P855/860

DISC MONITORS

PRELIMINARY - CONFIDENTIAL

In this book the two P855/860 monitors which may be used in a disc environment, are described.

The book is divided into three parts.

In Part I a description is given of the Disc Operating System (DOS), a non-real time, disc-typewriter oriented monitor.

Part II describes the Disc Real Time Monitor.

Part III contains descriptions of the operator control messages and monitor requests and under which monitor they can be used.

This book must be used in conjunction with other manuals, containing information on P855/860 assembler, Fortran compilers, linkage-editor, update package, text editor and debugging package.

PART I

DISC OPERATING SYSTEM

PRELIMINARY

CONFIDENTIAL

Introduction

The Disc Operating System is a non-real time disc system intended mainly as a tool for program development.
It is based on the following minimum configuration:

- P855/P860 central processor
- Memory of 8k words
- Operator's typewriter
- One disc unit. (Moving head !)

The multiply/divide option is mandatory for this system.
The memory protect option is recommanded.
This configuration can be extended with other peripherals and options, such as extra disc units, card reader, line printer, etc.

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 peripheral units in the configuration may be used without restriction.

The Disc Operatirg System 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 user files. The user may call such processors as an assembler, FORTRAN compiler, linkage-editor, update package, text editor, debugging package.
All the system components, including the processors are disc-resident.

PRINCIPLES OF OPERATION

In the Disc Operating System, the user works in a session.
Such a session is opened when the user types in his user identification, in reply to the system message USERID:, output after the system has been loaded or after a previous session has been closed.

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. This 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 SYS, with an entry in the disc catalogue 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, but if a card reader is included in the configuration, they may also be read from the card reader, thus making it a batch processing system. 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 of which the catalogue contains his user identification. This implies that one user can not 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 (output of LKE 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.

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 as well as when he puts files into his library. 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. It is also possible to keep programs which have been prepared for execution and have them executed under control of the Disc Real Time Monitor, as the DOS and the DRTM are compatible.

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 with the same or another user.

SYSTEM COMPONENTS

The Disc Operating System consists of three main components, the actual monitor which supervises all processing, the Control Command Interpreter which processes the control commands and acts as an interface between user and monitor, and the processors.

Monitor

The monitor supervises the processing of all user and system programs. It is always resident in memory. The monitor consists of several elements, which are defined at system generation time and include facilities for communication with the user and I/O drivers.

The user communicates with the monitor through monitor requests (LKM) in his program and, on option, through operator control messages. The latter is possible only if an operator communication package has been included in the monitor. Requests to this package are made through the control panel interrupt button (INT) which must be pressed in order to be able to enter an operator control message on the typewriter. If the operator communication package has not been included, pressing the interrupt button will result in abortion of the current job.

Another part of the monitor is formed by the I/O drivers. According to the physical configuration of the system they are all part of the monitor and include physical and logical disc I/O handling and blocking buffers.

All these elements must be defined at system generation time, where it is necessary to define a minimum of two blocking buffers.

Control Command Interpreter

This component is disc-resident, organized as a number of load modules in an overlay structure. Being loaded into memory to process control commands between job steps, it can use all of the available memory as any other program.

The Control Command Interpreter (CCI) asks, after every job step, for the next control command, then checks it for errors, processes it by itself or requests the monitor to load and run a specific system processor or user program.

Communication between the CCI and the monitor and within the CCI itself takes

place between two consecutive loading operations through a system table resident in memory, in the monitor. The CCI can use all the monitor facilities, such as IKM monitor requests, but it runs in master mode.

System Processors

All the system processors are disc-resident. They are loaded into memory on requests through the Control Command Interpreter. The processors include an assembler, a basic and a full FORTRAN compiler, linkage-editor, text editor, update package and debugging package.
They run in user mode in the same way as user programs, as all the processors are part of the system library which is considered the library of a particular user, i.e. SYS.

ORGANIZATION

Disc Organization

The disc organization described below applies not only to user libraries and files, but also to the system components which are part of the SYSTEM library. They are considered as belonging to a special user, with the user identification SYS.

Disc Space Allocation

The space on disc is divided into granules, track areas which are eight sectors long, each sector consisting of 200 words. The number of granules per disc depends on the type of disc used. All space allocation on disc takes place on the basis of these granules.

