Up to eight ASCII characters, left justified, right blank filled.
Bytes 10-49 (DIR&LBL)		Label. Forty ASCII characters.
Byte 50 (DIR&FTR)		First track. Starting track number of file data.
Byte 51 (DIR&SIF)		Sector interlace factor.
Byte 52 (DIR&LTR)		Last track. Number of last track containing data for this file.
Bytes 53-63		Unused and undefined.

The end of the file directory is marked by an entry whose first byte is set to X'00'. All subsequent entries also start with X'00'.

When a file is created, a partial entry is made with the following format:

Byte 0	(DIR&STA)	X'00'
Byte 1	(DIR&TYP)	File type ('D', 'O','R', or 'S')
Bytes 2-9	(DIR&FNM)	File name
Bytes 10-49	(DIR&LBL)	File label
Byte 50	(DIR&FTR)	First track
Byte 51	(DIR&SIF)	Not set (undefined)
Byte 52	(DIR&LTR)	Not set (undefined)
Byte 53	(DIR&EPT)	Erase pointer X'00' if no file to be erased when the file is closed. Else if there is a file to be erased on close: Bit 7 set to 1. Bit 6 set if directory entry is in second half of sector. Bits 5-0 sector number of directory entry of file to be erased.

When the file is closed, this entry is modified to become a normal active directory entry by setting the DIR&STA byte to blank and setting the DIR&SIF and DIR&LTR bytes from the FCB. If bit seven of the DIR&EPT byte is set, then the entry for the file pointed to by its remaining bits is set to erased status. If a file is never closed, then this partial entry acts as a normal end of directory sentinal.

Format of Saved Core Image

Track two is used to save part of the core image when a manual boot or abnormal nucleus return occurs.

The nucleus and all SPD/DOS utilities (except ASSEMBLE and RASSEMBL) use locations X'0000' - X'0DFF' and the top sector (whose physical address depends on the core size): SPD/DOS always addresses top secotr as if a 32K memory were present . These 4K bytes are saved on track two starting at sector zero using an interlace factor of five:

```
Track 2  Sector 0   Bytes X'0000' - X'007F'
Track 2  Sector 5   Bytes X'0080' - X'00FF'
Track 2  Sector 10  Bytes X'0100' - X'017F'
            .         .                  .
            .         .                  .
            .         .                  .
Track 2  Sector 7   Bytes X'0D80' - X'0DFF'
Track 2  Sector 12  Bytes X'xE00' - X'xE7F'
Track 2  Sector 17  Bytes X'xE80' - X'xEFF'
Track 2  Sector 22  Bytes X'xF00' - X'xF7F'
Track 2  Sector 27  Bytes X'xF80' - X'xFFF'
```

(x=0, 1, 3, 7, for 4K, 8K, 16K, 32K memory.)

The ZAP utility maintains its virtual memory image in this same format. Locations X'0E00' - X'xDFF' on an SPD 20/20 are maintained on core since these core locations are not modified by the nucleus or other SPD/DOS utilities, except ASSEMBLE and RASSEMBL which use all of available memory and thus destroy part of the ZAP memory image on machines with more than 4K bytes of storage.

Format of Data Files

Data files consist of a contiguous set of tracks whose internal format is completely under control of the applications program using the file and not restricted in any way by the operating system.

Format of Object Files

Object files are wirtten in a sequential manner with a sector interlace factor of five. In general, the sectors are considered to form a continuous sequence of bytes and logical records are written without regard to sector boundaries. However, the object text for each segment, including segment zero, must start on a sector boundary, and fill bytes with value X'FF' are used to ensure that this is the case.

The first four bytes of the object file are as follows:

Byte 0	CNFG	X'00'	CNFG 0
		X'0A'	CNFG 10
		X'14'	CNFG 20
		X'18'	CNFG 24
		X'19'	CNFG 25

The MSB of this byte is set for BOOT mode programs.

Byte 1 Number of overlay segments. X'00' for non-overlay programs.

Byte 2	Size	X'0F'	4K
		X'1F'	8K
		X'3F'	16K
		X'7F'	32K

Byte 3 External Size. Most significant byte of the third operand of the SIZE pseudo-op. X'00' if no external memory present (SIZE third operand omitted). Note that programs with external memory requirements cannot be loaded by the SPD/DOS nucleus.

Following these four bytes are the entries in the segment location dictionary. There is one entry for each overlay segment (no entries for an unsegmented program). Each entry is two bytes long:

Byte C	Track number of first sector containing load records for the segment minus the track number of the start of the object file (i. e., relative track number).
Byte 1	Sector number of first sector containing load records for the segment.

There can be up to 250 entries corresponding to the maximum permitted number of segments. Following the last entry X'FF' fill bytes are written as required to fill to a sector boundary. Thus the segment location dictionary occupies from 1-4 sectors depending on the number of overlay segments.

Following the segment location dictionary are text records with the following format:

Byte 0	Segment number
Bytes 1-2	Initial load location
Byte 3	N = count of text bytes +1
Bytes 4-(N+2)	Text data bytes

A count byte with a value of 1, followed by no data, is permissible for a null record.

