LEVEL 64

DATA MANAGEMENT UTILITIES

SUBJECT

Description of the File and Volume Utility Programs

SPECIAL INSTRUCTIONS

For users of Release 0400, this edition completely supersedes Revision 2, dated September 1978, and Addendum A. Change bars in the margins indicate new or changed information; asterisks denote deletions.

SOFTWARE SUPPORTED

Level 64 GCOS Release 0400

ORDER NUMBER

AQ20-03

July 1980

Honeywell

Preface

This manual describes the record file and volume utility programs for the Level 64 GCOS operating system. It is aimed at the programmer or analyst who has data management responsibilities.

Section 1 of this manual is an introduction to the utilities.

Section 2 describes the structure of the utilities.

Section 3 discusses file and volume identification.

In Section 4 there is a description of many keywords and parameters which are common to the utilities.

Section 5 contains a description of the record level utilities and the associated Data Services Language (DSL).

The file level utilities are described in Section 6, and the volume level utilities are described in Section 7.

Appendix A contains a brief description of the compatibility between files and volumes saved under the previous and current release of GCOS.

In this manual, all references to Series 60 apply specifically to Series 60 Level 64 computer systems running under the Level 64 GCOS Operating System.

The term GCOS used in this manual always refers to the Level 64 GCOS Operating System, unless otherwise stated.

Each section of this document is structured according to the heading hierarchy shown below. Each heading indicates the relative level of the text that follows it.

Level	
1 (highest)	ALL CAPITAL LETTERS, BOLD FACE
2	Initial Capital Letters, Bold Face
3	ALL CAPITAL LETTERS, MEDIUM FACE
4	Initial Capital Letters, Medium Face
5 (lowest)	ALL CAPITAL LETTERS FOLLOWED BY COLON: Text begins on the same line.

The following notation conventions for JCL statement formats are used in this manual:

UPPERCASE	The keyword item is coded exactly as shown.
lowercase	Indicates a user-supplied parameter value.
[item]	An item within square brackets is optional.
item 1 item 2 item 3	A column of items within brackets means that one value must be selected if the associated parameter is specified. If the parameter is not specified the underlined item is taken as the default value.
( )	Parentheses must be coded if they enclose more than one item.
. . .	An ellipsis indicates that the preceding item may be repeated one or more times.

The Level 64 Document Set follows.

LEVEL 64 DOCUMENT SET

Order Number	Title
AQ02	*Series 100 Program Mode Operator Guide*
AQ03	*Series 100 Conversion Guide*
AQ04	*Series 200/2000 Conversion Guide*
AQ05	*System 360/370 Conversion Guide*
AQ09	*System Management Guide*
AQ10	*Job Control Language (JCL) Reference Manual*
AQ11	*Job Control Language (JCL) User Guide*
AQ13	*System Operation Operator Guide*
AQ14	*System Operation Console Messages*
AQ20	*Data Management Utilities Manual*
AQ21	*200 Program Mode User Guide*
AQ22	*200 Program Mode Operator Guide*
AQ26	*Series 100 File Translator*
AQ27	*Series 200/2000 File Translator*
AQ28	*Library Maintenance Reference Manual*
AQ40	*IBM System/3 Conversion Guide*
AQ49	*Communications Network Control Terminal Operations*
AQ50	*Terminal Operations*
AQ52	*Program Checkout Facility*
AQ53	*Communications Processing Facility*
AQ55	*TDS/64 Site Manual*
AQ56	*TDS/64 User Guide*
AQ57	*TDS/64 Programmer's Reference Manual*
AQ59	*Unit Record Devices User Guide*
AQ60	*Interactive Operation Facility*
AQ63	*COBOL User Guide*
AQ64	*COBOL Language Reference Manual*
AQ65	*FORTRAN Language Reference Manual*
AQ66	*FORTRAN User Guide*
AQ67	*FORTRAN Mathematical Library*
AQ68	*RPG Language Reference Manual*
AQ69	*RPG User Guide*
AQ70	*Series 100 COBOL Translator*
AQ72	*Series 200/2000 COBOL to Level 64 COBOL Translator*
AQ73	*IBM System 360/370 to Level 64 COBOL Translator*
AQ77	*File Translator*
AQ80	*Series 360 (DOS) Program Mode Operator's Guide*
AQ82	*BFAS User Guide*
AQ83	*HFAS User Guide*
AQ84	*UFAS User Guide*
AQ85	*Sort/Merge*
AQ86	*Catalog Management*
AQ87	*Library Maintenance User Guide*
AQ88	*I-D-S/II User Guide, Volume 1*
AQ89	*I-D-S/II User Guide, Volume 2*
AQ90	*COBOL Pocket Guide*
AQ91	*System Summary Pocket Guide*
AQ92	*Operator's Reference Card*
AQ93	*RPG Listing Decoder*
AQ94	*FORTRAN Reference Card*
AQ95	*Telecommunications and TDS Reference Card*
AQ96	*I-D-S/II Pocket Guide*
AQ97	*Character Conversion Reference Card*

Order Number	Title
AQ98	*System Overview Manual*
CH73	*Query Driven System Reference Manual*
CQ17	*Error Messages and Return Codes Pocket Guide*
CQ31	*Error Messages and Return Codes Manual*
CQ35	*Remote Batch Facility*
CQ43	*BASIC*
DQ02	*Automated Update Facility*

Table of contents

1. Introduction

The Data Management Utilities of Series 60 Level 64 GCOS are programs which are simple to use and make data-file and volume management easier for the user. The utilities work on both UFAS (Unified File Access System) and BFAS (Basic File Access System) file formats.

Each utility (utility program) is run as a GCOS job step, and more than one utility can be run in a job. The utilities are accessed by means of Extended JCL (Job Control Language) statements, and this means that the user has to write only one statement ($PREALLOC, for example) and supply the appropriate parameters. The system expands the statement into a series of Basic JCL statements which make up a single job step.

There are three types of utility :

- Record-Level Utilities
- File-Level Utilities
- Volume-Level Utilities.

RECORD-LEVEL UTILITIES

The record-level utilities, described in detail in Section 5, operate on the logical records of a file. They allow the user to manipulate data fields within records, to select or omit certain records, and to change the organization of a file. There are three record-level utilities :

- $COMPARE : Compares the contents of two files
- $CREATE : Loads or reorganizes a file
- $PRINT : Prints logical records from a file.

FILE-LEVEL UTILITIES

The file-level utilities operate on entire files. Section 6 contains a detailed description of these utilities, which are listed in Table 1-1.

Utility name	Function
$DEALLOC	Deallocate a disk or tape file
$FILCHECK	Check the integrity of a disk file
$FILDESC	List file label information
$FILDUPLI	Copy the contents of a file into another file of a similar type
$FILMODIF	Change the name and/or the expiration date of a file
$FILPRINT	Print the physical blocks of a disk or tape file
$FILREST	Restore a $FILSAVE file onto disk
$FILSAVE	Save a disk file onto a tape file
$PREALLOC	Allocate space for a disk or tape file
$SORTIDX	Sort and load secondary indexes of a file

Table 1-1. The File-Level Utilities

VOLUME-LEVEL UTILITIES

The volume-level utilities operate on tape reels or disk packs. They are fully described in Section 7, and listed in Table 1-2.

Utility name	Function
$VOLCHECK	Check the integrity of a disk volume
$VOLCOMP	Compare two volumes
$VOLCONTS	List the Volume Table of Contents of a disk volume and edit file characteristics
$VOLDUPLI	Make a duplicate of a volume
$VOLPREP	Prepare a disk, cassette or tape volume
$VOLPRINT	List the physical blocks of a tape volume
$VOLREST	Restore a $VOLSAVE file onto a disk volume
$VOLSAVE	Save a disk volume into a tape file

Table 1-2. The Volume-Level Utilities

THE DATA SERVICES LANGUAGE (DSL)

With the record-level utilities, data can be manipulated by the DSL (Data Services Language). The DSL allows you to define selection rules for the record stream that is being loaded into a file. The DSL statements are provided through an input enclosure to the record level utilities. For further details of the DSL, see Section 5.

RUNNING UTILITY PROGRAMS

A utility program is run as a normal Level 64 GCOS job step, in the same way as, for example, a program translation job step or a user application program. A typical job to run a utility might be:

```
$JOB WXC, USER = FRED ;
   VOLCOMP      parameter-list ;
$ENDJOB ;
```

The format of $JOB and $ENDJOB is given in the JCL Reference Manual, and the precise definition of $VOLCOMP and its parameters is given in Section 7 of this manual.

There is no need to run each utility as a separate job, and several can occur in one job. For example, the job below prepares a disk volume, allocates space on it for a file, loads the file, and then prints the file to validate the data transfer.

```
$JOB CVB, USER = JOHN ;
   VOLPREP      parameter-list ;
   PREALLOC     parameter-list ;
   CREATE       parameter-list ;
   PRINT        parameter-list ;
$ENDJOB ;
```

REPORTS AND DIAGNOSTICS

When a utility is executed, a JOR (Job Occurence Report) and a Utility Report are produced (see Note, below). The JOR shows you how GCOS handled the utility, and the utility report is produced by the utility itself ; additional messages may be sent to the operator's console.

Error messages can be produced on two occasions, when the JCL is processed, and during the execution of the utility. Both kinds of message are printed on the JOR.

Messages which are delivered during the execution of the utility step belong to one of the four categories described below :

SEV0 Messages :

Messages in this category are merely information messages; they do not report any malfunctions. They only describe normal events, such as storage being allocated to a temporary file, and so on. They also provide statistical information such as the number of records that have been processed. SEVO messages do not affect the way job steps are processed by Job Management.

SEV1 Messages :

Messages in this category are WARNING messages. They usually support some unexpected event or events which have little or no effect on the results of the utility step execution.

The user must pay attention to WARNINGS and take appropriate action to ensure that they do not occur in future executions of the jobs.

A common example of a warning message is the one which occurs whenever an obsolete JCL statement is encountered by the JCL Translator. If no action is taken by the user to update the jobs, they will not work when the next version of the system is installed, as the obsolete JCL statement names will no longer exist. SEV1 messages, like SEV0 messages, have no effect on the way in which job steps are sequenced by Job Management.

