P800M Programmer's Guide 2
Volume I : DOM

A publication of

Philips Data Systems B.V.

Apeldoorn, the Netherlands

Publication number 5122 991 27373

June, 1977

Copyright (C) by Philips Data Systems B.V., 1977
All rights strictly reserved. Reproduction or issue to third parties in any form whatever is not permitted without written authority from the publisher.

Printed in the Netherlands.

This is the first of a five-volume set dealing with the Disc Operating System (non-real time and real time) for the P800M series. It describes the Disc Operating Monitor.

The other volumes of this set are:
II: Instruction Set
III: Software Processors
IV: Disc Real Time Monitor
V: Multi-Application Monitor

Other books pertaining to the P800M series are:

P852M System Handbook
P856M/P857M Handbook
P800M Operator's Guide
P800M Interface and Installation Manual
P800M Software Reference Data
P800M Data Communication User Manual

Great care has been taken to ensure that the information contained in this manual is accurate and complete. However, should any errors or omissions be discovered, or should any user wish to make a suggestion for improving this manual, he is invited to send his comments, written on the sheet provided at the end of this book, to:

SSS-DOC.
at the address on the opposite page.

Table of contents

	Page
Introduction	1
Chapter 1: Principles of Operation	3
Chapter 2: Memory Organization	7
Chapter 3: Interrupt System	11
Hardware Interrupt Lines	11
Software Priority Levels	12
Dispatcher	13
Stack	13
Chapter 4: Programming	15
Interrupt Routines	15
Software Level Programs	16
Level 63: Idle Task	17
Scheduled Labels	18
Program Segments and Overlay Structure (DOM)	21
Catalogued Procedures (DOM)	23
Job Parameter Table (DOM)	27
Chapter 5: Disc Organization	29
Sectors	29
Files	31
Access Modes	34
Random	34
Sequential	34
Record Format	35
Special Records	37
File Types	38
Load Module Size	39

Page

Disc Structure 40
Catalogue and Library Structure 40
Catalogue Structure 40
Directory Structure 42
DOM System Disc Structure 45

Chapter 6: Input/Output 47
DOM Logical Files and Units 48
File Codes 51
Access Modes 52

Chapter 7: Data Management 53
Sequential Access Method 55
Direct Access Method 62

Chapter 8: Operation 67
Loading Bootstrap and IPL 67
Initial Program Loader (IPL) 69
Starting a Session or Job 69
Batch Processing 72
Changing a Disc Pack 73
Copying or Updating the System Disc 73
System Messages 76

Chapter 9: DOM CCI Control Commands 79
Table of CCI commands 80

Chapter 11: DOM Operator Control Messages 117
Table of Operator Control Messages 117

Chapter 13: Monitor Requests 123

Page

APPENDICES A-1

Appendix A: System Generation A-3

Appendix B: Premark A-39
Disc Premark A-39

Appendix C: Peripheral Input/Output A-41

Appendix D: Control Unit Status Word Configuration . . . A-49

Appendix E: P852M Bootstrap A-51

Appendix F: Control Commands A-57

Introduction

The Disc Operating Monitor (DOM) is used in a non-real time disc system and is intended mainly as a tool for program development. Communication with the system normally takes place via the operator's typewriter by means of a comprehensive set of control commands, more or less as in time sharing, but not only disc storage but also the other peripherals in the configuration may be used without restriction.

This monitor allows the user to process and maintain on disc all kinds of user data, such as programs in source, object and load format as well as various files. This can be done in interactive as well as in batch processing mode.

All system components, including processors, are disc resident. The scheduled label feature allows some form of multiprogramming.

1 Principles of Operation

With the Disc Operating Monitor, there are two modes of operation:
- conversational, working in sessions
- batch processing, working in jobs.

A session is opened when the user types in his user identification, in reply to the message USERID:, which is output after the system has been loaded or after a previous session or job has been closed. The user may then decide on batch processing or conversational mode.

The user identification is an individual code name for each user, through which he gets access to the system and to his own library files. The user identification must have been declared previously and is then stored by the system in a catalogue on the disc. Thus, only those users whose identification is known to the system can access it. The system itself is also considered as a user, with user identification SYSTEM, with an entry in the disc catalogue (the first entry) and with its own library, i.e. the system software components.

The user communicates with the system through control commands which are normally typed in on the operator's typewriter in conversational mode, a card reader or punched tape reader in case of batch processing.
By means of these commands, the user may create files, call processors, handle his library, delete files, etc.

Per user there is one library on disc, pointed to by the user identification entry in the disc catalogue. All his permanent files are stored in this library, which is located on one disc only, i.e. the disc which contains the directory for that user. This implies that one user cannot have his files stored on several discs, unless, of course, he makes use of several user identifications. However, a user can have access to the system library and

to other user libraries by specifying the user identification of the user of those libraries, but he can only read the data in these files, not write in them.

The files contained in a Library may be of several types: source program files, object modules, load files (programs ready for execution) and files not belonging to any of these types, e.g. files created by user programs.

During a session the user will, outside his library, also need temporary storage space on the disc, either for the processors which he activates through his control commands or for his own programs. This storage space is always allocated to him on the same disc on which his library is located.
Files created during a session are always considered temporary. If the user wants to make them permanent, he must give a specific control command (KPF: Keep File) to have the file stored in his library. Temporary files which are not kept in a library with this control command are automatically deleted at the end of a session or by specific other commands, such as SCR (Scratch).

The allocation of disc storage space takes place dynamically, for library files as well as temporary files. A disc is divided into granules, areas of 8 sectors (one sector = 200 words) and the monitor handles the allocation of these granules to the user's files, when he is creating temporary files. For this purpose, the monitor keeps a table of the granules, to which files they belong and which ones are still available for allocation. On each disc, it also maintains a granule table for that disc.
These tables are updated each time a user gives a command to keep or delete a file. By means of the control commands the user can handle his files and library, call processors to update, assemble, compile, link-edit and debug his programs and he may execute them. In his programs the user may ask the monitor to perform certain functions by means of monitor request, which may be coupled to so-called scheduled labels to achieve a form of multi-tasking.

At the end of a session, the user must type in BYE to close his session. After this the system types out USERID: again and waits for the next session or job with the same or another user. In batch processing, the commands are processed sequentially and automatically until a program itself asks for operator intervention or until an error occurs, in which case the monitor looks for a new job or until a command is given to indicate the end of batch processing (BYE).
It is possible to switch from batch to conversational mode or vice versa during processing, the first choice of mode being made at the beginning of a session. If batch processing is chosen, the first command must be JOB.

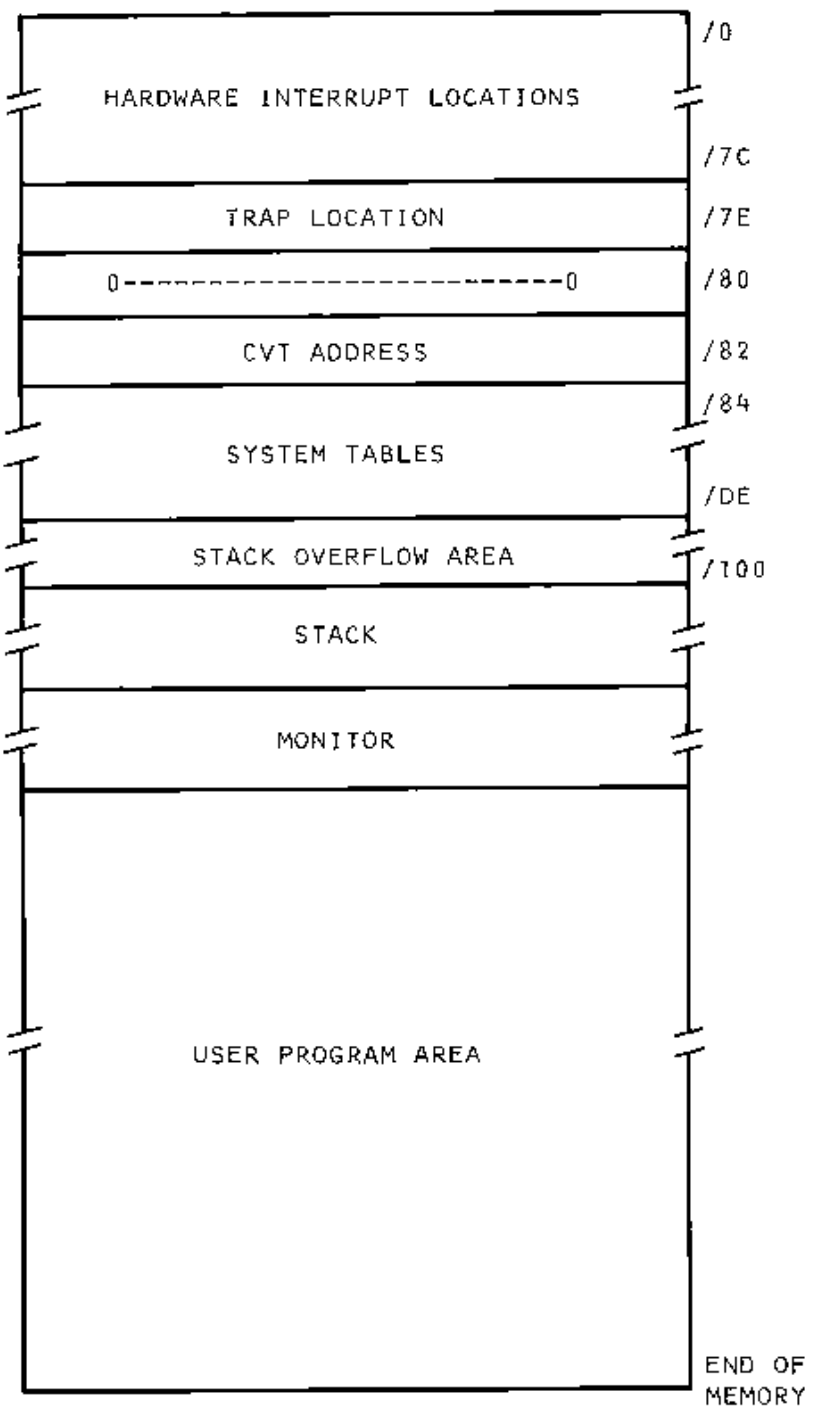
Control Command Interpretation

An interesting feature of the system are the catalogued procedures: it is possible to store a sequence of CCI Control Commands on disc under a procedure name. When it is desired to execute this command sequence or part of it, the procedure is called from the input stream and it is possible at that point to fill in certain parameters or indicate which commands are to be executed and which ones are to be skipped in the catalogued procedure.

CONVENTIONS

Hexadecimal numbers are written preceded by a slash (/), except when used in an operator control message. So:
LDKL A7,/44FE (instruction) but:
DM 44FE 4600 (operator command)
A granule on a disc is defined as an area of 8 consecutive sectors.
Control commands and processor calls are given to the Control Command Interpreter, which types out S: on the typewriter.
Operator control messages are given to the monitor, which types out M: on the typewriter after the INT button on the control panel has been pressed.
Clusters are object code records, the format of which is described in the System Software Manual.

The Memory Layout is as follows:

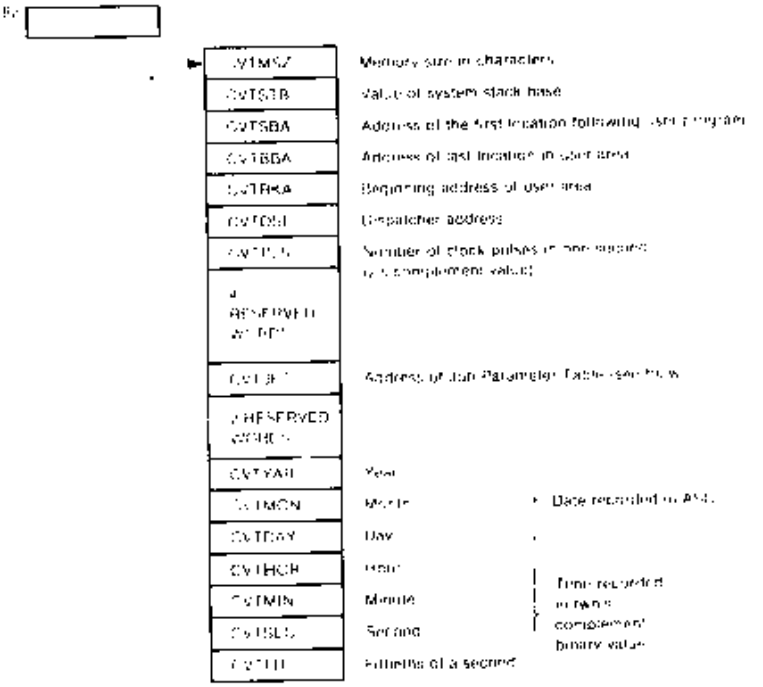


- Location /0 to /7C are hardware interrupt locations. They are hard-wired to internal and external interrupt lines. Each location contains the address of the interrupt routine required to service the interrupt connected to that location. The interrupt connected to location /0 has the highest priority (level 0).

- Location /7E contains the address of the trapping routine which handles simulation of certain instructions not included in the hardware (e.g. double add, double subtract, multiply, divide, multiple load, multiple store and double shift).

- Location /80 contains 0.

- Location /82 points to the Communication Vector Table. This is a system table which contains information which may be of use to the user program. This table has the following layout:



- Locations /84 to /DE are used for other system tables.

- The area occupied by the stack is defined at system generation time. When an interrupt occurs, P- register and PSW are stored here by hardware and a number of registers by software. The number of registers stored depends on whether the interrupt routine servicing the interrupt runs in inhibit mode (anywhere from 0 to 15 registers) or in enable mode or branches to the dispatcher (always 8 registers). The A15 register always points to the next free location in the stack (where all information is stored towards the lower memory addresses). When A15 reaches the value /100 or becomes lower a stack overflow interrupt is given.

- The area after the monitor area is the user area.
  In the user area, one program can be run at a time.
  The area remaining after the program area is reserved for dynamic memory allocation. From this area, blocks of memory space can be requested by the system or by the user. For this purpose the user must send a 'Get Buffer' monitor request. When he does no longer need the buffer, he must send a 'Release Buffer' request.

Programs and routines under DOM run on the basis of an interrupt and priority system consisting of up to 64 levels. These levels are subdivided as follows:

0-47:	levels for interrupt routines connected to the hardware interrupt lines	hardware interrupt levels
48 :	interruptable monitor service routines	software priority levels
49 :	disc file management	
55 :	operator routines	
61 :	abort module	
62 :	user program	
63 :	idle task	

Level 0 has the highest priority, 63 the lowest, so all hardware interrupts always have priority over the software levels.

HARDWARE INTERRUPT LINES

The interrupt lines are connected to memory location /0 to /7C. These locations contain the addresses of the interrupt routines which service the internal and external interrupts.
For the interrupts the user can define the priority levels at system generation time.

The following priorities are recommended for the various interrupt lines:

/0 :	power failure	- (interrupt location /00)
/1 :	LKM/stack overflow	- (interrupt location /02)
/2 :	real time clock	- (interrupt location /04)
/3 :	not used	etc.
/4 :	punched tape reader	
/5 :	tape punch	
/6 :	operator's typewriter	
/7 :	control panel	
/8 - /F:	free	
/10:	X1215 disc	
/11:	disc	
/12:	disc	
/13:	magnetic tape	
/14:	cassette tape	
/15:	card reader	
/16:	-	
/17:	line printer	
/18 - /1F:	free	

Software Priority Levels

Only the user program and some of the monitor modules operate on software priority levels: 62, 49, 55, 61 and 63.
The user program is activated by the CCI command RUN.

Level 63 is reserved for the idle task, an instuction sequence to use up idle central processor time. See also Chapter 4: Programs.

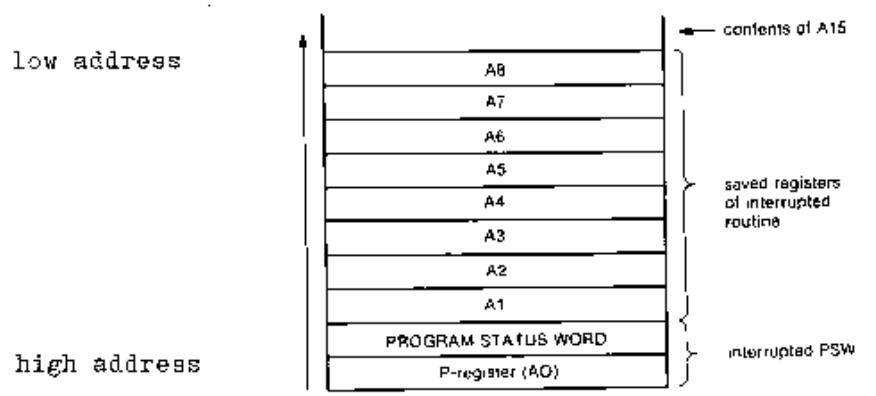
Dispatcher

The dispatcher is a monitor module (M:DISP) running on level 48, which divides central processor time by starting programs according to their priority.
The dispatcher can be entered only from an interrupt routine, i.e. from a level below 48, such as the I/O interrupt and monitor request handlers.

System Interrupt Stack

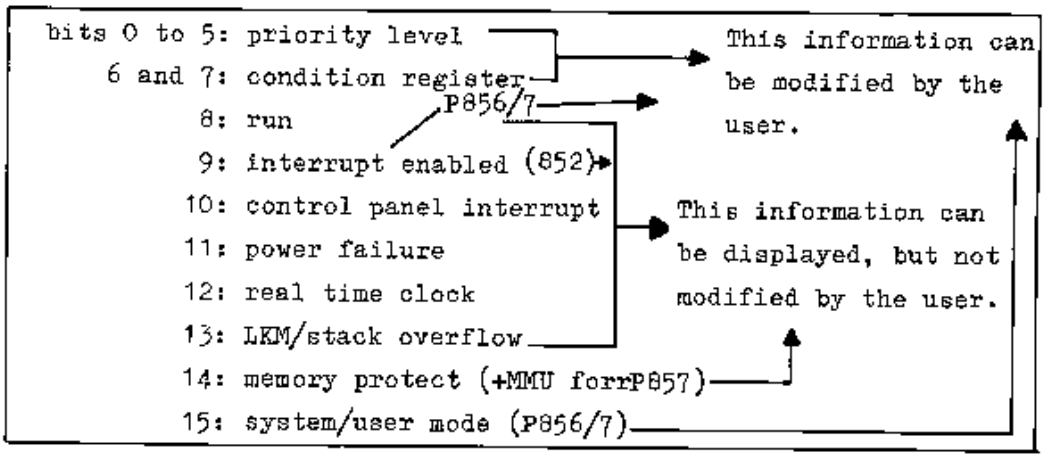
When an interrupt occurs, certain information about the interrupted program or routine must be saved before the interrupt can be serviced.
This is done in the system stack, the address of which is held in register A15. The start address of this stack is defined at system generation time and it is built in a downward direction in memory, i.e. towards the lower addresses. The A15 register always points to the first free location in the stack.
Upon interrupt, PSW and P-register are always saved in this stack and also a number of registers:

- any number if the interrupt routine runs in inhibit mode and takes care of restoring the registers itself;
- registers A1 to A8 if the interrupt routine runs in enable mode or ends with a branch to the dispatcher, because the dispatcher always handles the stack on this basis:



When the stack pointer (A15) reaches or becomes lower than the value /100, the last location before the stack overflow area, an interrupt occurs.

The Program Status Word, stored in the stack upon interrupt, contains the following information:



INTERRUPT ROUTINES

User interrupt routines must have been connected to one of the interrupt locations in memory (locations /0 to /70). That is, the address of the interrupt routine must be placed in one of these locations, which at the same time determines the priority level of the interrupt routine. i.e. the interrupt routine whose address is loaded in location /0 has the highest priority (0).
The interrupt routine address may be loaded into this location in several ways. One of them is to link-edit and include the routine with the rest of the system modules at system generation time.

When an interrupt occurs, the P-register and PSW of the interrupted program are stored by hardware in the system stack pointed to by the A15 register (see Ch.3) and the system is put in inhibit mode.
Then the interrupt routine receives control and from within the routine the user may store any other registers by software, if he wishes. The interrupt routine may now continue in inhibit mode, or if the user decides that other interrupts must be able to overrule the current one, he may set the system to enable mode by giving an ENB instruction. (Note: If an INH instruction immediately follows the ENB instruction, a dummy instruction must be inserted, because external interrupts are scanned every two instructions. This dummy instruction may, for example, be another ENB, so the correct sequence becomes: ENB-ENB-INH). This, however, entails a substantial difference in the handling of the system stack. If the routine runs in inhibit mode from beginning to end, any of the registers A1 and A14 can be used, provided the user first takes care of storing old contents in the A15 stack and restoring them at the end of the routine. This may, for example, be done as follows:

```
STR     A1,A15          (On P856/7 and when the simulation rou-
STR     A2,A15          tine for multiple load/store has been se-
STR     A8,A15          lected at sysgen for P852, the sequence is:
                        MSR 8,A15
  coding                  coding
                        MLR 8,A15
LDR     A8,A15          RTN A15     )
LDR     A7,A15
LDR     A1,A15
RTN     A15
```

This is the case of an interrupt routine in inhibit mode with a normal return via the A15 stack. The RTN via A15 results in an automatic enable.

However, if other interrupts are to be enabled during a routine or the user makes an absolute branch from the interrupt routine to the dispatcher (ABL M:DISP; for dispatcher address: see CVT), he must take care that before the ENB or ABL instruction is given, the A15 stack contains only P, PSW and registers A1 to A8 inclusive, because on this basis the A15 stack is handled.

Conventions

- Interrupt routines must start by saving the old contents of the registers to be used in the routine.
- Before returning via A15, the old register contents must be restored.

  If a branch is made to the dispatcher, the stack must contain P, PSW and register A1 to A8, so any other registers used, must have been restored before making the branch.
- In case of an interrupt routine for internal interrupts (LKM/ stack overflow, real time clock, power failure, control panel), an RIT instruction must be given at the beginning of the routine, to reset the interrupt. See Volume II.

SOFTWARE LEVEL PROGRAMS

User programs run on one level: software level 62.

Level 63: Idle Task

Level 63 is reserved for the idle task, an instruction sequence to use up idle CPU time. It is memory resident and consists of 2 instructions:

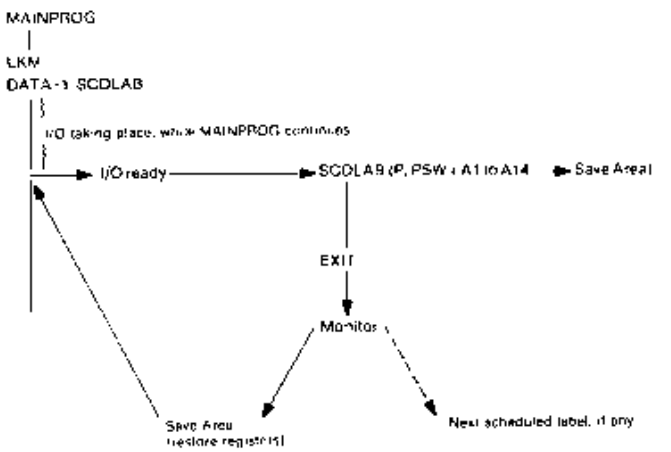
```
RF    *+2
RB    *-2
```

SCHEDULED LABELS

The scheduled label is a feature which allows the user to do a sort of multi-tasking by attaching a routine to a monitor request. To this end, the user specifies the monitor request as having the two's complement of the DATA number indicating the monitor function which is to be executed, followed by the label of the routine which must be executed upon completion of the request. For example:

```
normal I/O request:      with scheduled label:
  LDK   A7,CODE            LDK   A7,CODE
  LDKL  A8,ECBADR          LDKL  A8,ECBADR
  LKM                      LKM
  DATA 1                   DATA -1, SCDLAB
```

In this case, the routine SCDLAB is to obtain control on completion of the I/O monitor request specified. When a scheduled label is attached to an I/O request, one should not set the wait bit, so that the program can continue concurrently with the I/O operation requested. When that operation is finished, control is passed to the SCDLAB routine, and the P-register, PSW and registers A1 to A14 are stored in the 16-word save area (SAVADR) in front of the user program. (When entering a scheduled label routine, only the value of A8 (ECBADR) is significant.) When the routine is finished and exits, control is returned to the monitor, which passes control to a possible following scheduled label routine, or back to the main program by restoring the registers and PSW from the save area. This can be illustrated as follows:



Any number of scheduled labels may be given in a program. However, it is possible that one scheduled label is blocking execution of another one because it is active, i.e. using central processor time. In such cases the address(es) of the queued scheduled label routines are temporarily stored in a table (FILLAB). The maximum number of scheduled label addresses which may be stored in this table at any one time is the number defined at sysgen. This is the only restriction. Queued scheduled labels are treated on a first-in-first-out basis.

Note:

Although it is possible to give a Wait monitor request within a scheduled label routine, this is not normally recommended, for it blocks the whole program.

Example:

This example illustrates how scheduled labels are queued in the FILLAB table, when their execution is blocked in case of an I/O operation on a slow device.

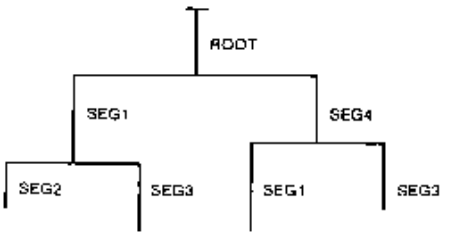
In a program there are three monitor requests for output: onto tape punch, typewriter and line printer. To each of these requests a scheduled label is attached. Each of these scheduled labels requests output on the typewriter. In such a case, it will be evident that the line printer output will be finished first and thus that the scheduled label attached to that request will receive control first. Now, when the other two output operations in the main program are finished, the scheduled labels attached to them will be queued in FILLAB, for they are also requesting output on the typewriter which is still busy with the last scheduled label. When that one is finished control is passed on the last one entered in the FILLAB table.

Note, that in this case the third scheduled label is the first to receive control, because the I/O to which it was attached was for line printer and terminated before the other two.

```
        IDENT SLAB
BUF1    DATA '1234567890'       SPECIFYING THE CHARACTERS TO BE
BUF2    DATA 'ABCDEFGHIJ'       OUTPUT
BUF3    DATA 'ZYXWVUTSRQ'
BUF4    DATA '1A2B3C4D5E'
BUF5    DATA 'BUF5'
BUF6    DATA 'BUF6'
DCB1    DATA 5,BUF1,5,0,0,0     DECLARING THE EVENT CONTROL
DCB2    DATA 5,BUF2,40,0,0,0    BLOCKS FOR THE OUTPUT OPERATIONS.
DCB3    DATA 3,BUF3,5,0,0,0     5 IS THE FILE CODE FOR TYPEWRITER,
DCB4    DATA 2,BUF4,12,0,0,0    3 FOR PUNCH. 2 LINE PRINTER.
DCB5    DATA 5,BUF5,6,0,0,0
DCB6    DATA 5,BUF6,6,0,0,0

START   EKD   A7,6              MAIN PROGRAM STARTS AND REQUESTS
        LDKL  A8,DCB1           STANDARD OUTPUT OF BUF1 ON THE
        LKM                     TYPEWRITER. SCHEDULED LABEL SLAB1
        DATA  1,SLAB1           ATTACHED THIS REQUEST.
        LDK   A7,6              MAIN PROGRAM REQUESTS STANDARD
        LDKL  A8,DCB3           OUTPUT OF BUF3 ON THE TAPE PUNCH.
        LKM                     SCHEDULED LABEL SLAB2 ATTACHED
        DATA  1,SLAB2           TO THIS REQUEST.
        LDK   A7,6              MAIN PROGRAM REQUESTS STANDARD
        LDKL  A8,DCB4           OUTPUT OF BUF4 ON THE LINE PRINTER.
        LKM                     SCHEDULED LABEL SLAB3 ATTACHED
        DATA  1,SLAB3           TO THIS REQUEST.
        LKM
        DATA  3                 MAIN PROGRAM EXIT

SLAB1   LDK   A7,6              SLAB1 REQUESTS STANDARD OUTPUT
        LDKL  A8,DCB2           OF BUF2 ON THE TYPEWRITER
        LKM
        DATA  1
        LKM
        DATA  3                 AND EXITS
SLAB2   LDK   A7,6              SLAB2 REQUESTS STANDARD OUTPUT
        LDKL  A8,DCB5           OF BUF5 ON THE TYPEWRITER
        LKM
        DATA  1
        LKM
        DATA  3                 AND EXITS
SLAB3   LDK   A7,6              SLAB3 REQUESTS STANDARD OUTPUT
        LDKL  A8,DCB6           OF BUF6 ON THE TYPEWRITER
        LKM
        DATA  1
        LKM
        DATA  3                 AND EXITS

        END   START
```

PROGRAM SEGMENTS AND OVERLAY STRUCTURE

If a program becomes very large, it may be possible to divide it into segments and execute it in an overlay structure. Each segment, in such a case, must be considered as a self-contained program. These segments are treated as programs: they are given names and they are catalogued with KPF (Keep File) commands, under the same user identification. The user must first decide on the overlay structure in which he will have his programs executed. To this end he sets up a so-called 'tree' of the segments which form his program. This must be done in such a way that those segments which are related to each other, through references, form a path i.e. a sequence of contiguous segments which can be in memory at the same time and start with the first segment (root). This is shown in the example below:



In this illustration we can discern the following paths:

ROOT - SEG1 - SEG2
ROOT - SEG1 - SEG3
ROOT - SEG4 - SEG1
ROOT - SEG4 - SEG3

These segments have been previously catalogued under the names SEG1, SEG2, SEG3 and SEG4 and they are declared as follows:

```
SEG SEG1,SEG2,SEG3,SEG4
RUN ROOT
```

The first control command defines these programs as segments of an overlay structure and the second command starts the execution of the program ROOT, which is the first segment of the overlay structure. ROOT takes care of loading SEG1 or SEG4, which, in turn, load the other segments.