After all the load records for an overlay segment is a transfer record:

Byte 0	Segment number
Bytes 1-2	Transfer address
Byte 3	X'00'

The text records for each overlay segment start on a sector boundary. X'FF' fill bytes being written as required. All text records for an overlay segment are contiguous and followed by the segment transfer record. The text records for the main segment (segment zero) may be interspersed through the file or may be collected at the start of the file. The former organization is the one output by the assembler. COPY with the O and L options rearranges the file into the more efficient second organization. Overlay segments need not necessarily appear in order.

The end of the object file is indicated by the occurrence of the last transfer record (the number of transfer records is equal to the number of overlay segments + 1). Remaining data to fill out the last track is undefined and never read.

Note once more that the X'FF' fill bytes occur only to fill out the last segment location dictionary sector and to place the start of text for each overlay segment on a sector boundary. They must not appear anywhere else.

Format of Source Files

Source files are written sequentially with an odd sector interlace factor (normally eleven). Successive sectors are considered to form a continuous stream of bytes and logical records are written without regard to sector boundaries.

Each logical record is up to 255 characters long and consists of ASCII graphic characters (X'20' - X'7F'). Records may be compressed on the file by deleting trailing blanks and replacing strings of up to 63 embedded blanks by X'80' + (number of blanks). Strings of other repeated characters (1-63 characters in length) may be replaced by a byte with value (X'C'0'+ string length) followed by a single byte containing the repeated character. Following the last character of the compressed record, a carriage return (X'0D') marks the end of the record.

The end of the source file is marked by the occurrence of EOT (X'04') following the carriage return of the last record. The remaining contents of this sector and of all sectors following to fill out the final track are undefined and never read since the EOT terminates the file.

The characters X'01' and X'02' appearing in a source file will cause the source input routines to return search check or read check status. This is used to mark bad locations in source files which are copied or packed. Such files are always marked with ? status in the file directory.

The data compression is performed automatically by the source output routines. However, it is not compulsory and the source input routines will correctly read source records containing uncompressed data. The code X'81' is a permissible replacement for a single blank.

FORMAT OF RELOCATABLE FILES

A relocatable file is basically a sequential string of bytes which fills consecutive sectors of the file at any non-zero interlace factor. The standard value of this interlace factor as generated by the RASSEMBL is 9. No assumption should be made about this value on reading a relocatable file, the value should be obtained from the directory entry as usual.

MODULE DIRECTORY

The first part of the file is the module directory consisting of a series of entries in the form:

1-8 bytes	Module name. MSB of last character set on.
1 byte	Relative track of start of module (i.e. track number of start of module preamble minus track number of start of file).
1 byte	Sector (0-31) of start of module preamble.
1 byte	Offset (0-127) of start of module preamble within sector.

As indicated, the length of each entry is variable from 4-11 bytes, the entries spilling over sector boundaries as required.

The end of the module directory is indicated by a dummy entry with a name of X'FF'. This dummy entry correctly indicates the first unused byte location in the file, and does not point to an actual preamble. The preamble for the first module either immediately follows this dummy entry or X'FF' fill bytes may intervene.

MODULE PREAMBLE

The preamble is a contiguous string of bytes of variable length starting at the location indicated by the associated module directory entry. It has the following format.

Bytes 0-1	Highest (i.e. first unused) relocatable origin in module. This is used to determine the location and load counter values to be set following an IN pseudo-operation.
Bytes 2-3	Highest (i.e. first unused) TOP origin in module. X'0000' if no TOP pseudo-operation appears in module.
Byte 4	CNFG setting of module.
Byte 5-N	External symbol dictionary.
Byte N+1	X'FF' terminator.

External Symbol Dictionary

A contiguous series of bytes containing one entry for each XTN or DEF pseudo-operation in the module source text, immediately following the five header bytes described above.

Either type of entry starts with the characters of the name, the MS bit of the last character being set on. In the case of DEF entries, a WS (see section on word specifications) follows which specifies the defined value of the symbol. An XTN entry is complete with the name (an examination of the codes involved will indicate that this arrangement causes no ambiguities).

The order of DEF entries has no effect since the only function of these entries is to make the appropriate entries in the symbol table of the absolute assembly.

The order of the XTN entries is significant. The first entry is numbered X'0001' and subsequent entries are assigned successively higher numbers. References in the module text to external symbols use these numbers.

End of Module Preamble

The end of the external symbol dictionary and hence of the preamble is marked by a single byte with value X'FF'. The module text either follows immediately, or after intervening X'FF' fill bytes.

MODULE TEXT

The ext for each module is a contiguous series of bytes in a special format designed to minimize the sapce required for relocatable libraries.

First we define some special sequences which are used throughout.

Word Specification (WS)

A WS is a specification of a word value or address operand consisting of a descriptor byte followed by a one or two byte value. The descriptor byte has the following form:

11XLIBTT

X	Normally set to 1. Set to 0 only for an indexed one word instruction operand.
L	Normally set to 0. Set to 1 only for a literal reference in a one word instruction (value given is literal operand).