SEV3 Messages :

Messages in this category report on unexpected events which will have FATAL consequences on the running of the utility step. These events are considered to have lasting effects on the execution of the step, and restarting the step from one of its recovery points (if there are any) will not give a better execution. Consequently, the step is aborted and no attempt is made to restart it, even if the REPEAT option is present. SEV3 messages cause Job Management to consider the step as aborted, and every step that follows is skipped until either an executable $JUMP statement is found, or a $ENDJOB, **is found. SEV3 messages often result from user errors.**

SEV4 Messages :

Messages in this category also report on unexpected events that have FATAL consequences to the execution of a utility step. Unlike SEV3 messages, SEV4 messages report on malfunction conditions that are not thought to have lasting effects. Such conditions are expected to disappear either as a result of normal activity of the system (such as the release of a resource), or as the result of some action by the operator (such as replacing a volume by another one or moving a volume onto another drive) ; therefore an attempt can be made **automatically by the system to** restart the aborted step from its recovery point.

If the REPEAT option was used in the STEPOPT parameter group, the utility step aborts and is restarted from its available recovery point. The abort and restart sequence can happen again and again until the operator refuses a new restart operation for that step.

If the REPEAT option was not used, or if the operator did not allow a restart operation for the step, Job Management handles step sequencing as it does for SEV3 messages.

Note :

There are a few utilities which produce very little printed output, and these utilities report exclusively to the JOR. A utility report is not produced. This happens, for example, with $PREALLOC, $DEALLOC, $VOLSAVE, $FILSAVE, $VOLREST and $FILREST.
With the utilities that do produce a utility report, the user can ask the system to direct this report to a permanent output file rather than to the standard SYSOUT file. This is done by means of the PRTFILE, PRTDEF, and PRTOUT parameters to the utilities. These parameters are described in Section 4.

2.Structure of the Utility Statements

The rules for writing utility statements are identical to those for writing Basic JCL statements. These rules are fully described in the JCL Reference Manual.

Most utility statements read from a file (the input file) and write to another file (the output file).

The input file is described by providing a file-description after the key-word INFILE. This file description is, in fact, given in terms of all the $ASSIGN parameters (except the internal-file-name). Some utilities can process more than one file, in which case a series of file-descriptions follows the key-word INFILES. The way in which this is done is identical to the file concatenation for $ASSIGN. In addition, some utilities may need to read more than one file, in which case the input files are described after the key-words INFILE 1 and INFILE 2.

Similarly, the output file is described after the key-word OUTFILE.

There are occasions when it is useful to provide the file characteristics using parameters from the $DEFINE Basic JCL statement . This is done by putting these parameters under the scope of the key-words INDEF or OUTDEF (or INDEF 1 and INDEF 2) as appropriate.

Most utilities produce a utility report. This utility report is normally treated as standard output (that is, temporarily stored in the SYSOUT file until it is printed). It is possible to send the report to a permanent SYSOUT file, which is described by the parameter PRTFILE ; the characteristics of this file can also be defined by the parameter PRTDEF.

This utility report will be printed by the output writer with all the standard options (possibly defined by an OUTVAL statement - see the JCL User Guide). If other characteristics are required for this output (for example, the use of a specific paper size, or a specific output class, or several copies), they can be supplied by placing the parameters of the $SYSOUT Basic JCL statement under the scope of the PRTOUT key-word for the PRTFILE file (the SYSOUT key-word will be used instead of PRTOUT for the OUTFILE file).

All the characteristics of the step corresponding to the execution of the utility (for example, execution priority, limits on time, etc...) are the default ones unless specific information is provided by placing $STEP Basic JCL statement parameters after the key-word STEPOPT.

For the record level utilities, some DSL statements may be provided, and these will be read from the input enclosure, sequential file, or library member described after a COMFILE key-word.

To summarize, the general form of the utility statements is:

Utility-name

INFILE = (file-description)	file read
[, INDEF = (define-parameters)]	
, OUTFILE= (file-description)	file written
[, OUTDEF= (define-parameters)]	
[, SYSOUT = (sysout-parameters)]	
[, PRTFILE= (file-description)]	utility report
[, PRTDEF = (define-parameters)]	
[, PRTOUT = (sysout-parameters)]	

```
[, COMFILE   = (file-description)]          DSL
[, STEPOPT   = (step-parameters)]           step parameters
utility-specific-parameters;
```

The most used form is likely to be :

```
Utility-name
INFILE = (file-description), OUTFILE = (file-description),
         utility-specific-parameters ;
```

3. Files and Volumes

This section discusses the rules for identifying files and the volumes and devices on which files are stored.

When specifying a file in a utility statement, it is usually necessary to also specify the disk or tape volume on which the file is stored, and the type of device that supports the volume.

FILE NAMES

The complete conventions for file names are described in the JCL Reference Manual.

FILE EXPIRATION DATES

For all permanent disk files and for all files on native labelled cassettes and tapes there is an expiration date maintained with the file. For disk files, the expiration date is set when the file space is allocated. For tape or cassette, the date is set when the first file is recorded onto the volume. Note that an expiration date is associated with the file and not with the volume.

The users attention is drawn to Section 4. This shows the complete file description which is applicable to the utilities.

VOLUME IDENTIFICATION

A distinction is made between a volume that is permanently online to the system (resident) and one that is loaded on a device only when it is needed (non-resident).

Resident Volume

A resident volume is always a disk, and is identified by the RESIDENT parameter. No other identification is necessary. Example:

```
$DEALLOC      CUIFEB,      RESIDENT;
```

The permanent file CUIFEB is held on a RESIDENT disk volume. For further details see the description of the parameter RESIDENT in Section 4. Note that when RESIDENT appears, FILESTAT = UNCAT applies by default.

Non-resident Volume

For a non-resident disk volume or a tape volume, the volume identifier (which is recorded in the volume label) must be specified using the MEDIA parameter. The volume identifier is expressed in alphanumeric (0-9, A-Z) and has a maximum of six characters (the maximum is 4 characters for a COMPACT cassette). Note that the keyword WORK may be specified to indicate that a work volume is to be used. Work tapes are prepared using the $VOLPREP utility statement. In the case of cataloged files, the volume type and name are given in the catalog description of the file.

MULTIVOLUME FILES AND VOLUME MOUNTING

This paragraph only deals with record and file level utilities, as volume level utilities cannot handle media-lists with several volume-names.

The MOUNT option of the $ASSIGN JCL statement is supported by the record level utilities without any restrictions, but if the file organization has any special requirements, they must be observed (for example, direct access or relative files require all the volumes to be mounted).

For disk files, the file level utilities sometimes override the user-supplied value of the MOUNT parameter. For $PREALLOC, all the volumes have to be mounted.

Tape files are, in general, supported by file level utilities with a user given volume mounting request.

When a file level utility is restarted at a checkpoint ($FILSAVE, for example), it is possible that the re-opening of a disk file causes the first volume of the file to be mounted without apparent justification, although the actual processing resumes with the volume which was being accessed when the checkpoint occurred. This is done to check the file label at file opening time.

When the $PREALLOC utility has already processed several volumes of a multivolume file and it encounters abnormal conditions on the current volume, it generally attempts to undo what has already been done to the file. It therefore requests that the already processed volumes be mounted again, to restore them to their previous state. The success of this restore, and the possibility of being able to restore are not guaranteed and, if unsuccessful, the file will be left in an inconsistent state.

Volume and File Sharing

Shared access only applies to input disk files and volumes.

The input volumes for the utilities $VOLCOMP, $VOLSAVE and $VOLDUPLI will only be shared by multiple executions of the utilities themselves. That is, an input volume to $VOLSAVE may be accessed concurrently by, say, $VOLCOMP.

$VOLCONTS can share its input volume with any file-level or record-level utility, but not with other volume-level utilities.

If the input volumes are RESIDENT, then this restriction does not apply ; all other jobs may access a RESIDENT volume concurrently with a volume-level utility.

Sharing of input files ($FILSAVE, $FILREST, $FILDESC, $CREATE, etc.) is always allowed. These utilities access the input files with user specified values of ACCESS and SHARE.

For further details on file sharing and the parameters ACCESS and SHARE, see the UFAS User Guide and the BFAS User Guide.

4. Common parameters

In the utility statement,definitions presented in Sections 5, 6, and 7, there are a number of parameters and keywords that occur frequently. To avoid repeated explanation of these, they are described separately here in this section. No further detailed explanation is given in the utility description, but they are shown to be present in the statement form, where applicable. These parameters are :

- define-parameters
- file-description
- INDEF
- INFILE
- OUTDEF
- OUTFILE
- PRTDEF
- PRTFILE
- PRTOUT and SYSOUT
- STEPOPT
- sysout-parameters.

They are described in alphabetical order on the following pages.

Define-parameters can be given in addition to the file-descriptions used for files accessed at record level. They usually do not apply to files which are handled at the file level.

The files which are accessed at record level are PRTFILE (for any utility which has one), and the input and output files of the record level utilities.

Thus a file accessed at record level can be fully described by a combination of two parameter groups ; one providing a file-description for the file, and the other providing a file-definition for the file.

The define-parameters are identical to those which may be given in a $DEFINE Basic JCL statement ; a full description of $DEFINE is given in the JCL Reference Manual.

The relation between define-parameters and the corresponding file is established through the internal-file-name, as follows :

INDEF is related to the internal-file-name infile

INDEF1 is related to the internal-file-name infile

INDEF2 is related to the internal-file-name infile2

OUTDEF is related to the internal-file-name outfile

PRTDEF is related to the internal-file-name prtfile

The most useful of the define-parameters are the following :

[,BLKSIZE = blksize]

[,RECSIZE = recsize]

[,RECFORM = { F | FB | U | V | VB }] [,FILEFORM = { UFAS | BFAS | ANSI | NSTD }] [,DATACODE = { BCD | H200 | ASCII | EBCDIC }]

[,NBSN]

[,NBBUF = { 1 | 2 }] [,JOURNAL = BEFORE]

[,BPB = blocks-per-buffer] [,KEYLOC = key-position][,KEYSIZE = key-length]

[,CISIZE = control-interval-size] [,DUMMYREC = rec-interval]

[,CASIZE = control-area-size]

[,CKPTLIM = number-of-records]

[,ERROPT = { ABORT | IGNORED | SKIP }] [,WRCHECK]

Other define-parameters can also be used, especially those which relate to the unit record devices PRINTER, READER and PUNCH.

The DATAFORM parameter is not supported. No conversion SARF ↔SSF is provided by any of the utilities. Such conversions are performed by $LIBMAINT.

No DATACODE conversion is provided except if it can be performed automatically by the system at the level of the device attachment mechanisms.