The design of such an overlay structure depends on several factors: the available memory space, the frequency of use of each segment, the relations between the different segments. The root segment is the program that gets control at the start of execution. It contains those program parts which have to be in memory throughout program execution. When a reference is made to a place in another segment, that segment must first be loaded. If another path is required by the root, it overlays the previously loaded segments or part of them.
The same segment may be implemented in different paths in a tree, if it is required by other segments in that path, as shown in the illustration above. Any overlay structure can thus be used, since the user himself is responsible for calling the loader and transferring control to other segments.

The memory area required for the overlay depends on the length of the longest path in a tree. If, in the illustration above, the path ROOT-SEG1-SEG3 is the longest, then at least the area required for those segments is necessary.
At link-edit time, if a map listing was requested (M parameter in the LKE control command), this map will contain the heading L= XXXX, indicating the program length. If the user wants to use Get and Release Buffer requests, he must reserve an a area, in the root, equal to the combined Lengths of all the segments in the longest path of his overlay structure. This also goes for the blank common. Buffers will be allocated after the last word loaded into memory when the root is started through a RUN command.

When a segment is loaded, it is always loaded at its loading address + 8, because the Linkage Editor generates four words preceding it and fills them with:
- Start address
- Number of sectors in the program
- Length of the program (in characters)

- Beginning address of the symbol table (for the debugging package).

This enables the calling program to branch indirectly to the address of the called program.

Below, an example is given of how a reference is made from one segment to another:

```
ROOT
          LDK    A7,n          Load segment number n
          LDKL   A8,LDAD       Load address
          LKM                  Monitor request to load segment n
          DATA   9
          ADK    A7,0          Test if request accepted:
          RF(4)  ERROR         No: branch to error routine
          CFI    A14,8+i,A8    Yes: Call entry point number i in
                                    segment n
SEGMENT N
             1st data word     DATA Entry point 0
                               DATA Entry point 1
                               DATA Entry point 2
             Entry point i     EQU  *
                               RTN  A14   return to calling program via
                                          register A14
```

CATALOGUED PROCEDURES

A catalogued procedure is a sequence of CCI control commands stored on disc under a procedure name. These commands are executed when the procedure is called from the input stream. It is possible to use or not use certain parameters of the commands in the procedure.

Catalogued procedures are kept in a special file named M:PROC, with file type UF. For each <userid> , one M:PROC file can be used. The M:PROC file may contain several procedures, each of which must begin with the procedure name (the first character of which must be $) and be terminated with an END command:

```
$PROC1
<commands of procedure 1>
END
$PROC2
<commands of procedure 2>
END
:EOF
```

As shown, the last procedure in the M:PROC file must be followed by an :EOF.
The first line of a procedure is the name used to identify the procedure call. The end of the procedure is indicated by the END command, so input commands will be read from the normal input stream.
Each user can update his M:PROC file without any difficulty. Adding new procedures is also done by updating the M:PROC file.

A call for a catalogued procedure is made by specifying the procedure name in the input command stream on the device with file code /EO. The Control Command Interpreter (CCI) first checks if it is the name of a standard CCI control command. If it is not, it assigns file code /DO to the file M:PROC of the current <userid> and starts looking for the name used to call the procedure. If it is found, the catalogued procedure is called. If it is not found, the CCI will assign file code /DO to the system M:PROC file and look for the procedure name there. Having found the procedure, the CCI will create a disc file with code /EE containing all the commands to be executed for the procedure.

Parameter Use

The use of parameters is allowed in the procedure body, but not in the first line (procedure name) nor in the END record. Comments are not allowed in these lines either.
In the catalogued procedure body, the parameter may appear in the first field (command name field) or in the second field (parameter field) of a line. The following forms of parameter specification are allowed:

- @⟨param⟩

  where ⟨param⟩ is a character string of up to 4 characters, identifying the parameter name. If this name is not specified after the name of the procedure called at execution time, no parameter will be used.

  For example, if procedure $PR1 is catalogued as:

```
$PR1
RDS
FRT /S,@NLST
LKE
RUN
END
```

  and is called as: $PR1

  it will cause compilation as: FRT /S

  but if called as: $PR1 NLST=NL

  compilation will be done as: FRT /S,NL

- @⟨param⟩ = default value

  where the default value will be taken if the parameter is not specified.

- @⟨param⟩ =

  where the whole line of the procedure will be ignored if the parameter is not specified.

  For example:

  A procedure $PR2 is defined as:

```
$PR2
@CM1=
LED @PNAM=
@CM2=FRT /S
INC @MOD=
LKE
RUN
END
```

  If this procedure is called as: $PR2 CM1=RDS

  the following commands will be executed:

```
                                    RDS
                                    FRT /S
                                    LKE
                                    RUN
```

If it is called as: `$PR2 CM1=RDS, CM2=ASM`

the procedure will be executed as:

```
                                    RDS
                                    ASM /S
                                    LKE
                                    RUN
```

If it is called as: `$PR2 PNAM=PRGRAM,MOD=MAIN`

The procedure will be executed as:

```
                                    LED PRGRM
                                    FRT /S
                                    INC MAIN
                                    LKE
                                    RUN
```

Note:

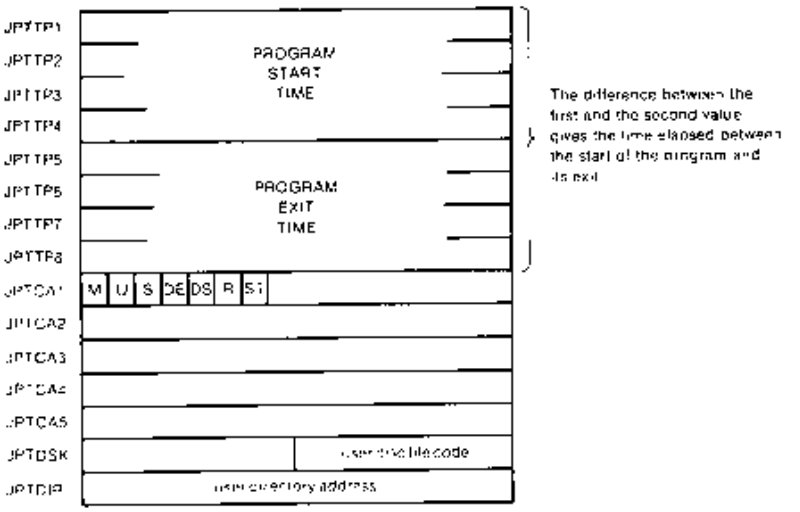
- all the parameters in the procedure call are keyword parameters, separated by commas.
- in the procedure body, parameters may be separated by spaces or commas: spaces are used to separate the command field from the parameter field; between parameters, only commas are allowed.
- in PSE and MES commands used in a catalogued procedure, no blanks are allowed within the message.
- Input commands for processors, e.g. LED are not allowed within a catalogued procedure but must be done on another file (e.g. /E0).
- when, from within a catalogued procedure, another catalogued procedure is called, no return is made to the first (calling) procedure.
  If the second (called) procedure is not found, an immediate return is made to the first procedure.
- the user can not use a catalogued procedure to open a new session, for the commands JOB and BYE implicitly cause the END of the procedure.
- the SCR command may not be used inside a catalogued procedure, except as SCR /S or SCR /O.

Error Messages:

ERROR IN PROCEDURE DEFINITION
ERROR IN PROCEDURE GENERATION (procedure is not correct)
PROCEDURE NOT CATALOGUED (procedure unknown in M:PROC file)
M:PROC NOT CATALOGUED
I/O ERROR

JOB PARAMETER TABLE

The Job Parameter Table consists of a 5-word communication area, followed by a certain number of words used by the system. The layout is as follows:



JPTCA1 to JPTCA5 are used by the system processors as follows and refer to the CCI commands ASM, LKE, FOR, FRT and their options:

Assembler: M = 0, if NL is present in the control command
M = 1, if NL is not present in the control command
The other bits are not meaningful.

Linkage Editor: M = 1, if the parameter M is present in the command
M = 0, if the parameter M is not present in the command.U and S are set as follows:

if N is present in the command, U = S = 0
if S is present in the command, U = 0 and S = 1
if U is present in the command, U = 1 and S = 0
if neither U nor S is present, U = S = 1
DE = 1, if the parameter DE is present in the command
DS = 1, if the parameter DS is present in the command
ST = 1, if a start address has been specified in the command
JPTCA2 contains the binary value of the common displacement.
If there is no common, this word is reset to zero.
JPTCA3 to JPTCA5 contains the entry point name of the start address, if it has been specified.
If the name consists of fewer than 6 characters, it is filled up with blanks and left justified.

FORTRAN Compilers:

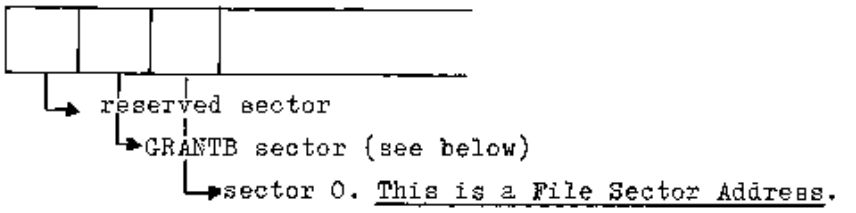
M = 1, if NL is not present in the control command
M = 0, if NL is present in the command
R = 1, if the parameter R is present in the command
R = 0, if the parameter R is not present in the command
The other bits are not used.

To understand this chapter it is necessary to define some concepts first.

The disc organization is based on sectors, which are sections of a track with a length of 205 words, of which 200 are usable.

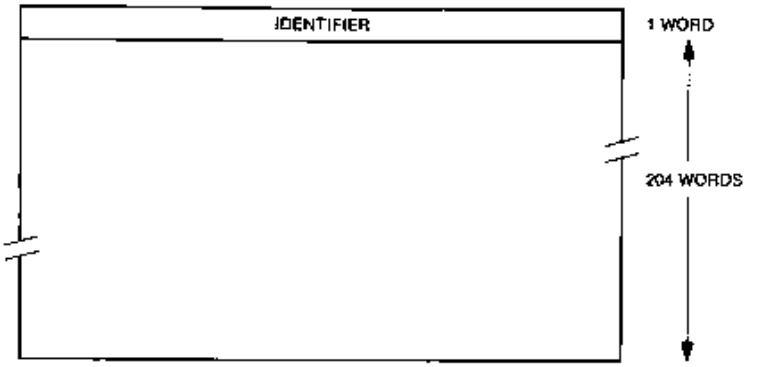
On the basis of this sector concept, the following definitions must be kept in mind.

- Disc Sector Physical Address (DSPA):
  the physical address of a sector on the disc, i.e. the address of a sector in a consecutive arrangement.
- Disc Sector Logical Address (DSLA):
  the address of a sector on the disc as calculated by interlacing, which is ordering the sectors in such a way as to make access as efficient as possible (see below).
- File Sector Address (FSA):
  the address of a sector inside a logical file as managed by Disc File Management (DFM).
  Inside a file, the sector numbering starts at the third sector:

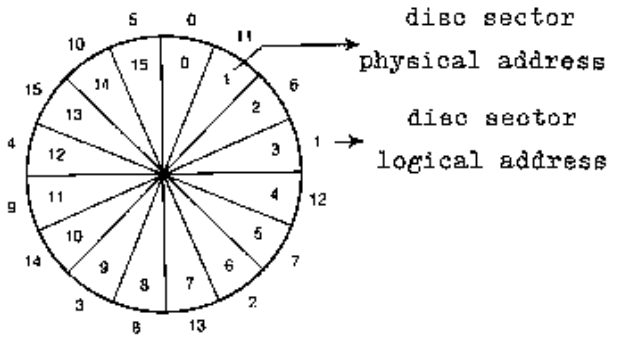


SECTORS

The basic unit in the organization of all types of disc used is a sector. All access operations take place at sector level. The sectors are numbered for 0 to n, consecutively. They can be accessed directly from any program through the I/O driver. A sector has the following format:



The identifier is written in the first word of every sector by the DISC PREMARK program. For moving head discs it contains the number of the cylinder in which this sector is located. For fixed head discs its value is not significant. The sector identifier is used by the system to check if a seek operation has been successful or not. The first word is set by the driver when a sector is written onto the disc. This word must not be modified during the I/O transfer, or a 'seek error' status may be returned later, when the same sector or another one in the same cylinder is accessed.
Although physically the sectors on a disc are numbered consecutively (Disc Sector Physical Addresses), for logical handling they are numbered in a different manner (disc sector logical addresses). This is done to give a requesting program enough time to process the current sector before requesting the next one in a sequential process.
For these logical sector addresses the sectors are interlaced, on a factor -3 basis for discs with 8 or 16 sectors per track.



16 sectors

The disc sectors are always, except in one case, handled according to their logical addresses, e.g. all Data Management operations take place on this basis and when a disc dump is made, the sectors are dumped in their logical order. Only with disc error messages is the sector address indicated the physical address.

FILES

We have seen that the term file sector address takes into account that the first two sectors of a file are reserved: one for file header and one for a table called GRANTB, so that sector addressing starts with the third sector, which gets file sector address 0. Space allocation on the disc for a file is done on the basis of granules, where each granule is an area of 8 consecutive disc sector logical addresses, i.e. 8 logically consecutive sectors. A file is always stored on an integer number of granules, so one granule cannot be shared by two or more files. The system allocates as many granules as necessary to a file which is being written. These granules are chained and attached to the file code assigned to that file.
The addresses of the granules allocated to a file are kept in the table GRANTB in the second sector of the first granule of that file.
The sector GRANTB is initialized when a file code is assigned.
GRANTB is filled with the addresses of the allocated granules, any remaining words being set to zero.

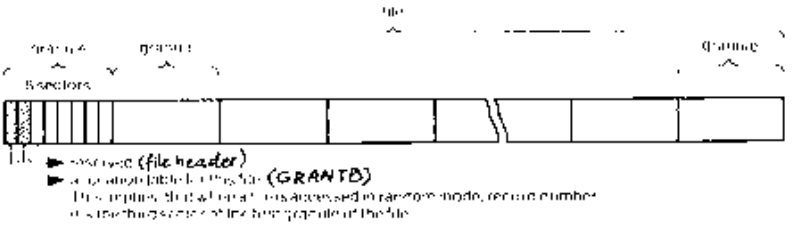
For sequential access, when an attempt is made to write onto a granule which has not yet been allocated, a new one is allocated and GRANTB is updated. The system manages a table called BITAB which contains one bit for each granule on the disc. This table is copied into memory when a disc pack is loaded and updated in core and written back onto disc when a Keep File command is given. A bit is set to 1 when the corresponding granule has been allocated and it is 0 when the granule is still free. Allocation takes place via this table, after which the granule address is stored in GRANTB.

GRANTB		
	0	IDENTIFIER
	2	LENGTH
	4	ADDRESS OF GRANULE 0 OF THE FILE
	6	ADDRESS OF GRANULE 1 OF THE FILE

In this table, location 2 (i+2) is the address of the ith granule. If the granule has not yet been allocated, the location contains the value zero.
The granule address is a binary value from 0 to n, representing the relative sector address, from the beginning of the disc, of the first sector of the granule. GRANTB is 200 words long, so the file length is limited to 320k words (200x8 sectors).

Note: Random access files of more than 200 granules are possible, but when such a file is deleted under DOM, only the first 200 granules are deleted.

Finally, a file is organized as follows:



Inside the file, the useful area in a sector is 200 words (see Record Format).

At the start of a session with the DOM, allocation begins from the first granule available and new granules, if any, are added in ascending order (higher sector addresses). No backward search is done to take into account any granules which may have been deallocated again, e.g. after deleting a file. On the other hand, a specific command provides the possibility to start allocation at the first granule (SCR: Scratch control command). The alloca-

tion table BITAB stored on the disc is updated only when a file is made permanent (KPF control command) and when a file is deleted (DEL control command).
Allocation is done for temporary files only when they are written. So, a file which has been made permanent (KPF control command) cannot be extended directly. Updating a sequential file is done in the normal manner by copying it through the Update processor (Line Editor). Updating a permanent file in random access can only be done directly if no extra granule is required (see Data Management).

The first two granules on a disc are always reserved and used by the system for catalogues and libraries.

ACCESS MODES

Files can be accessed in two ways: random and sequential, each access mode requiring a different organization of the file.

Random

This type of organization has the following characteristics:

- records are of fixed length: one logical record=one physical record=one disc sector.
- the records in the file may be organized in any manner. Access takes place according to the file sector address, i.e. by the relative position of the record in the file (not counting the first two sectors, which are the file header and GRANTB). The logical sequence of the records is not identical to the physical sequence. Such a file is a keyed file, the relative number of the record (sector) being the key.
- when a random file is created under DOM, one granule is allocated at a time, so extension of the file is possible as long as the file has not been made permanent (KPF control command).
- records are accessed directly by specifying the file sector address.
- records may be retrieved, updated and restored individually.
- random access files should be Assigned, Kept and Deleted under DRTM, not under DOM. See Note on page 32.

Sequential

This type of organization has the following characteristics:

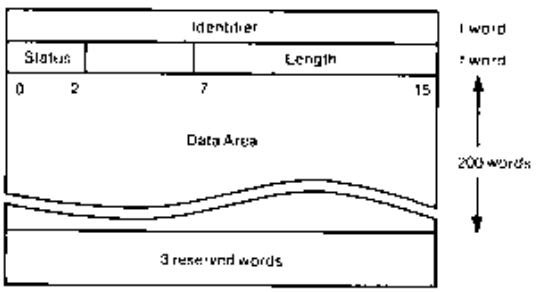
- records may be of any length, up to 16k words. The sector format of such records is described below.
- the only relation between the records in a sequential file is their sequence. The records of such a file must be presented in the order in which they must be written onto the disc, i.e. the

logical sequence of the records is identical to the physical sequence.
- to retrieve a record, the whole file must be scanned up to the desired record.
- to update a record, the file must be processed as a whole.

Additional details on Access Mode are given in chapter 7, Data Management.

Record Formats

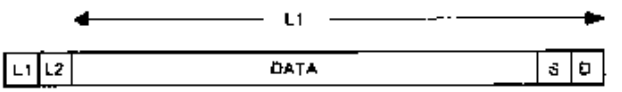
The format of physical record (sector) has already been described (see SECTORS). This is the record format for random files.
In sequential files a sector may contain a logical record or it may be part of a logical record. Sectors have the following format in these files:



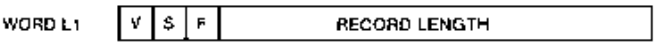
- status consists of 3 bits:
  bit 0: if 1, this sector has been deleted
  bit 1: if 1, an EOS (End-of-Segment) record has been written in this sector. This record is the last one in the sector, for the first record following an EOS always starts at a new sector.
  bit 2: if 1, EOF (End-of-File) record has been written in this sector. An EOF record requires a full sector without any other records in it. So when an EOF is written for a file, first the current sector buffer is written, if necessary, and then an additional sector for the EOF record.
- Length: specifies the length, in characters, of the used area of this sector.

- the data area contains data written in either sequential or random access mode.
- the last 3 words are not used.

The logical records in sequential access files are always compressed and blocked by the system to save disc space (trailing blanks are removed). The format of these logical records is as follows:



L1 is the length of the record, including the 2 words S and D, but excluding L1 and L2:



V = 1: this record has been deleted from the file
S = 1: this record is an EOS record
F = 1: this record is an EOF record

L2 is the initial record length, in characters. This is the requested length recorded in the ECB when the I/O request for writing this record was made.

S is the file sector address within the file containing the first word of the record.

D is the displacement in characters in the sector S, of the first word belonging to the record.
Data is up to 1600 words long (one granule), trailing blanks removed.

Special Records

There are some special records in sequential files, which have the following format (cf. Record Format) (values in hexadecimal):

:EOS: 4004 - 0 - S - D
(4004 = L1)

:EOF: 2004 - 0 - S - 4
(2004 = L1; 4 = D)

Blank card: 0004 - 0050 - S - D
(0004 = L1; 0050 = L2) (hexadecimal 50=decimal 80=card length)

Note: Records on disc always consist of an even number of characters. So, whatever the value of L2 given by the user at creation time, L1 always represents an even number of characters, because when the requested length is an odd value, a dummy character is added to the record. An EOS is always stored in one sector and must be the last record in the sector.
An EOF is always written in a separate sector.

FILE TYPES

Under the DOM, the following four file types can be distinguished:

- Source file (SC)

  Source files are sequential files input in source language or after an update of the source language. These files are used as input to one of the language processors.

- Object Files (OB)

  An object file is a sequential file with one record per object cluster. Each object module is followed by an EOS record. A new object module starts at a new sector. The final object module in an object file is followed by both an EOS and an EOF record.
  As an object module is not a file, it need not necessarily be stored on an integer number of granules. Therefore deletion of an object module need not result in deallocation of a granule. However, in such cases a flag is set in the second word of every sector of the deleted module (See Sector Format: Status). When an object file is read sequentially, the Data Management routines will automatically skip any deleted sectors.

- Load Files (LM)

  A load file is accessed in random mode. Each full sector of such a file contains 188 code words and 12 control words for relocation bits (see Linkage Editor).
  The first four words of a load file contain the following information:

  - start address of the load module
  - number of sectors in the load module
  - memory length required
  - address of entry points table (for debugging functions).

- Undefined Files (UF)

  This type comprises all other files, such as data files.

Load Module Size

All system or user programs must be kept on disc in the load format generated by the Linkage Editor or at system generation.
The number of granules required for keeping a program is
((S-1)/185+2)/8+1
S being the program size in words. Thus, the following table gives the maximum program size for a number of granules:

Number of Granules	Maximum Program Size (Words)
1	1128
2	2632
3	4136
4	5640
5	7144
6	8648
7	10152
8	11656
9	13160
10	14664
11	16168
12	17672

DISC STRUCTURE

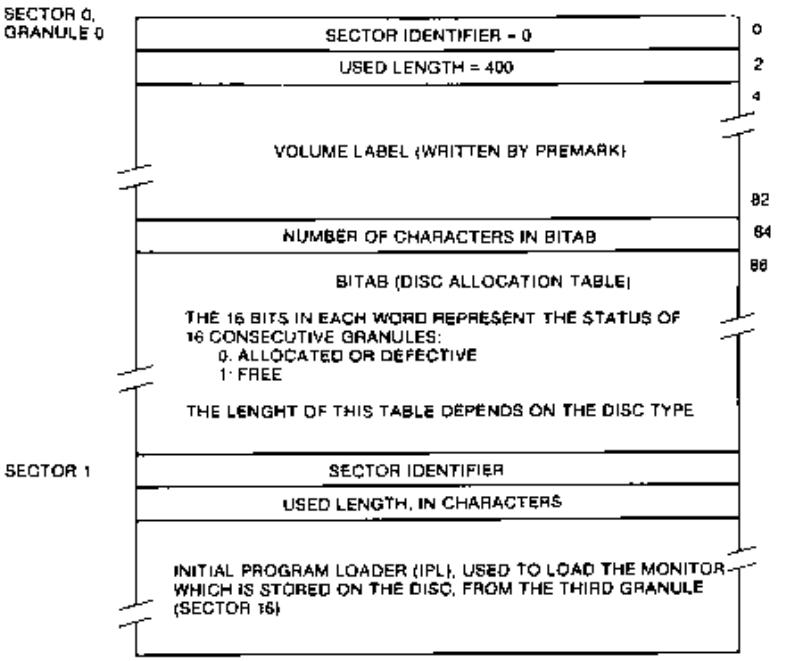
Catalogue and Library Structure

On a disc, the Catalogue contains one entry for each user identification declared on that disc. Each of these entries contains a pointer to a library, one for each user in the Catalogue. Each user library consists of a directory and the files in the library.

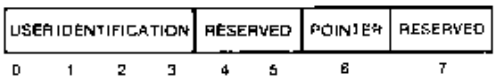
Catalogue Structure

The first granule on a disc contains a catalogue of the users of this disc. Its layout is as follows:

- Sector 0: Volume label and disc allocation table (see Premark: Appendix B).
- Sector 1: IPL (Initial Program Loader).
- Sectors 2 to 7: Catalogue.

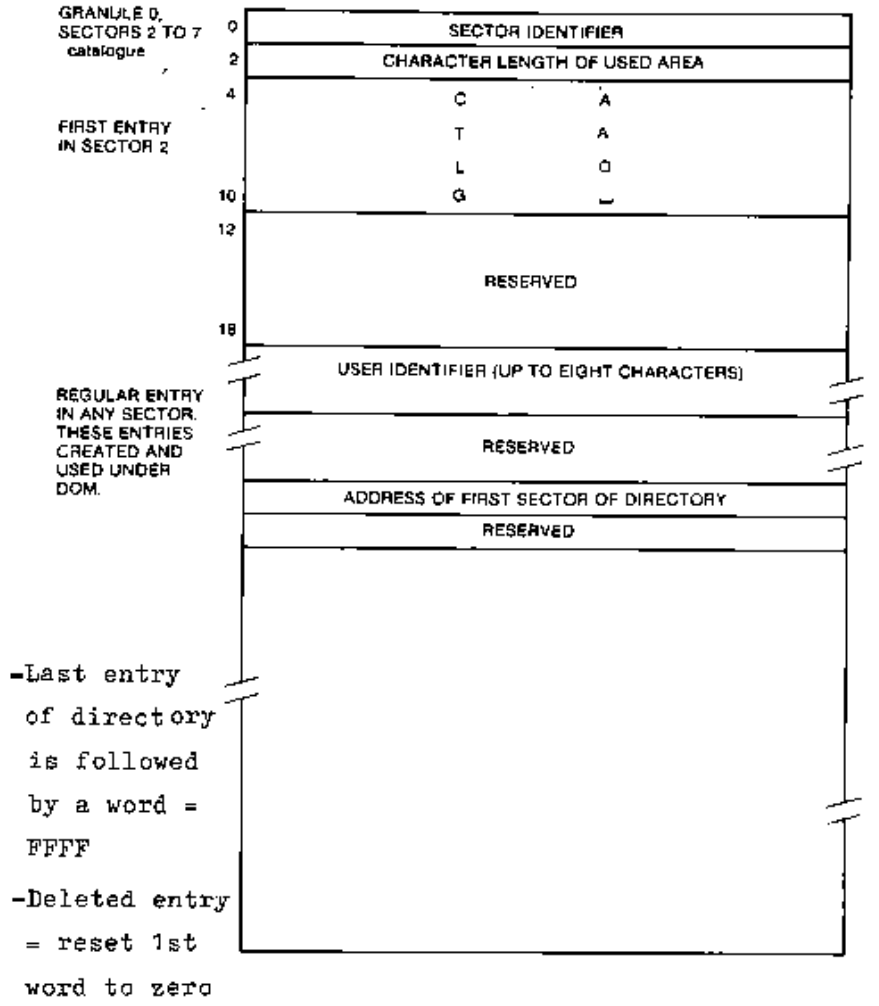


The Catalogue consists of entries occupying 8 words each; each entry relates to a user who has been declared for this disc (Declare User control command: DCU). An entry has the following format:



- words 0 to 3 contains the user identification, as declared in the DCU command
- words 4 and 5 are not used
- words 6 contains a pointer to the user directory; it is the disc sector address of the granule containing the user library directory.
- word 7 is reserved.

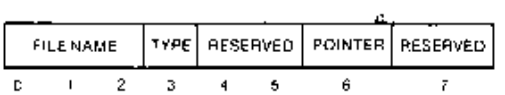
If the user identification is SYSTEM, the value of the pointer in word 6 is 8, because the system directory always occupies the second granule on the disc.
The Catalogue may contain up to 150 entries (6 sectors, 25 entries each).
When an entry has been deleted, the first word is filled with /0000.
The last entry in the catalogue is followed by a word containing /FFFF.



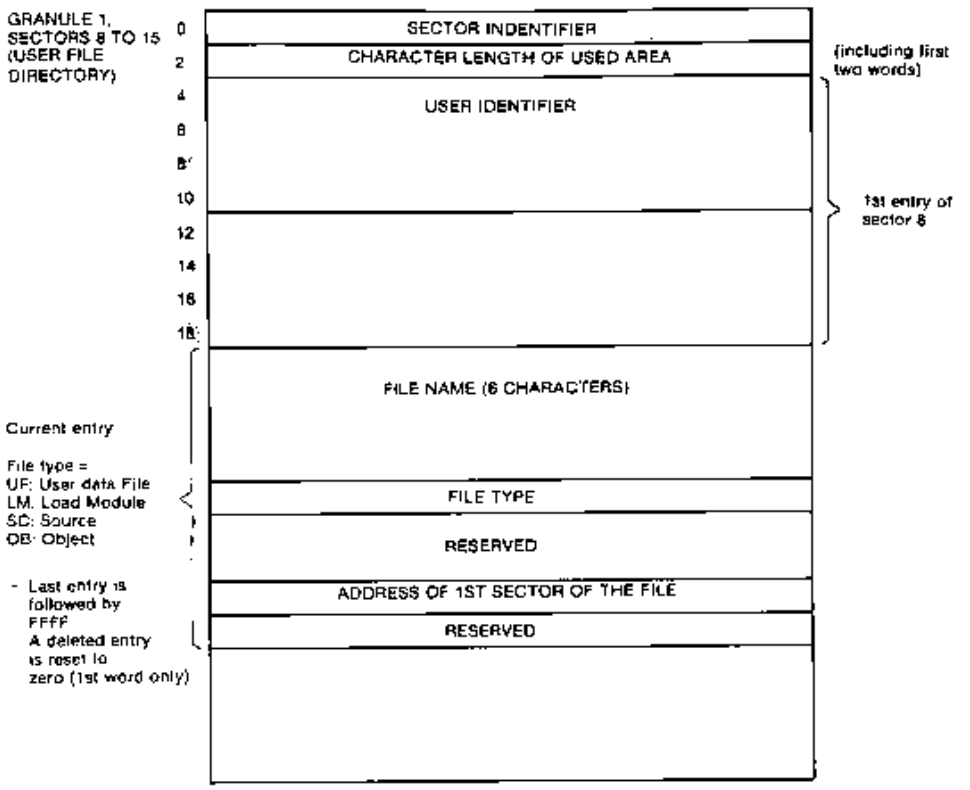
Format of user catalogue (sectors 2 to 7) -

Directory Structure

Each user is provided with his own directory and library. The directory occupies one granule and contains the names of and pointers to the user's files. The granule containing the user directory may be located anywhere on the disc, except when the user is SYSTEM. In this case it is the second granule on the disc.
Each entry in a directory consists of 8 words:



- words 0, 1 and 2 contain the file name
- word 3 contains the file type, which may be one of the following:
  . for source files: SC
  . for object files: OB
  . for load files: LM
  . for undefined files: UF
- word 6 is the file pointer; it contains the disc sector address of the first granule of the file
- words 4, 5 and 7 are reserved.