I	Normally set to 0. Set to 1 if the WS represents an address operand for an indirectly addressed instruction or pseudo-operation.
B	Set to 1 if the following value is two bytes. Set to 0 if the following value is one byte (i.e. MS byte of value is X'00').
TT	Type of Value

0 0	Absolute value
0 1	Relocatable value
1 0	TOP value
1 1	External symbol reference (in this case the "value" is the external symbol number).

If a WS represents an operand of the form: external $\pm$ offset, then the WS is immediately followed by the sequence: X'F9' a a

where a a is the (two-byte) absolute offset to be added to the value of the external reference.

Byte Specification (BS)

A BS specifies a byte value, as used, for example, in an immediate class instruction. The following possibilities exist:

(a) For an external reference, a BS has the same form as a WS.

Note that the descriptor byte has one of the two values X'E3' or X'E7'.

(b) Absolute byte values greater than or equal to X'E0' are represented by X'FF' followed by the byte value.

(c) Absolute byte values less than X'E0', the BS is simply the byte value in question.

One-Word (Word or Byte Class) Instructions

The normal form consists of one byte containing the opcode followed by a WS giving the operand address. The opcode byte has all addressing bits off.

Immediate Class Instructions

Opcode byte + BS giving the byte operand.

Jump on Condition Instructions

Two bytes giving the opcode and second byte of the instruction followed by a WS giving the jump operand.

An SKP instruction is represented as the word value X'8800' using the special code X'F5' (see section on special codes).

Compare and Jump Instructions

One byte giving the opcode followed by BS giving the immediate byte operand followed by WS giving the jump address operand.

Generic Instructions

Two bytes of the instruction.

Input-Output Instructions

The opcode byte is generated first followed by a BS specifying the function code and a BS specifying the channel. In the cases of JFACK, JTACK, a WS follows specifying the jump address.

Special Codes

X'01' text X'01'	Generate text (0 or more bytes from TEXT, TXT8, LTX8, LTXT pseudo-operation). Note that the opcode value X'01' never appears so no confusion arises.
X'F1'	BSS 1 (used for word alignment).
X'F2' a a	BSS aa where aa is a two byte absolute value giving the count.
X'F3' BS	Generate byte specified by BS once.
X'F4' a a BS	Generate byte specified by BS number of times indicated by two byte absolute value aa.
X'F5' WS	Generate word specified by WS once.
X'F6' a a WS	Generate word specified by WS number of times indicated by two byte absolute value aa.
X'F7'	Switch from relocatable to TOP section or vice versa.
X'F9' WS	Generate address constant WS (from DAC)
WS X'F9' a a	The WS is of the form external and offset and a a is the two byte absolute offset value.
X'FF' a	As a BS, generate the one byte absolute value a, which is greater than or equal to X'E0'.

End of Module Text

The module text is terminated by a single byte X'FF'. The preamble for the next module, if any, either immediately follows or X'FF' fill bytes may intervene. The bytes following the last byte of the last module text are unreferenced and undefined.

SECTION VI

EXTERNAL FILE FORMATS

This section contains exact details of the external file formats supported by the COPY utility. There is a sub-section for each device type for each file type. Note that magnetic tape and punched card file formats are the same, except for the hardware file mark (only on magnetic tape) and the bootstrap (only on punched card objects).

DATA FILE ON CASSETTE TAPE

Header Block

	2000 Leader bytes X'55'
	1 Sync byte X'77'
Byte 1	'D'
Bytes 2-41	Label
Byte 42	Sector interlace factor

Data Block

There is one data block for each sector of information. Thus the number of data blocks is a multiple of 32 since an integral number of tracks is involved. The sectors are written in the logical order specified by the sector interlace factor unless the interlace factor is zero, in which case the sectors are written as though the interlace factor was 5.

	30 Leader bytes X'55'
	1 Sync byte X'77'
Byte 1	X'FF'
Bytes 2-129	128 data bytes

End of File Block

	30 Leader bytes X'55'
	1 Sync byte X'77'
Byte 1	X'00' (end of file flag)
Bytes 2-129	All set to X'00'

OBJECT FILE ON CASSETTE TAPE

Bootstrap

For unsegmented programs, a bootstrap loader is written as the first block. This loader requires the following operating sequence:

MANUAL
REWIND
BOOT
AUTO BOOT

On input, the bootstrap record is completely ignored.

Header Block

	30 or 2000 leader bytes X'55' (depending on whether a loader is written)
	1 sync byte X'77'
Byte 1	'O'
Bytes 2-41	Label
Byte 42	Program CNFG, two hexadecimal digits, 00 = CNFG 0 0A = CNFG 10 14 = CNFG 20 18 = CNFG 24 19 = CNFG 25
Byte 43	Number of overlay setments. Two hexadecimal digits, 00 for unsegmented program.

Byte 44	Program size MSB (i.e., minimum required memory). Two hexadecimal digits (e.g., 1F=8K).
Byte 45	External size. Two hexadecimal digits, 00 if no external memory present, else it is the most significant byte of the third parameter to SIZE.