The keyword NBBUF may be coded in all cases. It specifies the number of buffers, and the default value is 1. To handle binary decks of cards NBBUF must be set to 2.

BLKSIZE, RECSIZE, RECFORM, and NBSN can only be used with tape or cassette files, and are used as follows :

For tape input :

- Specify BLKSIZE, RECSIZE, and RECFORM when LABEL = NONE, when LABEL = NSTD, or when the labels are incomplete (IBM DOS format).

For tape output :

- BLKSIZE, RECSIZE, and RECFORM must always be specified unless the file is cataloged. If no BSN's are required, then NBSN may be specified.

Handling of a deck of cards also requires BLKSIZE and RECSIZE, RECFORM be given.

The parameter BPB (Blocks per buffer) only applies to BFAS Sequential and indexed Sequential files. The default value is 1, and the parameter may be specified for input and output.

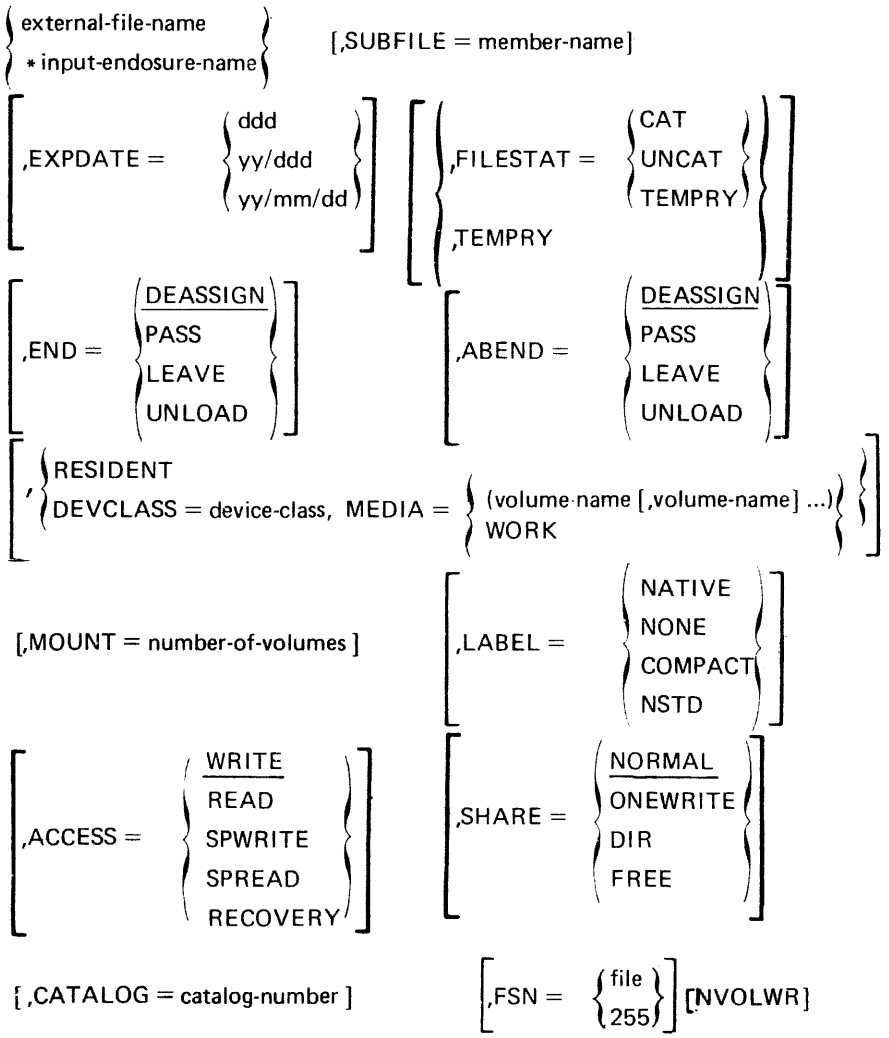
The DUMMYREC parameter only applies to BFAS Indexed Sequential files on output.

FILEFORM = NSTD can be used to handle tape files using the Foreign tape processor similarly, a Foreign cassette handler is available. Native standard files can also be processed using the foreign tape or cassette handlers, but certain services are not available. For example, there is no repositioning in the event of a malfunction or a restart, and there is no automatic track switching for cassettes. It should be noticed that the record level utilities, and in particular $CREATE, are not designed to be used as tape handlers which can do anything with any tape.

It should be noted that only a subset of the define parameters is applicable to any given file. Relevant information concerning a parameter, the given file format, file organization (or unit record device) can be found in the appropriate access method users guide or the Unit Record Devices User Guide.

The file-description that is applicable to the utilities is shown below. The parameters which are mandatory, not used, or which may be omitted with the input and output files, are shown in the description of each file.

The file-description corresponds to the file information which may be supplied using the $ASSIGN Basic JCL statement. The format of the most common file description options is as follows:



For other options and further details, see the $ASSIGN statement in the JCL Reference Manual.

Note :

The file-description is related to an internal-file-name which becomes visible to the user in various circumstances, including messages relative to step reporting. The internal-file-name is identical to the keyword which introduces the file-description.

INDEF

– Function :

To provide file attributes which are used by the access method to handle records from the input file that the INFILE parameter group describes.

– Parameter form :

INDEF = (define-parameters)

– Other similar groups are :

INDEF1 = (define-parameters)

which can be used instead of the INDEF parameter group

INDEF2 = (define-parameters)

which is related to the file that the INFILE2 parameter group describes.

– Restrictions :

The JOURNAL option has no meaning.

WRCHECK has no meaning.

Other restrictions depend upon the utility using the parameter group.

– Associated internal-file-names :

Keywords	Related file-description	Related internal-file-name
INDEF	INFILE	infile
INDEF1	INFILE	infile
INDEF2	INFILE2	infile2

–. Parameter description :

The define-parameters have been described earlier in this section. They can be separated into two classes : those related to the characteristics of a file, and those related to the way the file is to be processed by the utility (that is CKPTLIM, BPB, ERROPT).

The first class of define-parameters need not be used when the associated input file is a native disk or tape file ; this is because every characteristic required is available in the file label, and label information overrides user provided define-parameter values.

The second class of define-parameters is given a set of default values, thus avoiding the necessity to specify these parameters when their default values are convenient to use.

Consequently, none of the INDEF, INDEF1, or INDEF2 parameters are mandatory ; they can be omitted whenever the input files are native labelled files, and the default values for the file processing define-parameters are convenient.

If one of the input files is either an unlabelled file or a non-standard file, or if it is a Level 62 file (which does not have a HDR2 label), the corresponding INDEF parameters must be provided. An input cards deck also requires an INDEF parameter group.

Similarly, if the user wishes the input file, INFILE, to be accessed with double buffering, or if the user wants to specify that unreadable blocks be skipped, the INDEF parameter group must be present, even if the file is a native labelled file.

INDEF is a record level utility option. With the file level utilities, only $FILDUPLI allows this option. In this case, INDEF is to be used to set the file processing parameters **only**, according to the user's wishes.

INFILE, INFILES

– Function :

To describe the assignment to a utility step of a file which is given as input.

Parameter form :

INFILE = (file-description)
INFILES =((file-description) [,(file-description)] ...)

– Other similar parameter groups are :

INFILE1 = (file-description) or INFILES1 = (list-of-file-descriptions)
INFILE2 = (file-description) or INFILES2 = (list-of-file-descriptions)

INFILE1 is a synonym of INFILE. INFILE2 is used whenever two distinct input files are accessed at the same time (e.g. $COMPARE).

– Restrictions :

The parameter EXPDATE is not allowed with these parameter groups.

Internal-file-names :

File-description keyword	Related internal-file-name
INFILE	infile
INFILES	infile
INFILE1	infile
INFILES1	infile
INFILE2	infile2
INFILES2	infile2

– Parameter description :

Depending on the utility, the file will be accessed either at record level via an access method, or globally when the utility lists or modifies the file attributes or when it applies to the file in its entirety. In this last case, it most often happens that the file is accessed at basic or I/O level, that is, track by track or block by block.

In all cases the file is accessed in input mode, which means that no write operation is performed by the utility on the device that holds the file. The exception to this rule occurs if the file is a BFAS Indexed Sequential file, in which case, statistical information on the file usage is written when the file is closed.

The input file can be a disk, tape, or cassette file, and with the record level utilities, it may be an input enclosure.

The input file can be an input enclosure whose contents are card images. By choosing appropriate values for the CONTCHAR and ENDCHAR options of the $INPUT statement ,card images can hold records longer than 80 characters. Input enclosures are only supported by the record level utilities.

The file may be temporary, permanent uncataloged, or permanent cataloged, and it may be passed or not. File concatenation is available for tape files through the INFILES parameter group, and is supported by the record level utilities.

Before the utility is run, the input file must have been preallocated and loaded, and if it is a UFAS Indexed Sequential file which is to be accessed via secondary indexes, the utility $SORTIDX must have been used to sort the secondary indexes.

The option INFILES is to be used when descriptions are provided for several files which are intended to be viewed by the utility as a single file. Everything happens as if each described file was concatenated to other described files in the same order they are described.

OUTDEF

– Function :

To provide information needed by the access method to create the output file of the utility (when it does not previously exist), and to process it according to the user's requirements.

– Parameter form :

OUTDEF = (define-parameters)

– Restrictions :

ERROPT has no meaning.

Other restrictions may apply, depending on the utility and on the fileform and file organization of the file OUTFILE.

– Associated Internal-file-names :

Keyword	Related file-description	Related internal-file-name
OUTDEF	OUTFILE	outfile

– Parameter description :

The OUTDEF option cannot be used unless the OUTFILE option is also used.

When OUTFILE is a disk file, or a preallocated cataloged tape file, the OUTDEF option can only be used to change the file processing parameters to values other than the default values (that is, JOURNAL, BPB, NBBUF, etc. . .).

When OUTFILE is an ordinary tape file or a cassette file that does not yet exist, OUTDEF is mandatory to establish the file characteristics to the required values ; file label will be built from the user defined values. OUTDEF is also mandatory when OUTFILE is a temporary disk file that has not been passed by a previous step, or when it is a deck of cards.

With the record level utilities, OUTDEF can be used to provide both classes of define-parameters : file characteristics and file-processing parameters.

With the file level utilities, OUTDEF is only accessible to $FILDUPLI. In this particular case, the file characteristics are automatically set to the same values as the input file, and therefore, OUTDEF can only be used to access the file processing parameters of the define-parameters group.