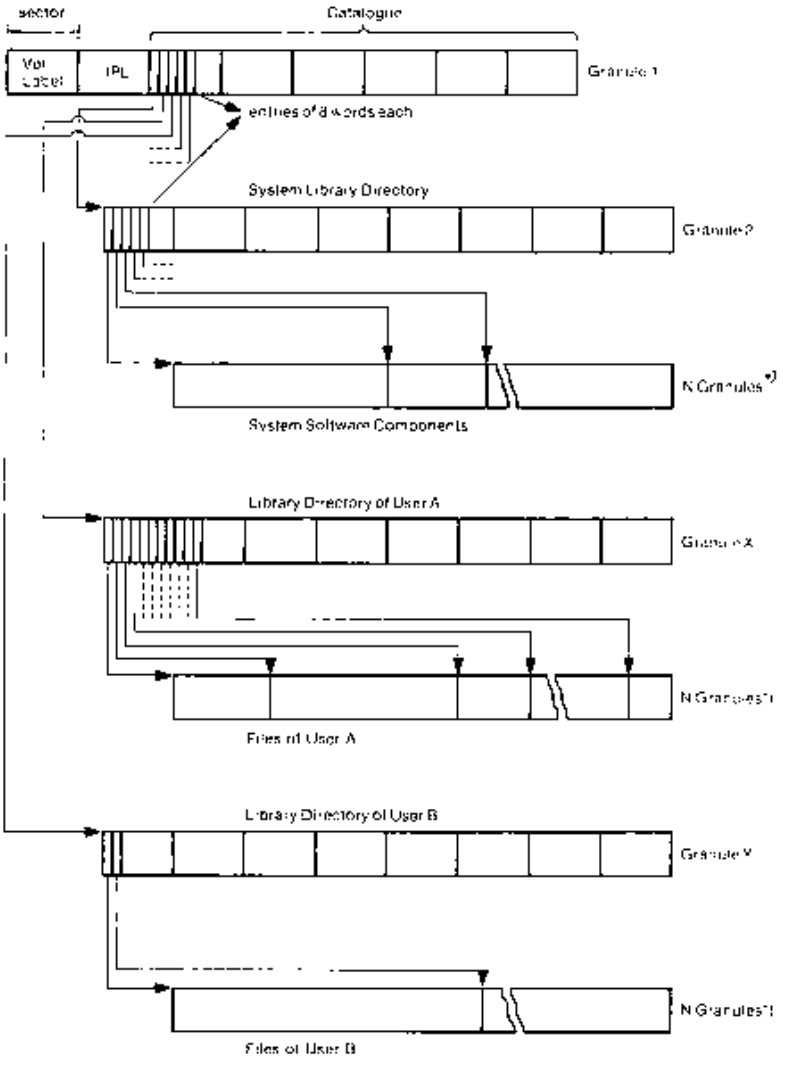
Format of Directory

A user directory may contain up to 200 entries (8 sectors of 25 entries each).
Each entry points to a granule table (GRANTB), containing the successive addresses of the granules allocated to the file represented by the entry in the directory. GRANTB may contain up to 200 granule addresses.

When an entry has been deleted, the first word is filled with the value /0000.
The last entry in a directory is followed by a word containing the value /FFFF.

The granule for the user directory is allocated at the time when a new user is declared to the system (DCU command) and entered in the Catalogue.

Below a drawing is shown of the DOM library structure.



*) These granules are not necessarily adjacent

Example of the Structure of Catalogue and Libraries

DOM System Disc Structure

The System Disc contains all the system software, i.e. the monitor, the system processors and the system object library. These software components are on the disc in the same manner as the files of the different users are, i.e. in a user library coupled to a user identification and a directory. For the system software the user identification SYSTEM is used. It is necessary that the first file in the library of the 'user' SYSTEM is the monitor and that all the granules it occupies are adjacent. The format of the monitor is the same as that of load modules and it may be catalogued under name and type.
The other SYSTEM software components must be present in the directory of this first user identified by SYSTEM, but the granules they occupy need not necessarily be adjacent. Thus the first granules of the System Disc are occupied as follows:

Granule 0: Sector 0: Volume Label
Sector 1: IPL=Initial Program Loader
Sectors 2 to 7: Volume Catalogue
Granule 1: Directory of the first user, i.e. SYSTEM
Granules 2 to n consecutively contain the Monitor.
The other software components use granules randomly.
Any granules remaining on the System Disc can be used by other users for their files.

All I/O operations are initiated by an I/O monitor request. At system generation time, the necessary tables for fulfilling this request must have been filled and the necessary modules loaded. When the request is given, with an LKM instruction, register A7 must have been loaded with parameters about the particular type of I/O function, while register A8 must contain the address of an Event Control Block which holds the necessary information about the data to be transferred.

There are several types of I/O request (as specified in A7):
- Random I/O requests: for random access I/O operations on disc devices.
- Basic I/O requests: for these requests the monitor will not do any character checking or data conversion, so they are used in case of binary I/O. The monitor handles only the control command initialization and signals the end of the I/O operation.
- Standard ASCII I/O requests: these requests provide more monitor facilities, such as error control characters, data conversion from external code to internal ASCII code and vice versa, character checking for end of data. Characters are stored 8 by 8 bits, two to a word.

Moreover, a number of control functions can be performed through a monitor request, such as writing EOS or EOF records, skipping forward or backwards, rewinding, etc.

In the Event Control Block (ECB), pointed to by register A8, the user specifies the file code (see below) of the device or file concerned with the I/O operation, and additional parameters such as buffer address and buffer length. At the end of the I/O operation, the monitor places information about the result of the I/O operation in this ECB, so that it may be verified by the user program.

For non-disc devices, I/O operations are done at record level, by I/O drivers running at level 48. No blocking- deblocking is performed. For disc devices, the user can use an I/O driver or he can access the disc through the Data Management package. If he uses the driver, he must specify an absolute sector address for the I/O operation. With Data Management, he specifies the relative sector address for direct access and Data Management will find the correct sector, Moreover, Data Management automatically provides additional functions, such as blocking-deblocking.
In a following chapter detailed information will be given about the I/O monitor requests.

DOM LOGICAL FILES AND UNITS

To facilitate use of the system, the control command language allows for implicit addressing of some system logical units and for calling some system temporary files by a predefined name.

Temporary Source File: /S

At creation time, a source file is always temporary. The monitor allocates the necessary granules to this file on the disc which contains the library of the current user.
A source file is sequential.
The disc temporary source file /S can receive a file read from the source input unit or a file which is the result of an updating process done for a library source file. The /S file may then be used as input to one of the language processors.
It is possible to have this file listed, printed or punched (LST, PRT and PCH control commands) or the file may be made permanent by keeping it in a library (KPF control command) for later use.

Temporary Object File: /O

An object file, i.e. an output file of one of the language processors, is always temporary at creation time. Disc space is allocated to it when it is being produced, in the same way as for tem-

porary source files, i.e. by the monitor. The temporary object file /O receives the object modules read from the object input unit, or object modules produced by one of the language processors or selected from the object files of user libraries.
The /O file is sequential.
The /O temporary object file is used as the main input file for the Linkage Editor.
By means of the control command POB it is possible to have this file punched on tape and by means of the control command KPF all or part of its object modules may be placed in a library.

Temporary Load File: /L

The output of a link-edit operation is the temporary load file /L. This file is random file. Disc space is allocated to it by the monitor as for the other temporary files. This file may be executed by means of the control command RUN or it may be stored in a library (KPF control command). It is also possible to have the /L file punched out (PLD command), but only for LKM output, not for OLE output. In this case the format is converted to standard object cluster format. Such a module cannot be directly introduced from the object input unit to the /L file. It first has to be put on the /O file and be link-edited.

Source Input Unit

As defined at system generation time, the source input unit may be:

- card reader
- punched tape reader
- ASR punched tape reader
- cassette tape unit
- magnetic tape unit.or disc, i.e. any input device.

By means of a specific control command (RDS) source programs can be read from this unit and be copied onto disc as a temporary file ready for assembly or compilation, or they may be copied into a library and made permanent (RDS, followed by KPF control command). There is no restriction in addressing the unit from the user program.

Object Input Unit

This unit is also defined at system generation time, and may be one of the following:

- disc
- punched tape reader
- ASR punched tape reader
- cassette tape unit
- magnetic tape unit.

By means of the command RDO, an object module file on punched tape can be read from this unit, to be copied onto disc as a temporary file ready to be link-edited or it may be copied into a library and made permanent for later use (RDO, followed by KPF control command).
Because no object code is punched on cards, this unit may not be a card reader.

Command Input Unit

This is the unit on which the control commands are entered. As defined at system generation time it normally is the operator's typewriter, but it may be any input device.

Print Unit

If a line printer is included in the configuration, this may be defined as the print unit at system generation time, otherwise it will be the print equipment of the operator's typewriter.

Punch Unit

One of the following two may be defined as the punch unit at system generation time:

- tape punch
- ASR tape punch
- cassette tape unit
- magnetic tape unit
- disc, i.e. any output device.

Note:
During a session, new assignments may be defined for different units, but when a system is loaded the assignments are the ones defined at system generation time.

FILE CODES

The following file codes are standard assignments for the logical files and units specified, as incorporated in the Disc Operating Monitor.
Depending on the configuration, different logical units may have the same physical assignment:

01:	user typewriter (answers from CCI S:); used by user program)
02:	print unit
03:	punch unit (output)
04-09:	reserved for peripheral devices
D0:	catalogued procedure input
D4:	/S file or library source file (Line Editor output)
D5:	/O file (ASM output, LKE input)
D6:	/L file (LKE output)
D7:	system object file (library)
D8:	user object file (library)

D3 and D9 to DF are reserved for system use.

E0:	control command input
E1:	source input
E2:	object input
EE:	catalogued procedure output
EF:	system operator's typewriter (system output in response to INT button: M:)

F0 to FF are logical addresses of disc units. These are reserved for system use, and are not to be used by the user.

The file codes 01,02,03,E0,E1,E2 and EF can be used without having been assigned in a previous ASG control command.
The file codes 04 to CF can be used by the user to address his own files and any additional peripheral devices in his configuration, but only after he has assigned these file codes through ASG commands, or, for 04 to 09, at SYSGEN time.
File codes 0A to DF are scratched when an SCR or BYE command (see chapter 9) is given.

ACCESS MODES

Two access modes are allowed for disc files: random and sequential.

- Random access is possible only for fixed length records of 200 words (one sector).
  A record is accessed by means of the record number (file sector address).
- Sequential access is possible for records of variable length of up to 3200 characters (1600 words = 8 sectors = 1 granule).
  The interface for sequential access in read or write mode is the same as for punched tape, cassette tape or magnetic tape, the blocking-deblocking function and blocking buffer allocation being handled by the system.

A file written in sequential mode can be accessed in random mode.
For further details see Data Management in Chapter 7.

Data Management consists of a set of routines to help the user transfer his records between the memory and the peripheral devices, to help him create files of a particular type and retrieve records from these files. The routines are selected and included in the monitor at system generation time.
Data Management is memory resident and runs at level 49. This implies, that a request coming from a program at level 49 can be processed only when this program, or any others connected to level 49, have given a Wait monitor request.
All operations on the peripheral devices take place through file codes, so the user need not know the type and physical address of each device. The system will find this out by translating these file codes with the aid of monitor tables.
There are two types of Data Management, i.e. two ways of writing or reading files:
Sequential Access Method and Direct Access Method.

The user creates a file by assigning a file code to a temporary disc file (ASG control command) and writing onto it by running a program.
Under the Disc Operating Monitor, if the user wants to make the file permanent for later use, he must do so by specifying a KPF (Keep File) control command at the exit of the program, otherwise this file will be scratched at the end of the session or when a scratch command is used. The file is then catalogued and may be consulted any time, after a file code has been assigned to it. Once the file has been made permanent, however, the number of sectors it occupies cannot be changed, because granule allocation is done only when the temporary file is created.
All Data Management operations, i.e. reading and writing a record, writing EOS or EOF, etc., are done by I/O monitor requests in the program for each record, in which the user can specify the access method and the data management function. It is not necessary to call special routines. The mode of access is determined by the first request for a whole run.

At system generation time the user has to define the number of buffers and their lengths for use by the Data Management package. In general 2, or at most 3, buffers of one sector length will suffice. These will then be included in the system to be allocated automatically.

The first sector on each disc contains a disc allocation table (BITAB) in which the status of each granule, free or allocated, is recorded. See page 31
The second sector of each file contains a granule table (GRANTB) with the addresses of all the granules of this file. See page 31
Four types of file exist:
- source (SC): only sequential access possible
- object (OB): sequential and direct access possible
- load module (LM): only direct access possible
- undefined (UF): sequential and direct access possible.

SEQUENTIAL ACCESS METHOD

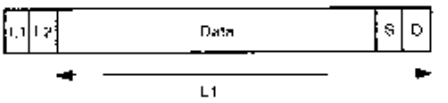
A file is sequential when the only relation between the different records is their sequence. When such a file is created, the records must be presented in the same order in which they must be written onto the disc. To access the file, it must be scanned sequentially until the desired record is found.
The logical sequence in a sequential file is identical to the physical sequence of the records in the file.

Record Structure

User records may contain up to 3200 characters. This implies that a record may be part of a sector or that it may occupy a number of sectors. When these records are written onto disc they are first blocked into a buffer.

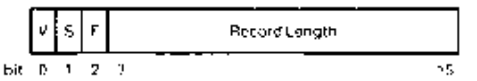
Logical Record Format

In order to save disc space, the system compresses and blocks the logical records used in sequential access; trailing blanks are removed. The format of a record is as follows then:



where:

L1 is the record length, including the words S and D, but not including L1 and L2:



V = not used
S = 1, if the current record is a segment mark (EOS)
F = 1, if the current record is a file mark (EOF)
L2 = the initial record length in characters, as specified in the user's ECB (word 2) in Write mode
S = file sector address of the first word of the record.
D = the displacement in sector S of the first word belonging to the record (number of characters).

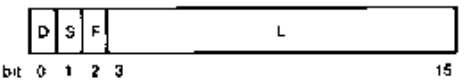
Sector Format

The format of a sector is as follows:

where:

ID is the cylinder identification: a number from 0 to 202, used to check seek operations on moving head discs.

L is the length of area used for data in the sector (0 to 400 characters)



where:

D = 1, if the sector has been deleted from the file.

S = 1, if the current sector contains a segment mark (EOS).

F = 1, if the current sector contains a file mark (EOF).

Special records

Some special records have the following format (compare with - Logical Record Format):

:EOS: 4004 - 0 - S - D
(4004 = L1)

:EOF: 2004 - 0 - S - 4
(2004 = L1; 4 = D)

: Blank card: 0004 - 0050 - S - D
(0004 = L1; 0050 = L2 (hexadecimal 50 = decimal 80 = card length).

Note:

Records on disc always consist of an even number of characters. So, whatever the value of L2 given by the user at creation time, L1 always represents an even number of characters, because when the requested length is an odd value, a dummy character is added to the record.

An EOS is always stored in one sector and must be the last record in the sector.
An EOF is always written in a separate sector.

File Creation and Processing

A sequential disc file is created by the program delivering the logical records with the aid of the Data Management Package. Each record is written by an I/O request up to 'Write EOF'. When this request is encountered the contents of the last blocking buffer are output to the disc and an EOF record is written in the next sector.
When a request is given to write an EOS, the current sector is terminated with an EOS record and the following record will be written at the beginning of the next sector. Before creating a sequential file the user must assign a file code to a temporary disc file. If the user wants to have the file catalogued, he must give a KPF control command before closing the session. Before reading such a catalogued file or writing onto it, an assign command has to be given for it.

Updating

If a user wants to update a sequential file he must first read it, then update it and finally write the updated file on another temporary disc file. Such an updated file can be made permanent in the same way as the original file and under the same name, by means of the KPF control command. If the user gives a new name to the updated file, the original file is not destroyed, which enables him to have several versions of one program belonging to the same Library.

Read a Record

To get a logical record from an input sequential disc file, the user may give a Read monitor request (LKM 1), as for any other device. The system automatically provides the disc buffer, fills it, deblocks the records and recovers any errors. Only the sign-

ificant part of the record will be stored in the record area specified by the user in his ECB, i.e. control words will be removed by the system.
When an EOS or EOF mark is encountered, this is indicated to the user in the Status word of the ECB (word 4). An attempt to read records beyond an EOF mark will cause an EOF status to be returned to the user.

Write a Record

To put a record on an output file may be done by means of a Write monitor request (LKM 1), as for any other device. When the record is moved to the disc buffer it will be formatted with control words, as it consists only of data words in the user area. The system will also take care of buffer allocation, record blocking and error recovery. For temporary sequential files, records can be written onto a file until the maximum number of granules has been allocated (200). After this, an 'End of Medium' Status will be returned in the user's ECB if an attempt is made to write over the number of granules already allocated.

Opening a File

Opening a file need not be requested by the user as this is implicit in the first read or write request.
When the ASG control command is given an entry is made in a file code table and a Logical File Table is built by the system to record information about the file used. This, however, does not imply that the file has been opened yet.

Closing a File

Closing a file is done in write mode, after the last record of a file has been written onto the disc, by giving a Write EOF monitor request. If the user wants to write the contents of the last buffer onto a disc without closing the file, he should give a Write EOS request. This is the case with the common object file created by the language processors, which do not have to close the /O file

as this is done by the system when the Linkage Editor is called:

Positioning a File

It is possible for the user only to position the file at the first logical record. This is done by giving an I/O request with the order to rewind the file.

Data Management Requests

The Data Management package is activated by an I/O monitor request (LKM 1). The function which has to be performed is loaded into the A7 register, while the A8 register must be loaded with the address of an Event Control Block, containing the necessary parameters. The calling sequence is as follows:

```
LDK   A7,CODE
LDKL  A8,ECB
LKM
DATA 1
```

where:

the word CODE is made up as follows:

bit 8 = 1: wait for completion of the event is implicit in the request.

= 0: control will be returned to the calling program after initialization of the operation. To check for completion the program must give a Wait monitor request (LKM 2).

bit 9 is not used, and must be reset to zero. (If it is 1, the status /C010 will be returned).

ORDER contains the function code:

/01,/02: Read a Record (Basic, Standard)

/05: Write a Record (Basic)

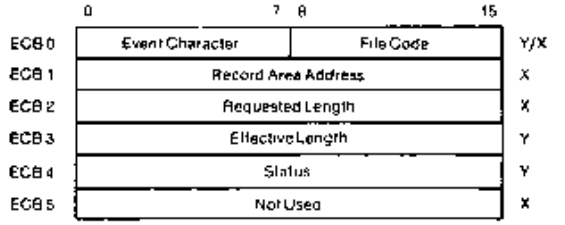
/06: Write a Record (Standard)
/07: Write a Record (Object, 4+4+4+4 tape format)
/08: Write a Record (Object, 8+8 tape format)
/22: Write EOF mark (Close a file)
/26: Write EOS mark
/30: Get information about a file code
/31: Rewind the file.

Notes:

When the order /30 is given, the user must specify the file in ECB word 0; the ASCII characters 'DK' (physical files F0 to FF) or 'DL' (logical files) will be returned in ECB word 1;
the other words will be reset to zero, because disc file codes are handled by the system.

Basic orders (01, 05) and Object Write orders (07, 08) are converted to Standard Read/Write for disc files.

The standard ECB, to which register A8 points, has the following layout:



The words marked X must be filled by the user.
Those marked Y must be reserved by the user, but will be filled by the system.

ECB 0: Event Character: Bit 0 is set to 1 on completion of the I/O operation. The other bits are not used and reset to zero.
File Code: Defines the logical reference to the file.
ECB 1: Specifies the beginning address of the area where the record is stored in memory.

ECB 2: Length in characters of the record area. (Words for basic read on cards).

ECB 3: Number of characters which has actually been moved from or to the record area. (Words for basic read on cards).

ECB 4: This word contains the status returned to the user program by Data Management. See page 127 . In addition, status /10 is returned when an attempt is made to write beyond the last allocated granule of a file (End of Medium) and for temporary files, when over 200 granules are written.

ECB 5: is not used with sequential access method.

DIRECT ACCESS METHOD

When the direct access method is chosen, the records within a file may be organized in any manner and accessing a record may be done at random, by specifying a file sector address from 0 to 1597. This is possible because with this method a logical record is equivalent to a physical record on the disc: one sector. When a direct access file is created, the records may be delivered in any order. The system will create a granule table and allocate granules to the records (sectors) that are delivered. Each granule address is noted in the table, and when the user wants to read a particular record, he specifies the file sector address, upon which the system will be able to find it with the aid of this granule table. Thus, such a direct access file may be considered a keyed file, where the relative number of the record (=sector) is the key. Although the great advantage of a direct access file is that the user can read, write or update individual records without having to scan or copy a whole file sequentially, he may nonetheless want to be able to access such a file sequentially. If this is the case, the individual records must be formatted in the same way as for a sequential file, i.e. the sector format must be the same and the records must be written sequentially and terminated with a 'WRITE EOF' request.

Record Structure

With direct access, a logical record is the same as a physical record: a record is equal to a sector.
The first word contains the cylinder identification (used by the disc driver to check the position of the head after a seek operation). The remaining 204 words may be used for data storage.

6 File Creation and Processing

A file is created when an Assign command or monitor request is given. This reserves the required number of granules on the disc where the records can then be written. Such a file can be made

permanent by means of a Keep File control command.
When a request is made, each record is transferred directly from the user area to the disc.
Only one granule is allocated initially, and the file is extended granule by granule during its creation.
Random access files do not have to be closed by the user.

File Retrieval

Here lies the main advantage of a direct access file, because once the file has been assigned, any sector record can be retrieved, erased or updated and rewritten individually.
The size of a file is fixed at creation time, when a granule table is also written with an entry for each granule of the file. When the file is catalogued by means of a KPF command, creation is terminated and no more granules can be added. Therefore the user must know the maximum size of the file at creation time (no more than 1598 sectors)

Read a Sector

This is done in the same way as for sequential access, the only difference being that the user must specify in ECB5 the relative number of the sector within the file (i.e. the file sector address, a number from 0 to 1597), and specify / A for the read order in register A7. Moreover, he must supply the system with a 205-word buffer in which the physical record will be stored. As mentioned above, the first word contains the cylinder identification.

Write a Sector

This is done in the same way as for sequential files, the only difference being that the user must specify in ECB5 the relative number of the sector to be written into the file (a number from 0 to 1597) and specify / B for the write order in register A7.

Moreover, he must supply the disc buffer in which the information is stored: a 205-word buffer of which the first word will be replaced by the cylinder identification by the system (required by the physical disc I/O driver).

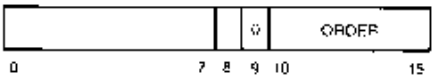
Data Management Requests

The Data Management package is activated by an I/O monitor request (LKM 1). The function which has to be performed is loaded into the A7 register, while the A8 register must be loaded with the address of an Event Control Block, containing the necessary parameters. The calling sequence is as follows:

```
LKD     A7,CODE
LDKL    A8,ECB
LKM
DATA 1
```

where:

the word CODE is made up as follows:



bit 8 = 1: wait for completion of the I/O operation is implicit in the request.

= 0: control will be returned to the calling program after initialization of the I/O operation. To check for completion, the program must give a Wait monitor request (LKM 2).

bit 9 is not used and must be reset to zero. (If it is 1, the status /C010 will be returned).

ORDER contains the function code:

/0A: Read (Random)

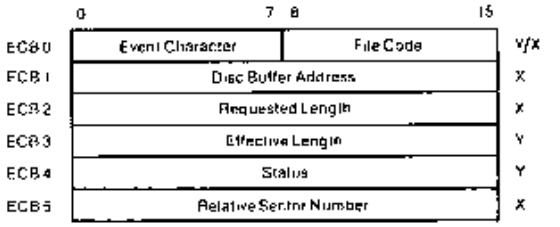
/0B: Write (Random)

/30: Get information about a file code.

Note: When the order /30 is given, the user must specify the file code in ECB word 0; the ASII characters 'DL' will be returned in ECB word 1.

The other words will be reset to zero, because disc file codes are handled by the system.

The standard ECB, pointed to by register A8, has the following layout:



The words marked X must be filled by the user.
Those marked Y must be reserved by the user, but will be filled by the system.

ECB0: Event Character: Bit 0 is set to 1 on completion of the I/O operation. The other bits are not used and reset to zero.
File Code: Defines the logical reference to the file.

ECB1: Specifies the beginning address of the 205-word disc buffer.

ECB2: Whatever the value of the requested length, 205 words will be written on the disc or read into memory.

ECB3: This word is not affected.

ECB4: This word contains the status returned to the user program by Data Management when the sector transfer is terminated. See page 127 . In addition, status /10 is returned when an attempt is made to write beyond the last allocated granule of a catalogued file and, for catalogued files, when over 200 granules are written.

ECB5: Specifies the relative position of the sector within the file, i.e. the file sector address.

Physical Disc Access

This is a direct access on a physical sector level. The Data Management module is not used for this type of access, but special orders are used in the monitor request (LKM1):

/11=physical read
/15=physical write

Moreover, the file code in the ECB must be one of the disc unit file codes /F0 to /FF.
Access is done at sector level, where the sectors are numbered consecutively from 0 to n. This implies that word 5 in the ECB must contain the disc sector logical address of the sector which is to be accessed. The disc which is accessed must not be shared by the user and Data Management, but it may be shared between a number of other programs doing direct physical access. An example of this type of access is 'Dump Disc'.

The initial loading procedure is very simple:
The bootstrap is loaded, either through the toggle switches or by pushing the IPL button on the control panel, then the Initial Program Loader (IPL) is loaded into memory, followed by the monitor.

LOADING BOOTSTRAP AND IPL

The bootstrap can be loaded in one of two ways, depending on whether the optional ROM bootstrap is included in the system or not:

If not, the procedure is as follows:

- switch on the CPU
- load the bootstrap into the first 64 memory locations manually, by means of the toggle switches and the Load Memory button on the control panel. The 64 bootstrap values can be found in Appendix E. Then check by reading these locations out.
- set up the device parameters on the toggle switches, as shown below, and load this value into the A15 register.
- put the disc containing IPL and monitor into the disc drive, push the START button on the drive and wait till the READY button lights
- push the MC button
- load 0 into the A0 register
- push the RUN button on the CPU control panel
- the IPL is now loaded into memory and it loads, in turn, the monitor, after which the monitor initialization phase is started with the typing out of the monitor identification.

If the ROM bootstrap is included in the CPU, which is highly recommended, the procedure is much simpler:

- switch on the CPU
- put the disc containing IPL and monitor into the disc drive, push the START button on the disc unit and wait for the READY button to light

- set up the device parameters on the toggle switches on the CPU, as shown below, and load this value into the A15 register
- push the IPL button on the control panel. This loads the bootstrap into memory, which immediately loads the IPL from disc. The IPL then loads the monitor and the monitor initialization phase is started with the typing out of the monitor identification.

The device parameters on the data switches must be set as follows:

0	1	2	3	4	7	8	9	10	15

where:

bit 0 = 0: tape format used is 8+8
= 1: tape format used is 4*4

bit 1 = 0: the device used is not a disc
= 1: the device used is a disc

bit 2 is used only if bit 1 = 1:
= 0: the disc used is a fixed-head disc
= 1: the disc used is a moving-head disc

bit 3 = 0: the device is connected to the I/O processor
= 1: the device is connected to the programmed channel

bits 4 to 7 are used to qualify the CIO Start command sent by the bootstrap and are transferred to the addressed control unit on lines BIO 12 to 15 when required

bit 8 = 0: the control unit involved is a single device control unit
= 1: the control unit involved is a multiple device control unit

bit 9 = 1: the disc used is an X1215 disc (P824)

bits 10 = 15 contain the device address.

Initial Program Loader (IPL)

The disc IPL program is written onto the disc when the disc is premarked. It is written in absolute binary, so, to enable it to run anywhere in memory it does not contain any memory direct reference.

When loaded, the IPL reads and loads into memory from the disc from which it has itself been loaded, starting from sector number /12 (the first two sectors of a file are reserved for the system). The first four words of sector /12 contain:

- start address of the load module
- number of sectors used by this module
  (This is the standard load format on disc; these words are generated by the disc linkage editor).

Note:
As long as it has been stored on the disc according to the IPL requirements, any stand-alone program, even if it does not use the disc, can be loaded into memory by the disc IPL.

Programs to be loaded by disc IPL
- must be in disc system load format (188 code words plus relocation per sector)
- must be catalogued on disc as a file starting at disc sector logical address /10
- must be built of consecutive granules.

STARTING A SESSION OR JOB

After the system has been loaded by IPL procedure, the operator must initialize the date and time. The system types out
DATE:
and the operator then answers by typing in the date as follows:

DD MM YY (LF)(CR) or YY MM DD (LF)(CR)

where DD,MM and YY are 2 characters giving day, month and year, separated by a delimiter, which may be any character on the keyboard. Then the system types

TIME:

and the operator answers by typing in the time as follows:

HH MM (LF)(CR)

where HH,MM are 2 decimal characters specifying hour and minute. The control panel key must be in CLOCK position, if the time must be updated automatically.

After the user has entered the time, the monitor types out:

BATCH PROCESSING? to which the user can reply with:

Y,<DNDA> if he wants to work in batch processing mode, where <DNDA> (device name + device address) is the new assignment for file code /E0; or

N (Y)

which will be followed by system output of

S:

after which the user can assign /E0:

AS␣<DNDA>, /E0

N if he wants to work in conversational mode.