Data Block

	30 leader bytes X'55'
	1 sync byte X'77'
Bytes 1-512	512 bytes of text data

Text and transfer records are of variable length and are packed into 512 byte blocks as indicated above, spilling across block boundaries as required.

Text Record:	Byte 1	Segment number
	Bytes 2-3	Initial load address
	Byte 3	Count of data bytes +1
	Bytes 4-N	(N-3) data bytes

Transfer Record:

Byte 1	Segment number
Bytes 2-3	Transfer address
Byte 4	X'00'

The end of tape is implied by the occurrence of the last transfer record. The number of transfer records is equal to the number of overlay segments plus one (for the main segment). The final physical block is padded out to 512 bytes with X'FF' fill codes.

The order of text and transfer records is subject to the following rules:

(a) All text records for an overlay segment (non-zero segment) must be contiguous and immediately followed by the corresponding transfer record.

(b) The segment zero transfer record must appear after (but not necessarily immediately after) the last segment zero text record.

(c) Segment zero text records may thus be contiguous at the start of the file or interspersed between overlay segment data in the natural assembly-order. The COPY utility will always generate the first of these formats when copying from an object file on diskette.

Boot Mode Programs

If COPY is used to write a BOOT mode object program from diskette to cassette tape, the output is generated in standard bootstrap format. There is no way to read a BOOT mode program from cassette tape onto diskette.

SOURCE FILE ON CASSETTE TAPE

Note: This format is compatible with the source format used by the J-101, J-102, J-103 and J-104 cassette editor and assembler programs.

Header Block

	2000 leader bytes X'55'
	1 sync byte X'77'
Byte 1	S
Byte 2	: (colon)

Bytes 3-28	First 26 characters of label. On input this is padded with blanks, on output the label is truncated to 26 characters.
Byte 29	X'FE'

Text Blocks

	30 leader bytes X'55'
	1 sync byte X'77'
Bytes 1-100	100 data bytes
or	
	30 leader bytes X'55'
	1 sync byte X'77'
Bytes 1-N	N data bytes (N<100)
Byte N+1	X'FE'

The short block format with the X'FE' may appear anywhere in the input, but is only used for the last block on output.

The source records are written in variable length form terminated by X'0D' (carriage return), packed into 100 byte blocks and spilling over block boundaries as required.

Consecutive blanks in a source record may be (input)/are (output) replaced by X'80' + count of blanks.

The end of file is marked by X'FF' following the last carriage return.

RELOCATABLE FILE ON CASSETTE TAPE

Header Block

	2000 Leader bytes X'55'
	1 Sync byte X'77'
Byte 1	'R'
Bytes 2-41	Label
Byte 42	Sector interlace factor

Data Block

There is one data block for each sector of information. Thus the number of data blocks is a multiple of 32 since an integral number of tracks is involved. The sectors are written in the logical order specifie` by the sector interlace factor.

	30 Leader bytes X'55'
	1 Sync byte X'77'
Byte 1	X'FF'
Bytes 2-129	128 data bytes

End of File Block

	30 Leader bytes X'55'
	1 Sync byte X'77'
Byte 1	X'00' (end of file flag)
Bytes 2-129	All set to X'00'

DATA FILE ON MAGNETIC TAPE

Data files are written to magnetic tape in 80 character blocks, odd parity, nine-track, ASCII code.

Header Record

Character 1	'D'
Characters 2-41	Label
Character 42	Sector interlace factor (hexadecimal value)
Characters 43-80	Blank

Data Records

Each data record is two blocks long and corresponds to a single sector of information. The number of data blocks is thus always a multiple of 64 since there is an integral number of tracks. The order of data records corresponds to the logical sector order as implied by the interlace factor except when the interlace is zero in which case the records are written as though an interlace of 5 was in use.

Block 1	Characters 1-2	Blanks
	Characters 3-66	Data bytes 0-63 in hexadecimal punch code
	Characters 67-80	Blanks
Block 2	Characters 1-2	Blanks
	Characters 3-66	Data bytes 64-127
	Characters 67-80	Blanks

End of File Record

Character 1	'.' (period)
Character 2	'E'
Character 3	blank
Character 4-43	label
Character 44-80	blanks

Followed by hardware File-Mark

OBJECT FILE ON MAGNETIC TAPE

Object files are written to magnetic tape in 80 character blocks, odd parity, nine-track, ASCII code. There are no bootstrap records.

Header Record

Character 1	'O'
Characters 2-41	Label
Character 42	Blank
Characters 43-44	Program confg. Two hexadecimal digits 00 = CNFG 0 0A = CNFG 10 14 = CNFG 20 18 = CNFG 24 19 = CNFG 25
Characters 45-46	Number of overlay segments. Two hexadecimal digits, 00 for unsegmented program.
Characters 47-48	Program size MSB (i.e., minimum required memory). Two hexadecimal digits (e.g., 1F=8K)
Characters 49-50	External size. Two hexadecimal digits, 00 if no external memory present, else it is the most significant byte of the third parameter to SIZE.
Characters 51-78	Blanks
Characters 79-80	Checksum. Ones complement (mod 255) sum of the four byte values on the header line as two hexadecimal digits.