OUTFILE

Function : To describe the assignment to a utility of a file which is used as an output file.

Parameter form :

OUTFILE = (file-description)

Restrictions :
The parameter NVOLWR is not allowed with this parameter group
The OUTFILE file description cannot refer to an input-enclosure.

Internal-file-names :

File-description keyword	Related internal-file-name
OUTFILE	outfile

Parameter description :

As for INFILE, the file is accessed either at record level, or globally when the utility modifies the file attributes or applies to the file in its entirety (for example, $FILREST and $FILDUPLI).

In all cases, the file is accessed in output mode, which means that write operations are performed by the utility on the device which contains the file.

The file may be a disk, tape, or cassette file, but preferably should not be a stream attached to a unit record device such as a card punch or a printer, as proper driving of such devices is not fully insured by utilities themselves (no error handling, etc.).

The file can be a temporary file, an uncataloged permanent file, or a cataloged permanent file, and may be a passed or an unpassed file.

File concatenation cannot be used with an output file.

The output file can of course reside on one or several volumes.

The output file is accessed at record level using standard access methods or the Foreign tape processor by record level utilities. It is accessed at I/O level (track by track or block by block) by file level utilities.

PRTDEF

– Function :

To provide the information needed by the access method to create and load the file where the utility is to store the report that it produces.

– Parameter form :

PRTDEF = (define-parameters)

– Restrictions :

ERROPT is meaningless with this parameter.

The only define parameters that can be used are those which are allowed with sequential output files.

– Associated Internal-file-names

Keyword	Relative file-description	Internal-file-name
PRTDEF	PRTFILE	prtfile

– Parameter description :

The PRTDEF option cannot be used unless the PRTFILE option is used.

By default, the PRTFILE file is the sysout file established by the system to hold the printed reports that result from the execution of the utility step. Note that some utilities do not use a sysout file and do not produce a printed report ; some utilities only produce JOR messages - see the descriptions of each utility for details.

The utilities that use a sysout file set the following default values for tape files :

BLKSIZE = 604, RECSIZE = 600, RECFORM = VB, NBBUF = 1,
DATAFORM = SSF, FILEORG = SEQ.

For further details on how to establish the various parameter values refer to the Output Writer section of the COBOL User Guide.

The use of the COMPACT option is of interest when prealiocating a PRTFILE file.

PRTFILE

– Function :

To describe the assignment of a permanent file, to be accessed through the SYSOUT access method, to a utility.

– Parameter form :

PRTFILE = (file-description)

– Restrictions :

The NVOLWR option is not allowed with PRTFILE.

The output file described cannot be an input enclosure, and DUMMY cannot be specified in the file-description.

The option FILESTAT = TEMPRY should not be used because the file may well have disappeared by the time the Output Writer processes it.

– Associated Internal-file-name :

Keyword	Internal-file-name
PRTFILE	prtfile

– Parameter description :

The PRTFILE may be a sequential tape or disk file which can be printed, after the step has finished, by the $WRITER Basic JCL statement.

If PRTFILE is a disk file or a cataloged tape file, it must have been preallocated before the utility is executed.

When the PRTFILE option is used with a utility, the printed report that this utility produces is sent into this private file instead of being loaded into the system file SYS.OUT. The report can be printed from this private file at any later time more convenient to the user.

PRTOUT or SYSOUT

– Function :

To indicate to the system Output Writer how a file is to be printed.

– Parameter form :

PRTOUT = (sysout-parameters)
or
SYSOUT = (sysout-parameters)

– Associated Internal-file-names :

Keyword	Related file-description	Related internal-file-name
PRTOUT	PRTFILE	prtfile
SYSOUT	OUTFILE	outfile

– Parameter description :

The sysout-parameters for PRTOUT and SYSOUT consist of the parameters to the Basic JCL statement $SYSOUT, and are described in this section.

PRTOUT can be used whenever PRTFILE is allowed. When PRTFILE is not used, PRTOUT applies to the standard default SYS.OUT file.

When no sysout-parameters are given, the PRTFILE file is not printed, and stays available (as a permanent file) for a possible off-line printing performed via the Basic JCL statement $WRITER.

SYSOUT can be used whenever OUTFILE is allowed, and only in connection with a record level utility. SYSOUT must be used to inform the Output Writer that it is to process the contents of the file OUTFILE. The contents of this file will then be printed or punched without any editing process occurring. Note that care should be taken if OUTFILE contains binary data.

STEPOPT

– Function :

To specify the execution conditions of the utility.

– Parameter format :

```
STEPOPT = ([load-module-name], [load-module-library-name]
          [, XPRTY = step-exec-priorty]
          [,CPTIME = cplim]
          [,ELAPTIME = elaplim]
          [,LINES = sysout-print-lim]
          [,CARDS = sysout-card-lim]
          [,DUMP = { NO   } ]
                   { DATA }
          [,REPEAT] )
```

– Parameter description :

The parameters to the STEPOPT parameter are described briefly below. For a complete description, see the $STEP basic JCL statement in the JCL Reference Manual. The parameters in the STEPOPT parameter group are :

XPRTY — This specifies a one-digit step execution (dispatching) priority between 0 (the highest priority) and 9 (the lowest priority). The default value of XPRTY is determined by the CLASS of the job containing the step.

CPTIME — Specifies the maximum permitted CPU usage time for the step, in units of one-thousandths of a minute. The value cplim may consist of up to seven digits.

ELAPTIME — This parameter specifies the maximum permitted clock-time during which the step can execute. The unit of time is one minute, and the elaplim value can consist of up to four digits. By default, there is no limit on step execution time.

LINES — This specifies the maximum number of printer records that may be written to each SYSOUT file by the step. The value of sysout-print-lim must not exceed eight digits. If the limit is exceeded, the return code ERLMOV is produced. By default, there is no limit.

CARDS — Specifies the maximum number of card images that may be written to the standard SYSOUT file by the step. The value of sysout-card-lim must not exceed eight digits. If the limit is exceeded, the return code ERLMOV is produced. By default, there is no limit.

DUMP — This specifies if a dump listing is to be produced in the event of abnormal step termination, and, if so, what its contents will be.

- NO — By default, no dump listing is produced.
- DATA — Only segment containing data are dumped.

REPEAT — This specifies that the step is to use the Checkpoint/Restart facilities. This provides the step with a recovery point at which it can possibly be restarted after a system crash or abnormal termination of a step. If REPEAT is not specified, by default the step has no recovery points and cannot be restarted. For further details on when checkpoints are taken and how a step can be restarted, see the utility descriptions.

SYSOUT - PARAMETERS

– Function :

To define how an output file is to be printed by the Output Writer.

– Parameter form :

[,CLASS = output-class]
[,PRIORITY = output-priority]
[,WHEN = { JOB | STEP | IMMED | DEFER }]
[, { HOLD | NHOLD }]
[,NAME = symbolic-name]
[, { BANNER | NBANNER }]
[BANINF = (name1 [,name2] ...)]
[,COPIES = number-of-copies]
[,DEVCLASS = device-class [,MEDIA = media-name]]
[,DEST = station-name]
[, { SLEW | NSLEW }]
[,DELETE]

– Parameter description :

These parameters correspond to those of the $SYSOUT Basic JCL statement. For full details, see the JCL Reference Manual.

5. Record Level Utilities and the DSL

This section contains descriptions of the Record Level Utilities in alphabetic order by statement name. After the specifications, there is a detailed description of the Data Services Language (DSL) which can be used with these utilities. Each utility specification consists of :

1. The function or purpose of the utility.
2. The statement formats, using the conventions defined in the preface
3. A description of the utility operation
4. A description of the parameters and keywords
5. A set of one or more examples.

A summary of these utilities is given in Table 5-1.

Statement name	Function
$COMPARE	Compare the contents of two files
$CREATE	Load or reorganize a file
$PRINT	Print logical records from a file

Table 5-1. Record Level Utilities

The record level utilities are identical in the way they handle files and in their step completion conditions. A common description of these features is given at the beginning of this section.

STEP COMPLETION CONDITIONS FOR RECORD LEVEL UTILITIES

When a fatal error occurs during the translation of the utility statement, files are left untouched, and are not opened.

When a syntax error or any other user error is detected during the translation of the DSL statements from the COMFILE, the utility step aborts with a SEV3 termination status. As the files have been opened, they are closed, and the OUTFILE will not hold any records unless it was previously loaded with data records and the APPEND option was used. If OUTFILE contained records before the utility step was executed, and the APPEND option was not used, it will be left with no records in it.

When an error occurs during the processing of the input file(s), and if the error is FATAL, every file is closed at the point that has been reached, and OUTFILE is partly loaded. A JOR message informs the user how many records were processed from each file (the message key is DU03.01). Whenever possible, the utility tries to print the last processed record into the sysout or PRTFILE file.

If the error occurred during a READ operation, the printed buffer will not necessarily show significant contents, but if the error was during a WRITE operation, the contents of the printed buffer will be significant.

If the REPEAT option is used in the STEPOPT parameter group, checkpoints are taken automatically after every 100,000 records are processed for disk files, and at every end-of-reel for tape files. If the described default values for checkpoint taking are not satisfactory for the user, the CKPTLIM option of the define-parameters for the file(s) can be used to define when checkpoints are to be taken.

Note that even if the REPEAT option is used, the step is only restarted if it has aborted with a SEV4 completion status.

COMPARE

Function :

To logically compare the contents of two files. The user data part of each record in the first file is compared with that of the corresponding record from the other input file. The differences are printed.

Statement form :

```
$COMPARE {INFILE1  = (file-description)                                          }
         {INFILES1 = ( (file-description-1) [,(file-description-2)] ...)         }
         [,INDEF1  = (define-parameters)]
        ,{INFILE2  = (file-description)                                          }
         {INFILES2 = ( (file-description-1) [,(file-description-2)] ...)         }
         [,INDEF2  = (define-parameters)]
         [,OUTFILE = (file-description) ]
         [,OUTDEF  = (file-definition) ]
         [,SYSOUT  = (sysout-parameters)]
         [,PRTFILE = (file-description)]
         [,PRTDEF  = (define-parameters)]
         [,PRTOUT  = (sysout-parameters)]
         [,COMFILE = (file-description)]
         [,START = {n}]
                   {1}
         [,INCR  = {m}]
                   {1}
         [,HALT  = p]
         [,PRINT = ([,FORMAT = {ALPHA}] [,TITLE = 'character string'])]
                               {BOTH }
                               {HEX  }
         [,TAPEND = {q}]
                    {1}
         [,EQUAL]
         [,STEPOPT = (step-parameters)];
```

Statement description :

The files to be compared may be of different organizations, as the comparison only involves the user data part of the records. The usual BFAS and UFAS file organizations are supported, as are IBM DOS files and Level 62 files.