If batch processing mode is chosen the system starts reading the control commands (on cards or punched tape) immediately. The first of these commands must be a JOB command.

If conversational mode is chosen, the system reads the user identification from the device with file code /E0. This will usually be the typewriter, so the system types out:

USERID:

and the user types in his user identification in one of the two following ways:

/<disc number>,<userid> or

<userid>

In the first case, the system will scan only the disc specified by /<disc number> to find the given user identification.

In the second case, the system will scan the catalogue of each online disc, starting with disc unit F0, until it finds the user identification specified. If SYSTEM is specified as user identification, the first user of disc unit /F0 will be taken, whatever name may be stored for this user on the disc. In such cases we have a SYSTEM session.

If the user is not yet present on a disc and has no entry in the Catalogue he must first declare himself to the system in order to be registered in the Catalogue and to obtain space for a directory for his own library. This must be done in a so-called system session, i.e. after the system message USERID: the user types in SYSTEM and gives a DCU control command with his own user identification. The system will then make an entry for him in the Catalogue and allocate a granule for his library directory. Then the user closes this session with the control command BYE, after which the system again types out USERID: Now the user may start with his own user identification and proceed as he wishes.

Note:

If the user reply to the message USERID: is

SYSTEM

a system session is opened; however, if the user types in

/FO, SYSTEM

the CCI will look for a user named SYSTEM on disc/FO and the session opened is not a system session.

In cases of errors, the following messages may be output:

- INPUT COMMAND I/O ERROR

An I/O error has been detected during the reading of the user identification. The user must type in a new userid on the typewriter.

- I/O ERROR

An I/O error has been detected during the loading of the disc allocation table from the disc into memory. The user must type his userid again on the typewriter.

- USERID UNKNOWN

The userid specified has not been found on any of the discs. The user must type in a new userid on the typewriter.

BATCH PROCESSING

Batch processing is started at system initialization time or when the user switches from conversational to batch processing mode by giving a BYE␣BYE control command.
The first command of a batch must always be JOB. The control command input stream, i.e. the sequence of CCI commands necessary to perform one or a series of jobs, is punched on cards or paper tape or output on disc and executed sequentially, without operator intervention.
The advantage of batch processing is, that when an error or abort occurs, the system simply looks for and starts the next job. When an error occurs, only the commands JOB, BYE and END are recognized and other commands up to one of these three are skipped. Because a job often requires many CCI commands, in more or less the same sequence, these commands can be stored on disc as catalogued procedures, in which case the user will only have to specify the procedure name and any necessary parameters in the input stream.
For this feature see page 23
The following restrictions must be taken into account:
- a program is not aborted when it reads the next job;
- there is no limit on the number of cards read, lines printed or records punched, nor on the execution time.

A batch is started as follows:
- put the batch command sequence on the input stream device (tape or card reader)
- first command must be a JOB command

Note: When a program exits, register A7 contains an exit code in its right character. If bit 8 of A7 is set to 1, in batch mode the whole batch will be aborted up to a following JOB or BYE␣BYE control command. The exit codes are specified under the monitor requests.

CHANGING A DISC PACK

The procedure is different depending on the type of disepending on changed:

- If it is the system disc, changing it must be followed by reloading the system by IPL procedure as described in the previous paragraph.
- If it is not the system disc, as soon as the disc becomes ready, an interrupt is sent to the monitor. Until the Control Command Interpreter is loaded again to re-initialize the system, reading or writing on the new disc is not possible and a non-operational status will be returned.

All file codes which had been assigned to the disc unit which becomes ready, are scratched, for catalogued as well as for temporary files. The codes assigned to the other disc units will not be affected.

Note:
If the user changes the disc pack containing the current session user, the system will close the session by simulating an automatic BYE control command and ask for the next user identification by typing USERID:

COPYING OR UPDATING THE SYSTEM DISC

Two examples are given to show how a copy can be made of the System Disc onto another disc and how a System Disc can be updated.

Example 1
After a new disc has been formatted by means of the PREMARK disc initialization program, it is mounted on the unit assigned to file code /F1. The system is assigned to the disc with file code /F0. /F0 must be copied onto /F1. It is assumed that the first user of

the System Disc (i.e. the system itself) is catalogued as OLDSYSTM. The following command sequence must be given:

USERID: /F1,NEWSYSTM (NEWSYSTM must have been declared during Premark)
SVU OLDSYSTM,/FO
It is possible to replace the command SVU by
MOV
KPF
for each of the system components as they appear in the directory of OLDSYSTM. The monitor must be the first file and be stored on consecutive granules.

Note:
This procedure may be used only once and it must be done on a clean disc, i.e. one which has only been Premarked. If this procedure is done a second time for the same disc, the disc will be destroyed. To replace the supervisor a second time it is better to use the procedure described below, in example 2.

Example 2
It is assumed that disc number 1 contains the new system and that disc number 2 has to be updated. The system to be used is that of disc number 1. So, /FO is assigned to disc number 1, /F1 is assigned to disc number 2. The monitor of disc number 1 is catalogued as MON1 in the directory.

The following command sequence should be given:
USERID: SYSTEM
MOV MON1,/L
RSU /F1
BYE
It is also possible to use the system of disc number 2, in which case the command sequence is different:
/FO is assigned to disc number 2, /F1 is assigned to disc number 1. The first user (i.e. the system) of disc number 1 is catalogued as

NEWSYSTM and its first file is MON2. Then:

```
USERID: SYSTEM
MOV MON2,/L,NEWSYSTM
RSU /FO
```

SYSTEM MESSAGES

Apart from the messages which may result from an error in a control command the following messages may be output by the system:

Abort Messages

When a program is aborted for any reason, messages are output for the user on the devices with file code 01 and/or file code 02, specifying:

- the location where the abort occurred in the program:
  PROG ABORTED AT XXXX
- the reason for the abort:
  NOT WIRED INSTRUCTION
  OVERFLOW IN SIMULATION ROUTINE SAVE AREA
  BUFFER AREA DESTROYED
  TOO MANY SCHEDULED LABELS
  OPERATOR ABORT
  BUFFER ALLOCATION OVERFLOW
  DISK OVERFLOW
  DISK QUEUE OVERFLOW
  MEMORY OVERFLOW DURING LOADING PHASE
- the contents of the PSW and the registers at the moment of the abort.

Peripheral Unit Error Messages

- When an error occurs during an I/O operation, messages are sent to the operator, giving the status of the operation as follows:

  PU DNXX,ST,RY

  where:
  DN is the device name
  XX is the device address
  ST is the device status
  RY invites the operator to correct the error and retry the last operation. In this case the operator must press the interrupt (INT) button on the CPU control panel and type in

  RY XX

  after he has fixed the error or, if the error cannot be corrected,

he may release the operation on that device by pressing the INT button and typing

RD XX

- DKER␣<disc address>␣<cylinder/track/physical sector number>␣<status> is a message to signify a disc error. It provides the address of the disc as well as the cylinder, track and sector number (given in one number) and the hardware status (see Appendix D). If the status is /8000, the disc has become ready, but has not been reinitialized by the Control Command Interpreter.

9 CCI Control Commands

Control commands start with a mnemonic of three letters, followed by one space, possibly followed by one or more parameters.
The parameters are positional and separated by commas.
No additional spaces are allowed.
When the commands are entered via the typewriter, each command must be terminated with **CR LF**
Some of the parameters have a fixed meaning:
<userid> is the user identification, a string of 1 to 8 alphanumeric characters, not starting with / (slash). The first character must be a letter.
<name> is a file name or object module name of 1 to 6 characters, not starting with / (slash), or a numeric character.
<disc number> or ,FX is one of the file codes F0 to FF (see File Codes).

Each time the system wants a new control command it will type out S: After that, the control command may be typed in by the user.
It is possible to add a comment statement after the last parameter of a command. It must be separated from the command by at least one space. On the following pages the control commands are listed in alphabetical order, according to their mnemonics.
They are followed by the processor calls, which are used in the same manner.

In the syntax descriptions, Backus Normal Form is used for the notation, i.e.:

|	means:	or
[ ]	means:	optional component; any or all items within these brackets may be omitted. [+|-]<integer> can mean -<integer>, -<integer>, or <integer>.
[ ]	means:	alternative components; one of the items within these brackets must be selected: [+|-] 426 can mean +426 or -426.
< >	means:	these brackets contain a syntactic item.
␣	means:	one space.

Command	Meaning	Page
ASG	Assign a File Code	82
ASM	Call Assembler	109
BYE	End of Session	83
DCU	Declare User	83
DEB	Call Debugging Package	110
DEL	Delete File	84
DLU	Delete User	84
DUF	Dump File	85
END	End of Catalogued Procedure	85
FBS	Space File Backwards	90
FFS	Space File Forward	90
FOR	Call Full Fortran Compiler	111
HSF	High- Speed Fortran	112
INC	Include an Object Module	86
JOB	Start Batch Processing	86
KPF	Keep File	87
LED	Call Line Editor	114
LIC	List Catalogue	88
LKE	Call Linkage Editor	113
LSD	List Directory	88
LSF	List File Code	88
LST	List File	89
MES	Send Message	91
MOV	Move a File	91
NOD	Define Node	92
OLE	Overlay Linkage Editor	116
PCH	Punch a File	93
PLB	Print label	90
PLD	Punch Load	94
POB	Punch Object	94
POD	Print Object Directory	95
PRC	Print Catalogue	98
PRD	Print Directory	98
PRT	Print File	99
PSE	Pause	100

Command	Meaning	Page
RBS	Space Record Backwards	90
RDA	Read Data	100
RDO	Read Object	101
RDS	Read Source	102
REF	Rewind File	90
REW	Rewind to Load Point	90
RFS	Space Record Forward	90
RSU	Replace Supervisor	103
RUN	Run a Program	103
SCR	Scratch	104
SDM	Save Disc onto Magnetic Tape	105
SEG	Define Segments	106
SKF	Skip Form	107
SVD	Save Disc onto another Disc	107
SVU	Save User Files	108
ULD	Unlock Device	90
UPR	User Processor	116
WES	Write Device	90
WEF	Write EOS	90
WEV	Write End-Of-Volume	90
WLB	Write Label	90

syntax ASG␣/<file code 1> [,/<file code 2> [,<device name> [[,<name> [,<userid>[,<disc number>[,NP[[,NP]]]]]]]]

use: This command is used to assign a file code to a peripheral unit, a disc file or a temporary area on disc.
The parameters have the following meaning:
<file code1>: file code which is to be assigned.
<file code2>: if this parameter is used, the assignment previously made for this file code has to be made for the first one (<file code1>) also. As a result the assignments for the two file codes specified will be equated.
<device name>: if this parameter is used, <file code1> is assigned to the peripheral unit specified here by two characters for the unit type and 2 hexadecimal digits for the address. If the device is the disc, only DK need be specified, without address.
<name>: this parameter is used only when DK is specified for <device name>. It specifies the name of the library file to which the file code must be assigned. If DK is used without this parameter, the file code will be assigned to a temporary disc area.
<userid>: this parameter is used only when <name> is specified. With <disc number>, it allows assigning a file code to a file in another user's library on the disc specified. The file will be set to write-protected, unless the parameter NP is specified, in which case it will not be protected, to allow writing on a file of a different <userid>.

If the file code to be assigned has already been assigned previously, the old assignment is deleted.

Note:
As mentioned previously, file codes 01 to 09 and D0 to FF are reserved for the system or have standard assignments. The following restrictions apply, however:
- file code 01 cannot be assigned
- file codes 02 to 09 can be assigned only to non-disc devices.
- file codes D0 to DF cannot be assigned.
- file codes E0 to EF cannot be assigned to a disc file.
- file codes F0 to FF cannot be assigned.

errors:
FILE CODE ERROR (1st parameter)
2nd FILE CODE ERROR
DEVICE UNKNOWN
TOO MANY PARAM
DEVICE NAME MISSING (2nd parameter)
FILE CODE NOT ASSIGNED (2nd file code)
FCT OVERFLOW (file code table overflow)
FILE CODE ABSENT
FILE NAME ERROR
USERID ERROR
INVALID FILE CODE
USERID UNKNOWN
DEVICE NAME ERROR
DEVICE ADDRESS ERROR
I/O ERROR (encountered during a read/write to/from disc)
LFT OVERFLOW (disc logical file description table overflow)
FILE NAME UNKNOWN
DISK OVERFLOW (no free granule available to allocate to the temporary disc file)
TOO MANY FILE CODE EQU (more than 7 file codes have been assigned to the same disc file)

BYE — END OF SESSION — BYE

syntax: BYE␣[BYE[,<DNDA>]]

use: In *batch processing mode*, this command indicates the end of the job and the system looks for the following job; if the parameter BYE is also specified, the system will switch from batch processing to conversational mode. If, in this case, <DNDA> (device name + device address) is also specified, this becomes the new assignment for file code ; E0.

In *conversational mode*, the user must give this command at the end of the session to indicate that he is leaving the system: the system is re-initialized and will again ask for identification in order to start a new session, unless the parameter BYE is also specified, in which case the system will switch to batch processing mode and automatically start reading the job control commands (on the card or punched tape reader).

DCU — DECLARE USER — DCU

syntax: DCU␣ userid',' 'disc number'

use: This command can only be used in a system session, i.e. when the user gives, at the start of the session, the user identification SYSTEM. Then, through this command, a new user identification is added to the Catalogue of the disc specified. A directory granule is allocated to this user and initialized with 'FFFF'. An entry for this user is filled in the Catalogue. The allocation table is updated.

errors: INVALID USERID (the user identification does not start with a letter)
USERID ABSENT (no parameter is given in the command)
INVALID FILE CODE (the disc number cannot be a disc file code, for it is not in the range from 'F0 to 'FF)
DISK FILE CODE ABSENT (the disc file code is not present in the command)
DISK NOT OPERATIONAL (the disc unit is not ready)
USERID ALREADY CATALOGUED (the userid specified has already been catalogued previously on the disc specified)
CATALOG OVERFLOW (too many userids have been catalogued on the disc specified)
DISK I/O ERROR (an I/O error has been detected during a read/write operation to, from disc)
DISK OVERFLOW (no free granule is available to be allocated to the userid directory)
TOO MANY PARAM (an illegal parameter follows the disc number)
COMMAND NOT ALLOWED (the current session is not a SYSTEM session)
DISK FILE CODE UNKNOWN (the specified disc file code has not been declared at SYSGEN or the generated system does not contain the disc specified).

DEL

DELETE FILE

DEL

syntax: DEL␣[<name>[/OB][,[/S]/O]/L]]

use: This command is used to delete a file or object module from a library.
<name> indicates the name of the file or module.
/OB indicates that the whole object file of the library must be deleted. If /OB is used, /S, /O or /L may not be specified.
/S, /O and /L specify the type of file: source, object or load. When <name> is used as the first parameter and no second parameter is specified, the type of file is UF (user file). If a /S, /O or /L file is to be deleted, this must be specified in the second parameter.
When <name> is used with /S or /L, a check is made on the types source or load to find the file which is to be deleted.
When <name> is used with /O, <name> is considered as being an object module in the object library.
When DEL /O is given the user object directory OBDIR is regenerated (see POD command).

errors: PARAM ERROR
INVALID PARAMETER
MISSING PARAMETER
FILE NOT CATALOGUED
I/O ERROR
TOO MANY PARAM
ERROR ASSIGN
PROGRAM NOT CATALOGUED

DLU

DELETE USER

DLU

syntax: DLU␣<userid>,/<disc number>

use: This command can only be used in a system session, i.e. when the user gives, at the start of the session, the user identification SYSTEM. By means of this command, the user specified is deleted from the disc specified.
<userid> specifies the user to be deleted.
/<disc number> gives the file code of the corresponding disc.
The corresponding entry in the Catalogue, the directory granule and all the granules of the library files are released and the allocation table of the disc is updated.
The DLU command may not be used to delete the first user on the disc with disc number /F0 (SYSTEM).

errors: COMMAND NOT ALLOWED (the current session is not a SYSTEM session)
USERID ERROR (the first parameter is not a userid)
USERID MISSING (no parameter is given)
DISK FILE CODE ERROR (the second parameter is not numeric)
DISK FILE CODE MISSING (no disc address specified in the command)
INVALID DISK FILE CODE (the value of the second parameter is not in the range from /F0 to /FF)
DISK FILE CODE UNKNOWN (the on-line system does not contain the specified disc)
DISK NOT OPERATIONAL (the disc unit is not ready)
TOO MANY PARAM (more than two parameters specified in the command)
USERID NOT CATALOGUED (the specified userid has not been catalogued on the disc specified)
I/O ERROR IN CATALOG (an I/O error has been detected during a read or write operation in the catalogue)
DISK I/O ERROR (an I/O error has been detected during the de-allocation of the user files).

DUF — DUMP FILE — DUF

syntax: DUF␣[/<file code>|/O|/L|<name>][,<sect.nb1>[,<sect.nb2>]]

use: This command is used to get a hexadecimal dump on the print unit of a library file or a file with a file code or a disc at physical level (/FX)
<file code>: file code of the file which must be dumped.
<name>: name of a library file which must be dumped. This is a user file of the type UF.
/O and /L cause dumping of the object and load files respectively. This is mainly useful for system debugging purposes.
It is also possible to get a selective dump by specifying two sector numbers. <sect.nb1> and <sect.nb2>, as the beginning and ending sectors of the dump.
A dump is made up to an EOF record, up to End-Of-Volume (last granule) or <sect.nb2>. The <sect.nb.> is a disc sector logical address.
Users must be careful about the position of the file after a DUF command. For a sequential file, they must rewind it before using it again. This means that the following command sequence will give an error:

```
ASM /S
DUF ,O
LKE
```

After assembly, the sequential 'O file is positioned to the next free sector of the file, but when a DUF command is executed, it is positioned to the last dumped sector, in random mode. This means that the EOF record written by the system when the LKE command is encountered does not immediately follow the object code and the result will not be correct. The command sequence should be as follows:

```
ASM /S      or:    ASM 'S
LKE                KPF ;O
DUF ;O             DUF ;O
                   LKE
```

The DUF command must normally be performed only after the completion of execution of a job step.

errors:
FILE NAME ERROR
FILE NAME MISSING
FILE CODE ERROR
INPUT FILE ASSIGN ERROR (followed by a message giving the reason for the error)
FILE CODE NOT ASSIGNED
DISK NOT OPERATIONAL
TOO MANY PARAM
PARAM ERROR (error in sector number)
I/O ERROR (an I/O error has been encountered while reading the disc file)
SECTOR DELETED

END — END CATALOGUED PROCEDURE — END

syntax: END

use: This command must be specified by the user at the end of a catalogued procedure to indicate its termination to the system (see page 23). This implies that the user cannot define a procedure named END, nor can he use any other CCI command name for a catalogued procedure.

INC **INCLUDE OBJECT MODULE** INC

syntax: [INC␣[/OBJCT]<name>][,<userid>[,<disc number>]]

use: By means of this command it is possible to select an object module from the library of the current (<userid>) or another (<userid>,<disc number>) user and to copy it in the temporary file /O.
<name> is the name of the object module which is to be included.
<userid> is to be used only if this object module is to be searched in the library of a user other than the current one.
If /O (temporary object file) exists, but has not been terminated by an EOF yet, this module is written at the end of this file.
If /O exists and has already been terminated with an EOF, this file is lost and a new assignment is made for a new /O.
If /O does not exist, an assignment is made for it and this module will be the first one of this new /O.
When the module has been copied, no EOF is written. Thus, it is possible to use this command several times, mixed with RDO commands and language processor calls to build a /O file.

Note:
- If /OBJCT is specified, the whole object library will be copied onto /O.
- When filling the /O file, no RDO, ASM or FRT command must appear after an INC command.
  It may only be followed by another INC command.

errors:
MISSING PARAMETER
PARAM ERROR
ERROR ASSIGN
UNKNOWN USERID
NO OBJECT LIBRARY
INVALID NAME
PROGRAM NOT CATALOGUED
I/O ERROR

JOB **START JOB** JOB

syntax: JOB␣[<userid>|,<disc file code>,<userid>]

use: This command is used to define the beginning of a new batch.
When it is encountered, the system will automatically switch to batch processing mode and implicitly close the preceding session, if any.
If <userid> only is specified, the system will scan the catalogue of each on-line disc, starting with disc unit /F0, until it finds the user identification specified.
If ,<disc file code>,<userid> is specified, the system will look for the user identification only on the disc of which the file code is specified.

KPF **KEEP FILE** KPF

syntax: KPF␣[S][O][L][,<file code>][,<name>]

use: This command is used to keep a file or module, which has previously been created as temporary, in a library, i.e. to make this file or object module permanent.

,S, ,O, ,L specify the type of the file which is to be kept.

<file code> is the file code of the file which is to be kept.

<name> is the name which is to be given to this file in the library and which will be placed in the directory.

If the first parameter is ,S, the file will be of the type source. If <name> is not specified, ,S is assumed to contain a source program of which the name can be found in its IDENT statement. Otherwise the name specified is taken as the file name. In case a file of the same name and type already exists in the library, it is deleted.

If the first parameter is ,L, <name> must be specified. An old file of the same name and type will be deleted, as with ,S.

If the first parameter is <file code>, <name> must be specified. An old file with the same file code will be deleted. Of course, <file code> must apply to a file which has previously been created as temporary.

If the first parameter is ,O, <name> is optional. If it is not specified, all object modules of the ,O file must be kept in a library, otherwise only the module <name> will be kept. Keeping object modules implies copying them from the ,O file onto the user object file. If any modules of the same name already are included in the existing object file, they will be deleted by the system setting the 'sector deleted' flag in the sectors containing these modules. If possible, the copying is done in the same physical area which the deleted modules occupied previously.

when KPF /O is given, the system generates a user object directory, the layout of which is described under the CCI command POD.
Note: Any NOD commands in the /O file will be ignored.

errors:
PARAM ERROR
INVALID PARAMETER
MISSING PARAMETER
DIRECTORY OVERFLOW
FILE EMPTY
I/O ERROR
IDENT MISSING
FILE CODE NOT ASSIGNED
FILE ALREADY CATALOGUED
MODULE UNKNOWN
DISK OVERFLOW
FILE OVERFLOW
ASSIGN ERROR

LIC **LIST CATALOGUE** LIC

syntax: LIC␣/<disc number>

use: This command can only be used in a system session, i.e. when the user gives, at the start of the session, the user identification SYSTEM. It provides for printing out the catalogue of the disc specified on the typewriter.
/<disc number> gives the file code of the disc of which the catalogue must be listed.

errors: SYSTEM SESSION COMMAND
PARAM MISSING
PARAM ERROR
FILE CODE NOT ASSIGNED

LSD **LIST DIRECTORY** LSD

syntax: LSD␣[,OB]

use: This command provides for a listing of the directory of the user library on the typewriter.
If ,OB is specified, only the names of the modules of the object file are listed.

errors: PARAM ERROR
NO OBJECT FILE CATALOGUED

LSF **LIST FILE CODES** LSF

syntax: LSF

use: When this command is given, a list is output on file code 1 of all the assigned file codes and the devices corresponding to them.

syntax: LST␣|<file code>|'S|'S,<name>|<name>||,<line nb1>[,<line nb2>]]

use: This command causes a listing of the specified disc file on the operator's typewriter. The file must be sequential and consist of ASCII records. If a record is longer than a print line it will be printed on several lines.
The file can be either:
- a catalogued source file: 'S,<name>
- a catalogued user data file: <name>
- a temporary data file: <file code>
- the temporary source file: 'S.

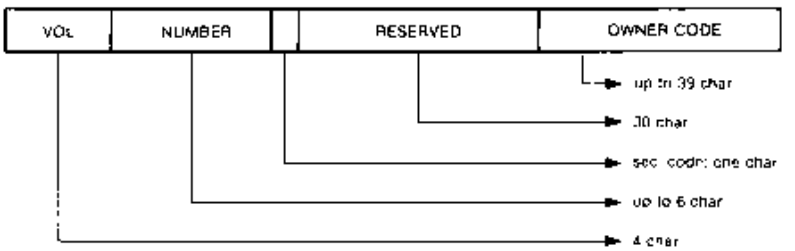
<line nb1>,<line nb2> if specified provides for a listing of the file from the first line number to the second line number specified.

The maximum record size allowed is 80 characters. Records which are longer will be truncated. Non-printable record characters will be replaced by spaces and trailing blanks will be removed. The listing will stop when either <line nb2>, an EOF or End-Of-Volume (the last granule of the file) is reached. End-Of-Volume occurs when no EOF has been written for this file, which normally is the case for temporary files created by a program in the debugging phase. After the listing, the file is positioned at the last record listed.

errors: FILE NAME ERROR (the first parameter is neither 'S nor a file code nor a character string)
FILE NAME MISSING
LINE NUMBER ERROR
TOO MANY PARAM
INPUT FILE I/O ERROR
OUTPUT FILE I/O ERROR
FILE CODE ERROR
OUTPUT NOT ASSIGNED (-02 is assigned to NO device or has not been assigned at all)
INPUT FILE CANNOT BE ASSIGNED (the system has to assign a temporary work file to the file which must be listed but this turns out to be impossible. A message will follow explaining the error).
EOV ON INPUT FILE, MOUNT NEW TAPE THEN RESTART (the input file code is assigned to a magnetic or cassette tape and its end-of-volume is encountered before the whole file has been listed. To continue the operation, the operator must mount the next tape or turn over the cassette and restart the program. The EOV mark is not considered as a record, so it is not listed).

MAGNETIC TAPE CONTROL COMMANDS

- REW␣<file code> is used to rewind the tape on the device to which <file code> has been assigned.
  ULD␣<file code> is used to switch the device specified by <file code> to manual state by sending an 'unlock' command to the cassette tape unit or a 'switch off' command to the magnetic tape unit.
- FFS␣<file code>[,<number>|ALL] is used to position the device after a tape mark. <number> if used, indicates the number of tape marks which must be skipped. If ALL is specified, the device will be positioned after two consecutive tape marks. Default value=1.
- FBS␣<file code>[,<number>] is used to space the tape backwards across the previous tape mark or across a number of tape marks as specified by <number>.
  RFS␣<file code>[,<number>] is used for spacing forward over a number of records, if <number> is specified. If it is not, a forward spacing is done until immediately after the next physical record.
- RBS␣<file code>[,<number>] is used for backwards spacing, either until immediately past the previous record, or, if <number> has been specified, past the specified number of previous records.
- WES␣<file code>[,<number>] is used to write one or a <number> of EOS records onto the device specified by <file code>.
  WEF␣<file code>[,<number>] is used to write one or a <number> of tape marks onto the device specified by <file code> (EOF records for devices other than tape).
- WLB␣<file code>,<serial number>,<sec. code>,<owner> is used to write a volume label on a magnetic or cassette tape, indicated by <file code>.
  <serial number> is the volume serial number, up to 6 numeric characters.
  <security code> consists of one hexadecimal character.
  <owner> is a character string of up to 39 characters, where blanks and commas are accepted.
  The label is written as follows, in ASCII:



  The label is followed by a tape mark.
  If <security code> is specified as NO, no volume label but only a tape mark is written.
- PLB␣<file code> is used to have the volume label of a magnetic or cassette tape printed on the operator's typewriter and the tape positioned at the first record of the file following the label. The CCI does not check the label. If the label is absent and a tape mark is read as the first record on the tape, the message NO LABEL is printed.
- WEV␣<file code> is used to write an End-Of-Volume mark on magnetic or cassette tape.
  REF␣<file code> is used to position the file specified at the first record. It can be used for cassette tape, magnetic tape and disc.

The commands REW, FFS, FBS, ULD can be used only with cassette and magnetic tape units.

The following error messages are possible for these commands:
FILE CODE MISSING (no <file code> specified)
FILE CODE ERROR
INVALID FILE CODE
FILE CODE UNKNOWN
PARAM ERROR (error in the second parameter)
PARAM MISSING
TOO MANY PARAM
I/O ERROR

MES **SEND MESSAGE** MES

syntax: MES␣<message to the operator>

use: This command is used especially in batch processing mode or catalogued procedures to have the message specified typed out to the operator.

MOV **MOVE A FILE** MOV

syntax: MOV␣<name>[,'S|'L|,<file code>][,<userid>[,<disc number>]]

use: This command is used to move a file from a library - generally of another user [,<userid>,,<disc number>] to a temporary file 'S or 'L, or to a file indicated by a file code.

<name> is the name of the library file which is to be moved.

'S: the file must be moved to temporary file 'S.

'L: the file must be moved to temporary file 'L.

<file code>: file code of the temporary file to which the file <name> must be moved.

<userid>: user identification of the user whose file must be moved (if the file belongs to another user than the current one).

The type of the file must be compatible with the type of the receiving temporary file:

'S: source

'L: load

<file code>: undefined.

If 'S or 'L or <file code> has already been assigned to a temporary file, the latter file is lost.

After a file has been moved, it can be used directly as a temporary file, or it can be kept in a library (KPF command).

errors: FILE TYPE MISSING (the second parameter is absent)
FILE TYPE ERROR (the second parameter is neither a file code, nor 'S, nor 'L)
USERID ERROR
USERID UNKNOWN
TOO MANY PARAM
INPUT FILE ASSIGN ERROR
OUTPUT FILE ASSIGN ERROR (these messages are followed by another one specifying the cause of the error)
I/O ERROR (an I/O error has been encountered during the read or write operation and the file is not copied. A new MOV command has to be given).

DEFINE NODE

NOD

syntax: NOD␣<name>

use: By means of this command, the user can define a node in an overlay structure (see Programmer's Guide 2, Vol II: System Software).
<name> must be a string of 6 ASCII characters for acceptance by the OLE. The Control Command Interpreter, however, will copy any number of characters onto the object file.
The command is written as it is encountered, after which an: EOS is written onto the /O file.

errors: UNKNOWN FILE (/O file has not yet been opened)
NO OBJECT FILE (file on which command must be written is not /O)
CLOSED OBJECT FILE (/O file has already been closed)
I/O ERROR (impossible to write on /O file).