Text Records

Characters 1-2	Segment number, two hexadecimal digits.
Characters 3-6	Starting load address for data on this record given as four hexadecimal digits.
Characters 7-76	1-35 data bytes, each given as two hexadecimal digits. If less than 35 bytes are present, blanks fill the unused character positions.

Characters 77-78	Blanks
Characters 79-80	Checksum. Two hexadecimal digits representing the ones-complement (mod 255) sum of all hexadecimal data previously output on this record, previous text records (including checksums) and header record. Thus the checksum is cumulative and is reset only by a transfer record.

<u>Transfer Record</u>

Characters 1-2	Segment number, two hexadecimal digits.
Characters 3-6	Transfer address, four hexadecimal digits.
Characters 7-78	Blanks
Characters 79-80	Checksum. As for text record. The checksum is reset to zero for the start of the next record.

<u>End of File Record</u>

Character 1	'.' (period)
Character 2	'E'
Character 3	Blank
Characters 4-43	Label (not required on input decks)
Characters 44-80	Blanks

Followed by Hardware File-Mark

The order of text and transfer records is subject to the following rules:

(a) All text records for an overlay segment (non-zero segment) number must be contiguous and immediately followed by the corresponding transfer record.

(b) The segment zero transfer record must appear after (but not necessarily immediately after) the last segment zero text record.

(c) Segment zero text records may thus be contiguous at the start of the file or interspersed between overlay segment data in the natural assembly order. The COPY utility will always generate the first of these forms when copying from an object file on diskette.

Boot Mode Programs

BOOT mode programs are output to tape as they appear on disk (i.e., with a CNFG value whose MSB is set). They may be read back from tape to disk in this form.

SOURCE FILE ON MAGNETIC TAPE

Source files are written to magnetic tape in 80 character blocks, odd parity, nine-track, ASCII code.

Header Record

Character 1	: (ASCII colon)
Characters 2-41	Label
Characters 42-80	Blanks

On input to COPY, the header may be omitted in which case the label is taken from characters 2-41 of the first source record.

Source Records

Characters 1-80	Source image data in ASCII punched code.

End of File Record

$$\left.\begin{array}{ll}\text{Character 1} & \text{'.' (period)}\\ \text{Character 2} & \text{E}\\ \text{Character 3} & \text{Blank}\\ \text{Characters 4-43} & \text{Label}\\ \text{Characters 44-80} & \text{Blanks}\end{array}\right\}\quad\text{optional}$$

Followed by Hardware File-Mark

RELOCATABLE FILE ON MAGNETIC TAPE

Relocatable files are written to magnetic tape in 80 character blocks, odd parity, nine-track, ASCII code.

Header Record

Character 1	'R'
Characters 2-41	Label
Character 42	Sector interlace factor (Hexadecimal value)
Characters 43-80	Blank

Data Records

Each data record is two blocks long and corresponds to a single sector of information. The number of data blocks is thus always a multiple of 64 since there is an integral number of tracks. The order of data records corresponds to the logical sector order as implied by the interlace factor.

Block 1	Characters 1-2	Blanks
	Characters 3-66	Data bytes 0-63 in hexadecimal punch code
	Characters 67-80	Blanks
Block 2	Characters 1-2	Blanks
	Characters 3-66	Data bytes 64-127
	Characters 67-80	Blanks

End of File Record

Character 1	'.' (period)
Character 2	'E'
Character 3	blank
Characters 4-43	label
Characters 44-80	blanks

Followed by Hardware File-Mark

DATA FILE ON PAPER TAPE

Header Block

All characters up to X'81' starting this record are ignored on input.

Byte 1	X'81'
Byte 2	Sector interlace factor
Bytes 3-42	Label in ASCII code
Byte 43	Checksum. Ones-complement (mod 255) sum of all data in the header block including the X'81'.

Data Blocks

There is one data block for each sector of information. Thus the number of data blocks is a multiple of 32 since an integral number of tracks is involved. The sectors are written in the logical order specified by the sector interlace factor unless the interlace factor is zero, in which case the sectors are written as though the interlace was 5. Nulls or deletes (or any characters other than X'82' or X'83' may appear between blocks.

Byte 1	X'82'
Bytes 2-129	128 bytes of data, full 8 bit ASCII code.
Byte 130	Checksum. Ones-complement (mod 255) sum of all data in the block including the X'82'.

End of File Block

Byte 1	X'83'

OBJECT FILE ON PAPER TAPE

Note: This format is compatible with output generated by the H716 assembler system.

Header Block

All characters up to the X'81' starting this record are ignored on input, including any bootstrap record.