When he writes a file onto disc the user need not reserve disc space himself, nor does he need to know the expected length of the file, because disc space is allocated by the monitor. The monitor allocates as many granules as necessary to the file which is being written. These granules are chained and attached to the file code assigned to that file. Allocation is done one granule at a time, so a file always occupies an integer number of granules and one granule can not be shared by several files.

On each disc the system maintains an allocation table which enables the system to know which granules are still available for allocation.

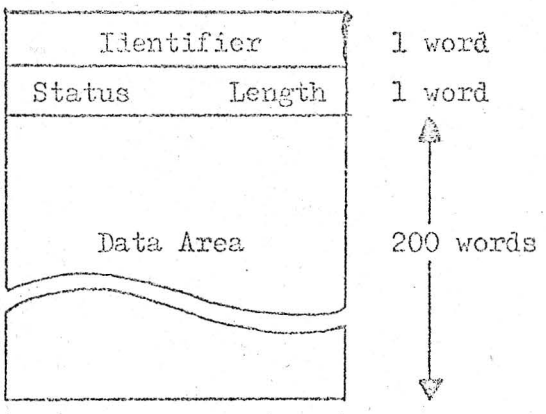
At the start of a session, allocation starts from the first granule available and new granules, if any, are added in ascending order (higher sector addresses). No backward search is done to take into account any granules which may have been deallocated again, e.g. after deleting a file. On the other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

It will be clear that the granules allocated to a file will not necessarily be consecutive. Therefore an allocation table is also maintained for each file and attached to it. This table contains the addresses of the granules allocated to the file. It is 200 words long, thus limiting the file length to a maximum of 320k words (200 x 8 sectors).

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. Updating a permanent file in random access can only be done directly if no extra granule is required.

Sector Format

The layout of a sector on disc is as follows:

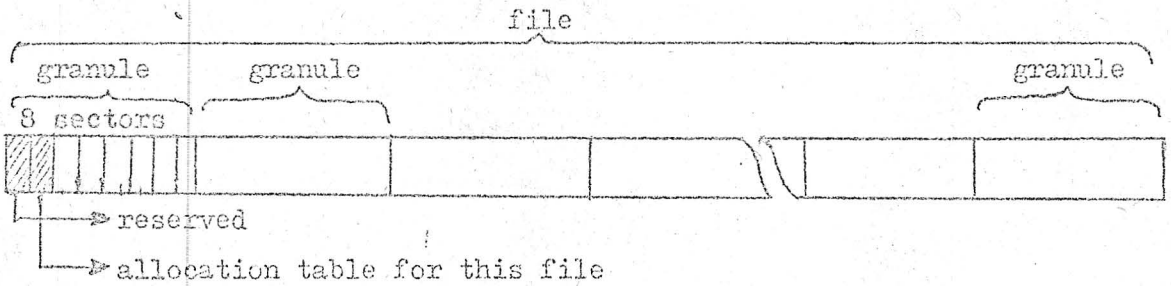


where:

- identifier is a sector identifier written at disc initialization time by a utility program called Premark.
- the second word is filled by the system only for files written in sequential mode. In other cases this word remains unused.
  - Status: bit 0 = 1: this sector has been deleted (See File Structure: Object Files).
    bit 1 = 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 = 1: An 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: length of the data part in this sector (up to 400 characters).
  - data area: contains data written in either sequential or random access mode (see Record Structure).

File Structure

A file is always stored on an integer number of granules, which means that two or more files cannot share the same granule. The structure of a file is as follows:



This implies, that when a file is accessed in random mode, record number 0 is the third sector of the first granule of the file.

- Object Files:

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:

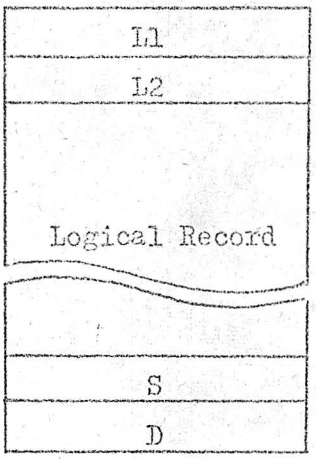
A load file is accessed in random mode. Each full sector of such a file contains 188 code words and 12 control words for relocation bits (see Linkage Editor).

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).