PCH **PUNCH A FILE** PCH

syntax: PCH␣[/<file code>|/S|<name>|<name>,/S] [,<new file code>]

use: This command is used to have a file punched on the punch unit. The file must be sequential with a maximum record length of 132 characters. The file may be of one of the following types, as specified in the command:
<file code>: file code of a temporary user file which must be punched.
/S: the source file must be punched.
<name>: name of a catalogued user data file which must be punched.
<name>,/S: name of a catalogued source program which must be punched. If any records of over 132 characters are encountered in the file, they are truncated. The file must be closed by an EOF record, otherwise the file will be punched up to the End-Of-Volume, where the last records of the last granule may be wrong.
<new file code>: output file code. Default value is /03.

errors: /S EMPTY
FILE NAME ERROR
FILE NAME MISSING
INVALID FILE CODE
FILE CODE NOT ASSIGNED
TOO MANY PARAM
INPUT I/O ERROR
OUTPUT I/O ERROR
FILE TYPE ERROR (the parameter following name is not /S)
INPUT FILE CANNOT BE ASSIGNED (it is impossible for the system to assign a work file code to the input file. A message will follow explaining the cause of the error).
EOV ON OUTPUT FILE, MOUNT NEW TAPE THEN RESTART (an end-of-volume has been encountered on magnetic or cassette tape output device. The operator must mount a new tape or turn over the cassette and restart the program to continue the output operation).
OUTPUT FILE CODE ERROR

PLD — PUNCH LOAD — PLD

syntax: PLD␣[<name>][,/L][,<file code>][,/L,<file code>]

use: This command is used to punch a load file from the library or from the temporary file /L. The file will be punched on the punch unit in object code format.
<name>: name of a load file in a library. If ,/L is specified, the temporary file /L will be punched with ident record built from the name specified in the command, if this name is different from the IDENT name of the program.
<file code>: output file code. Default value is /03.

errors:
PARAM ERROR
FILE CODE NOT ASSIGNED
PARAM ABSENT
FILE NAME NOT CATALOGUED
NO LOAD MODULE
OUTPUT FILE CODE ERROR

POB — PUNCH OBJECT — POB

syntax: POB␣[<name>][<name>|<file code>|<name>,<file code>]

use: This command is used to punch an object module from the object file of the library or to punch the whole contents of the temporary object file /O.
<name>: is the name of the library object module which must be punched. If no name is specified, the whole temporary object file /O will be punched.
When the temporary file /O must be punched and no EOF has yet been written after it, the EOF mark is first written, before the file is rewound and punched.
<file code>: output file code. Default value is /03.

errors:
OBJECT MODULE NAME ERROR
/O EMPTY
/O CLOSE ERROR (error detected during writing of EOF and rewinding of /O file)
/O INPUT ERROR (error detected during reading of /O file)
OUTPUT I/O ERROR (I/O error encountered during the output operation)
INPUT I/O ERROR (error detected during reading of the module which is to be output)
ILLEGAL EOS IN INPUT FILE (the first record of a module is an EOS)
OBJECT MODULE NOT CATALOGUED
IDENT MISSING (the first record of a module in the object library is not an IDENT)
OBJECT LIBRARY ASSIGN ERROR (it is impossible for the system to assign a work file to the user object library. A following message will explain the cause of the error).
EOV ON OUTPUT FILE, MOUNT NEW TAPE THEN RESTART (an end-of-volume has been encountered on a magnetic or cassette tape output device. The operator must mount a new tape or turn over the cassette and restart the program to continue the output operation).
OUTPUT FILE CODE ERROR

POD

PRINT OBJECT DIRECTORY

POD

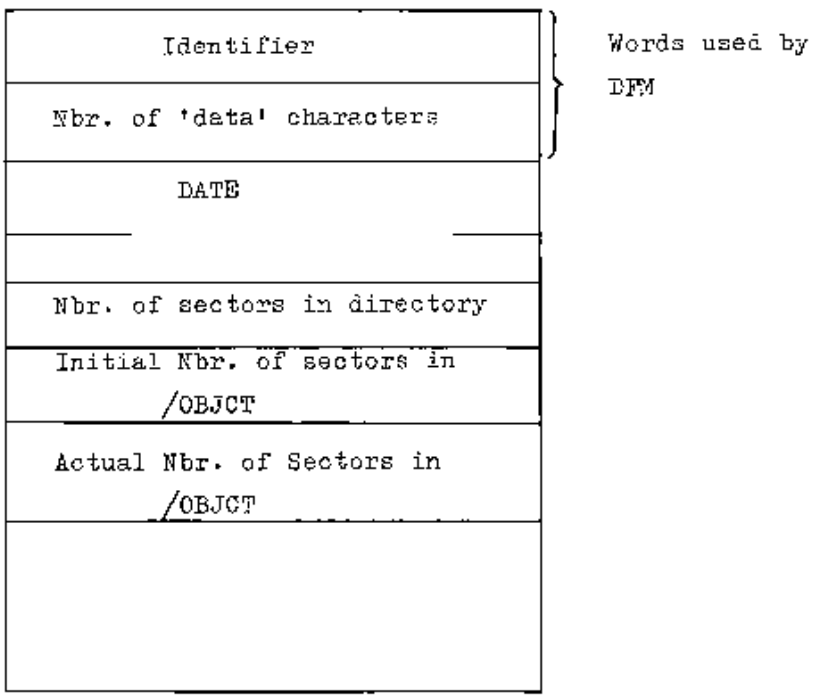
syntax: POD

use: This command is given to obtain a listing of the OBDIR user object directory file which has been created when a KPF /O command was given.
In the print-out, the first line is self explanatory. Under the second line the name of the module will be found, then under STRT, INIT and ACTL the sector number on which the module starts, the initial number of sectors filled by the module and the actual number of sectors filled by the module respectively.

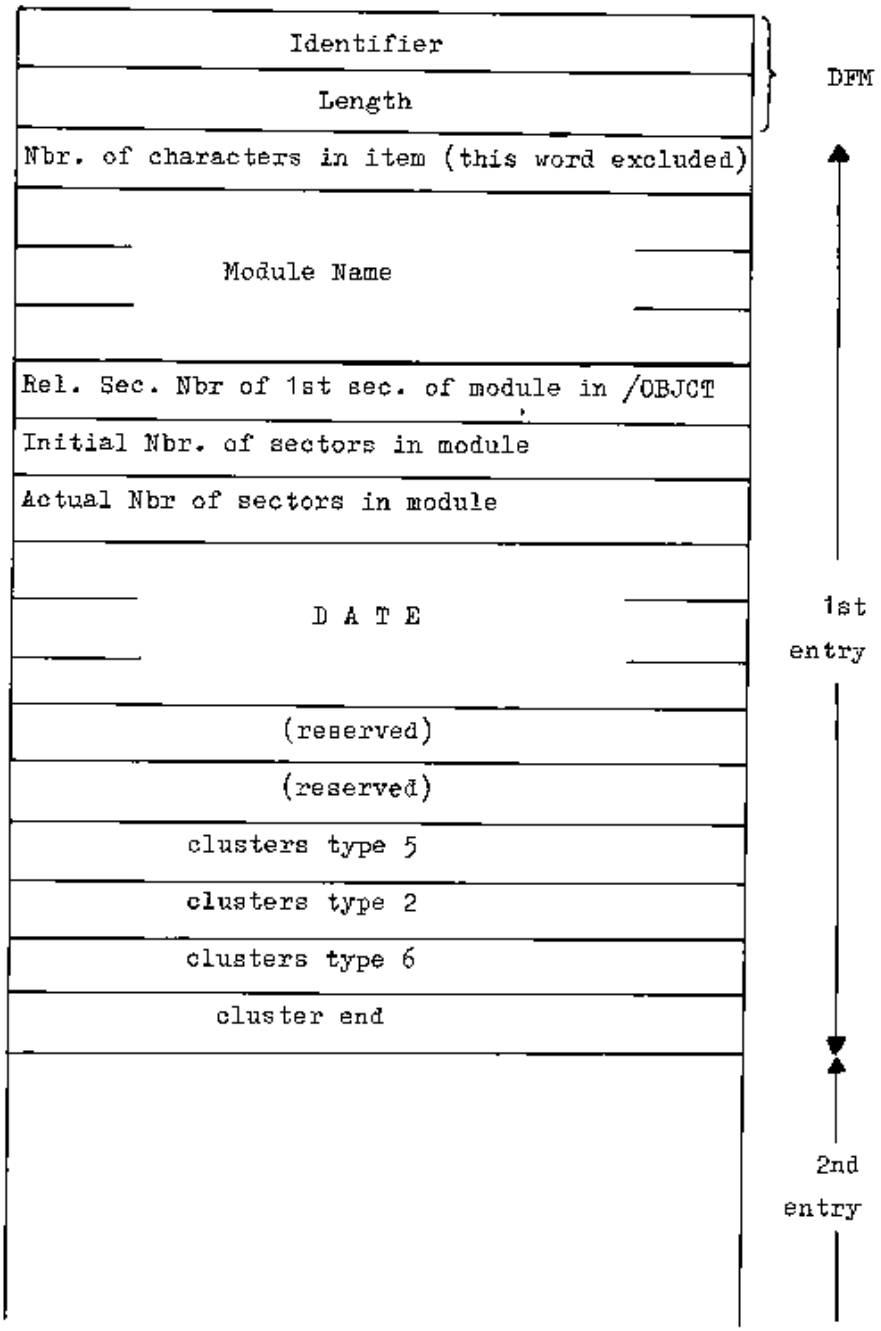
errors: I/O ERROR (incorrect I/O request)
INVALID FILE (OBDIR is not assigned to /D8)
INVALID OBJECT (an EOF has been encountered)
INVALID OUTPUT (there has been an abnormal sequential I/O)
DIRECTORY ERROR (OBDIR format incorrect)
ASSIGN FILE CODE ERROR (/D8 could not be assigned)
DISK NO ASSIGN ERROR (/D8 deletion incorrect)

Structure of Directory

- Sector 0



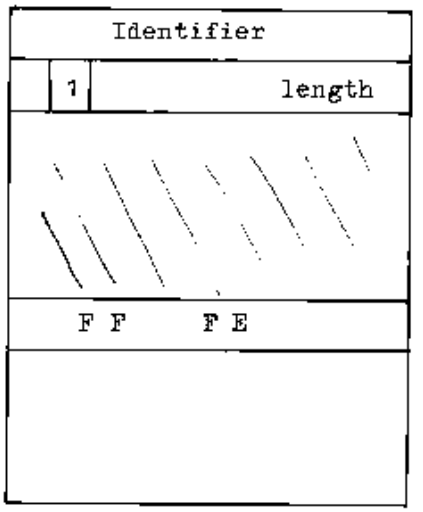
- Following sectors



Remarks:

- The last 3 words of sectors are not used.
- Bit 1 is set in last sector's 2nd word to indicate the end of directory.

Last sector:



- In this last sector, the last directory's entry is followed by the constant /FFFE.

PRC **PRINT CATALOGUE** **PRC**

syntax: PRC␣/<disc number>

use: This command can only be used in a system session, i.e. when the user gives, at the start of the session, the user identification SYSTEM. It causes a print-out on the print unit of the catalogue contained on the disc specified. /<disc number> gives the address of the related disc.

errors: SYSTEM SESSION COMMAND (the current session is not a SYSTEM session)
PARAM MISSING
PARAM ERROR
FILE CODE NOT ASSIGNED.

PRD **PRINT DIRECTORY** **PRD**

syntax: PRD␣[,OB]

use: This command causes a print-out of the user's library directory on the print unit. If ,OB is specified, only the names of the object modules in the object file are printed, including any comments in the IDENT statements.

errors: PARAM ERROR
NO OBJECT FILE CATALOGUED

PRT **PRINT FILE** PRT

syntax: PRT␣[< file code > | < S | / S, < name > | < name >][, < line nb 1 > [, < line nb 2 >]]

use: This command causes a listing of the specified disc file on the print unit. The file must be sequential and consist of ASCII records. If a record is longer than a print line it will be printed on several lines.
The file can be either:
- a catalogued source file: < S, < name >
- a catalogued user data file: < name >
- a temporary data file: < file code >
- the temporary source file: < S, >

< line nb1 >, < line nb2 > if specified, provides for a listing of the file from the first line number to the second line number specified.

The maximum record size allowed is 132 characters. Records which are longer will be truncated. Non-printable record characters will be replaced by spaces and trailing blanks will be removed. The listing will stop when either < line nb2 >, an EOF or End-Of-Volume (the last granule of the file) is reached. End-Of-Volume occurs when no EOF has been written for this file, which normally is the case for temporary files created by a program in the debugging phase. After the printing, the file is positioned at the last record printed.

errors: FILE NAME ERROR (the first parameter is neither < S nor a file code nor a character string)
FILE NAME MISSING
LINE NUMBER ERROR
TOO MANY PARAM
INPUT FILE I/O ERROR
OUTPUT FILE I/O ERROR
FILE CODE ERROR
OUTPUT NOT ASSIGNED (.02 is assigned to NO device or has not been assigned at all)
INPUT FILE CAN NOT BE ASSIGNED (the system has to assign a temporary work file to the file which must be printed but this turns out to be impossible. A message will follow explaining the error).
EOV ON INPUT FILE. MOUNT NEW TAPE THEN RESTART (the input file code is assigned to a magnetic or cassette tape and its end-of-volume is encountered before the whole file has been printed. To continue the operation, the operator must mount the next tape or turn over the cassette and restart the program. The EOV mark is not considered as a record, so it is not printed).

PSE **PAUSE** PSE

syntax: PSE␣[<message to the operator>]

use: This command is used to put the machine in pause state and have the specified message typed out for the operator.
The operator must restart the program to have the next command read.

RDA **READ DATA** RDA

syntax: RDA␣<disc file code>[,<input code>]

use: This command is used to read and transfer data to a temporary user file.
<disc file code>: the temporary user file to which the data are transferred. This file code does not have to be assigned by the user; an implicit assignment will automatically be made by the system.
<input file code>: file code from which the data are read. Default value is 'FF (source input).
The data will be read until an EOF is encountered.
The file codes specified must be in the range from '01 to 'EF.

errors: DISK FILE CODE MISSING
DISK FILE CODE ERROR
INVALID DISK FILE CODE
INVALID PARAMETER (the file code is not a numeric value in the range from '01 to 'EF)
FILE CODE NOT ASSIGNED (the file code is assigned to NO device or has not been assigned at all)
TOO MANY PARAM
DISK ASSIGN ERROR (the system is unable to assign a disc temporary work file. A message will follow giving the cause of the error)
INPUT I.O ERROR
OUTPUT I.O ERROR (an error has been detected during the last read operation from the sequential file or during the last write operation to the disc temporary file).

syntax: RDO␣[<file code>]

use: By means of this command it is possible to copy an object file from the object input unit or from another sequential input unit onto the disc as a ;O file or as a complement to the ;O file.
<file code>: file code of the input unit from which the object file is to be read. If not specified, the object file is read from the standard object input unit.
No EOF record is written onto the disc.
If there is already a ;O file on the disc which has not been closed with an EOF record, the object file is copied after this ;O file.
If there is no ;O file on the disc or if it has already been closed with an EOF record, a new assignment is done for ;O starting at a new granule and the old ;O file is lost.
It is possible to give several RDO commands to fill ;O with several object files. Mixing RDO commands with language processor calls in order to fill the ;O file is allowed also.
Note:
An EOF mark is written after the ;O file when one of the following commands is given:
KPF ;O[,<name>]
POB
LKE

errors: INVALID PARAMETER (the file code is not a numeric value in the range from ;01 to ;EF)
FILE CODE NOT ASSIGNED (the file code is assigned to NO device or has not yet been assigned at all)
TOO MANY PARAM
DISK ASSIGN ERROR (the system is unable to assign the disc temporary file. A message will follow giving the cause of the error)
INPUT I/O ERROR
OUTPUT I/O ERROR (an error has been detected during the last read operation from the sequential file or during the last write operation to the disc temporary file).

RDS **READ SOURCE** RDS

syntax: RDS␣[/<file code>]

use: By means of this command it is possible to copy a sequential source program file or a sequential data file from the source input unit or from another sequential input unit onto the disc as a /S file.
If a /S file had already been built previously and not been made permanent in the library (KPF command), the old file is lost because a new assignment is made for the /S file starting at a new granule.
Records are 80 characters long and terminated by an EOS or EOF record. The last 8 characters of the input records are replaced by spaces.
When an EOS or EOF is encountered, the system writes the EOS on the /S file, followed by the EOF.

errors: INVALID PARAMETER (the file code is not a numeric value in the range from /01 to /EF)
FILE CODE NOT ASSIGNED (the file code is assigned to NO device or has not yet been assigned at all)
TOO MANY PARAM
DISK ASSIGN ERROR (the system is unable to assign the disc temporary file. A message will follow giving the cause of the error)
INPUT I/O ERROR
OUTPUT I/O ERROR (an error has been found during the last read operation from the sequential file or during the last write operation to the disc temporary file).
EOV ON INPUT FILE, MOUNT NEW TAPE THEN RESTART (the input file is a magnetic or cassette tape and the end-of-volume has been encountered before the end-of-file mark. The operator has to mount the next reel of tape or turn over the cassette and restart the program).

RSU — REPLACE SUPERVISOR — RSU

syntax: RSU␣[<disc number>]

use: This command can only be used in a system session, i.e. when the user gives, at the start of the session, the user identification SYSTEM.
Then this command may be used to change the monitor on the disc, by copying the contents of the .L file onto the disc starting from absolute sector number 18. The system ensures that <disc number> is a system disc, i.e. contains a load module file starting at address /10 built of consecutive granules and that the file size is ≥ /L.

errors: COMMAND NOT ALLOWED
PARAM ERROR (the parameter must be numeric)
DISK ADDRESS MISSING
INVALID DISK ADDRESS
TOO MANY PARAM
DISK UNKNOWN
DISK NOT OPERATIONAL
.L EMPTY
DISK I-O ERROR (an I-O error has been detected during the reading of the .L file or during the writing onto the specified disc).
/FX NOT A SYSTEM DISC
/L TOO BIG.

RUN — RUN A PROGRAM — RUN

syntax: RUN␣[<name>][,O]

use: This command must be used to start a program after it has been loaded from a library or from the .L file.
<name>: name of the program, if it is a program which is loaded from library. If no name is specified, the program is loaded from the .L file.
O, is specified to indicate that the program runs in system mode (P856/P857 only).

errors: PROGRAM NAME ERROR
.L EMPTY
PROGRAM NOT CATALOGUED
TOO MANY PARAM

syntax: SCR␣[[/S|/O|/L]/ file code]]

use: This command can be used to release previously made user assignments.
It is processed as follows:
file codes /0A through /DF are deleted
file codes /01 through /CF and /E1 through /EF are deleted when assigned to a disc file
file code /E0 is deleted when assigned to a temporary disc file
If no parameter is specified, all user assignments are released and the pointer for the granule allocation table is reset to the first available granule, i.e. the system is reset to the state it was in at the beginning of the session.
If /S, /O, /L or file code is specified, this indicates the file whose assignment must be released. In such a case, the corresponding entry in the file table of the monitor is made available for other assignments. This may be very useful in cases where table overflow problems must be avoided, or it can be done after an error message for table overflow has been received (the number of entries in the file table is defined at system generation time).

Note:
– The file codes from /01 to /09 and from /E0 to /FF can not be scratched, because they are reserved or system file codes.
– SCR may be defined in catalogued procedures only as SCR␣/S or SCR␣/O

errors: INVALID PARAM
INVALID FILE CODE
I/O ERROR (an I/O error has been detected during the loading of the allocation table).

SDM **SAVE DISC ONTO MAGNETIC TAPE** **SDM**

syntax: SDM␣<disc number>,<file code>[,CK]

use: This command is used to save the contents of a disc onto a magnetic tape. <disc number> is the file code (from ,F0 to ;FF) of the disc which must be saved.
<file code> is the file code of the magnetic tape on which the disc is saved. When the SDM command is given, the Control Command Interpreter will write a stand-alone, self-loadable program at the beginning of the tape to IPL the saved contents from MT to DK, then output the contents of the disc onto it, sector by sector. The tape is closed by a tape mark. Thus, the contents of a disc can be saved or duplicated, which is very convenient in configurations with only one disc unit.
CK: If CK is specified, the magnetic tape is rewound and compared to the disc.

errors: PARAM MISSING
DISK FILE CODE ERROR (first parameter is not a file code)
INVALID FILE CODE (Ffirst parameter is not a value from F0 to FF)
FILE CODE NOT ASSIGNED (disc file code is unknown)
2ND FILE CODE ERROR (the second file code is not assigned to a manetic tape or it is not a correct value for a file code)
TOO MANY PARAM
DISK I/O ERROR (an irrecoverable I/O error occurred while reading the disc)
I/O ERROR (an irrecoverable I/O error occurred on the magnetic tape).

SEG **DEFINE SEGMENTS** **SEG**

syntax: SEG␣<name list>

use: This command enables the user to build his program in a kind of overlay structure. It defines the library program names of the program parts which will be used as segments by a root program which is started in a following RUN command.
<name list> consists of one or several library program names, separated by commas. The list may not contain more than 15 names.
The segments specified will be called at run time by the root program or by other segments through the Load monitor request (I.KM DATA 9). The numbers of the segments are implicitly defined by the order in which they appear in the <name list>, i.e. the first program name specified will be segment number 1.
The following command must necessarily be a RUN command.

Note:
Special care must be taken with respect to the dynamic allocation area when using segmented programs. After the Root segment has been loaded, the start address of the dynamic area is the first address following the root. Now, if another segment were loaded without any provisions the dynamic area would be overwritten, which must be avoided if any Get and Release Buffer requests are used. The user must take care that the start address of the dynamic allocation area is behind the longest segment which the root program is going to load. This may for example be done by giving a request for area reservation (RES) in the Root program to reserve an area behind the root segment into which the other segments will be loaded. The dynamic area will then start behind this reserved area.

errors: SEGMENT NBR. 01 MISSING (no parameter present)
SEGMENT NBR. XX NOT CATALOGUED (the segment number XX is not catalogued or has been declared more than once in the command; all the <names> of the list must be different)
SEGMENT NBR. XX ERROR
TOO MANY PARAM (the number of segments in the command is greater than the maximum number of segments declared at system generation time).

SKF **SKIP FORM** **SKF**

syntax: SKF␣[<number>]

use: By means of this command, a number of pages may be skipped on file code '02, i.e. the line printer.
<number> is the number of pages to be skipped. Default value is 1.

SVD **SAVE DISC ONTO ANOTHER DISC** **SVD**

syntax: SVD␣<disc number1>,,<disc number2>

use: This command can only be used in a system session, i.e. when the user gives, at the start of the session, the user identification SYSTEM.
It is used to copy the contents of one disc onto another one.
All allocated granules of the disc specified by <disc number1> are copied onto the disc specified by <disc number2>.
If the capacity of the discs is different, only as many sectors will be duplicated as the smaller disc contains.
The discs are assumed not to contain any defective tracks, for the copying is done sector per sector, sequentially.

Note:
SVD ,FX,,F0 is not allowed.
The volume label of <disc number2> is **not** destroyed.

errors: COMMAND NOT ALLOWED (the current session is not a system session)
FIRST FILE CODE MISSING
SECOND FILE CODE MISSING
FIRST FILE CODE UNKNOWN
SECOND FILE CODE UNKNOWN
FIRST FILE CODE ERROR
SECOND FILE CODE ERROR
TOO MANY PARAM
INPUT I/O ERROR
OUTPUT I/O ERROR
INVALID DISK TYPE (the disc to which one of the two file codes is assigned, is not supported by the system).

SVU **SAVE USER FILES** SVU

syntax: SVU␣<userid>,/<disc number>

use: This command is used to copy all the files of the user specified and contained on the disc specified, into the library of the current session user. This may be the same user as the one specified by <userid>.
<userid>: user identification, a string of up to 8 characters.
/<disc number>: file code of the disc containing the files which must be saved.
The files will be copied one at a time, up to 'end of medium'.
The new file will be kept in the directory of the current user under the same name and type. If the name has already been kept previously in the directory, the old file is scratched and replaced by the new one. All the sectors of the file are copied, one at a time, except for the deleted sectors of the object library file. The files are copied in the same order as they appear in the directory of the user specified by <userid>.

errors: INVALID USERID (the first parameter is not a character string)
DISK FILE CODE ERROR (the second parameter must be a binary value)
USERID MISSING (the command contains no parameter)
DISK FILE CODE MISSING (the second parameter specifying the disc file code is not specified in the command)
INVALID DISK FILE CODE (the second parameter is numeric but not in the range from /F0 to /FF)
DISK FILE CODE UNKNOWN (the file code specified can not be found in the system file code table. It has not been declared at SYSGEN)
DISK NOT OPERATIONAL (the disc to which the file code has been assigned, is not operational. If it has just become ready, a retry is possible, otherwise the error must be corrected)
TOO MANY PARAM
USERID NOT CATALOGUED (the userid given in the command does not exist on the disc specified)
INPUT DISK I/O ERROR (an I/O error has been detected during a read operation from the disc specified in the command)
OUTPUT DISK I/O ERROR (an I/O error has been encountered during a read/write operation from/to the disc used in the current session)
INPUT FILE ASSIGN ERROR
OUTPUT FILE ASSIGN ERROR (in order to save all the files of the user specified, the system assigns a temporary work file code to a file of this user and another one to the disc file of the current session. This may be impossible, in which case a message will follow explaining the cause of the error)
DIRECTORY OVERFLOW ON XXXXXXFT (there is an overflow of the directory of the user of the current session, while the file with the name XXXXXX and the type FT is being catalogued).
The file type is SC for source files, OB for object files, LM for load modules and UF for user data files. This file can not be catalogued. The user may give a PRD command to find out which files have been copied so far, since all files are copied in the order in which they appear in the directory of the user specified by <userid>.

ASM — ASSEMBLER — ASM

syntax: ASM␣[/S|<name>][,NL]

use: This command must be used to assemble a source program from a library or from the temporary file /S.
/S: the source program must be assembled from the temporary file /S.
<name> : the source program to be assembled can be found in the library. <name> gives the name of this program.
NL: if specified, no listing will be provided of the assembled program. Otherwise, the listing is output on the print unit. Any error messages will be output on the operator's typewriter as well.
The object code is produced on the temporary file /O. If /O does not exist, an assignment will be made for it. If /O does already exist, the object module will be written at the end of this file, unless /O has already been closed with an EOF record. In that case a new assignment is made and the old /O file is lost.

Note:
If a fatal error occurs during assembly, the whole /O file will be deleted and a request for Link Edit will be refused.

errors: FILE NAME ERROR
FILE NAME MISSING
INVALID PARAM
/S EMPTY (no temporary source file exists)
/S ASSIGN ERROR (it is impossible to assign the file code /D4 to the catalogued source file. A message will follow to explain the error)
/O ASSIGN ERROR (an attempt to assign the temporary object file /O is refused. A message will follow to explain the error)
NL OPTION ERROR (NL has been declared more than once in the command)
PROCESSOR NOT CATALOGUED (a segment of the assembler has not been catalogued).

syntax: DEB␣[<name>]

use: This command must be used to call the Debugging Package and run a load module under its control.
<name> is the name of the load module if it is to be found in a library. If no name is specified, the temporary file :L is considered to be the load module. The Debugging Package loads the load module into memory by means of the Load monitor request. This implies that it is not possible to debug a segmented program. During the debugging phase a temporary file is created on the user disc for a copy of the Debug Processor on the system disc. This file, together with the user program which must be debugged are considered as a segmented program where the temporary file is the root and the user program is segment number 1.

errors: PROGRAM NAME ERROR (the parameter is not a character string)
:L EMPTY (there is no program in the temporary load file)
PROGRAM NOT CATALOGUED (the program <name> cannot be found in the directory)
TOO MANY PARAM
PROCESSOR NOT CATALOGUED (the debug processor has not been catalogued on the system disc)
SYSTEM DISK I/O ERROR
USER DISK I/O ERROR (an I/O error has been detected during the copying of the debug processor from the system library to a temporary file on the user disc)
ASSIGN ERROR (the system is unable to perform an assignment. A message will follow explaining the reason for this error).

FOR **FULL FORTRAN COMPILER** **FOR**

syntax: FOR␣[/S|<name>][,NL]

use: This command must be used to compile a FORTRAN source program from a library or from the temporary file /S.
/S: the program must be compiled from the temporary file /S.
<name> : the program to be compiled can be found in the library. <name> gives the name of this program.
NL: if specified, no listing will be provided of the compiled program. Otherwise, a listing is output on the print unit. Any error messages will be output on the operator's typewriter as well.
The object code is produced on the temporary file /O. If /O does not exist, an assignment will be made for it. If /O already exists, the object module will be written at the end of this file, unless /O has already been closed with an EOF record. In that case a new assignment is made and the old /O file is lost.

Note:
If a fatal error occurs during compilation, the whole /O file will be deleted and a request for link edit will be refused.

errors: FILE NAME ERROR
FILE NAME MISSING
INVALID PARAM
/S EMPTY (no temporary source file exists)
/S ASSIGN ERROR (it is impossible to assign the file code /D4 to the catalogued source file. A message will follow to explain the error)
NL OPTION ERROR (NL has been declared more than once in the command)
PROCESSOR NOT CATALOGUED (a segment of the compiler has not been catalogued)
/O ASSIGN ERROR (an attempt to assign the temporary object file /O is refused. A message will follow to explain the error).

HSF **HIGH-SPEED FORTRAN** HSF

syntax: HSF[,/S[<name>][,NL]]

use: This command is used to load the high-speed FORTRAN compiler into memory to start compilation of a source module from the temporary file /S or from the user library.

/S: the program must be compiled from the temporary file /S.
<name>: the program to be compiled can be found in the library.
<name> is the name of this program in its directory.
NL: if specified, no listing will be provided of the compiled program. Otherwise, a listing is output on the print unit. Any error messages will be output on the operator's typewriter as well.
The object code is produced on the temporary file /O. If /O does not exist, an assignment will be made for it. If /O does already exist, the object module will be written at the end of this file, unless /O has already been closed with an EOF record. In that case a new assignment is made and the old /O file is lost.

errors:
FILE NAME ERROR
FILE NAME MISSING
INVALID PARAM
/S EMPTY
/S ASSIGN ERROR
/O ASSIGN ERROR
NL OPTION ERROR
PROCESSOR NOT CATALOGUED

syntax: LKE␣[N|S|U][,M][,DE|DS][,<address>][,<start address>]

use: This command must be used to call the linkage editor. The parameters may be given in any order.