You can compare entire files or selected records of files. The selection can be done by the DSL (described in this Section) and by the parameters START, INCR, and HALT. The process of record selection is described in Figure 5-2. Basically, record number n (START) is found, and the records after this are used by the DSL to produce a group of records for input to INCR. INCR selects every m'th record from this group, and these are passed onto $COMPARE for processing. After p (HALT) records have been printed, processing will stop. The DSL can also be used to rearrange records. This feature allows the sorted keys to be chosen as part of the records, and thus produces a better listing of mismatches. The DSL statements are applied identically to each input file.

When the records have been DSL processed, they are assumed to have the same lengths (even if they were of different lengths to start with), and to be ready for comparison. Records in the same relative position are compared according to the normal sequence. User defined collating sequences are not allowed.

When the compared records are different, and if the PRINT option is used, a message is sent to the output listing file. The message states and describes the difference. If the EQUAL option is used, records which are the same are printed.

Once a difference has been found, it is necessary to make sure that the input files are still "synchronized" before the comparison process continues. To do this, $COMPARE assumes that the two input files have been sorted into two compatible ascending sequences of records before processing starts. If this is done, the file which delivered the lowest record of the "not equal" pair is read until a "not lower" record is found. At this point, either the current pair of DSL processed records are equal, or they are not, in which case, the synchronization process is carried out on the other file. This is repeated until the DSL processed records are equal.

One or both of the input files may be a card deck, which can be introduced as an input enclosure. This deck can contain records which are longer than 80 characters if the appropriate CONTCHAR and ENDCHAR parameters are given with the $INPUT statement.

The PRINT parameter is used to ask for a printed report, and enables you to choose the format of the printed report and its title. The margin, output line length, number of copies, and so on, can be supplied by the PRTDEF and PRTOUT parameter groups.

The outfile file contains those records which are in INFILE2, but not in INFILE1 ; if EQUAL has been used, then it will contain the records in INFILE2 which are the same as those in INFILE1. The outfile file is optional, and, when present, always contains records from INFILE2, and never contains records from INFILE1.

$COMPARE will also compare non-standard magnetic tape and cassette files. This is done by specifying FILEFORM = NSTD in the INDEF1 and INDEF2 parameter groups. The comparison is between physical blocks, and tape marks are treated as one byte blocks, conventionally shown as hexadecimal FF. Because there is no logical end to the processing, the HALT or TAPEND parameter must be used to stop processing. Using the DSL, it is possible to select blocks and rearrange their contents before comparison. Note that with FILEFORM = NSTD, the file is accessed through the Foreign Tape Processor, and the usual repositioning of the tapes and cassettes does not occur ; and tape marks show up as a one byte value (FF).

If the default values of the parameters are used, they will require the minimum of resources, but will give the maximum elapsed time. The execution speed of $COMPARE can be improved by setting NBBUF and BPB from the INDEF1 and INDEF2 parameter groups to values higher than their defaults.

If there is a lot of (printed) output, this can also be done in the PRTDEF and OUTDEF parameter groups.

COMPARE

Parameter description :

INFILE1
INFILES1
INDEF1 — These parameter groups describe the first input file.

INFILE2
INFILES2
INDEF2 — These parameters describe the second input file.

OUTFILE
OUTDEF — These optional parameters describe the output file.

SYSOUT — This optional parameter allows you to inform the Output Writer that it must process the contents of OUTFILE.

PRTFILE
PRTDEF
PRTOUT — When the PRINT option is used, these optional parameter groups define and describe a private output file to which the printed output is to be sent.

COMFILE — This parameter is optional, and describes a file which is to contain the DSL statements. A complete description of the DSL statements is given in this section. If the parameter is not specified, all the input records are transmitted to $COMPARE without any restructuring of the data fields.

START — This optional parameter which is an eight digit decimal number with a default of 1, indicates that the first n-1 records of the input files are to be ignored.

The first record to be considered for DSL and further utility processing are the records numbered n in the sequence of input records ; all records, including the n'th, after the n-1'th are processed. If the parameter is omitted, no records are ignored.

INCR — Those records which have been DSL selected are counted, starting with one. From these records, only one every m is selected for further processing by the utility. Thus records numbered 1, m+1, 2m+1, 3m+1, ... are the only ones to be compared. If INCR is omitted, all DSL selected records are passed on for further processing.

HALT — This optional parameter limits the amount of output produced by $COMPARE, and also terminates processing. Each output record (or prtfile record) is counted, starting at one. When p records have been counted, processing stops.

PRINT — This option must be used if the PRTFILE is to be loaded with a printed report of the mismatching (or matching records if EQUAL is used).

FORMAT TITLE

These options can be used to specify a format and title for the printed report (refer to PRINT parameter description, see below).

TAPEND — This option can be used when the input files are non-standard tape or cassette files. The processing stops when n tape marks have been read. The default value is 1, but TAPEND should always be specified when non-standard input tapes are being processed.

COMPARE

EQUAL — This optional parameter specifies that DSL processed-records that are equal in the input files are to be sent to the output files (OUTFILE or PRTFILE).

STEPOPT — If REPEAT is used, checkpoints are taken every 100,000 records of every file, unless otherwise specified by the user via the CKPTLIM parameter of the define-parameter group.

Examples :

1) To compare two catalogued files and obtain the count of mismatches :

```
COMPARE INFILE1 = .FIRSTFILE, INFILE2 = .SECONDFILE;
```

2) To compare two files position 10, length 3, position 50 length 4, position 1 length 9. The mismatches will be printed in alphanumeric (default) in a permanent sysout file.

```
COMPARE INFILE1 = (UFASINDEXED, DEVCLASS = MS/M400,
                   MEDIA = C056),
        INFILE2 = (SEQTAPE, DEVCLASS = MT/T9/D1600,
                   MEDIA = 1520),
        PRINT = ( TITLE = 'A TITLE'),
        PRTFILE = (PRIVATE.OUTPUT,RESIDENT),
        COMFILE = *DSL;
$INPUT DSL, PRINT;
        RECORD: ARRANGE= (10,3) (50,4) (1,9) END:
$ENDINPUT;
```

3) To compare two files and select and punch the equal records and place them in a third file. These records are to be printed in both alpha and hexadecimal format.

```
COMPARE INFILE = *CARD,INFILE2=(MYFILE,TEMPRY),
        OUTFILE=(OUT, DEVCLASS= CD/P/C80, MEDIA = PUNCH),
        OUTDEF= (BLKSIZE=80,RECSIZE=80,RECFORM=F),
        PRINT = (FORMAT = BOTH),
        PRTFILE=(PRIVATE.OUTPUT,RESIDENT),
        PRTDEF = (PRINTER = (MARGIN = 5)),
        PRTOUT = (DEVCLASS = PR/H155) EQUAL ;
```

4) To compare two sets of files:

```
COMPARE INFILES1 = ((OLD36,DEVCLASS= MT/T9/D1600,MEDIA = 1006,
                     FSN=3), (OLD38,DEVCLASS=MT/T9/D1600,
                     MEDIA=1006,FSN=6)),
        INFILES2 = ((NEW36,DEVCLASS=MT/T9/D1600,MEDIA = 1008,
                     FSN = 10), (NEW38,DEVCLASS = MT/T9/D1600,
                     MEDIA = 1008,FSN=13)),
        PRINT = (FORMAT = HEXA);
```

CREATE

Function :

To load or reorganize a UFAS or BFAS file.

Statement form :

```
$CREATE { INFILE  = (file-description)                          }
        { INFILES = ((file-description-1), (file-description-2) ...) }
        [,INDEF    = (define-parameters)]
         ,OUTFILE = (file-description)
        [,OUTDEF  = (define-parameters)]
        [,SYSOUT  = (sysout-parameters)]
        [,PRTFILE = (file-description)]
        [,PRTDEF  = (define-parameters)]
        [,PRTOUT  = (sysout-parameters)]
        [,COMFILE = (file-description)]
        [,START = { n }]
                  { 1 }
        [,INCR  = { m }]
                  { 1 }
        [,HALT  =   p]
        [,FILLER = { 'character' }]
                   { hexvalue    }
        [,PRINT = ([FORMAT = { HEX   }]  [TITLE = 'character-
                             { ALPHA }              string'])
                             { BOTH  }
        [,APPEND]
        [,KEYLOC = value]
        [,TAPEND = { 9 }]
                   { 1 }
        [,STEPOPT =  (step-parameters)];
```

Statement description :

The utility copies the records of the input file into the output file. The input file may be a disk, tape, or cassette file, a card deck, or a card deck image contained in an input enclosure. The input file organization may be BFAS (sequential, indexed sequential, or direct) or UFAS (sequential, indexed sequential, or relative). IBM DOS files (sequential, indexed sequential, or direct) and Series 60 Level 62 files (sequential, indexed sequential, or direct) are also accepted.

The files may have the record format fixed or variable, blocked or unblocked, or undefined.

When input records are larger than the output records, the input records are truncated on the right to conform to the maximum output record length.

When input records are shorter than the output records, and the output records have the format fixed blocked or fixed unblocked, the input records are expanded to match the output records. In this case, the space which is added to the input records is filled with space characters or with the character or hexadecimal code which has been specified in the FILLER parameter. If the output record format is variable (variable blocked or unblocked, or undefined), then the FILLER option is ignored, and the output record length is the same as the input record length.

For input files which have no labels (LABEL = NONE, cards, IBM DOS files and Level 62 files, except indexed sequential, or cassettes); the file characteristics must be given in the INDEF parameter.

The utility accesses the input file in a sequential manner. For output to a UFAS Indexed file or a BFAS Indexed file, records must be delivered in the ordered sequence based upon the key field. The position and length of the key field are those given in the $PREALLOC utility which allocated the file. Note that for creations between BFAS Direct files and UFAS Relative files (direct-direct, direct-relative, relative-direct, relative-relative) the access is direct in order to preserve any deleted records in the output file.