Record Structure

The record structure in a sequential file is as follows:



where:

- L1: record length in characters (L2, S and D included). This is the actual length on the disc, i.e. any trailing blanks have been removed by Data Management when the record was written.
- L2: initial record length in characters. This is the length requested in the I/O request when the record was written.
- S: sector address of the first word of the record (for backspace handling).
- D: displacement in characters in the sector S of the beginning of the record.
- Logical record: data part, up to 1600 words (one granule) long, trailing blanks removed.

A sequential file is always closed with an EOF record.

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 a library body.

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).
- 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 (DCU control command). An entry has the following format:

USER IDENTIFICATION				PASSWORD		POINTER	RESERVED
0	1	2	3	4	5	6	7

- words 0 to 3 contain the user identification, as declared in the DCU command
- words 4 and 5 contain a password
- word 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 SYS (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 X'0000'.
The last entry in the catalogue is followed by a word containing X'FFFF'.

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 SYS (system). In this case it is the second granule on the disc.

Each entry in a directory consists of 8 words:

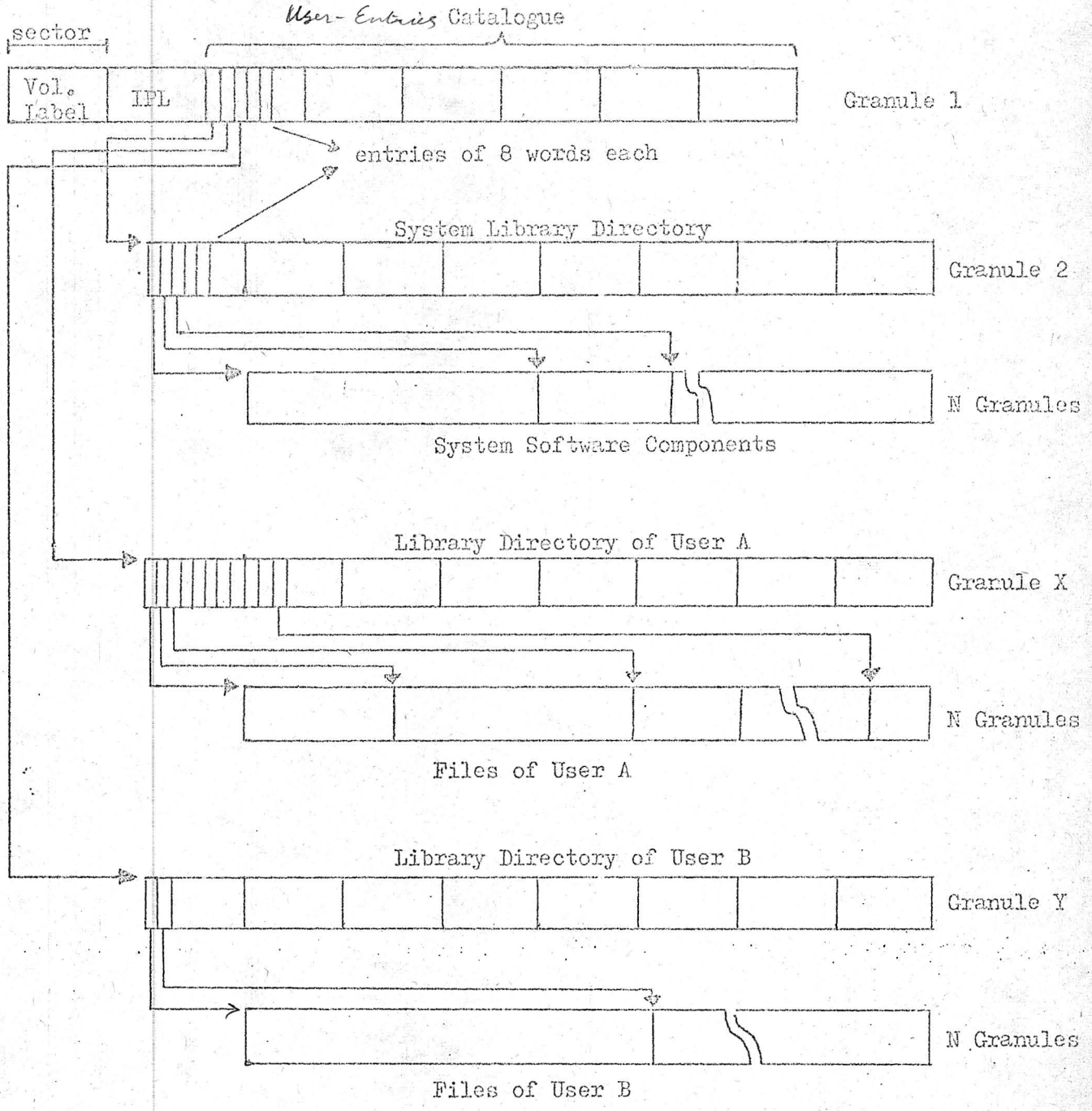
FILE NAME			TYPE	POINTER	RESERVED		
0	1	2	3	4	5	6	7

- 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: /S
  - for object files: /O
  - for load files: /L
  - for undefined files: file code.
- word 4 is the file pointer; it contains the disc sector address of the first granule of the file.
- words 5, 6 and 7 are reserved.

A user directory may contain up to 200 entries (8 sectors of 25 entries each). This limits the maximum length of a user's files to 320k words.
When an entry has been deleted, the first word is filled with the value X'0000'.
The last entry in a directory is followed by a word containing the value X'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.

On the following page a drawing is shown of the DOS library structure.



Example of Structure of Catalogue and Libraries.

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 temporary 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. 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 on tape (PLD control command). In this case the format is converted so as to produce a tape with the file in standard object cluster format.

A load module may not 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.

Note: A load module is entirely relocatable; therefore the directive AORG (see Assembler) should not be used in user programs.

Source Input Unit

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

- card reader
- punched tape reader
- ASR punched tape reader.

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 of 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. The RDS command also allows for reading and copying onto the temporary file a data file which is not a source program.

Object Input Unit : *No Object on cards.*

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

- punched tape reader
- ASR punched tape reader.