N: no library scanning is desired.
S: only the standard library has to be scanned.
U: only the user library has to be scanned.
Default value: both libraries will be scanned, the user library first.
M: a print-out of the map is wanted. Default: no map will be printed.
DE: all entry points must be preserved for the Debugging Package.
DS: only the pointers (implicit entry points) to the symbol tables must be preserved for the Debugging Package. The symbols have been preserved at assembly time
Default value: no debugging.
,<address>: hexadecimal displacement value of the blank common from the beginning of the load module. The address must be specified in characters. If this address is an even number, the blank common address is assumed to be relocatable; if it is odd, the address is assumed to be absolute. This option may be used for communication between several load modules making use of the same blank common. In such a case the load modules must be called by the Load monitor request (LKM DATA 9) and have been defined previously in a SEG command.
The address of the blank common must be defined in such a way that it will not be destroyed when a load module is loaded. The address must be defined at link-edit time for each one of the load modules of the overlay structure.
Default value: the blank common, if any, will be located at the end of the load module.
<start address>: name of the start address defined as an entry in one of the modules in the ;O file. The name must be different from the option parameters used in the LKE command: N, S, U, M, DE and DS.
Default value: the last start address encountered in the ;O file.
Before starting LKE, an EOF record is written on the ;O file if it has not already been closed with an EOF previously. Moreover, an assignment is made for the ;L file onto which the Linkage Editor will write the load module.

Note:
If a fatal error occurs during the link-edit operation, the ;L file is deleted and a RUN command will be refused.

errors:
INVALID PARAMETER
COMMON VALUE REDUNDANT
LIBRARY OPTION REDUNDANT
DEBUG OPTION REDUNDANT
MAP OPTION REDUNDANT
START ADDR. REDUNDANT
USER LIB ASSIGN ERROR:
SYSTEM LIB ASSIGN ERROR
;L ASSIGN ERROR:
;O EMPTY
;O CLOSE ERROR
PROCESSOR NOT CATALOGUED

syntax: LED␣<name>[[,<file code 1>[,<file code 2>]]][,,S],<file code 2>]][,xx]

use: This command is used to call the Line Editor and update a source file located in a library.
<name is the name of the source file.
<file code1> is the output file code. If specified, the file is written as a UF file.
<file code2> specifies the input command file code from which the control commands are entered. Default value: /E0. If this file code is assigned to a typewriter, the Line Editor prints L: before reading a record.
xx are two characters, specified if the user wants command control characters other than !! (see below).
The updated output file is written as a temporary /S file. If /S already exists, a new assignment is made and the old /S file is lost.

errors: FILE NAME ERROR
FILE NAME MISSING
INPUT FILE CANNOT BE ASSIGNED (followed by message explaining the error)
/S CANNOT BE ASSIGNED (followed by message explaining the error)
INVALID FILE CODE
FILE CODE NOT ASSIGN
TOO MANY PARAM
DSK INPUT ERR. UPD ABORTED
DSK OUTPUT ERR. UPD ABORTED
UNKNOWN COMMAND, TRY AGAIN
I/O ERR ON LAST RECORD, TRY AGAIN
SEQUENCE ERR. TRY AGAIN
SYNTAX ERR. TRY AGAIN
EOF, UPD TERMINATED (the EOF mark has been encountered on the input source file before reaching the specified line, thus terminating the update process)
AUX. INPUT CANNOT BE ASSIGNED, TRY AGAIN (auxiliary file used in JN command cannot be assigned)
CMND NOT ALLOWED IN EXE MODE, TRY AGAIN (definition mode command)
TABLE O'FLOW, TRY AGAIN (character string table is overflowing)
EOF IN AUXI INPUT (JN command is terminated but operation continues).

When the message TRY AGAIN is printed, the user may correct the erroneous command or data record from the device with file code 01. If he types only CR, input is resumed from the device with the normal Input File Code.

With the Line Editor, there are two phases of operation: definition and execution. The user first defines the modifications to be made to the file. The commands given are not executed immediately but recorded until the updated file is created. When the Line Editor is called, it is flagged as being in definition mode, to be closed implicitly when an execution mode command is encountered.
Execution mode commands are executed immediately, i.e. creation of the updated file is started. After executing the execution mode commands, the Line Editor will look for any pre-recorded definition mode commands. Once execution mode has been started, it is impossible to return to definition mode.

Definition mode Commands:

!!CH␣ $$ <character string1> $$ <character string2> $$
is used to have <character string1> replaced by <character string2> wherever it appears in the program, no

matter how many times it is encountered. To replace only one line, see !!RE below. This command is pre-stored in memory but not yet executed.

!!LS␣$$ <character string> $$

causes a listing of the lines containing the <character string> specified to be printed immediately.

Note: When giving a !!CH command, take care that with a !!RE command <character string1> could be affected by the !!CH command.

Execution mode Commands:

!!JN␣⌈<line number>⌉,<name>,<line1>,<line2>

is used to insert the lines from <line1> to <line2> inclusive of the module named <name> after the <line number> of the current input. If <line number> is not specified, the lines are inserted behind the comment line of the main input.

!!RE␣<line number>,$$ <character string1> $$ <character string2>$$

is used to have <character string1> replaced by <character string 2> in the line of which the <line number> is specified.

Note: After !!RE immediate insertion of new lines is possible. An !!IL on the same line is not possible.

Further LED commands are:

!!DL␣<line1>⌈,<line2>⌉

which is used to delete the specified line or the lines form <line1> to<line2>inclusive.

!!IL␣⌈<line number>⌉

which is used to insert one or more lines after the specified line number or, if no parameter is specified, after the current statement. It is possible to insert a line before the first line of a coding sequence, e.g. if an IDENT statement has been forgotten.

!!AB is used to abort a line edit operation

!!EN which is used to terminate the line edit session.

Note:

- The commands must be given in ascending order of line numbers.
- Only sequential input files consisting of records of up to 80 characters can be handled.
- The definition commands are pre-stored in memory, not on disc.

OLE

OVERLAY LINKAGE EDITOR (P857 only)

OLE

syntax: OLE␣[N|S|U][,M][,[DE|DS]][,/<address>][,<start address>]

use: See Linkage Editor (LKE) command

errors: same as for LKE, plus:
USER OBDIR ASSIGN ERR:
SYST DBDIR ASSIGN ERR:
(error detected when system attempts to assign a file code to the system or user object directory. A following message will indicate the cause of the error).

UPR

USER PROCESSOR

UPR

syntax: UPR␣<processor name>[/S|<file name>][,NL]

use: This command allows for calling a user-made processor from the user library. <processor name> is the name it has as a load module file under the user directory. The other parameters have the same meaning as under ASM.
This feature is useful in conjunction with the GEN directive of the DOS Assembler, allowing the use of assemblers with predefined mnemonics (FORM) or symbols (equivalence tables) without affecting the standard Assembler in the system area.

example: - creating a special Assembler:

```
USERID: USER
MOV     ASM,/L,SAG
ASG     /20,DK
PLD     ASM,/L,/20
REF     /20
RDO     /20 (store assembler on /O file)
ASM     GEN (assemble GEN section with FORM directives and symbol
        definition)
LKE     N,M
KPF     /L,ASMSPE
```

- use of special assembler:

```
USERID: USER
UPR     ASMSPE, SOURCE (SOURCE is module to be assembled)
```

11 Operator Control Messages

There are a number of control messages which the operator can enter via the typewriter. To do so, he must first press the interrupt button on the control panel of the CPU (INT). The system then types out M; and the operator can type a control message.
Such a message consists of two characters identifying the message, followed by a space, possibly followed by one or more parameters, followed by CR - LF. The parameters are separated by spaces or commas.

Note:

- The operator communication package is an optional module. If this module is not included in the monitor, the operator must take care not to press the interrupt button (INT) on the control panel, because then the running program will be aborted.
  Every numeric value in an operator control message must be a hexadecimal value, specified with or without a slash (/). Between parameters, blanks or commas are allowed.
- If a message contains an error, the system types out a message ER. The operator may then press the INT button and type in a correct message.

In the following paragraphs, the syntax and use of the available messages are given. In the descriptions of the syntax, Backus Normal Form is used for the notation, i.e.

|	means:	or
[]	means:	optional component; any or all items within these brackets may be omitted: [+] [-] <integer> can mean -<integer>, +<integer> or <integer>.
	means:	alternative components; one of the items within these brackets must be selected: [+] [-] 426 can mean +426 or -426
␣	means:	space.
<>	means:	these brackets contain a syntactic item.

In the following paragraphs the operator control messages are specified in alphabetical order according to their mnemonics.

Message	Meaning	Page
AB	Abort a Program	118
AS	Assign a File Code	118
DM	Dump Memory	119
HD	Halt Dump	119
MC	Manual Device Control	119
PS	Pause	120
RD	Release Device	120
RS	Restart	120
RY	Retry an I/O operation	121
WM	Write into Memory	121

Table of Operator Control Messages

AB — ABORT A PROGRAM — AB

syntax: AB

use: This message definitively stops a running program. A message is sent to the operator by the system, specifying the program location where the abort occurred and the reason for the abort, in this case: OPERATOR ABORT. If a PU error message is given with retry possibility (RY) and the retry works, the abort will not be effectuated.

AS — ASSIGN FILE CODE — AS

syntax: AS␣<file code>␣<device address>

use: By means of this message a new file code can be created or the assignment of a previously created file code can be modified.
<file code> is the file code to be assigned, consisting of two hexadecimal characters from 01 to CF or from E0 to FF, except for F0 which is always the system disc.
<device address> is the physical address of the device to which the assignment is made, consisting of device name and number.
The following device names can be used:

TR:	ASR tape reader
TP:	ASR tape punch
TY:	operator's typewriter
PR:	punched tape reader
PP:	tape punch
LP:	line printer
CR:	card reader
TK:	magnetic tape cassette
NO:	no device; an operation on this file will have no effect.

example: AS 08 LP0D
File code 08 is assigned to the line printer with physical address 0D.

DM — DUMP MEMORY — DM

syntax: DM␣<address1>␣<address2>

use: This message causes a memory dump on the print file, which is usually the line printer. The dump is made in full lines.
<address1> is the beginning address of the memory dump (up to 4 hexadecimal characters)
<address2> is the ending address of the memory dump (up to 4 hexadecimal characters)

example: DM 0002 0004
The contents of memory will be dumped from hexadecimal address 0000 to address 000B.

HD — HALT DUMP — HD

syntax: HD

use: If this message is given, the output of a memory dump is stopped. This can be very useful if the dump is taking place on the operator's typewriter as this is a very slow device.

MC — MANUAL DEVICE CONTROL — MC

syntax: MC␣<device address>␣<order>[␣<repeat factor>]

use: This message can be given if the operator wants to do a manual operation on a magnetic tape device.
<device address> is the physical address of the tape unit, consisting of two hexadecimal characters.
<order> must be one of the following hexadecimal numbers, each one of which indicates a specific operation:

16: skip forward to EOF mark
22: write EOF mark
24: write EOV mark
26: write EOS mark
31: rewind to load point
33: backspace one block
34: skip forward one block *(not allowed on cassette)*
36: skip back to EOF mark

38: Unlock

<repeat factor> allows the operator to have the required function performed as many times as specified here, with only one MC message.

example: MC 15 34
The device with physical address 15 is to skip forward one block.

PS **PAUSE** **PS**

syntax: PS

use: This message causes the running program to be stopped temporarily.
In order to restart the program the operator must give the message RS (see below).

RD **RELEASE DEVICE** **RD**

syntax: RD␣<device address>

use: When the monitor has typed out a PU error message (peripheral unit failure) after an I/O operation and thus requests operator intervention, the operator may type in this message if he wants to release the operation on the device which resulted in the error message (see System Messages).
<device address> gives the address of the device which the operator wants to release and consists of two hexadecimal characters, as specified at system generation time.
After release of the device, control is returned to the user with the error status in word 4 of the Event Control Block.

example: RD 02
Release the last I/O operation on the device with physical address 02.

RS **RESTART A PROGRAM** **RS**

syntax: RS[␣<value>]

use: This message causes a program which has been stopped temporarily by a Pause monitor request or PS operator message, to be restarted.
<value> is a hexadecimal value of up to 4 characters which, if specified, will be loaded into the A7 register by the monitor.
This new value for the A7 register may be specified only if the program has been stopped by the monitor request Pause.

RY — RETRY AN I/O OPERATION — RY

syntax: RY␣<device address>

use: When the monitor has typed out a PU error message (peripheral unit failure) after an I/O operation and thus requests operator intervention, the operator may type this message to retry that same I/O operation, after he has taken any necessary steps (see System Messages).
<device address> gives the address of the device on which the I/O operation has to be retried and consists of two hexadecimal characters, as defined at system generation time.
If the operation succeeds now, control is returned to the user with the status in word 4 of the Event Control Block.
If it still does not succeed, a new error message may be output by the monitor if another retry is possible, or the operator may release the device (RD operator message).

example: RY 02
Retry the last I/O operation on the device with physical address 02.

WM — WRITE INTO MEMORY — WM

syntax: WM␣<address>␣<value1> [␣<value2>␣.....<valuen>]

use: This message can be used to correct the contents of one or more memory locations.
<address> is the first location of which the contents are to be modified (a hexadecimal address of up to 4 characters).
Values 1 to n are values which are to be entered into the memory locations starting from <address>, i.e.
<value1> is put in location <address>
<value2> is put in location <address> + 2, etc.

example: WM 4FE 44F 3FE4
The value 44F is placed in memory location 4FE
The value 3FE4 is placed in memory location 500.

13 Monitor Requests

The user program can request the monitor to perform certain functions. A request takes the form of an LKM (Link to Monitor) instruction followed by a DATA directive.
The directive has a number as operand, which specifies the function to be executed. If this number is negative, the user is scheduling a label on completion of the request.
Preceding a request, certain parameters may need to be loaded into the A7 and A6 registers.
After the monitor has processed the request it loads a return code in the A7 register. If the requested service module is not available, A7 will always contain the value -1.

Note: It is possible to use scheduled labels in conjunction with a monitor request. See chapter 4.

I/O REQUEST

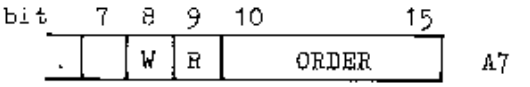
Calling Sequence

```
LDK     A7,CODE
LDKL    A8,ECBADR
LKK
DATA 1
```

Use

The user can ask the system to start a particular I/O operation on a peripheral device.
Processed at level 48 for physical I/O requests, at level 49 for logical I/O requests (Data Management).

Register A7 is loaded with a CODE specifying the details of the I/O function, as follows:



W and R specify the mode of operation:

W = 1: the requesting program wants to wait for the completion of the requested I/O operation. Only after completion of the requested function, will the return to the calling program take place.

W = 0: a return to the calling program will be made as soon as the transfer has been initiated. The program will give a Wait request later on for synchronization.

R = 1: the program itself will process any abnormal condition concerning the requested operation (possible only with Basic Read/Write). The system will return the hardware status in ECB word 4. No retry is possible.

R = 0: any abnormal conditions will be processed by the system. The software status is returned in ECB word 4.

ORDER specifies which I/O function is required, by giving one of the following hexadecimal values:

01: Basic Read

05: Basic Write.

For Basic I/O requests the system does not provide for character checking or data conversion, only for control command initialization and end of operation signals.

02: Standard Read

This order can be used to input standard object code records in 4 x 4 (4 rows of 4-bits: 4 track tape) or 8 + 8 (2 rows of 8 bits: 8-track tape) format, as well as ASCII character strings.

06: Standard Write

Standard (ASCII) I/O requests provide, by means of standard conversions, for special features such as error control characters, conversion from external code to internal ASCII and vice versa.

07: Object Write (4+4+4+4 tape format)

08: Object Write (8+8 tape format)

Object I/O requests provide, by means of standard conversions, for special features such as error control characters, checksum and data conversion from external 4+4+4+4 or 8+8 tape format to internal 16-bit format.

0A: Random Read

0B: Random Write

14: Skip forward to EOS mark

16: Skip forward to EOF mark

22: Write EOF mark

24: Write EOV mark

26: Write EOS mark

30: Get information about a file code

31: Rewind file.

33: Backspace one block

34: Space one block forward (not allowed for cassette)

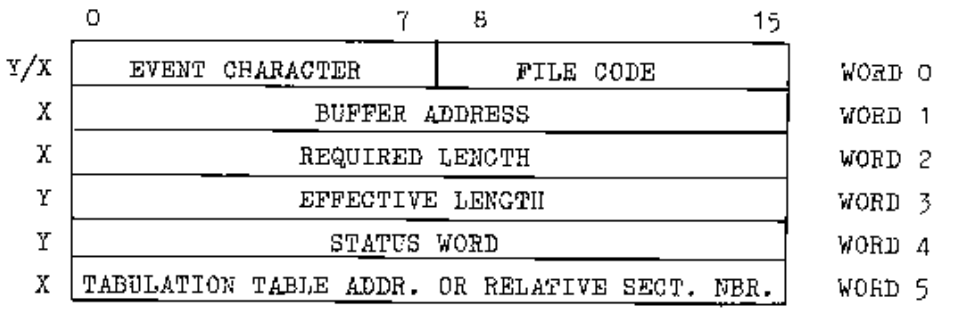
36: Skip backward to EOF mark

38: Unlock.

(Physical disc access on sector level is possible with orders /11 (Read) and /15 (Write). The Disc File Management module is not used in these cases. The file code in ECB0 must be one of the disc file codes F0 to FF. ECB5 must contain the absolute sector number of the sector which is to be accessed. Therefore, when a disc is shared by Disc File Management and user physical disc access, extreme caution must be exercised).

For each of these request orders, specific information applies to the various peripheral devices. This information is given in Appendix C at the end of the book.

The Event Control Block, of which the address must have been loaded into the A8 register, has the following format:



X: these words must be filled by the user
Y: these words are filled by the monitor

where:

WORD 0: event character:
bit 0 = 1: end of operation has occurred for the ECB.
The other bits remain unused.
WORD 1: address of the user buffer
WORD 2: requested length to be read or written, in words (basic read on card reader) or characters (other devices). The first character is always the character given by the buffer address. For standard write on typewriter or line printer,

two characters must be added, at the beginning of the buffer.

WORD 3: effective length which has been transmitted, in words (basic read on card reader) or characters (others). Stored here by the monitor upon completion of the I/O operation.

WORD 4: status word, stored here by the monitor upon completion of the requested I/O operation.

- For Basic orders, this word will be filled with the hardware status by the control unit. However, if the monitor detects an error in the calling sequence, bit 0 will be set to 1 and the other bits will contain the software status of page 128. Hardware status: Appendix D).
- For the other orders, the software status will be returned:

  bit 0 = 0: the operation has been successfully completed:
  bit 7 = 1: no data (tape cassette)
  bit 8 = 1: End-Of-Volume } cassette or
  bit 9 = 1: End-Of-Tape } magnetic tape
  bit 10 = 1: beginning of tape encountered.
  bit 11 = 1: end of input medium (disc only).
  bit 12 = 1: requested length is incorrect.
  bit 13 = 1: illegal character code.
  bit 14 = 1: an EOS mark has been read.
  bit 15 = 1: an EOF mark has been read.

  When the operation was not successfully completed, bit 0 is set to 1 and bit 1 is set to 0 (retry also was not possible). In this case bits 2 to 15 give the hardware status.
  When the monitor has detected an error in the calling sequence, bits 0 and 1 are both set to 1 and bits 2 to 15 have the following significance:

bit 10 = 1: file is write-protected.
bit 11 = 1: function is unknown or not compatible with the device or bit 9 in A7 set for a DFM request
bit 12 = 1: illegal buffer size or address
bit 13 = 1: illegal ECB address
bit 14 = 1: device is attached to another program.
bit 15 = 1: an illegal file code has been used.

WORD 5: this word is used by the user to store:

- the relative sector number to be exchanged in the case of a random disc file (see Data Management).
- the tabulation table address in the case of a standard read operation on ASR or punched tape equipment. This tabulation table has the following format.

Number of Tackets	First Tacket
Second Tacket	Third Tacket

The tackets indicate an absolute position in the print line. Characters up to the following tacket are filled with blanks.

Example:

3	10
20	30

Input line: LABEL \OPER \OPERAND \COMMENT
Line in buffer:
LABEL␣␣␣OPER␣␣␣␣␣OPERAND␣␣␣COMMENT
1 10 20 30
At completion of the input, the buffer is filled with spaces, but the returned length is the length effectively entered and stored, including the spaces replacing the tabulation codes (\).
Note: If word 5 is used for tabulation, the required length in word 2 must contain both the characters and the blanks in the tackets, i.e. for the above example word 2 must be filled with 36.

WAIT FOR AN EVENT

Calling Sequence

```
LDKL   A8,ECBADR
LKM
DATA   2
```

where:

ECBADR gives the address of the Event Control Block (see I/O requests). The first character of the ECB is the event character. If the first bit of this character is set to 1, the event has been completed.

Use

This request causes a program to stop and wait for the completion of an event which has to take place in another program (user or system). If the event has occurred, the dispatcher returns control to the requesting program. If the event has not occurred, the program is put in wait state, to be restarted when the event has occurred.

Note:

It is recommended not to use a Wait request inside a scheduled label routine, as this causes the whole program to be blocked temporarily.

EXIT

Calling Sequence

```
LKM
DATA 3
```

Use

This request is used to specify the end of a program. The program exit is effected after completion of all I/O operations and after all labels, if any, have been scheduled. A scheduled label exit passes control to the next scheduled label, if one is present, otherwise control passes to the main program.

In batch processing, register A7 contains an exit code in its right character (see under each monitor request).
If bit 0 of A7 is set to 1, the current job will be aborted up to the following JOB or BYE␣BYE command.

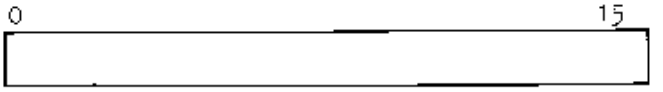
GET BUFFER REQUEST

Calling Sequence

```
LDK     A7, LENGTH
LKK
DATA 4
```

where:

LENGTH is the length, in characters, to be allocated to the buffer area (maximum 32k characters):

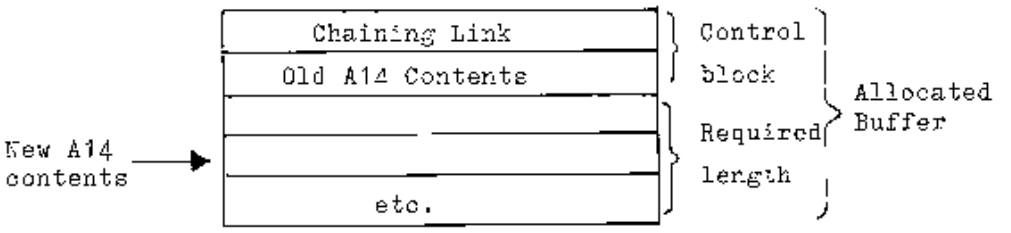


If 0 is loaded into A7, the monitor will return the highest address of memory in A7, except when the memory size is 32k, in which case 0 will be returned in A7.

Use

By means of this request, the user can allocate a memory area for temporary use, in the dynamic memory allocation area.

When the allocation is made, a control block is created by the system at the beginning of the allocated area. This block will contain a chaining link and the old contents of the A14 register:



The user must not destroy this control block

Upon completion of the request, the system responds as follows:

A7 = 0: the buffer is allocated

= 1: there is no memory space available (bit 0 in LENGTH = 0)

A14: contains the address of the fourth word of the allocated buffer, so that, as soon as the buffer is allocated, the user may give a Call Function instruction with the A14 register without having to update the A14 register first.

However, the user must then provide for stack handling.

RELEASE BUFFER REQUEST

Calling Sequence

```
LDKL   A14, BUFADR
LKM
DATA 5
```

where:

BUFADR points to the second word in the buffer as given in register A14 after the Get Buffer request.

Use

To release the memory space previously reserved by a Get Buffer request. The A14 register is reloaded with the value it contained before the Get Buffer request was made.
The system responds as follows:

A7 = 0: the memory space is released.
-1: the memory space has already been released.
If the A14 pointer is incorrect, or if the buffer area has been destroyed, the system issues a Halt.

PAUSE

Calling Sequence

```
LDK    A7,MESLGT
LDKL   A8,MESBLK
LKM
DATA 6
```

where:

MESLGT is a constant, specifying the length of a message which the program may output on being put in Pause state (in characters).
MESBLK is the address of a message block containing the message to be output.

Use

This request causes a temporary halt of the running program. It is put in wait state and, if specified, a message is printed out on the operator's typewriter. The program can be restarted only by an operator control message RS. When the program is restarted, it may be given an additional parameter in register A7 (see RS operator control message).

Notes:

- It is very useful if the user mentions, in the output message, that the program now is in pause state.
- The message must start with a control character.

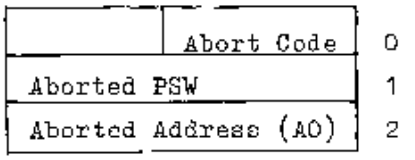
KEEP CONTROL ON ABORT CONDITION

Calling Sequence:

```
LDKL A7,PARAM
LDKL A8,LABADR
LKM
DATA 7
```

where:

LABADR is the address of a user label to be scheduled on abort.
PARAM is the address of a 3-word block which will receive the parameters of the abort condition. It has the following format:



where:

- the abort code has the same meaning as in the regular AB abort message.
- word 1 gives the PSW of the aborted program.
- word 2 gives the address where the abort took place (register A0 contents).

Use

By means of this request the user can himself handle abort conditions, i.e. his own routine replaces the abort handling of the monitor. Thus, the program will not be declared aborted by the monitor, but control will be transferred normally to the user routine attached to this request (register A8). The parameter of the abort condition will be placed in a 3-word block and, according to the returned parameters, the user routine can take action. This request is to be given once in a program, at the point from where the user wants control over any abort conditions, so mostly at the beginning of the user program. It is usable only once, so

if an abort takes place and the program is restarted, the request has to be given again. When a user decides that, at a certain point in his program, he wants to return to regular abort handling by the monitor, he may do so by provoking an 'artificial' abort.
The abort code returned in the PARAM block may be one of the following:

01: simulated routine save area overflow
02: non-available instruction used
04: buffer area destroyed or block was bigger than 16k words
05: label could not be scheduled
07: buffer overflow
08: disc overflow
09: disc queue overflow
0A: memory overflow during loading phase.

Note:
An operator abort (AB) can never be blocked.

LOAD A SEGMENT

Calling Sequence:

```
LDK   A7,NUMBER
LDKL  A8,ADDRESS
LKM
DATA 9
```

where:

NUMBER is the number of the segment which is to be loaded, as specified in the SEG control command. ADDRESS is the loading address of this segment.

Use

If a program is built up in segments and to be executed in an overlay structure, this request is used in the root segment or any segment, to load another segment into memory. For more details, see the section Programming and CCI control command SEG. The system response to the request can be found in A7:

A7 = 0: loading has been completed
A7 = 1: parameter error, e.g. segment number error
A7 = 2: memory overflow (segment too large)
A7 = 3: error in loading address.

APPENDICES

To standardize the system generation procedure for all systems, a set of system generation processors has been developed which provides great flexeibility, extensive logging of the process and improved efficiency.
The three steps inherent in any system generation process, i.e. monitor tables generation, monitor body generation and system medium generation are handled by a number of generation processors which are loaded and started successively.

For the generation of a Disc Operating Monitor these are:

- GENMON, a generation monitor used only during the system generation process to run the generation processors.
- DOMGEN, which generates the monitor tables from the answers it receives from the user in a conversational process with a standard list of questions.
- PREMDK, which is used to premark the system disc and write an Initial Program Loader on it, as the first module on the system disc. PREMDK runs under any monitor.
- GENLKE, which runs under GENMON and scans the library of system modules (DOMLIB), to select the ones requested during the DOMGEN phase and link them with the tables generated during DOMGEN.
- DISLOD, which, running under GENMON, is used to record the system processors on disc in load module format.

Depending on his configuration, the user may receive his sysgen tools, i.e. the above-mentioned processors, on punched tape or on cassette. In the first case, each processor is contained on a separate punched tape, in the second case all the processors are contained on one cassette.
In the description which follows in the paragraphs below, the cassette case is the basis, as this is the assumed standard for this sysgen process. It is very easy for the user, however, to redefine these standards (under GENMON) in case he works with punched tape. Apart from the redefinition of the standards, the main difference in the description is that from cassette the successive processors can be loaded and started without any manual

operation, whereas with punched tape, for each following step a new tape with the following sysgen processor must be put on the tape reader and then loaded and started.
Note: For DataCom SysGen, see Data Communication User Manual
In the following paragraphs, the whole set of operations necessary for the generation of a DOM is described in a number of sections corresponding to the system generation processors listed above. At the end of each section a number of notes and rema remarks is given, which the user must carefully read before starting the operation.

OPERATION

The minimum configuration required for generating a DOM is:
- CPU with 16k memory
- typewriter with address 10
- paper tape reader and punch, or
  two magnetic tape cassette drives on 1 control unit
- one X1215 disc unit

If the configuration is paper tape-oriemted, the user receives 25 tapes, containing:
- IPL + GENMON
- DOMGEN
- PREMDK
- GENLKE
- DOMLIB (the DOM Standard Library)
- DISLOD
- one tape for each of the monitor segments and system processors:
  CCI, CSEG1, CSEG2, CSEG3, CSEG4, CSEG5, LED (Line Editor), CSEG7, CSEG8, CSEG9, CSEGA, CSEGB, CSEGC, CSEGD, CSEGE, ASM (Assembler), LKE (Linkage Editor), DEB (Debugging Package), IPLGEN (IPL Generator).