Byte 1	X'81'
Byte 2	Program CNFG X'00' = CNFG 0 X'0A' = CNFG 10 X'14' = CNFG 20 X'18' = CNFG 24 X'19' = CNFG 25
Byte 3	Number of overlay segments. X'00' for unsegmented program.
Byte 4	Program size.
Byte 5	Program size MSB (i.e., minimum required memory). Two hexadecimal digits (e.g., 1F=8K).
Byte 6	External size. Two hexadecimal digits, 00 if no external memory present, else it is the most significant byte of the third parameter to SIZE.
Bytes 7-46	File label. This may be less than forty characters long, in which case the label is filled out with blanks. On input the parity (MS) bit is stripped off.
Byte 47	X'9D' IGS to terminate label
Byte 48	X'00'
Byte 49	Checksum. Ones complement (mod 255) sum of all bytes in block including the X'81'.

Text Blocks

Nulls or deletes may appear between blocks.

Byte 1	X'82'
Byte 2	Segment number
Bytes 3-4	Initial load address for text in this record
Bytes 5-N	Text data, see below
Byte N+1	X'9D'
Byte N+2	X'00'
Byte N+3	Checksum. Ones complement (mod 255) sum of all bytes in block including the X'82').

The text is in 8-bit ASCII code except that the following subsequence is used for strings of duplicated characters.

Byte 1	X'9D'
Byte 2	Duplication count (non-zero)
Byte 3	Character to be duplicated

To prevent confusion, the X'9D' character itself in text always appears as byte 3 of a duplication subsequence (with a duplication count of one if necessary).

Transfer Block

Byte 1	X'83'
Byte 2	Segment number
Bytes 3-4	Transfer address

Byte 5	X'9D'
Byte 6	X'00'
Byte 7	Checksum. Ones complement (mod 255) sum of first six bytes in block.

The end of tape is implied by the occurrence of the last transfer block. The number of transfer blocks is equal to the number of overlay segments plus one (for the main segment).

The order of text and transfer blocks is subjrct to the following rules:

(a) All text blocks for an overlay segment (non-zero segment) must be contiguous and immediately followed by the corresponding transfer block.

(b) The segment zero transfer block must appear after (but not necessarily immediately after) the last segment zero text block.

(c) Segment zero text blocks may thus be contiguous at the start of the file or interspersed between overlay segment blocks in the natural assembly order.

Patch Format

Patch records are inserted in the tape to replace the existing transfer block for the segment to be patched. All data in patch mode is ASCII letters or digits or carriage returns with the parity bit ignored. All other characters are ignored and may be used for layout purposes.

The letter P, if read when byte 1 of a block was being serached, causes patch mode to be entered. Text patch records are then entered as follows:

Characters 1-4	load address as four hexadecimal digits
Characters 5-6	First data byte (two hexadecimal digits)
Characters 7-8	Second data byte
.	
.	
.	
Characters (2N+3)-(2N+4)	N'th data byte
Character 2N+5	Carriage return

Following the last text patch record is the transfer patch record.

Characters 1-4	Transfer address as four hexadecimal digits
Character 5	Letter S

Note that the segment number is not given. It is assumed to be the same as the current segment number. Thus patches can only appear following at least one normal format text block for the segment to be patched.

SOURCE FILE ON PAPER TAPE

Note that all nulls, deletes and line feeds on a source tape are ignored.

Header Block

Byte 1	S
Bytes 2-41	Label (if less than 40 characters, filled out with blanks)
Byte 42	Carriage return

On input to COPY, the header may be omitted, in which case the label is taken from characters 2-41 of the first source record, filled out with blanks if necessary.

Source Record Blocks

Bytes 1-N	1-N ASCII characters, parity bit ignored
Byte N+1	Carriage return

Deleted Source Record Blocks

Bytes 1-N	1-N ASCII characters
Byte N+1	ASCII Back Arrow (X'5F' or X'DF')
Byte N+2	Carriage Return

End of File Block

Byte 1	. (period)
Byte 2	E
Byte 3-N	Option character data

<u>or</u>

Byte 1	X'03' or X'83' (ETX)

<u>or</u>

Byte 1	X'04' or X'84' (EOT)

RELOCATABLE FILE ON PAPER TAPE

Header Block

All characters up to the X'81' starting this record are ignored on input.

Byte 1	X'81'
Byte 2	Sector interlace factor
Bytes 3-42	Label in ASCII code
Byte 43	Checksum. Ones-complement (mod 255) sum of all data in the header block including the X'81'

Data Blocks

There is one data block for each sector of information. Thus the number of data blocks is a multiple of 32 since an integral number of tracks is involved. The sectors are written in the logical order specified by the sector interlace factor. Nulls or deletes (or any characters other than X'82' or X'83' may appear between blocks.

Byte 1	X'82'
Bytes 2-129	128 bytes of data, full 8 bit ASCII code.
Byte 130	Checksum. Ones-complement (mod 255) sum of all data in the block including the X'82'.