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 if a

card reader is included in the configuration, this can also be used for reading the control commands.

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.

File Codes

The following file codes are standard assignments for the logical files and units specified, as incorporated in the monitor.
Depending on the configuration, different logical units may have the same physical assignment:
01: operator's typewriter
02: print unit
03: punch unit
04 to CF may be assigned by the user
D4: /S file or library source file
D5: /O file
D6: /L file
D7: system object file (library)
D8: user object file (library)
D9 and DA are reserved for system use.
E0: control command input
E1: source input
E2: object input
EF: operator's typewriter (normally the same as 01)
F0 to FF are logical addresses of disc units. These are reserved for system

use.

The file codes 01, 02, 03, EC, 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 control commands.

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.
- 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 for punched tape, cassettte 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.

OPERATION

The user may communicate with the monitor through monitor requests in his program, requesting certain monitor functions. If an operator communication package is included in the monitor, he may also make use of a number of operator control messages.

These monitor requests and operator control messages are described behind the chapter on the Disc Real Time Monitor in a chapter containing all the monitor requests and operator control messages which may be used under both disc monitors. For each request and message, a separate indication is given under which of the two monitors it may be used. For the DOS monitor they are the following:

Monitor Requests

- I/O requests
- Wait
- Exit
- Get Buffer
- Release Buffer
- Pause
- Control Abort
- Load a Program
- Assign a File Code

Operator Control Messages

- Abort a Program (AB)
- Assign a File Code (AS)
- Dump Memory (DM)
- Halt Dump (HD)
- Load a Program (LD)
- Manual Device Control (MC)
- Pause (PS)
- Release Device (RD)
- Restart (RS)
- Retry (RY)
- Start (ST)
- Write into Memory (WM)

As for the control commands entered via the operator's typewriter, after the system has been loaded by IPL procedure or after a previous session has been closed, the Control Command Interpreter asks for the user identification by typing out USERID:

Then the user types in his user identification, whereupon the system will scan the catalogue of each on-line disc, starting with the disc unit F0, until it finds the corresponding user identification. If it is correct, the CCI types out S: and waits for the first control command.

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 SYS and then he 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.

An error may occur when a user identification can not be found in any Catalogue. In such a case an error message is given and USERID: is typed out once more.

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).
<name> is a file name or object module name of 1 to 6 characters, not starting with / (slash).
<disc number> 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.

On the following pages the control commands are listed in alphabetical order, according to their mnemonics.