If the APPEND parameter is present, then records are added to the output file. These additional records are written sequentially after the last record currently present. The APPEND parameter is not allowed with COMPACT cassette files or with non-standard tape or cassette files. Note that if the files are not preallocated, the allocation parameters must be given in the OUTDEF parameters (BLKSIZE, RECSIZE, RECFORM, FILEFORM, and, optionally, NBSN). This often has to be done when the output files are on tapes or cassettes, or are cards to be punched.

It is possible to load or append a file with the contents of a card deck which contains records whose length can be greater than 80 characters, or variable. To do this, the card deck should be contained in an input enclosure and the correct values chosen for the CONTCHAR and ENDCHAR parameters of the $INPUT statement.

It is not possible to load a complete library file, but each subfile of the library file can be loaded with $CREATE. Note that $CREATE does not automatically insert the control records that text to be processed later by $LIBMAINT must contain,to conform to SSF format. Conversion from SARF to SSF is done with the MOVE command of $LIBMAINT.

CREATE

The Data Services Language (DSL) available from within the COMFILE allows the user to select (and load into the output file) only certain records from the input file. It also allows these selected records, or all the records if no selection took place, to be modified by inserting new fields or by updating and moving existing fields. The records which can be loaded into the output file may be the selected records, or all of the records including the selected (and possibly modified) ones. This feature allows correction of a file at record level by selection of incorrect records, correction of these records, and then loading the corrected file. A complete description of the DSL is given in this section.

In addition to the record selection performed by the DSL, you can ask $CREATE to ignore the first n-1 records of the input file and start processing at the n'th input record. This is done by using the START = n parameter. Similarly, you can ask the utility to stop processing the input file as soon as p records have been printed (HALT = p).

It is possible to load «sample» files by selecting one input record every m records (INCR=m).

The $CREATE utility will print the output file while it is loaded if the PRINT parameter is specified.

A UFAS Indexed Sequential file can be accessed according to any of its secondary indexes by using the parameter KEYLOC=secondary-key-location in the file-description.

When the input file contains deleted records, these records are transmitted to the output file unless the output file was preallocated with the NDLREC parameter present. This means that $CREATE will not change the relative numbers within a file if it is used carefully.

Non-standard input tapes and cassettes are accepted by $CREATE, but there are a number of restrictions, such as no automatic track switching for cassettes, and no automatic repositioning at restart. With non-standard tapes and cassettes, each block is read and written as a record, and FILEFORM = NSTD should be specified in the INDEF and OUTDEF parameters. A tape mark is printed and written as a one byte block (hexadecimal FF). Every block is read and written, including the labels, so there is no natural end to the processing, and the TAPEND or HALT parameter must be used to end processing. Using the DSL record selection facility, it is possible to produce a standard output tape or cassette from a non-standard input file. It is also possible to produce a standard disk file with RECFORM=U from a non-standard tape or cassette file by writing the output file with one record per block. The user should take care when using these facilities, because of the lack of safeguards mentioned above.

In all cases with $CREATE, the default values of the parameters require the minimum of resources, but give the maximum elapsed time. The execution of $CREATE can be speeded up by setting the NBBUF and BPB parameters (INDEF, OUTDEF, and PRTDEF) to greater than 1.

CREATE

Parameter description :

INFILE
INFILES
OUTFILE
INDEF
OUTDEF
SYSOUT
PRTFILE
PRTDEF
PRTOUT

These parameters describe the input and output files which are used by $CREATE. The file-description and define-parameters are described in Section 4.
INFILE(S) and OUTFILE are mandatory, but the others are optional.

COMFILE

This parameter is optional, and describes a file which is used to contain the DSL statements. A complete description of the DSL statements is given in this section. If the parameter is not specified, all the input records will be copied without any restructuring of the data fields.

START

This optional parameter indicates that the first n-1 records of the input file are to be ignored. The first record to be considered for DSL and further utility processing is the record numbered n in the sequence of input records ; all records, including the n'th, after the n-1 'th are processed. If the parameter is omitted, no records are ignored (default value of 1), and it may be up to eight digits long.

INCR

This parameter allows record selection by sampling. Those records which have been DSL selected are counted, starting with one. From these records, only one every m is selected for further processing by the utility. Thus of the records which are passed on by the DSL for further processing, only those numbered 1, m+1, 2m+1, 3m+1, ... are written into the output file. If it is omitted, all the records are passed on for further processing (default value of 1), and it may be up to eight digits long (refer to Figure 5-2).

HALT

This optional parameter limits the amount of output produced by $CREATE.

Each record processed is counted, starting at one. Processing stops when p records have been counted. The parameter may be up to eight decimal digits long, and the default value causes all the records to be processed (refer to Figure 5-2)

FILLER

This optional parameter specifies the padding character to be used when the output records are longer than the input records. The padding character may be specified as a character between quotes. If the character required is a quote, or if there is no character corresponding to the padding value required, a two digit hexadecimal may be given. The default padding character is a space.

PRINT

This optional parameter specifies whether the output file is to be printed while it is being loaded. When PRINT is omitted, the only output produced is that displayed on the JOB. When it is present , and if the PRTFILE parameter is not given, a standard SYSOUT subfile is produced. When PRTFILE is given, it describes a permanent sysout file which must have been previously preallocated (if it is a disk file). The user has control over this file and the way it is processed by Output Writer, via the PRTOUT parameter.

CREATE

APPEND This parameter is optional, and causes the output records to be added to the existing output file. If OUTFILE is not a sequential file, the records to be added must be presented in ascending sequence by key value, and their key values must be greater than the highest key value already present in the file. APPEND cannot be used with COMPACT cassette files and non-standard tape and cassette files. This parameter can only be used when the OUTFILE is a sequential BFAS or UFAS file. OUTFILE can also be a BFAS indexed file but one must remember that in order to load such a file the records must be delivered in a sorted sequence.

KEYLOC This optional parameter provides the user with a means of specifying which secondary index of a UFAS Indexed Sequential input file is to be used to access the records of the file. When this is omitted, the records are accessed in the primary index order. This parameter cannot be used with BFAS Indexal Sequential files. It may be up to five digits long.

TAPEND This optional parameter is used when non-standard files are being processed. Processing stops after no tape marks have been read. The default vaiue is 1, but TAPEND should be considered mandatory for non-standard tape and cassette files.

STEPOPT This parameter describes the processing options for the step.

Note :

Under the previous release of Level 64 GCOS, it was possible to produce a printed report with $CREATE. This option has been replaced as follows:

- AUDIT information is now automatically produced in the JOR.
- PARAM is now replaced by the PRINT option of the $INPUT Basic JCL statement. When this option is used, the contents of the input enclosure are printed out by the system.

Examples :

1) To create a cataloged file of any organization from another cataloged file :

```
CREATE INFILE = .THISFILE, OUTFILE = .OTHERFILE;
```

2) To load a temporary file with the contents of an input enclosure with variable records, and to simultaneously load a permanent sysout file where the records will be formatted in hexadecimal. A title will be created, and it will be printed on each page of the output during the processing of the $WRITER statement :

```
PREALLOC PERM.OUTPUT, RESIDENT = (SIZE = 1), FILESTAT = UNCAT,
BFAS = (SEQ = (BLKSIZE = 1204, RECSIZE = 600, RECFORM = VB)) ;
CREATE INFILE = *CARDS, OUTFILE = (MYFILE, TEMPRY, END = PASS) ,
        OUTDEF = (BLKSIZE = 5000, RECSIZE = 256, RECFORM = VB,
                  FILEFORM = BFAS),
        PRTFILE = (PERM.OUTPUT, RESIDENT),
        PRINT = (FORMAT = HEX, TITLE = 'THIS IS MY FILE');
$INPUT CARDS, PRINT, CONTCHAR = '/', ENDCHAR = ';';

AAAA ...          A;
BBBBBBBBB ...     B/
BBBBBBB ...       B/
BBBBBBBBBBB;
C;
$ENDINPUT;
```

As can be seen, it is easy to create test files which can be used to debug new programs. These files can have variable length records, and the record length can be longer than 80 characters.

3) To append onto a cassette file records from a separate deck of cards with printing of the created records. The printing of these records will be in both formats : alphanumeric and hexadecimal ; each line will start in column 5 and stop at column 155.

```
CREATE INFILE = (CARDS, DEVCLASS = CD/R/C51, MEDIA = READER),
        INDEF =(BLKSIZE = 51, RECSIZE = 51, RECFORM = F),
        OUTFILE = (K7FILE, DEVCLASS = CS, MEDIA = K7), FILLER = ' * '
        APPEND, PRTDEF = (PRINTER =  (MARGIN = 5)),
        PRTOUT = (DEVCLASS = PR/H155),
        PRINT = (FORMAT = BOTH);
```

4) To create a native file from a foreign tape. This file has 10 tapemarks, and the file to be created is between the 30'th block and the 10'th tape mark ; the tape marks are not to be copied onto the new created standard file :

```
CREATE INFILE = (MYTAPE, DEVCLASS = MT/T9/D1600,
                 MEDIA = 1070, LABEL = NONE),
        INDEF = (BLKSIZE = 1000, RECSIZE = 1000, RECFORM = U,
                 FILEFORM = NSTD),
        OUTFILE = (STANDARD, DEVCLASS = MS/M400, MEDIA = C036),
        TAPEND = 10, COMFILE = *DSL;
$INPUT DSL, PRINT;
RECORD: OMIT = 1,1 EQ 'FF'HEXA
        OMIT = RECNB LT UBIN '30'
END:
$ENDINPUT;
```

5) To update a file which has a bad byte 19 of the word SMITH. The selection is made on the bad name SMITZ DAVE. The byte 19 is corrected, and all the records of the file are copied by the TRANSMIT = ALL parameter.
The copy will be faster if the number of buffers and the blocks per buffer are greater than 1 :

```
CREATE INFILE = (OLDFILE, RESIDENT),
        OUTFILE = (NEWFILE, RESIDENT), INDEF = (NBBUF=2, BPB = 3),
        OUTDEF = (NBBUF = 2, BPB = 3), COMFILE = *DSL;
$INPUT DSL, PRINT;
RECORD: INCLUDE = 15,10 EQ ' SMITZ DAVE'
         ARRANGE 1,18 ' H' 20,200
         END: TRANSMIT = ALL
$ENDINPUT;
```

6) To create a sequential file from a UFAS Indexed Sequential file, where the records are being taken from a secondary index :

```
PREALLOC TEST. SORTIDX, DEVCLASS = MS/M400, UNIT = CYL,
            FILESTAT = UNCAT, GLOBAL = (MEDIA = C062, SIZE = 50),
            UFAS = (INDEXED = (CISIZE = 1024, RECSIZE = 200,
                    RECFORM = FB, KEYLOC = 1, KEYSIZE = 6,
                    SECIDX    = ((KEYLOC = 134,KEYSIZE = 10,
                                  DUPREC))));
CREATE INFILE = (TEST. PERF2SQ, DEVCLASS = MS/M400,
                 MEDIA = C062),
        OUTFILE = (TEST. SORTIDX, DEVCLASS = MS/M400,
                   MEDIA = C062);
SORTIDX OUTFILE = (TEST. SORTIDX, DEVCLASS = MS/M400,
                   MEDIA = C062);
CREATE INFILE = (TEST. SORTIDX, DEVCLASS = MS/M400,
                 MEDIA = C062),
        OUTFILE = (SEQUENTIAL, DEVCLASS = MS/M400,
                   MEDIA = C083),
                   KEYLOC = 134;
```

7) To create a new file from a set of files with checkpoints every 50,000 records on INFILES and OUTFILES :

```
CREATE INFILES = (K71, DEVCLASS = CS, MEDIA = K7ONE )
                 (K72, DEVCLASS = CS, MEDIA = K7TWO )
        OUTFILE = (COLLECT, DEVCLASS = MS/M400, MEDIA = C036),
        INDEF = (CKPTLIM = 50000),
        STEPOPT = REPEAT;
```

PRINT

Function :

To print records from a file.