End of File Block

Byte 1	X'83'

DATA FILE ON PUNCHED CARDS

Header Record

Column 1	'D'
Columns 2-41	Label
Column 42	Sector interlace factor (hexadecimal value)
Columns 43-80	Blank

Data Records

Each data record is two cards long and corresponds to a single sector of information. The number of data cards is thus always a multiple of 64 since there is an integral number of tracks. The order of data records corresponds to the logical sector order as implied by the interlace factor except when the interlace is zero in which case the records are written as though an interlace of 5 in use.

Card 1	Columns 1-2	Blanks
	Columns 3-66	Data bytes 0-63 in hexadecimal punch code
	Columns 67-80	Blanks
Card 2	Columns 1-2	Blanks
	Columns 3-66	Data bytes 64-127
	Columns 67-80	Blanks

End of File Record

Column 1	'.' (period)
Column 2	'E'
Column 3	Blank
Column 4-43	Label (not required on input decks)
Columns 44-80	Blanks

OBJECT FILE ON PUNCHED CARDS

Bootstrap

The bootstrap punched on output for unsegmented programs consists of seven cards in boot load format with the last card filled out with end boot codes (zeros). On input, any bootstrap loader cards are ignored providing that the last one is completely filled out with zeros.

Header Record

Column 1	'O'
Columns 2-41	Label
Column 42	Blank
Columns 43-44	Program CNFG. Two hexadecimal digits. 00 = CNFG 0 0A = CNFG 10 14 = CNFG 20 18 = CNFG 24 19 = CNFG 25
Columns 45-46	Number of overlay segments. Two hexadecimal digits, 00 for unsegmented program.
Columns 47-48	Program size MSB (i.e., minimum required memory). Two hexadecimal digits (e.g., 1F=8K).
Columns 49-50	External size. Two hexadecimal digits, 00 if no external memory present, else it is the most significant byte of the third parameter to SIZE.
Columns 51-78	Blanks
Columns 79-80	Checksum. Ones complement (mod 255) sum of the four byte values on the header line as two hexadecimal.

Text Records

Columns 1-2	Segment number, two hexadecimal digits.
Columns 3-6	Starting load address for data on this record, given as four hexadecimal digits.

Columns 7-76	1-35 data bytes, each given as two hexadecimal digits. If less than 35 bytes are present, blanks fill the unused columns.
Columns 77-78	Blanks
Columns 79-80	Checksum. Two hexadecimal digits representing the ones-complement (mod 255) sum of all hexadecimal data previously output on this card, previous text cards (including checksums) and header card. Thus the checksum is cumulative and is reset only by a transfer record.

Transfer Record

Columns 1-2	Segment number, two hexadecimal digits.
Columns 3-6	Transfer address, four hexadecimal digits.
Columns 7-78	Blanks
Columns 79-80	Checksum. As for text record. The checksum is reset to zero for the start of the next record.

End of File Record

Column 1	'.' (period)
Column 2	'E'
Column 3	Blank
Columns 4-43	Label (not required on input decks)
Columns 44-80	Blanks

The order of text and transfer records is subject to the following rules:

(a) All text records for an overlay segment (non-zero segment) must be contiguous and immediately followed by the corresponding transfer record.

(b) The segment zero transfer record must appear after (but not necessarily immediately after) the last segment zero text record.

(c) Segment zero text records may thus be contiguous at the start of the deck or interspersed between overlay segment data in the natural assembly-order. The COPY utility will always generate the first of these forms when copying from an object file on diskette.

Patch Format

An object deck may be patched by hand as follows:

(1) All patch cards are in standard text format except that columns 79-80 are left blank.

(2) Patch cards must immediately precede the transfer record of the segment to be patched.

(3) The transfer record of any patched segment must be recopied (possibly modifying the transfer address) with columns 79-80 blank.

The bootstrap loader will correctly load unsegmented programs in this manner. Object decks to be read onto disk using COPY may have patches applied to any segment.

Boot Mode Programs

If COPY is used to punch a BOOT mode object program from diskette, the output is generated in standard boot load format. There is no way to read a BOOT mode program from cards onto diskette.

SOURCE FILE ON PUNCHED CARDS

Header Record

Column 1	: (ASCII colon)
Columns 2-41	Label
Columns 42-80	Blanks

On input to COPY, the header may be omitted, in which case the label is taken from columns 2-41 of the first source record.

Source Records

Columns 1-80	Source image data in ASCII punched code.

End of File Record

Column 1	'.' (period)
Column 2	E
Column 3	Blank
Columns 4-43	Label (not required on input decks)
Columns 44-80	Blanks

RELOCATABLE FILE ON PUNCHED CARDS

Header Record

Column 1	'R'
Columns 2-41	Label
Column 42	Sector interlace factor (hexadecimal value)
Columns 43-80	Blank

Data Records

Each data record is two cards long and corresponds to a single sector of information. The number of data cards is thus always a multiple of 64 since there is an integral number of tracks. The order of data records corresponds to the logical sector order as implied by the interlace factor.