Command	Meaning	Page
ASG	Assign a File Code	21
ASM	Call Assembler	35
BYE	End of Session	22
DCU	Declare User	22
DEB	Call Debugging Package	36
DEL	Delete File	23
DLU	Delete User	24
EDT	Call Text Editor	37
FOR	Call Full Fortran Compiler	38
FRT	Call Basic Fortran Compiler	39
INC	Include an Object Module	24
KPF	Keep File	25
LIC	List Catalogue (= general catalogue only part of that is used) → TY.	26
LKE	Call Linkage Editor	40
LSD	List Directory (= of user lib.)	26
LST	List File (hexadecimal)	27
MOV	Move a File	28
PCH	Punch a File	29
PLD	Punch Load	29
POB	Punch Object	30
PRC → LPR	Print Catalogue (= general catalogue that is used and not used)	30
PRD	Print Directory	31
PRT	Print File	31
RDO	Read Object	32
RDS	Read Source	33
RUN	Run a Program	33
SCR	Scratch	34
SEG	Define Segments (overlay)	34
UPD	Call Update Package	41

TABLE OF CONTROL COMMANDS AND PROCESSOR CALLS

Assign

syntax: ASG␣ /<file code1>,[/<file code2>|<device name>[,<name>[,<userid>]]]

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. It allows assigning a file code to a file in another user's library.

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

Errors:
- table overflow
- <name> cannot be found
- <userid> unknown
- <device name> unknown
- <file code2> had not been assigned previously.

End of Session

syntax: BYE

use: The user must give this command at the end of a session to indicate that he is leaving the system. After this command, the system is re-initialized and will again ask for user identification in order to start a new session.

Declare User

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 SYS. 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 X'FFFF'. An entry for this user is filled in the Catalogue. The allocation table is updated.

Errors:
- <userid> already exists on this disc (no check is done on the other discs)
- there is no entry available in the catalogue
- there is no granule available for a user directory.

Delete File

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 first file encountered with the name <name> is deleted without any check on the type.
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.

Errors:
- file unknown
- object module unknown.

Delete User

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 SYS. 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 address 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.

Errors:
- <userid> unknown.

Include an Object Module

syntax: INC␣<name>[,<userid>]

use: By means of this command it is possible to select an object module from the library of the current or another user and to copy it in the temporary file /0.
<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 /0 (temporary object file) exists, but has not been terminated by an EOF yet, this module is written at the end of this file.
If /0 exists and has already been terminated with an EOF, this file is lost and a new assignment is made for a new /0.
If /0 does not exist, an assignment is made for it and this module will be the first one of this new /0.
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 /0 file.

Errors:
- <name> unknown
- <userid> unknown.

Keep File

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

use: This command is used to keep a file or object 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 containgn these modules. If possible, the copying is done in the same physical area which the deleted modules occupied previously.

Errors:

- /S, /O or /L does not exist
- <file code> has not yet been assigned
- <name> can not be found in the /O file.

List Catalogue

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 SYS.
It provides for printing out the catalogue of the disc specified on the typewriter.
<disc number> gives the address of the disc of which the catalogue must be listed.

List Directory

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:
- /OB has been specified, but there is no object file in the library.

List File

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

use: This command causes a listing of a specified disc file on the operator's typewriter. The file must be sequential and comsist of ASCII records. If a record is longer than a print line it will be printed on several lines. The parameters have the following meaning:

<file code> : file code of a temporary file which must be listed.

/S: the system file must be listed.

<name> : name of a library file which must be listed.

<line nb1>,<line nb2> : if specified, the file will be listed from the first line number specified to the second line number.

The maximum record size allowed is 132 characters. Records which are longer will be truncated. Non-printable record characters will be replaced by blanks.

The listing will stop when either <line nb2>, an EOF or the End-Of-Volume (the last granule of the file) is reached. End-Of-Volume occurs when no EOF has been written for the file, which normally is the case for temporary files created by a program in the debugging phase.

Errors:

- <file code> has not yet been assigned, or /S does not exist
- <name> unknown
- errors in the line numbers.

Move a File

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

use: This command is used to move a file from a library - generally of another user - 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, this 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:

- <name> unknown
- <userid> unknown.

Punch a File

syntax: PCH␣[/<file code>|/S|<name>]

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 80 characters. The file may be 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 file in the user library which must be punched.

If any records of over 80 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.

Errors:

- <file code> has not been assigned, or /S does not exist
- <name> unknown.

Punch Load

syntax: PLD␣[<name>]

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 no name is specified, the temporary file /L will be punched.