If the configuration is cassette tape-oriented, the user receives two so called generation cassettes, containing:
- cassette 1: side A: IPL
  GENMON
  DOMGEN

PRENDK
GENLKE
side B: DOMLIB
DISLOD
- cassette 2: side A: monitor segments and system processors:
CCI
CSEG1
CSEG2
CSEG3
CSEG4
CSEG5
LED
CSEG7
CSEG8
CSEG9
CSEGA
CSEGB
CSEGC
CSEGD
CSEGE
ASM
LKE
DEB
IPLGEN
side B: ------

The user needs two cassettes or paper tapes of his own, to be used for intermediate storage of sysgen output.

Note:
Throughout this chapter, user replies typed in response to questions output by one of the sysgen processors, are underlined.

To start the process:
- switch on the CPU
- for cassette:
  - load generation cassette 1 (hereafter called cassette G1) in cassette drive TK05 with side A up
  - set the data switches on the CPU control panel to allow the bootstrap to load from TK05:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	1	1	1	0	0	0	0	1	0	1

(hexa 0785)

(for the significance of the bits, see your Programmer's Guide, Vol. 1; suffice it to mention here that bit 3 is 0 if the cassette drives are connected to the I/O processor and 1 if they are connected to the programmed channel, and bits 10 to 15 contain the device address.)

- for paper tape:
  - put the tape containing IPL +GENMON on the tape reader and make it operable
  - switch on the paper tape punch and feed tape
  - set the data switches on the CPU control panel to allow the bootstrap to load from the tape reader:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0

(hexa 1020)

(for the significance of the bits, see page 68 ; suffice it to mention here that bit 3 is 1 because the reader is connected to Programmed channel and that bits 10 to 15 contain the device address which is here assumed to be 20).

Then:
- press the MC button
- press the IPL button

Now the bootstrap is loaded which loads the first sysgen processor from either the cassette in drive TK05 or the paper tape reader into memory:

GENMON

GENMON is a special monitor used only during the sysgen process. To be able to do this, it must know the system configuration, the device addresses, interrupt levels and file codes. It is here that this system generation procedure shows its flexibility, for the great number of definition possibili-

tieswhich the user has at this point. GENMON outputs two questions, allowing the user to give his definitions and assignments. However, the system generation will handle a set of built-in standards if these are acceptable to the user. In this case he does not have to define anything, but if one of the user's assignments or definitions is different from the standard ones as listed below, he must redefine under GENMON.

When GENMON is loaded its identification is output on the typewriter:
GENMON
When loading is terminated,
:EOS
:EOF
is output on the typewriter, followed by the question
STANDARD CONFIGURATION?
The reply to this question can be Y[ES] or N[O].
If the user replies Y or YES followed by (LF) (CR), GENMON assumes the following standard configuration definition:

- typewriter : TY10 at level /6
- tape reader : PR20 at level /4
- tape punch : PP30 at level /5
- line printer : LP07 at level /17
- card reader : CR06 at level /15
- cassette tape : TK05 at level /14
  TK15 at level /14
  TK25 at level /14
- magnetic tape : MT04 at level /13
  MT14 at level /13
  MT24 at level /13

- X1215 disc : BM02 at level /10 (removable cartridge).

(It is not necessary for the user to have all these devices in his configuration to be able to answer YES; but the ones he does have must then correspond to these standards.)

If the user replies N or NO, i.e. if one or more of the device addresses or levels is different from the standards above, GENMON outputs the following list on the typewriter, thereby allowing the user to define the configuration himself:

```
TY:
PP:
PR:
LP:
TK:
MT:
CR:
DK:
DISK TYPE:     (type in B)
LKM LEVEL:                  (standard = 1)
RTC LEVEL:                  (standard = 2)
PANEL INTERRUPT LEVEL:      (standard = 7)
```

For each of the devices listed, the user can reply as follows:

- (CR) if he wants the standard address and level (see above);
- <address>,<level> if one of these is different from the standards, followed by (CR)
- N or NO followed by (CR) if he wants the device excluded from the system.

When the user has terminated his reply to the first question, GENMON types out:

```
STANDARD FILE CODE ASSIGNMENT?
```

The procedure here is the same as for the first question: the reply may be either Y[ES] or N[O].
If the user replies Y or YES followed by (LF) (CR) GENMON assumes the following standard file code assignments:

- file code 1 : TY10 (system keyboard)
- file code 2 : LP07 (listing output)
- file code 3 : TK15 (object output)
- file code 4 : TK05 (object input)
- file code 5 : TY10 (system keyboard)
- file code 6 : TK05 (object input)
- file code 7 : TK25 (object input)
- file code 8 : PR20 (object input)
- file code A : TY10 (sysgen source input)

- file code B : TK15 (sysgen object output)
- file code E0: TY10 (system keyboard)
- file code E2: TK05 (Disload object input)
- file code F0: BM02 (Disload disc output;i.e. X1215)

If the user replies N or NO to this second GENMON question, i.e. if one or more of his file code assignments is going to be different from the standard list above, which is the case when the user works with paper tape, GENMON outputs the following list on the typewriter, thereby allowing the user to give his own file code assignments (since the GENMON processor is the same for all monitors, some of the file codes given in this list are irrelevant to the generation of a DOM and the user must type in NO after those):

```
LOAD INPUT DEV. AND MAIN LKE INPUT DEV. F.C./4: (standard = TK05)
SYSGEN INPUT DEV.                       F.C./A: (standard = TY10)
SYSGEN OUTPUT DEV.                      F.C./B: (standard = TK15)
AUX. LKE INPUT DEV 1                    F.C./6: (standard = TK05)
AUX. LKE INPUT DEV 2                    F.C./7: (standard = TK25)
AUX. LKE INPUT DEV 3                    F.C./8: (standard = PR20)
AUX. LKE INPUT DEV 4                    F.C./9: (no standard)
IPLGEN/LKE/CASLOAD OUTPUT DEV.          F.C./3: (standard = TK15)
LISTING OUTPUT DEV                      F.C./2: (standard = LP07)
CASLOAD INPUT DEV.                      F.C./C: (type in NO)
DISLOAD INPUT DEV.                      F.C./E2:(standard = TK05)
```

For each of the file codes listed, the user can reply as follows:

- (CR) if he wants the standard assignment (see above);
- <dev.name><dev.address> if one of these is different from the standards, followed by (LF) (CR)
- N or NO followed by (CR) if he does not want this file code taken into account
- if there is only one device of its kind, e.g. one PP, or in case a device which must be taken is the first of a series encountered in the standard list above, e.g. TK05, it suffices to specify only the device name, i.e. PP or TK.

When the user has terminated his reply to this question, GENMON types out:

END OF GENMON INITIALIZATION

READY TO LOAD PROGRAMS

Now the user can proceed to the next phase: DOMGEN.

Notes on GENMON:

For the question STANDARD CONFIGURATION:

- From the time the MC button has been pushed up to the end of GENMON initialization, no ready interrupts should occur.
- If the user has answered N or NO to this question, in the list typed out by GENMON, specification of a level is mandatory for LKM LEVEL. For RTC LEVEL it is mandatory if the CPU key is in the RTC/ON or LOCK position. For PANEL LEVEL it is also mandatory.
- The standards imply that the system disc will be the removable cartridge of the X1215 disc unit; therefore, if the user wants his system on the fixed cartridge, he must redefine DK under this question, e.g. as BM22.

For the question STANDARD FILE CODE ASSIGNMENT:

- File code /4: all programs will be loaded from this file code. During the syslink phase it is used as GENLKE object input, so it must be cassette or punched tape.
- File code /A: from this file code the parameters for table generation will be read. This may be done in interactive mode (e.g.TY) or not (PTR, cassette, magnetic tape, card reader).
- File code /B: this may be cassette, punched tape or magnetic tape.
- AUX. INPUT DEV.: these may be assigned in advance, especially for syslink if libraries are to be scanned on various devices.
- File code /3: this is the main output device (sequential), i.e. cassette tape, punched tape or magnetic tape.
- File code /2: for logging of the sysgen operation: LP.
- File codes /4, /A, /B and /3 are mandatory; file codes /6 up to /9 and /2 are optional.

When answers to GENMON questions are given on an ASR type-

writer, they must not be typed in before the bell signal because of the low speed of the GENMON I/O module. This is not necessary for devices on V-24 interface such as the matrix typewriter P842, because they work in echo mode.

DOMGEN

When GENMON initialization is terminated and the message READY TO LOAD PROGRAMS has been output the user can start the second phase, DOMGEN, the building of the DOM tables. This is done by typing replies to questions output on the typewriter by DOMGEN. From these replies DOMGEN builds the tables and records them on the medium with file code /B. i.e. in standard cases the cassette in drive TK15.
When the questions and answers are handled via the typewriter, this phase is done in conversational mode. It is also possible however, to do it in non-conversational mode, for example by having the questions and answers pre-recorded on punched tape. In such a case, file code /A must have been assigned to a tape reader during the GENMON phase under the question STANDARD FILE CODE ASSIGNMENT?

The following actions must now be taken:
- when working with paper tape, take the IPL+GENMON tape from the reader, put the DOMGEN tape on it and make it operable; prepare the tape punch for output by switching it on and feeding tape
- when working with cassettes, put the first working cassette (hereafter called cassette W1) in cassette drive TK15; this cassette will receive the DOMGEN output.
- load a disc cartridge in the X1215 disc drive (this is the disc which will later on receive the generated system, depending on the disc defined under DK in the question STANDARD CONFIGURATION during GENMON)
- push the INT button on the CPU control panel
- on output of M: type in
  LD
- now the following sysgen processor, i.e. DOMGEN is loaded from the cassette or paper tape and its identification is output:
  IDENT DOMGEN

- when loading is terminated,
  :EOS
  :EOF
  is output on the typewriter

(At this point, if file codes /A and/or /B have been redefined under GENMON, i.e. if they are not assigned to TY10 or to a cassette respectively, prepare the relevant device. Normally, the cassette in drive TK15 should now be ready to receive the tables output by DOMGEN according to the replies given on TY10.)

- push the INT button
- on output of M: type in
  ST
- now DOMGEN is started and outputs the following messages on the typewriter:
  TABGEN INITIALIZATION
  IDENT SYSTEM DEFINITION
- then, if the user works in conversational mode, a series of questions is output, to which the user must type in the replies:

IDENT

Reply: Specify a character string of up to 6 alphanumeric characters, to be punched at the beginning of the module. This may be followed by a blank and a comment field of up to 73 characters.

Error Message: TG03: the first charactersis not alphabetic.

STACK SIZE

Reply: Specify 4 hexadecimal characters, giving the size, in words, of the stack which is used by the system to save registers when an interrupt occurs. Note that 10 decimal words are required in the stack for each accepted interrupt.

Error Message: TG03: the reply is not a hexadecimal number.

USER INTERRUPT ROUTINES

Reply: Specify all user interrupt routines (usually drivers for special devices) which must be link-edited with the monitor for inclusion in the system. as follows:

<L>,<ENTRYi>
:
:
END

L, ENTRYi specifies the name (up to 6 characters) of a routine connected to the single level L.

END indicates that all routines have been declared or is used if none are entered.

Error Messages: TG03: syntax error
TG06: the first character of the name (ENTRYi/j) is not alphabetic
TG09: error in the level declaration.

POWER FAILURE

Reply: Specify the level of power failure option as follows:
<L> giving the level number 1 or 2 hexadecimal digits.

Specify N if this feature is not available in the system.

Note: The standard system power failure routine performs only a HALT.

Error Message: TG03: parameter error.

REAL TIME CLOCK LEVEL

Reply: <L>
where L is the level to which the clock is connected

Reply: First specify the level to which the LKM interrupt is connected in the same manner as for the previous question. Then, define the optional monitor requests derived for your system (other monitor requests provided in the software are standard). DOMGEN will print the names of the optional ones, after which the user may type Y if he wants it to be included or N if he does not.
For each of the monitors a different list appears:

IORM (I/O Requests)
WAIT (Wait for Event) } necessary for standard system
EXIT (Exit)
BUFF (Get/Release Buffer): required for FORTRAN
PSE (Pause): required for cassette and magnetic tape handling and for the PSE control command.
ABRT (Abort Control): required for Debug
SLD (load a Segment): } necessary for
DFM (Disc File Management) } standard system

Note: If the user gives, later on in a running program, a request for a function which he has not included at this stage, the value -1 is returned in the A7 register.

Note: The user must not issue a request in his programs for a function he has not included. For example, if he has typed N after DFM at DOMGEN, an I/O monitor request (LKM1) on a logical disc file may cause a system hang-up. Therefore users should define exactly the required functions.

Error Message: TG06: the reply is neither N nor Y. Try again.

USER LKM

Reply: The user may specify his own set of monitor requests for inclusion in the monitor, as follows:

<N1>,<ENTRY1>
<Ni>,<ENTRYi>
<Nn>,<ENTRYn>

END

where Ni consists of two hexadecimal digits defining the DATA number which follows the LKM instruction, and ENTRY is the symbolic entry point of the routine processing the LKM DATA Ni function. The system will extend the monitor request table in which word i contains the address of ENTRYi. Therefore, during SYSLINK the user must provide GENLKE with the module containing all entry points ENTRYi specified here.
END indicates that all user LKM functions have been declared or that none are wanted.
All user LKMs are processed by core resident programs working at level 4B.

Note: Ni must not be equal to any of the standard LKM DATA numbers.

Error Message: TG05: the first parameter is not hexadecimal
TG07: the first character of the second parameter is not alphabetic
TG08: syntax error

PANEL LEVEL

Reply: When operator communication (OCOM) modules are selected, the user must specify the level to which the control panel interrupt is connected.
The level is specified as:
<L> a level number of 1 or 2 hexadecimal digits

Specify N if no operator communication feature is used.

Error Message: TG03

Reply: Specify, if the operator communication package must be included in the system by typing:
Y if it must be included, or:
N if it must not be included.
If the reply was Y, DOMGEN will print, one by one, the operator commands available. By typing Y or N after each one, the user indicates whether he wants it to be included or not.
The following list is output by DOMGEN:

DM (Dump Memory)
MC (Manual Control) (required for magnetic and cassette tape handling)
HD (Halt Dump) (recommended if DM has been selected)
WM (Write into Memory)
RY (Retry Device) } (see Note below)
RD (Release Device) }
AS (Assign a File Code)
AB (Abort a Program)
PS (Pause)
RS (Restart) (necessary with cassette or magnetic tape)

Note: - When the commands RY and RD are not selected, the system will stop when an I/O error occurs which requires operator intervention. In a configuration with only one typewriter these two commands are needed, but if more devices are used, they are highly recommended.
- RS is required if PS or PSE (LKM) has been selected.
- If N has been replied to the question OCOM OPTIONS, the abort option may be selected as an LKM option or under the question ABORT MODULE. (Pressing the INT button will then abort a running program). If, in add-

ition, Y has been replied under DUMP IN ABORT, pressing the INT button during the memory dump output will simulate an HD command.

USER COMMANDS

Reply: Specify any user-made operator commands which must be included in the system, as follows:

⟨CM1⟩,⟨ENTRY1⟩
⋮
⟨CMn⟩,⟨ENTRYn⟩
END

Where CMi consists of 2 characters defining the operator command and ENTRYi specifies the entry point of the module which processes command CMi. END indicates that all commands have been declared or that none are required.

Note: CMi must not be equal to any of the standard operator commands.

Error Message: TGO3: syntax error.

DEVICES ON PROGRAMMED CHANNEL

(Note: devices on channels must be declared according to Release Bulletin)

Reply: Specify, which devices are connected to the programmed channel, as follows:

⟨DNDA⟩,⟨L⟩
⋮
END

where:
DN is the device name, in two ASCII characters.
DA is the device address, in two hexadecimal digits.
Note: When the configuration contains only 2 cassette drives (TK), the physical addresses (DA) of these drives must be /0X and /1X, X being the

control unit address.
<L> specifies the level to which the device is connected.

END indicates that all devices have been declared.

Note: For the operator's typewriter, three devices(TY, TP, TR) may be declared with the same address, if they are all used. No check is made on device declaration.

Error Messages: TG01: the device specified is not supported by the system.
TG02: device address error
TG09: level error
TG10: device not declared.

Device names used:

TR : ASR tape reader
TP : ASR tape punch
PR : high-speed tape reader
PP : high-speed tape punch
TY : operator's typewriter
CR : card reader
LP : line printer
TK : cassette tape unit
MT : magnetic tape unit

DEVICES ON MULTIPLEX

Reply: Specify, which devices are connected to the multiplex in the same manner as for the devices on the programmed channel.
Note: For line printer there are additional parameters:
- Line printer:
  LDPA,L,LG, number

where LG=[S/L] : S = 80-column printer
L = 132-column printer.

<number> : a hexadecimal number specifying the number of lines per page wanted.

No check is made on the device declaration.

DISK UNITS

Reply: Specify all disc units connected to the I/O processor as follows:

DNDA,L,Q

END

where:

DN is the device name, giving the type of disc on 2 ASCII characters as:

BM: X1215 moving-head disc (model 1).

Note: For the X1215, each disc pack, not disc unit must be declared, for there are 2 packs to one unit.

DA is the device address, on 2 hexadecimal digits.

L specifies the interrupt level to which the disc is connected (1 or 2 hexadecimal digits)

Q defines, on 2 hexadecimal digits, the length of the disc queue table, in number of entries

Note:

- The disc queue table is used only if DFM has been selected. If it has not, this parameter must be 0. The number of entries in the disc queue table must be equal, for each disc unit, to the maximum number of disc logical I/O which may be processed simultaneously on the same unit. The answer must therefore be equal to the one given under DISC LOGICAL FILES.

SPECIFY FILE CODES

Reply: Specify all file codes used by the programs. If system processors are used, their standard file codes must be declared.
Declaration is done as follows:

```
<FC>,<DNDA>
     ¦
<FC>,<DNDA>

   END
```

where FC is a file code (2 hexadecimal digits) assigned to the device indicated by DN with address DA.

Note

File Code EF and 01 are used by the system for input/output to/from the operator's typewriter. It cannot be assigned or reassigned by AS command. So, if used, it is necessary to declare at least this file code. All other file codes may be assigned at execution time by means of the operator command AS or the ASG control command.

Some file codes, such as 02, 03, E0, E1, E2 are used by the system. These must also be specified, if any of the standard processors are used.
At least one file code between F0 and FF should be defined for each disc.

FILE CODES ASG TO USER DEVICES

Reply: The user may declare all file codes assigned to his non-standard devices. The related I/O drivers (including Device Work Tables) and the interrupt routines for these devices must be written by the user and be incorporated in the system during the GENLKE phase. (The entry points for the interrupt

routines must be declared under USER INTERRUPT ROUTINES). These file codes must be declared here, as they cannot be assigned by AS operator message or ASG control command. The reply must be as follows:

```
<FC>,<DWTi>
    :
   END
```

where:
FC is a two-digit hexadecimal file code which will be generated in the File Code Table.
DWTi is the entry point (a name of up to six characters) of the Device Work Table (DWT) associated with this device.
END indicates that all file codes have been declared.
Any number of file codes can be assigned to the same DWTi. The system checks only if the file code is a two-digit hexadecimal number, but not if it has already been used or is one of the standard file codes used by the system processors.

Error Message: TG03: syntax error, e.g. the file code has not been specified as hexadecimal.
TG07: the first character of the DWTi is not alphanumeric.

SPARE ENTRIES IN FCT

Reply: Specify the number of spare entries reserved in the File Code Table (FCT), on one or two hexadecimal digits.
These entries are required for assignment of user file codes at execution time or for assignment of temporary system file codes.
Each entry takes up two words.

Apart from those already defined, the system requires at least five entries, and one for the Line Editor JN command and one for the catalogued procedures, if used.

DISC LOGICAL FILES

Reply: This message is printed only in case the user has selected the Disc File Management (DFM) option (see under LKM LEVEL), to find out the number of disc logical files used in system.
Specify two hexadecimal digits, giving the maximum number of logical files which can be assigned (and not yet opened) at the same time.
Each entry takes up 28_{10} words, as the system will reserve a Logical File Table for each disc logical file, to store all the information about it.

The system itself requires at least five files, and one for the Line Editor JN command and one for catalogued procedures, if used.

DISK BUFFER POOL

Reply: This message is printed only in case the user has selected the Disc File Management (DFM) option (see under LKM LEVEL). Specify the number of blocking buffers used for blocking and deblocking logical records of sequential files or for the granule table in case of random access files. The system requires at least two and one for the Line Editor JN command and one for the catalogued procedures, if used. (Not more buffers than the maximum number of logical files opened at the same time. (See Data Management)).

DISK ALLOCATION TABLE

Reply: Specify the length, in characters, of the disc allocation table on two hexadecimal digits. (For X1215: /66).

DUMP IN ABORT

Reply: Specify Y if a post-mortem dump is wanted in case of abort, or N if it is not wanted.

MAXIMUM NUMBER OF SEGMENTS

Reply: Specify the maximum number of segments used by any of the programs, on two hexadecimal digits.

Note:
The Control Command Interpreter (CCI) uses 14 segments.

ABORT MODULE (This question is printed only if N was replied for AB under OCOM OPTIONS and ABRT under LKM LEVEL)

Reply: If neither the monitor request 'Keep Control on Abort' nor the operator command AB have been selected the user may include the System Abort Module by typing Y.
If the reply is N, the abort module will not be included. Abort of a user program will, if the module has been selected, result in an output message and the user may enter a new command to run the next job. If the module has not been selected, an abort will stop the machine, through a HLT instruction.

Error Message: TG06: invalid reply.

SIMULATED INSTRUCTIONS

Reply: Y or N, depending on whether the simulation package must be included or not. If the reply was Y, the following list is output:
MULTIPLY:
DIVIDE:
D ADD:
D SUB:
For P856/P857 the answer is N, for these instructions are included in the instruction set.

After each item the user must type in Y or N to indicate whether he wants that instruction simulation routine included or not.

Note: If the CCI package is to be used, the reply for this question must be Y.

SIMULATED ROUTINES SAVING AREAS NB

Reply: This question is output only if the reply to the previous question was Y.
The user must type in the number of save areas required by the simulation package.
This is 1, or, if scheduled labels are used, 2.

MAX NUMBER OF SCHEDULED LABELS

Reply: Type in a two-digit hexadecimal number, specifying the maximum number of scheduled labels which may be in queue at the same time. This will be the length of the FILLAB table described in the paragraph on Scheduled Labels in Chapter 4.

Note: This is not the maximum number of scheduled labels used in the program, which may be a higher number.

TABGEN ENDED

PREMDK

This is a program which checks the disc on which the generated system must be written for defective tracks, writes sector identifiers on it and an Initial Program Loader (IPL) in the second sector.
PREMDK outputs a number of questions on file code /EF (i.e. TYIO) to which the user must type in the replies. Output is done onto file code /F0, i.e. the system disc.
Operation is as follows:

- with paper tape, put the PREMDK tape on the reader; make it operable
- push the INT button on the CPU control panel
- on output of M: type in
  LD
- now PREMDK is loaded as the next processor from the cassette G1 in drive TK05 or the tape reader and its identification is output:
  IDENT PREMDK
- when loading is terminated,
  :EOS
  :EOF
  is output on the typewriter.
- push the INT button
- on output of M: type in
  ST
- now PREMDK is started and the following messages and questions are output on the typewriter:
  INITIALIZATION OF PREMRK
  NBR OF CYLINDERS = (type in 4 decimal digits giving the number of cylinders on the disc, followed by (LF) (CR) Example: 0203)
  NBR OF TRACKS = (type in 4 decimal digits giving the number of tracks per cylinder, followed by (LF) (CR) Example: 0002)
  NBR OF SECTORS/TRACK = (type in 4 decimal digits giving the number of sectors per track, followed by (LF) (CR) Example: 0016)
  DISK TYPE = (for the X1215, this must be CM followed by (LF) (CR))
  DISK UNIT PHYSICAL ADDRESS = (type in two hexadecimal digits giving the physical address of the disc, followed by (LF) (CR) Example: 32)

LABEL = (type in a volume label, consisting of up to 8 characters, followed by (LF) (CR) Example: TONTO)

DATE = (type in 6 decimal characters, separated by delimiters, e.g. spaces or slashes, followed by (LF) (CR) Example: 01/01/84)

PACK NBR = (type in 3 characters giving the number of the disc pack, followed by (LF) (CR) Example: F32)

SYSTEM USERID = (type in up to 8 ASCII characters, specifying the first user of this disc pack, i.e. the user identification of the system. This is to enable the pack to be loaded by DISLOD. It must be followed by (LF) (CR) Example: SILVER) It is mandatory to answer this question.

When the user has answered these questions, PREMDK starts premarking the disc and outputs the messages:

WRITING THE IDENTIFIERS

CHECKING THE IDENTIFIERS

END OF CHECK

NBR OF BAD GRANULES = XXXX

RUN AGAIN? (to this last question the user must type in NO if he wants no other disc premarked; if he types in OK, the above procedure is repeated; after NO PREMDK types out its last message:)

END OF PREMRK

Error Messages:

- BAD GRANULE 0: the disc pack is not usable
- NO SYSTEM USER POSSIBLE: granule 1 is bad and therefore no system can be stored on this disc pack.

Now the disc is premarked and ready to receive the generated system. But first we have the GENLKE phase, during which the tables generated under DOMGEN are linked with the modules required from the DOM Library and, possibly, user and/or extension libraries to generate the user's monitor.

GENLKE

During this phase the final user monitor is obtained by linking the tables generated under DOMGEN with the monitor modules required from the DOM Library and/or any user library or extension libraries (see note at the end of this section).
The input to GENLKE is done from file code /4, i.e. cassette or paper tape. The output is done onto file code /3, in standard cases working cassette W2 in drive TK15, but it may also be punched tape or magnetic tape.
In any case, the GENLKE processor must now be loaded:

- when using paper tape, take the PREMDK tape from the reader, put on the GENLKE tape and make the reader operable
- push the INT button on the CPU control panel
- on output of M: type in LD
- now GENLKE is loaded from the cassette tape or the tape reader and its identification is output:
  IDENT GENLKE
- when loading is terminated,
  :EOS
  :EOF
  is output on the typewriter.

With cassettes:

On the basis of the availability of two cassette drives, the cassettes are now handled as follows (with three drives, see below):

- take cassette G1 from drive TK05 after GENLKE has been loaded
- take cassette W1 (containing the tables generated under DOMGEN) from drive TK15 and put it into drive TK05 and wait for it to be rewound
- put the second working cassette W2 into drive TK15

- now start GENLKE as follows:
- push the INT button
- on output of M: type in
  ST
- GENLKE outputs
  L:
  and the user must type in the link-edit command as follows:

E[:<decimal number>],<module name>[,[8|4]]

where <decimal number> consists of three digits:

- the first is the file code for the object output device
- the second is the file code for the listing device
- the third is the file code for the object input device.

<module name> is the name of the user's system

8 or 4 is used if the monitor is punched on paper tape to indicate whether it must be punched in 4-track or 8-track format.

If the standard file codes are used (see under GENMON, i.e. /3, /2 and /4 respectively), they need not be specified and the command can be given as:

E,<module name>

- then GENLKE outputs

L:

to which the user must reply with

P

The tables generated during DOMGEN are now recorded from the cassette W1 in drive TK05 onto cassette W2 in drive TK15.

- when this is finished take cassette W1 from drive TK05
- put generation cassette G1 into drive TK05 with side B up (DOMLIB) and wait for it to be rewound, so that it is positioned correctly for the scanning of the DOM Library.
- in response to the

L:

output by GENLKE, now type in

L

upon which GENLKE will start scanning the DOM Library, select the required modules and record them onto the cassette W2 in drive TK15. The names of the selected modules are output on the listing device, together with their base addresses and any comments included in the identifiers.

- when this is finished, GENLKE again types out

L:

The user must now type in

U

to check if there are any unsatisfied references. The last module to be included must be INIMON, so if GENLKE types

INIMON

after the user has typed in his first U, it is correct.
The GENLKE processor then types out
L:
and the user can type
L
to solve his last unsatisfied reference.
(If there were more unsatisfied references, the user must repeat this L:L process until INIMON is the last unsolved reference and then give his last L command.)

- now, after all modules have been included, GENLKE again types
  L:
- the user once more types
  U
  to make sure that all references have been solved. Then, on the next
  L:
  the user types
  T
  to indicate the end of the GENLKE phase.
- GENLKE then outputs the symbol table of the generated DOM on the listing device and on the typewriter it outputs monitor length (L = XXXX), monitor start address (S = XXXX) and the first free location after the monitor in memory (E = XXXX).

Note: If the user has three cassette drives and wants to use them all during this phase, the procedure is as follows:

- after loading GENLKE, take cassette C1 from drive TK05 and put it back in with side B up (DOMLIB). Wait for it to be rewound.
- put cassette W1 (tables output by DOMGEN) into drive TK25 and wait for it to be rewound
- put the second working cassette W2 in drive TK15
- start GENLKE by pushing the INT button and, on output of M: typing in ST
- on output of L: type in the option message as follows:
  E:327,<module name>,8
  where 3 and 2 are the normal object output and listing file codes but 7 is assigned now to the input file code , so not /4 = TK05, but /7 = TK25. This is because TK25 contains

the cassette W1 with the tables generated under DOMGEN. See also under GENMON for the file code assignments.

- when this is accepted, GENLKE outputs L: and the user types H

  Upon this command, GENLKE scans the input file (i.e. the cassette W1 with the tables) up to EOF and then goes into Pause state.

- now GENLKE is restarted with an RS command with a new file code, switching it back from /7 to /4, the normal input file code and the one assigned to TK05, which contains the DOM Library which must now be scanned. So:

- push the INT button and on output of M: type

  RS 04

  GENLKE now starts scanning and selecting the required modules from the DOM Library and the rest of the procedure is the same as described above, starting after the user's first L command.

With paper tape:

- in this case the procedure is basically the same as with cassettes (see description above), but input is done from the tape reader and output onto tape punch. On the tape punch an IPL has already been generated and the paper tape should be left as it is.
- first the GENLKE tape must be put on the reader and GENLKE is loaded into memory by LD command. Then the tape containing the tables generated under DOMGEN is put on the reader and GENLKE is started with an ST command. Having entered the E: option command, the user types P and the tables are processed.
- then the DOMLIB is put on the tape reader and the user types L after which this library is scanned and the selected modules are output on the punched tape.
- having checked if INIMON is the last unsolved reference with a U command and having typed L to solve it, the user then types T to terminate the process.
- on the tape punch, a DOM has now been generated.