Card 1	Columns 1-2	Blanks
	Columns 3-66	Data bytes 0-63 in hexadecimal punch code
	Columns 67-80	Blanks
Card 2	Columns 1-2	Blanks
	Columns 3-66	Data bytes 64-127
	Columns 67-80	Blanks

End of File Record

Column 1	'.' (period)
Column 2	'E'
Column 3	Blank
Column 4-43	Label (not required on input decks)
Columns 44-80	Blanks

APPENDIX A

LIST OF OTHER USEFUL PUBLICATIONS

TITLE	ORDER NUMBER
PUBLICATIONS	
PUBLICATIONS CATALOG - SECOND EDITION	MS-7159
SPD/DOS MANUALS	
SPD D-250 DISKETTE REFERENCE MANUAL	MS-7143
SPD/DOS DISKETTE OPERATING SYSTEM OPERATORS REF. MAN.	MS-7177
SPD/DOS DISKETTE OPERATING SYSTEM PROGRAMMERS REF.	MS-7178
ASSEMBLER MANUAL	
SPD SYMBOLIC ASSEMBLY LANGUAGE REFERENCE MANUAL	MS-7215
SPD 10/20 MANUALS	
SPD 10/20 INTELLIGENT TERMINAL SYSTEM DESCRIPTION	MS-7145
SPD 10/20 PROGRAMMERS REFERENCE MANUAL	MS-7110
SPD 10/25 MANUALS	
SPD 10/25 INTELLIGENT TERMINAL SYSTEM DESCRIPTION	MS-7199
SPD 10/25 INTELLIGENT TERMINAL SYSTEM PROGRAMMERS REF.	MS-7217
SPD 20/20 MANUALS	
SPD 20/20 MULTI STATION DISPLAY PROGRAMMERS REFERENCE	MS-7144
SPD 20/20 MULTI STATION DISPLAY SYSTEM DESCRIPTION	MS-7165
SPD 20/20 MULTI-STATION DISPLAY SYSTEM OPERATORS MAN.	MS-7190
SPD 320/325	
SPD 320/325 VIDEO TERMINAL SYSTEM DESCRIPTION	MS-7158
SPD 320 VIDEO TERMINAL SYSTEM IBM 3270 COMP. "PLUS" B.	MS-7172
COMMUNICATIONS MANUALS	
COMMUNICATION CONTROLLER REFERENCE MANUAL	MS-7152
INCOTERM DATA COMMUNICATIONS MANUAL	CS-015
CONTROLLERS	
REMOTE LOAD CONTROLLER REFERENCE MANUAL	MS-7121
CYCLIC CHECK CONTROLLER REFERENCE MANUAL (with adden.)	MS-7122
COMMUNICATION CONTROLLER REFERENCE MANUAL	MS-7152
CYCLIC CHECK	
CYCLIC CHECK CONTROLLER REFERENCE MANUAL (with adden)	MS-7122
DATA ENTRY	
INCOFORM SOURCE DATA ENTRY SYSTEM DESCRIPTION MANUAL	MS-7205
INCOFORM SOURCE DATA ENTRY SYSTEM DESCRIPTION BROCHURE	MS-
INCOFORM SOURCE DATA ENTRY SYSTEM OPERATORS MANUAL	MS-7208
INCOFORM FORMS GENERATION OPERATORS MANUAL	MS-7209
PERIPHERAL EQUIPMENT MANUALS	
PRINTER	
SPD P-100 PRINTER REFERENCE MANUAL	MS-7123
SPD P-165/SPD P-165B PRINTERS REFERENCE MANUAL	MS-7218
Punch	
SPD PRP-45/200 PRINTING READER PUNCH OPERATORS MANUAL	MS-7154
SPD PRP-45/200 PRINTING READER PUNCH PRODUCT BULLETIN	MS-7204
TAPE	
MAGNETIC TAPE UNITS REFERENCE MANUAL	MS-7153
SPD-MT TAPE UNITS PRODUCT BULLETIN	MS-7162
SPD-T TAPE CASSETTE PRODUCT BULLETIN	MS-7201

VÄXJÖ DATA SYSTEM AB

VDS

Distributör för Incoterm i Sverige, Norge, Finland och Danmark

VÄXJÖ
Adress: Box 3034, Smedjegatan 37, 350 03 VÄXJÖ
Telefon: 0470/10070
Telex: 52 138

HELSINGBORG
Adress: Landskronavägen 23, 252 32 HELSINGBORG
Telefon: 042/14 94 30

MALMÖ
Adress: Södergatan 12, 211 34 MALMÖ
Telefon: 040/724 50

STOCKHOLM
Adress: Skeppargatan 8, .114 52 STOCKHOLM
Telefon: 08/14 22 35

CORONADATA AB
Adress: Box 5143, Åvägen 18, 402 23 GÖTEBORG
Telefon: 031/20 03 80

CORONADATA A/S
Adress: Park Alle 296, DK - 2600 GLOSTRUP
Telefon: (02) 45 88 22

CORONADATA OY
Adress: Notstigen 4 C, SF - 00330 HELSINGFORS 33
Telefon: 0 - 48 87 22

CORONADATA A/S
Adress: Torggatan 7, OSLO 1
Telefon: 2 - 33 42 60

LANPRODUKTER, Växjö 1976