Statement form :

```
PRINT  {INFILE  = (file-description)                                          }
       {INFILES = ((file-description-1) [,(file-description-2)] ...)          }
       [,INDEF    = (define-parameters)]
       [,PRTFILE  = (print-file-description)]
       [,PRTDEF   = (define-parameters)]
       [,PRTOUT   = (sysout-parameters)]
       [,COMFILE  = (file-description)]
       [,TITLE    = 'character-string']
       [,FORMAT = {ALPHA}]
                  {HEX  }
                  {BOTH }
       [,START  = {m}]
                  {1}
       [,INCR   = {n}]
                  {1}
       [,HALT   = p]
       [,KEYLOC = key-position]
       [,TAPEND = {9}]
                  {1}
       [,STEPOPT = (step-parameters)] ;
```

PRINT

Statement description :

This utility will print records for a disk, cassette, or a tape file, from an input enclosure, or from a card deck. Through parameters the user may specify that only every n'th record be printed, where n is coded by the user. In the same manner, a user may request that printing is to start or finish after the m'th record.

Input records are retrieved sequentially, and with UFAS files only, they may be retrieved according to the order of a specified secondary index.

The input file may be any BFAS file, any UFAS file, any IBM DOS file, any Level 62 file, any input enclosure, or any card deck. All record formats are accepted.

Records may be printed in alphabetic form, hexadecimal form, or both. The user may specify the line length, margin, number of copies and so on, depending on the values of the parameters of PRTOUT.

A title can be specified, and this is placed as a heading on each page of the output.

The $PRINT utility accepts a permanent SYSOUT file into which the output will be written, instead of having the output dispatched direct to the System Output Writer for printing.

Since $PRINT accepts DSL statements, it is possible to select only certain records for printing. It is also possible to re-arrange or omit data fields within records before each record is printed.

Cards containing more than 80 characters per record can also be printed, if the correct values of CONTCHAR and INCHAR are chosen with $INPUT.

The default options of $PRINT require from the system the minimum of resources, but give the maximum elapsed time. Execution of $PRINT can be speeded up by giving the NBBUF and BPB options of the INDEF and PRTDEF parameters a value greater than 1.

PRINT

Statement parameters :

INFILE
INFILES
INDEF — These keywords describe the input file from which the records are to be printed. The complete definition of the file-description is given in Section 4.

PRTFILE
PRTDEF — These optional keywords are followed by a parameter group which describes a permanent file into which the printed output is written.

PRTOUT — This optional parameter is followed by a parameter group which contains options which control the printing of PRTFILE by Output Writer.

COMFILE — This optional keyword designates the source for the DSL statements.

TITLE — This optional keyword may be coded to provide a title which will appear at the head of each page of the listing. The keyword is followed by the title contained within single quotes. The character string may not contain a single quote. The maximum length is 114 characters. If the title string is continued from one card to the next then the continuation character (–) must be the last character on the card to be continued.

For example :

```
TITLE = «THIS LINE IS TOO LON-
G TO FIT ON ONE CARD»
```

The character (G) above is in column 1 of the second card.

FORMAT — This optional keyword allows the user to choose how the records are printed. If FORMAT = ALPHA then the records are printed as alphabetic characters. If FORMAT = HEX then the record contents are printed in hexadecimal notation, two hex digits to a byte. If FORMAT = BOTH then the contents of records are printed in both alphabetic and hexadecimal formats.

If FORMAT is not coded then FORMAT = ALPHA is assumed.

START — This optional keyword allows the user to specify that printing starts at a given record number. If start is omitted then printing commences at the first record.

INCR — This optional keyword allows the user to specify that only every n' th record be selected for printing. For example if INCR = 8 then only records 1, 9, 17, 25, etc. will be printed.

If INCR is not specified then all input records will be selected for printing, subject, of course, to any selection instructions provided through DSL. Note that INCR selection occurs after INCLUDE/ OMIT processing (refer to figure 5-2).

HALT — This optional keyword allows the user to specify that printing will finish after a given number of records have been listed. For example if HALT = 86 then printing will terminate after the 86th record is displayed. If HALT is omitted then printing will continue until an end-of-file is encountered on input (refer to figure 5-2).

KEYLOC — This optional parameter provides the user with a means of specifying which index of an UFAS Indexed Sequential input file is to be used to access the records of the file. When this is omitted, the records are accessed in the primary index order.

TAPEND This optional parameter is a two digit number that specifies the number of tape marks that are to be read before processing stops. This option is only available when the input file definition holds the parameter FILEFORM = NSTD. The default value is 1, but the option should be considered mandatory.

STEPOPT This parameter describes the processing option for the step.

Example :

1) To print a cataloged file of any organization :

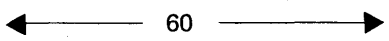
```
PRINT INFILE = THISFILE ;
```

2) To print a passed temporary file into a permanent sysout file with formatted output in alpha and hexadecimal with a title printed on each page. The records 50, 70, 90, 110, 130, 150, 170, 190, 210, 230 and 250 will be written into the permanent sysout file. A $WRITER statement can print this file :

```
PREALLOC PRIVATE. OUTPUT, RESIDENT = (SIZE = 1),
          FILESTAT = UNCAT, BFAS = (SEQ = (BLKSIZE = 1208
          RECSIZE = 600, RECFORM = VB));
PRINT INFILE = (MYFILE, TEMPRY, END = PASS),
       PRTFILE = (PRIVATE. OUTPUT, RESIDENT) ,
       FORMAT = BOTH, TITLE = 'THIS IS MY FILE' ,
       START = 50, INCR = 20, HALT = 11 ;
```

3) To print a cassette file with labels, but of any foreign organization, and with processing stopping when the fourth tape mark is found or if 200 records (blocks) have been printed.

The output format will be alpha (default), the paper width is 70 characters, and the margin width is 10 characters; this means that the blocks will be printed from column 11 to column 70. The record selected will be the first 9 records and the records with a number greater than 50. These records will be formatted as follows :



xxxxxxxxxxxxxxxxxxxxxx . . . xx

```
PRINT  INFILE = (MYK7, DEVCLASS = CS, MEDIA = K7 ),
       INDEF= (BLKSIZE = 200, RECSIZE = 200, RECFORM = U,
               FILEFORM = NSTD), TAPEND = 4, HALT = 200,
               PRTFILE = (PRIVATE. OUTPUT, RESIDENT),
               PRTOUT = (DEVCLASS = PR/H70),
               PRTDEF = (PRINTER = (MARGIN = 10)),COMFILE = *DSL;
$INPUT DSL ;
RECORD: INCLUDE = RECNB  LT  UBIN '10'
        INCLUDE = RECNB  GT  UBIN '50'
        ARRANGE = :'1, 10,:'50, 8:'19, 12:'END:
$ENDINPUT ;
```

4) To print a UFAS indexed file according to one of its secondary indexes.

```
PRINT  INFILE = (TEST. SORTIDX, DEVCLASS = MS/M400,
                 MEDIA = C062), KEYLOC = 134 ;
```

5) To print a set of files :

```
PRINT  INFILE = ((CAB, DEVCLASS = MS/M400, MEDIA = C036),
                 ( CAC, DEVCLASS = MS/M400, MEDIA = C038));
```

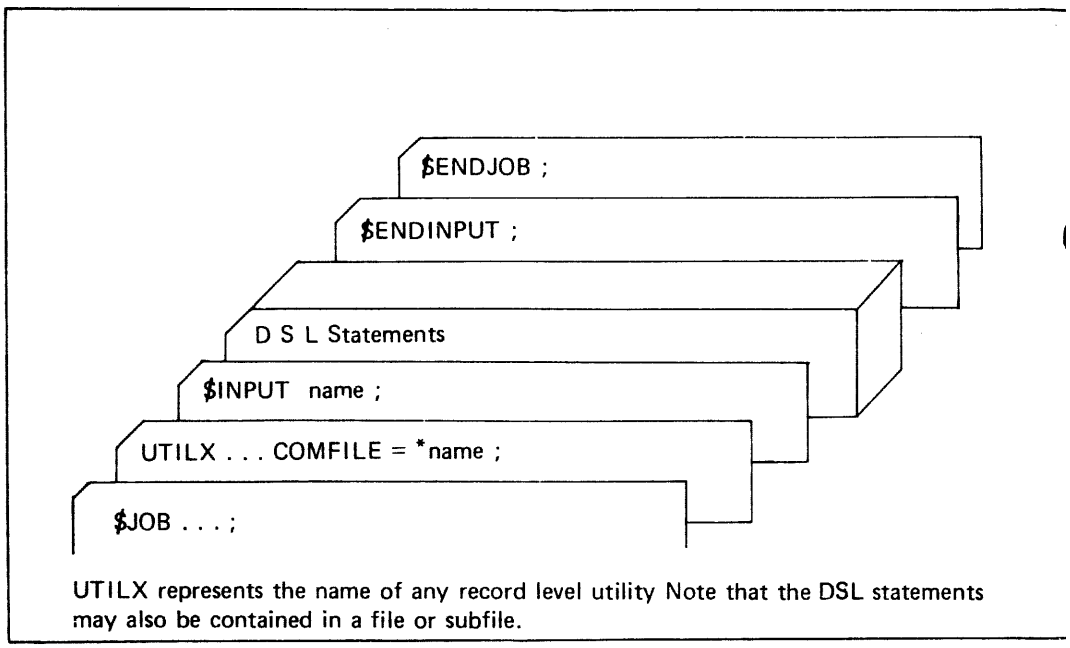
With the utility programs $COMPARE, $CREATE, and $PRINT the user may supply a GCOS input enclosure or any file containing DSL (Data Services Language) statements. Through DSL statements the user may request the following services :

- that certain input records be omitted from the output record stream ;
- that only certain input records be included in the output record stream ;
- that the order and contents of data fields be changed within records.