Note:
If modules from other (paper or cassette) tapes beside the DOMLIB tape must be link-edited during this phase, the tapes must be scanned in a defined order, which is:

- User Library tape(s), if any
- Extension tape(s)
- DOMLIB tape

When more references than INIMON remain unsatisfied, rescanning must start at the first step in this sequence.

Now the user has generated his Disc Operating Monitor on the device with file code /3.
The only thing that remains to be done now is to store the monitor together with the system processors on the system disc. This is done during the following phase, DISLOD.

DISLOD

DISLOD is the last system generation processor. It is used to record the generated monitor, monitor modules and system processors onto the system disc specified under GENMON, in load module format accepted by the DOM.
The standard object input file code for DISLOD is /E2 (standard TK05), the output disc file code is /F0 (BM02), the listing file code is /2 (LP07) and all conversational processes are done through file code /EF (TY10).

- when working with paper tape, put the DISLOD tape on the tape reader and make it operable
- with cassette, the cassette tape in file code /E2 is positioned after the DOMLIB, i.e. before DISLOD
- push the INT button on the CPU control panel
- on output of M: type in LD
- now DISLOD is loaded from the tape reader or the cassette and its identification is output:
  IDENT DISLOD
- when loading is terminated,
  :EOS
  :EOF is output on the typewriter.
- with paper tape, DISLOD tape must now be taken from the reader and the newly generated DOM tape must be placed on it, ready to be read
- with cassettes, take cassette G1 from file code /E2 (normally TK05) and replace it by cassette W2 (containing the new DOM) which must first be taken from file code /3 (normally TK15).
- push the INT button
- on output of M: type
  ST
  to start the DISLOD processor.
- DISLOD then outputs the message
  SYSLOAD XX P852
  on the typewriter, followed by the question
  NEXT ACTION:
- at this point, the user has three possibilities for action:
  - if he types (LF) (CR) the next program or module on the input file (/E2) will be recorded onto the system disc. DISLOD will output the program name onto the typewriter (PROG.NAME = XXXXXX) and list name, length in sectors,

start address and program length on the listing device (/2).

After this, DISLOD will again output the message NEXT ACTION:.

- if he types PS , DISLOD goes in Pause state, enabling the user to modify an assignment, operate on the cassettes or tapes manually, etc. To restart DISLOD, the message RS must be typed in (after having pushed the INT button), which may be followed by a parameter containing a new object input file code.
- if he types HT , DISLOD performs an Exit and the process is stopped.

- so, after the first NEXT ACTION : message output by DISLOD, the user types (LF) (CR) and his newly generated monitor is recorded from cassette W2 or paper tape onto the system disc specified by the user under GENMON and premarked under PREMDK.
- when this is finished, DISLOD again outputs
  NEXT ACTION:
- with paper tape, now put the first of the monitor segment and system processor tapes on the reader, i.e. the CCI tape (see list of sysgen tapes at beginning of chapter)
- with cassettes, now take cassette W2 out of the drive and replace it by the second generation cassette (G2), with side A up, containing the disc-resident parts of the monitor (CCI segments) and the system processors LED, ASM, LKE, DEB and an IPL generator IPLGEN.
- now, after each NEXT ACTION: message, the user must type (LF) (CR) to include successively the disc-resident monitor parts and the system processors. With cassettes, this requires no further manual operations, but with paper tape, the user must put the following tape in the reader each time before typing in (LF) (CR)
- when the last one (IPLGEN) has been recorded onto the system disc and DISLOD again types
  NEXT ACTION:
  the user must type in
  HT
  to terminate the DISLOD operation.

The user now has a complete system on disc. If he wants to record any other processors, such as FORTRAN and its Library, on it, this can be done with the relevant CCI commands of the generated Disc Operating Monitor. For this purpose, press the MC button and load the DOM.

System Generation under DOM Control

If the user already has a disc system and a Disc Operating Monitor, he can use this DOM to generate any system : BOS, DOS, BRTM or DRTM. If his DOM includes the Cassette File Management Package, it can also be used to generate a Cassette Operating System.

The necessary elements are on punched tape or cassette tape:
- the XXXGEN processor (BOMGEN, DOMGEN etc.)
- the Monitor Library

These must be recorded and arranged on disc.
For each system to be generated a special user identification must be declared (userid).

Let us assume such a userid is declared as GENUSER. This userid must then include the XXXGEN processor and the monitor library, except for INIMON (+ ASEX for DRTM):
- the tape containing XXXGEN is put on the tape reader
  (if it is on cassette tape, the procedure is:
  - load cassette C1, side A
  - assign file code /E2
  - give the CCI command FFS /E2
    This will position the cassette tape after the tape mark following GENMON, i.e. just before the XXXGEN processor. The monitor library always starts at the beginning of a cassette side.)
- give the CCI command RDO
  followed by LKE
  and KPF /L,XXXGEN
  to store the XXXGEN processor on the disc.
- then the library tape is put on the reader (or cassette)
- give the command RDO for each tape (if there are more tapes or more than one cassette side)
- then the command KPF /O . Keep the last tape on the reader.
- give the command BYE
  followed by the userid SYSTEM to declare a system session
- then the command RDO to read INIMON (+ ASEX for DRTM) as the last

modules and keep them with the command

KPF /O

Note: If systems including extensions such as Data Communication or Cassette File Management must be generated, these extensions must be read at this point:

- give the command BYE
  followed by the userid GENUSER
  followed optionally by RDO <extension> commands to include the extension or user library modules. This is done in this way because the extension may contain modules with the same names as processed during the previous phase. Under GENLKE only the first module encountered would then be taken.
  They are kept by KPF /O command.
- now we have the XXXGEN processor and the monitor library on disc and we can generate the monitor tables and link them with the required modules. First we must assign the correct file codes (see GENMON: all input devices plus disc sequential file may be used):
- the following commands must be given:

ASG /A,DK,<file name>	assuming that the XXXGEN commands have been stored in a permanent disc file
ASG /B,DK	(a temporary file)
RUN XXXGEN	(object code will be output on /B)
REF /B	(rewind /B, i.e. the generated tables)
RDO /B	(into the /O file)
LKE M	(there are 2 libraries: a user library with the monitor modules and a system library consisting of INIMON (+ ASEX), so INIMON will always be the last module selected)
KPF /L,<file name>	

- now the monitor has been generated and kept in the /L file under <file name>.

Note: If an error occurs during the XXXGEN process, an exit will be made with bit 8 set in register A7, so in batch processing mode an abort will occur.

- to generate a complete system then, a user must be created under which the other modules, e.g. for a DOS system the disc-resident monitor segments and the system processors, must be kept.
- declare a userid, for example DOSGEN
- premark a scratch disc pack and declare a system userid for it, for example WHOME.

- reload the System by means of IPL procedure.
- enter this userid WHOME
- then:

  BYE

  WHOME

  MOV <file name>,/L,GENUSER (see above)

  KPF /L,SUP (there must of course be no module of this name among the system parts to be included.
- the first module will now be catalogued on disc starting from sector address /10.
- then give the command

  SVU DOSGEN,/F0

  to implement all the other system modules and processors.

Note: It is also possible to use Catalogued Procedures for users who must do sysgens regularly. See page 23

Note: When the DOS Linkage Editor is used instead of GENLKE during the linking phase, the addresses given by the DOS LKE in the MAP output must be subtracted by 8 to get the exact memory addresses, whereas the addresses output by GENLKE are correct.

Note: IPLGEN is a DOS module which is required when generating a BOM or BRTM, because the monitor which is then generated on tape must be preceded by an IPL. The sequence of operations in such a case is:

ASG /3,<output device>

RUN IPLGEN

PLD <monitor>

The generated IPL can be used for any device.

DISK PREMARK

PREMRK is a standalone program to be used for formatting a disc pack before it will actually be used. It divides the disc into sectors and writes identifiers in them and it checks for bad tracks.

PREMRK is an absolute format, so it is loaded by [illegible].
After it has been loaded it starts to type out the following questions on the operator's typewriter and the user can type in his answers:

NBR OF CYLINDERS
Type in 4 decimal characters, specifying the number of cylinders on the disc, followed by LF CR

NBR OF TRACKS
Type in 4 decimal characters, giving the number of tracks per cylinder, followed by LF CR

NBR OF SECTORS TRACK
Type in 4 decimal characters, specifying the number of sectors per track, followed by LF CR

DISK TYPE
Type in 2 characters, specifying the type of disc unit, and the type of channel used, as follows:

CM X1215 *on I/O processor*
[illegible] fixed-head disc *on I/O processor*

then type in LF CR

DISK UNIT PHYSICAL ADDRESS
Type in two hexadecimal characters, followed by LF CR

LABEL
Type in 8 characters, giving the volume label, followed by LF CR

DATE
Type in 6 decimal characters, giving the date, followed by LF CR

PACK NBR
Type in 3 characters, giving the pack number, followed by LF CR

SYSTEM USERID =
Type in the identification of the system generated, followed by LF CR.

RUN AGAIN?.
Type in NO if no other discs are to be formatted, or OK if another disc must be done. In the second case, the above procedure is repeated.

Example:

```
INITIALISATION OF PREMRK
NBR. OF CYLINDERS - 0203
NBR. OF TRACKS = 0002
NBR. OF SECTORS/TRACK = 0016
DISK TYPE: CM
DISK UNIT PHYSICAL ADDRESS = 02
LABEL = EXAMPLE
DATE - 12/02/74
PACK NBR. = 001
SYSTEM USERID = SAG
  WRITING THE IDENTIFIERS
- CHECKING THE IDENTIFIERS
  END OF CHECK
  NBR. OF BAD GRANULES = 0000

RUN AGAIN?: NO
END OF PREMRK
```

The user replies are given in boldface.

Error Messages:
BAD GRANULE ZERO (granule zero of the pack is bad and therefore this pack is not usable)
NO SYSTEM USER POSSIBLE (granule one of the pack is bad and therefore no system catalogue can be opened).

This information applies to the order, which the user must specify in an I/O monitor request

BASIC READ (/01)

Operator's Typewriter
All characters are entered on 8 bits until the requested length is reached.

ASR Tape Reader
All characters are entered on 8 bits. The reader stops one character after an Xoff code has been read.

High-speed Tape Reader
All characters are entered on 8 bits, without checking of special features, until the requested length is reached
Basic Read does not ensure a stop on character

Card Reader
All the words are entered and stored in Hollerith code on 12 bits (4 to 15). In each word the column image is right-justified. The words are stored until the requested length is reached. The length is given in words.

Disc
With the aid of Data Management all the sector words are entered in the memory buffer.

Magnetic Tape Cassette:
All Read/Write operations (Basic, Standard, Object) are the same, with the following characteristics.

maximum record length: 256 characters
required length: block length

effective length: block length (without control character).

all read/write operations are done on the requested length
incorrect length after read operation: no error, if requested length is greater than block length and the returned status is correct.
throughput error or data fault: retry is made automatically, up to five times
after read: backspace - read
after write: backspace - erase - write.

Magnetic Tape:
Same as for cassette tape, with the following differences:
- maximum record length: 4095 characters; minimum 12 characters.
  required length: block length without control word.

- physical block length: required length + 2.
- effective length: block length.
  12 characters are always transferred, in any case
  incorrect length: see cassette tape above.

BASIC WRITE (/05)

Operator's Typewriter
All characters are output without checking or special features. This order can be used to print something and have the answer on the same line.

ASR Tape Punch
All characters are output without checking or special features.

Line Printer
All characters are output without checking. There is no control character.

High-speed Tape Punch
All characters are output without checking or special features.

Disc
With the aid of Data Management all the sector words are output onto the disc

Cassette and Magnetic Tape
See under Basic Read (/01).

STANDARD READ (/02)

Operator's Typewriter
ASCII characters are entered on 8 bits, with the following special features:

the special characters, coded from /0 to /1F, are ignored.
code /7F (Rub-out or Delete character) is ignored.
- code /5F (←—) can be used to delete the preceding character. If several ←— are used consecutively, an equal number of preceding characters will be deleted.
- code /5E (↑) is used to delete the line preceding it, up to the next carriage-return. code /0D (carriage return) indicates end of block. It is the last character to be entered. It is not transmitted to the user's buffer.
- code /0A means 'line feed'.
- code /5C (\ ) is used as a tabulation symbol (see ECB word 5). If the address of the tabulation table is zero, or if the number of tackets is zero, or if the storage address is greater than the last tacket, the code /5C is stored in the buffer. In other cases, /5C is not stored and replaced by spaces, as indicated by the tackets in the tabulation table.

ASR Tape Reader
For ASCII characters, the same features apply as for the keyboard: the code for carriage return must be preceded by the code for Xoff.
For object code in 4-4-4-4 tape format, the first character identifies the object format. It must be in the range from /8 to /1F and is converted to a number from /0 to /7 and stored on one character. The second character contains the word-count of the input block, excluding the first word and the checksum. Each punched row (4 bits) entered after this identifier is stored on one half-character up to the checksum. When the checksum has been read, input is stopped. The 8-8 tape format cannot be read on the ASR tape reader. To start the reader, an Xon code is sent by the system before entering the characters.

High-speed Tape Reader
Same as for the ASR tape reader. In addition: for object code in 8-8 format, the first character, identifying the object code format, must have one of the following values: /10, /11 to /14 or /151o /17. It is converted to a number from /0 to /7. Each punched row (8 bits) entered after this identifier is stored on one character up to the checksum. The second character is the length of the block, in words, excluding the first word and the checksum.

Card Reader
All words are read in Hollerith code, on 12 bits, converted and stored in ASCII code, on 8 bits, until the requested length is reached. Words which are not in Hollerith are converted into the ASCII code for / 20 and a 'data fault' status is returned in the software status (ECB word 4: bit 13 is 1). There is no special code. However, EOS and EOF marks are detected (bits 14 and 15 in the software status).

Cassette and Magnetic Tape
See under Basic Read (/01).

STANDARD WRITE (/06)

Operator's Typewriter
All characters, except '0 to '1F (special code characters) are output without checking. At the end of a line, a carriage return and line feed are output. The first word in the buffer contains a right-justified control character, as for the line printer (see below). If it equals ,30 or ,31, it is output as line feed; if it is different, it is not output.

ASR Tape Punch
Same features as for the keyboard. At the end of a line, the following character sequence is output: LF - Xoff - CR - Rubout.

High-speed Tape Punch
Same as for ASR tape punch.

Line Printer
All characters are output without checking, except for the control code. The first word in the buffer contains a right-justified control character. This control code may have one of the following three values.
- (-2B): print the line without advancing the paper (superposition).
0 (-30): advance two lines before printing.
1 (-31): skip to top of page before printing.

All other control codes are used as normally; advance one line and print. At the end of the buffer, after the requested length, one character must follow to be used by the system for a print code.
If the requested length is more than one line, the system puts a print code after the maximum length and the buffer will be printed on two or more lines.

Cassette and Magnetic Tape
See under Basic Read (,01).

OBJECT WRITE 4+4+4+4 TAPE FORMAT (/07)

ASR Tape Punch
The first character is output on one row, converted from ,0 - ,7 to ,18 - ,1F. Each following character is output on two rows; to avoid special (ASCII) code each row is converted. The second character contains the length of the block in characters, excluding the first character. At the end an 8-bit checksum is performed and punched, followed by an Xoff code.

High-speed Tape Punch
Same as for ASR tape punch, except that the second character contains the length in words.

OBJECT WRITE 8+8 TAPE FORMAT (/08)

High-speed Tape Punch
The standard object code is output in 8+8 format, where the first character is a format character and is output on one row, converted as follows:
,0 → 10
,1 to ,4 → ,01 to ,04
,5 to ,7 → ,15 to ,17
The second character contains the length in words, excluding the first word. An 8-bit checksum is performed and punched.

Cassette and Magnetic Tape
See under Basic Read (/01).

WRITE EOF MARK (/22)

Operator's Typewriter
An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

ASR Tape Punch
An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

High-speed Tape Punch
An end-of-file mark is output as follows: :EOF LF Xoff CR Rub-out

Line Printer
An end-of-file mark is output as follows: :EOF

WRITE EOS MARK (/26)

Operator's Typewriter
An end-of-segment mark is output as follows: :EOS LF Xoff CR Rub-out

ASR Tape Punch
An end-of-segment mark is output as follows: :EOS LF Xoff CR Rub-out

High-speed Tape Punch
An end-of-segment mark is output as follows: :EOS LF Xoff CR Rub-out

Line Printer
An end-of-segment mark is output as follows: :EOS

Magnetic Tape Cassette
An end-of-segment mark is written as '6F6F.

Magnetic Tape
An end-of-segment mark is written as :EOS + 8 blank characters.

READ UP TO END-OF-SEGMENT (/14)

High-speed Tape Reader
The tape is read until an :EOS statement has been read.

Card Reader
The cards are read until an :EOS statement has been read.

READ UP TO END-OF-FILE (/16)

High-speed Tape Reader
The tape is read until an :EOF statement has been read.

Card Reader
The cards are read until an :EOF statement has been read.

WRITE END-OF-VOLUME (/24)

End-of-tape management for Magnetic and Cassette tapes is a user program responsibility.
When the physical end of a tape is encountered during a write operation, a status is returned in ECB word 4 with the EOT bit set. The user may then issue a Write EOV request (/24; Write End-Of-Volume; see under I/O monitor requests), before requesting the operator to mount a new tape. When a new tape is mounted, for magnetic tape the unit must first be switched off by pressing the OFF LINE button, while for cassette tape a Manual Control (MC) operator command 'Unlock' must be given to enable the operator to remove the cassette.
Then the operator can mount a new tape reel or cassette and restart the program.
To ensure that all records will be retrieved when the file is read, the EOT (end-of-tape) status also returned in the status word of the ECB should be ignored and only the EOV status must be taken into account.

Note:
In case the EOT is detected while reading an EOV, only the EOV status is returned.

RETURN INFORMATION ABOUT A FILE CODE (/30)

By means of this order it is possible to find out the assignment of a file code. The information will be returned in the Event Control Block:

ECB - word 0: File Code
ECB - word 1: Device Name (2 ASCII characters):
TY = operator's typewriter (listing)
TR = ASR tape reader
TP = ASR tape punch
PR = tape reader
PP = tape punch
LP = line printer
CR = card reader
MT = magnetic tape

TK = cassette tape

If NO device is assigned, the ECB contents are set to zero.
ECB - word 2: maximum record size.
ECB - word 3: left character: unused.
ECB - word 3: right character: device address.
ECB - word 4: status = 0. For line printer, it contains the number of lines specified for this printer at system generation time.

Note: (physical files; F0-FF)
If this order is used for a disc file code, the system will return the device name DK or DL (logical files) in ECB word 1 and fill the other words of the ECB with zeros.

OFF LINE (/38)

Magnetic tape:
This order switches the machine off.

Cassette Tape:
This order unlocks the cassette from the drive unit.

Appendix D Control Unit Status Word Configuration

Bit	Description	Control unit								
		ASR	CR		DM	LP	PTP	PTR	CASS Tape	MT
0	-									
1	has become ready				x				x	x
2	rewinding									x
3	tape mark has been read								x	x
4	no data								x	
5	on cylinder				x					
	beginning of tape								x	
6	seek error				x					
	write unable								x	x
7	A or B side								x	
8	Device Address								x	x
9	Device Address				x				x	x
10	EOT							x	x	x
	tape low						x			
11	Program error				x				x	x
12	incorrect length		x		x				x	x
13	Parity error								x	x
	data fault		x		x					
14	throughput error	x	x		x			x	x	x
15	not operable (only significant bit for TST)	x	x		x	x	x	x	x	x

DM = moving-head disc

```
00000                    IDENT    BEBOOT
00001          *
00002          *
00003          *         DISPLAY THE KEYS AS FOLLOWS:
00004          *
00005          *         BITS     MEANING
00006          *         0=1      IPL LOADED FROM ASR , FORMAT 4=4
00007          *         1=1      DISK,THEN
00008          *         2=1      MOVING HEADS
00009          *         2=0      FIXED HEADS
00010          *         3=1      PROGRAMMED CHANNEL
00011          *         3=0      I/O PROCESSOR
00012          *         4 TO 7   BOU LINES ( 4 RIGHTMOST BITS )
00013          *         8=1      MULTI DEVICE CONTROLLER
00014          *         8=0      SINGLE DEVICE CONTROLLER
00015          *         9=1      X1215 DISK
00016          *         10 TO 15 DEVICE ADDRESS
00017          *
00018          *
00019          *
00020          *         DESCRIPTION
00021          *
00022          *         BEBOOT LOADS ONE RECORD ONTO LOCATION /80 THEN START AT /84
00023          *         THE RECORD IS THE SECTOR # 1 IF DISK,OR 254 CHARACTERS OF THE
00024          *         INPUT DEVICE,LEADING NULL CHARACTERS IGNORED
00025          *
00026          *
00027          *
00028          *         USED REGISTERS :
00029          *
00030          *         A1 BOU LINES,NOT TO BE DESTROYED IF BOOT IS CALLED AGAIN
00031          *         A2 ADDR OF INR INSTRUCTION
00032          *         A3 ADDR OF CIO INSTRUCTION(WHICH IS DESTROYED AND NEEDS TO BE
00033          *                          RESTORED IF BOOT IS CALLED AGAIN)
00034          *         A4 ADDR OF SST INSTRUCTION
00035          *         A5 MULTIPLEX : CONTENTS OF 1ST WORD TO BE SENT TO EXT REGISTER
00036          *            IO BUS : CHARACTER COUNT,INITIALIZED AT 254 AND DECREMENTED
00037          *         A6 MULTIPLEX : CONTENTS OF 2ND WORD TO BE SENT TO EXT REGISTER
00038          *                          (LOADING ADDR)
00039          *            IO BUS : ADDR OF NEXT CHAR TO BE LOADED,INIT AT /80 AND
00040          *                          INCREMENTED
00041          *         A7,A8 WORK REGISTERS
00042          *         A9 WORK REGISTER
00043          *         A10 TO A14 NOT USED
00044          *         A15 CONTAINS THE KEYS'VALUE
00045          *
00046          *
00047          *
00048          *
00049          *
```

```
**
00050
00051
```

```
00052                                  EJECT
00053                                  AORG    0
00054                          *
00055                          *
00056                          BOOT    EQU     *
00057                          *       INITIALIZE REGISTERS
00058  0000  0200  F                   LDX     A2,INR          ADDR OF INR INSTRUCTION
00059  0002  0300  F                   LDX     A3,CIO          ADDR OF CIO INSTRUCTION
00060  0004  0400  F                   LDX     A4,SST          ADDR OF SST INSTRUCTION
00061                          *                               EXTRACT DEVICE ADDR AND INIT I/O COMMANDS
00062  0006  861E                      LDR     A6,A15
00063  0008  263F                      ANK     A6,/3F          DEVICE ADDR
00064  000A  9629                      ADRS    A6,A2           INITIALIZE I/O INSTRUCTIONS
00065  000C  9623                      ADRS    A6,A3
00066  000E  9641                      ADS     A6,MIO
       0010  0000  F
00067                          *                               EXTRACT CONTROLLER ADDR AND INIT WER INST
00068  0012  871E                      LDR     A7,A15
00069  0014  3FC6                      SLC     A7,6            MULTI OR SINGLE DEVICE CONTROLLER
00070  0016  5600  F                   RF(6)   INIT20          SINGLE ONE
00071  0018  260F                      ANK     A6,/F           MULTIPLE ONE
00072                          INIT20  EQU     *               INITIALIZE MULTIPLEX DBLE WORDS
00073  001A  9631                      ADRS    A6,A4           SST INSTRUCTION
00074  001C  3E41                      SLL     A6,1
00075  001E  9641                      ADS     A6,WER1         SET UP WER INSTRUCTIONS
       0020  0000  F
00076  0022  9641                      ADS     A6,WER2
       0024  0000  F
00077                          *                               LOAD A1 WITH BOOT CONTENTS
00078  0026  811C                      LDR     A1,A7
00079  0028  0550                      LDX     A5,80           MULTIPLEX DOUBLEWORD: LOAD 80 CHAR INTO
00080                          *                               LOCATION /80
00081  002A  0080                      LDX     A6,/80
00082                          *                               CHECK IF DISK
00083  002C  3FE7                      SRC     A7,7
00084  002E  5600  F                   RF(6)   NODISK          NO
00085                          *                               FIXED HEADS 7
00086  0030  3FC1                      SLC     A7,1
00087  0032  5600  F                   RF(6)   NOSEEK          YES
00088  0034  0103                      LDX     A1,3            SEEK ZERO
00089  0036  41C0              CIO     CIO     A1,1,0          CIO SEEK ZERO
00090                          NOSEEK  EQU     *
00091  0038  811E                      LDX     A1,A15
00092  003A  3466                      SRL     A1,6            SECTOR NUMBER
00093  003C  213C                      ANK     A1,/3C
00094  003E  8520                      LDXL    A5,/80CD        1ST WORD OF MULTIPLEX FOR DISK DEVICE
       0040  80CD
00095                          NODISK  EQU     *
00096                          *                               EXECUTE WER,WHATEVER THE CHANNEL IS
00097                          WER1    EQU     *
```

```
00098  0042  7500            WER     A5,0
00099                NER2    EQU     *
00100  0044  7601            WER     A6,1
00101  0046  F031            EXR*    A4          SST
00102  0048  F02D            EXR*    A3
00103  004A  5C06            RB(4)   *+4
00104  004C  871E            LDR     A7,A15
00105  004E  3F43            SLL     A7,3
00106  0050  5600  F         RF(6)   SST         MULTIPLEX
00107                *
00108                *                           IO BUS
00109                *
00110  0052  8194            LDR     A9,A5       IF A9=A5 IGNORE LEADING CHAR,
00111                INR     EQU     *
00112  0054  4F00            INR     A7,0,0      READ ONE CHAR
00113  0056  5C04            RB(4)   *+2
00114  0058  E994            CWR     A9,A5       LEADING CHAR?
00115  005A  5400  F         RF(4)   INR10       NO
00116  005C  27FF            ANR     A7,/FF      CHECK IF NULL
00117  005E  580C            RB(0)   INR         YES,IGNORE
00118                *                           NO,CHECK IF 4+4
00119                *
00120                INR10   EQU     *
00121  0060  879E            LDR     A15,A15     4+4
00122  0062  5600  F         RF(6)   STORE       NO 8+8
00123  0064  3F44            SLL     A7,4        YES,4+4
00124  0066  809E            LDR     A8,A7       SAVE LEFT BITS
00125  0068  F029            EXR*    A2          READ NEXT CHARACTER
00126  006A  5C04            RB(4)   *+2
00127  006C  270F            ANR     A7,/F       GET 4 RIGHT MOST BITS
00128  006E  9702            ADR     A7,A8
00129                STORE   EQU     *
00130  0070  E734            SCH     A7,A6       STORE CHAR
00131  0072  1601            ADR     A6,1        NEXT CHAR ADDR
00132  0074  1D01            SUR     A5,1        COUNT DONE ?
00133  0076  5924            RB(1)   INR         NO
00134                *                           YES
00135                *
00136  0078  4180            HIO     CIO     A1,0,0
00137                *
00138                STATUS  EQU     *
00139  007A  4FC0            SST     SST     A7,0
00140  007C  5C04                    RB(4)   *+2
00141  007E  0F84                    A8      /84
00142                *
00143                *
00144                *
00145                         END     BOOT
```

```
          SYMBOL TABLE

  BUOT    0000  A   INR     0054  A   CIO     0036  A   SST     007A  A
  HIO     0078  A   INIT20  001A  A   MEM1    0042  A   MER2    0044  A
  MUDISK  0042  A   MUSLEA  0038  A   IMR10   0060  A   STORE   0070  A
  STATUS  007A  A

          ASS,ERR,    00000
:EOF
PROG ELAPSED TIME:  00H-00M-00S-000MS-
BEA IPL525
DATE     /   /     TIME 24H-60M-60S-
```

Command	Meaning	Page
ASG	Assign a File Code	82
ASM	Call Assembler	109
BYE	End of Session	83
DCU	Declare User	83
DEB	Call Debugging Package	110
DEL	Delete File	84
DLU	Delete User	84
DUF	Dump File	85
END	End of Catalogued Procedure	85
FBS	Space File Backwards	90
FFS	Space File Forward	90
FOR	Call Full Fortran Compiler	111
HSF	High- Speed Fortran	112
INC	Include an Object Module	86
JOB	Start Batch Processing	86
KPF	Keep File	87
LED	Call Line Editor	114
LIC	List Catalogue	86
LKE	Call Linkage Editor	113
LSD	List Directory	88
LSF	List File Code	85
LST	List File	89
MES	Send Message	91
MOV	Move a File	91
NOD	Define Node	92
OLE	Overlay Linkage Editor	116
PCH	Punch a File	93
PLB	Print label	90
PLD	Punch Load	94
POB	Punch Object	94
POD	Print Object Directory	95
PRC	Print Catalogue	96
PRD	Print Directory	98
PRT	Print File	99
PSE	Pause	100

Command	Meaning	Page
RBS	Space Record Backwards	90
RDA	Read Data	100
RDO	Read Object	101
RDS	Read Source	102
REF	Rewind File	90
REW	Rewind to Load Point	90
RFS	Space Record Forward	90
RSU	Replace Supervisor	103
RUN	Run a Program	103
SCR	Scratch	104
SDM	Save Disc onto Magnetic Tape	105
SEG	Define Segments	106
SKF	Skip Form	107
SVD	Save Disc onto another Disc	107
SVU	Save User Files	108
ULD	Unlock Device	90
UPR	User Processor	116
WES	Write Device	90
WEF	Write EOS	90
WEV	Write End-Of-Volume	90
WLB	Write Label	90

Comment Sheet

P800M Programmer's Guide 2, Volume I (5122 991 27373)

Name _____

Company _____

Address _____

Telephone Number _____ Ext. _____

Comments or suggestions