Errors:

- <name> unknown
- /L does not exist.

Punch Object

syntax: POB␣[<name>]

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 /0.
<name> is the name of the library object module which must be punched. If no name is specified, the whole temporary object file /0 will be punched.
When the temporary file /0 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.

Errors:
- <name> unknown
- /0 does not exist.

Print Catalogue

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 SYS.
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.

Print Directory

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.

Errors:
- /OB has been specified while there is no object file in the library.

Print File

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

use: This command is used to obtain a listing of a 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 parameters have the following meaning:
<file code> : file code of a temporary file which must be listed.
/S: the system file must be printed.
<name> : name of a library file which must be printed.
<line nb1>, <line nb2> : if specified, the file will be listed from the first line number specified to the second line number.
The maximum record size allowed is 132 characters. Records which are longer will be truncated. Non-printable record characters will be replaced by blanks. The print-out will stop when either <line nb2>, an EOF or the End-Of-Volume (last granule of the file) is reached. End-Of-Volume occurs when no EOF has been written for the file, which normally is the case for temporary files created by a program when debugging.

Errors:
- <file code> has not yet been assigned, or /S does not exist
- <name> unknown
- errors in the line numbers.

Read Object

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 /0 file or as a complement to the /0 file.
<file code> : file code of the input inut 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 an /0 file on the disc which has not been closed with an EOF record, the object file is copied after this /0 file.
If there is no /0 file on the disc or if it has already been closed with an EOF record, a new assignment is done for /0 starting at a new granule and the old /0 file is lost.
It is possible to give several RDO commands to fill /0 with several object files. Mixing RDO commands with language processor calls in order to fill the /0 file is allowed also.

Errors:
- <file code> has not been assigned.

Read Source

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 gramule.

Errors:
- <file code> has not been assigned.

Run a Program

syntax: RUN␣[<name>]

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.

Errors:
- <name> can not be found
- /L does not exist
- memory overflow (program too large).

Scratch

syntax: SCR␣[[/S|/O|/L|/<file code>]]

use: This command can be used to release previously made user assignments. 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).

Define Segments

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 10 names.
The segments specified will be called by the root program at run time through a Load monitor request (LKM 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.

Errors:
- <name> can not be found.

Assembler

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. 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 /0. If /0 does not exist, an assignment will be made for it. If /0 does already exist, the object module will be written at the end of this file, unless /0 has already been closed with an EOF record. In that case a new assignment is made and the old /0 file is lost.

Errors:

- /S does not exist
- <name> unknown.

If a fatal error occurs during assembly, the whole /0 file will be deleted and a request for link-editing will be refused.

Debugging Package

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.

Errors:
- <name> unknown
- /L does not exist.

Text Editor

syntax: EDT␣<name>

use: This command must be used to call the Text Editor in order to edit a source file from the library.

<name> : gives the name of the source file which must be edited.

Errors:

- <name> unknown.

Full Fortran Compiler

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. 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 /0. If /0 does not exist, an assignment will be made for it. If /0 already exists, the object module will be written at the end of this file, unless /0 has already been closed with an EOF record. In that case a new assignment is made and the old /0 file is lost.

Errors:

- /S does not exist
- <name> unknown.

If a fatal error occurs during compilation, the whole /0 file will be deleted and a request for link-editing will be refused.

Basic Fortran Compiler

syntax: FRT␣[/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. 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 /0. If /0 does not exist, an assignment will be made for it. If /0 does already exist, the object module will be written at the end of this file, unless /0 has already been closed with an EOF record. In that case a new assignment is made and the old /0 file is lost.

Errors:

- /S does not exist
- <name> unknown.

If a fatal error occurs during compilation, the whole /0 file will be deleted and a request for link-editing will be refused.

syntax: LKE␣[N|S|U][,M][,[DE|DS]][,/<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. 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.

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.

Errors:

- /O file does not exist
- user library can not be found.

If a fatal error occurs during link-editing, the /L file will be deleted and a RUN command will be refused.

Update Package

syntax: UPD␣<name>

use: This command must be used to call the Update Package and update a source file located in a library.
<name> is the name of the source file.
Updating takes place in ascending order of line numbers, through Insert and Delete lines commands. These commands are not handled by the Control Command Interpreter, but by the Update Package.
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:
- <name> unknown.