The selection/omission process may be based on a comparison between two data fields or between a data field and a constant.

The job deck for a utility execution, with DSL, will have the format shown in Figure 5-1.

DSL statements may also be entered from a source library in which they have been previously loaded. They should be loaded as TYPE = DATASSF by $LIBMAINT SL ; for further details see the Library Maintenance Reference Manual.



UTILX represents the name of any record level utility Note that the DSL statements may also be contained in a file or subfile.

Figure 5-1. Job Deck for Utility with DSL

DSL DECK FORMAT

The format of a DSL deck is as follows :

```
RECORD:  [ { INCLUDE = condition [AND condition] ...          }  ]
         [ {[INCLUDE = condition [AND condition] ...] ...     }  ]
         [ { OMIT = condition [AND condition] ...             }  ]
         [ {[OMIT = condition [AND condition] ...] ...        }  ]
           [ ARRANGE = arrange element [arrange element] ...]
END:     [ TRANSMIT = { SELECTED }                            ]
                      { ALL      }
```

Thus a DSL deck consists of an optional set of INCLUDE statements or an optional set of OMIT statements, followed independently by an optional ARRANGE statement. The start and end identifiers, RECORD: and END: must always be present. Every DSL deck must be in the above order. After the END, an optional TRANSMIT statement may be given.

The entity «condition» in the above definition is described in detail below. A «condition» describes a comparison or test to be made on records processed by the utility. The value of a condition is true (T) or false (F).

DSL Syntax Rules

DSL statements are coded in free format. There must be at least one separator between each element. A separator may be either a space character, or an opening parenthesis " (", or a closing parenthesis ")", or a comma. Any number of spaces may appear before or after a separator. The main advantage in using parenthesis and commas is in legibility. Using these rules the following are all equivalent :

```
INCLUDE = 36 4 NE PDEC '0'
INCLUDE = 36,4 NE PDEC '0'
INCLUDE = (36 4) NE (PDEC, '0'
INCLUDE = 36(4) NE PDEC ('0')
```

Note that left and right parenthesis do not have to balance.

Continuation from one card to the next is automatic : the next column after 80 is column 1.

The INCLUDE Statement

If a DSL deck contains one or more INCLUDE clauses then, for a record to be accepted, the condition values must satisfy at least one of the INCLUDE statements. An INCLUDE statement is satisfied only when all the conditions in it are true (T).

Consider the DSL example below :

```
RECORD :   INCLUDE = ca
           INCLUDE = cb AND cc
           INCLUDE = cd AND ce AND cf
END:
```

There are three INCLUDE statements. Records will only be accepted from the input stream if :

ca = T

or :

cb = T and cc = T

or :

cd = T and ce = T and cf = T

Input records which do not satisfy any of the three condition sets above will be skipped.

The tests are performed in the order in which the INCLUDE statements occur. The process of comparison and testing terminates as soon as any INCLUDE statement is satisfied. Therefore for efficiency in execution it is recommended that the INCLUDE statements most likely to be satisfied be placed before others which will be satisfied less frequently.

INCLUDE statements must not be mixed with OMIT statements in the same DSL deck.

The OMIT Statement

If a DSL deck contains one or more OMIT clauses then a record will be omitted if the condition values satisfy at least one of the OMIT statements. OMIT statements are provided as an alternate path of record selection, to be used when it is more convenient to describe unwanted records. An OMIT statement is satisfied only when all the conditions in it are true (T).

Consider the example below :

```
RECORD :   OMIT = ca
           OMIT = cb AND cc
           OMIT = cd AND ce AND cf
END :
```

There are three OMIT statements present. Input stream records will be excluded from the output stream if :

ca = T

or :

cb = T and cc = T

or :

cd = T and ce = T and cf = T

Input records which do not satisfy any of the three sets of conditions above will be included in the output stream.

The tests are performed in the order in which the OMIT statements occur. The process of comparison and testing terminates as soon as a statement is satisfied. Therefore for efficiency in execution it is recommended that OMIT statements most likely to be satisfied be placed before others which will be satisfied less frequently.

OMIT statements must not be mixed with INCLUDE statements in the same DSL deck.

DSL Condition Elements

The condition elements which appear in OMIT and INCLUDE statements contain parameters which describe the type of data field being addressed. The five data types available are shown in Table 5-2.

Parameter	Description	Maximum length (bytes)
CHAR	Alphanumeric Character Data	256
PDEC	Packed Decimal Data	16
UDEC	Unpacked Decimal Data	31
SBIN	Signed Binary Data	2 or 4
UBIN	Unsigned Binary Data	2 or 4
HEXA	Bit String Data	4

Table 5-2. Data Field Types in DSL

The conditions can have one of three possible forms. The first form is applicable only to decimal or binary data.

position length { POS / NEG / ZERO } { UDEC / PDEC / SBIN }

The «position» is the byte location at which the data field starts in the record. The position of the first byte in a record is 1. The «length» gives the field length in bytes. The next field gives the condition to be satisfied for the condition element to have the value true. The value true may be associated with positive field contents, or zero, or negative. The last keyword specifies the data format, Unpacked Decimal or Packed Decimal or Signed Binary.

Examples :

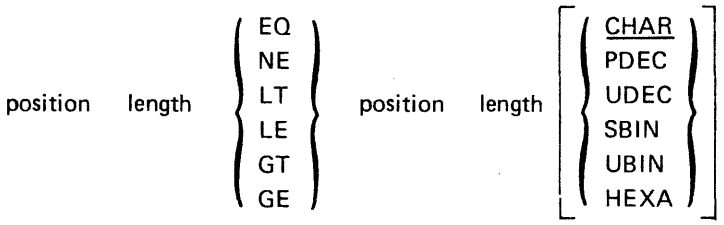
3, 7 ZERO UDEC

This element will be true when the 7 byte long Unpacked Decimal field starting at byte 3 of input records is zero.

1, 4 POS SBIN

The value is true when the signed binary full word data field starting at the first byte of the record is positive.

The second form of condition element is :



In this form the position and length of two data fields is given. The value is true if the asserted relationship between the fields is correct.

There are six possible relationships :

EQ	Equality
NE	Not Equal
LT	Less Than
LE	Less Than or Equal
GT	Greater Than
GE	Greater Than or Equal

If no data format is supplied then the fields are assumed to be CHAR.

Examples :

(4 , 4) EQ (28 , 4) SBIN

The condition is true if the signed binary numbers at locations 4 and 28 are equal. Both fields are fullword (4 bytes long).

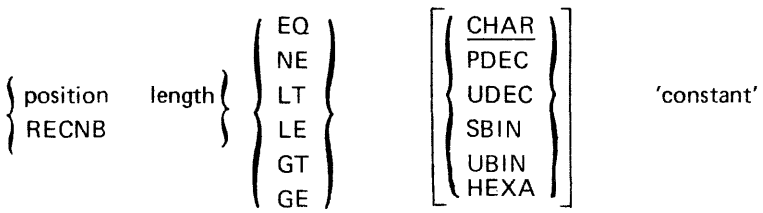
18,6 GE 5,7 UDEC

Two Unpacked Decimal fields are compared, one located at byte position 18, the other at 5. The first field is 6 bytes long, the second is 7. The condition will be true if the first decimal value is greater than or equal to the second.

(5 , 7) LT (18 , 6 UDEC)

This example has exactly the same meaning as the previous one.

The third form of condition element is used when the comparison is to be performed between a data field or the record number and a constant value.



RECNB is the name of a variable whose current value is the record number of the current input record. The data type of this variable is UBIN and it is 4 bytes long.

The value of this condition form is true if the asserted relationship between the field and the constant is correct. The data formats supported are the same as for the second condition form.

The constant is enclosed in apostrophes. If the value is numeric it has the form of a signed (SBIN), or unsigned (UBIN, UDEC) integer decimal value. In these cases a conversion is first made on the constant to produce the internal data format which is used for comparison. If the constant is CHAR in type then it must be the same length as the field against which the comparison is made. The sum of the constant lengths occurring in a RECORD: paragraph must not exceed 512 bytes. If the value is hexadecimal, it has the form of 2n hexadecimal digits (HEXA), where n is between 1 and 4.

INCLUDE and OMIT Examples

In a file transfer operation it is required that only those records with non-zero fields at locations 24 and 36 (4bytes PDEC) be moved.

```
INCLUDE = (36 4 NE PDEC '0' ) AND (24 4 NE PDEC '0' )
```

This statement will cause a record to be skipped if either field is zero. The equivalent operation using OMIT is :

```
OMIT = (36 , 4)  ZERO     PDEC,
OMIT = (24 , 4)  ZERO     PDEC
```

The ARRANGE Statement

The function of an ARRANGE statement is to re-order the contents of records and to insert or delete from a record data fields or constant values. It is an optional statement. If no ARRANGE statement is provided then the structure of ouput records is exactly the same as on input. Only one such statement may be given in a DSL deck. The format of the ARRANGE statement is :

```
ARRANGE = { position length                } [ , { position length                } ] ...
          { data-field-type 'constant'     }     { data-field-type 'constant'     }
```

Where "position" and "length" define fields of the input record (in units of bytes). Data field types are given in Table 5-2, the default value being CHAR. The position of the first byte in a record is 1. Through the ARRANGE statement the user provides a complete description of the output record, in terms of input record fields. It is possible to have certain input fields repeated two or more times or to have other input fields omitted.

Example :

```
ARRANGE = (18 , 40) 10 , 4) (61 , 20)
```

This ARRANGE statement describes output records to be 64 bytes long (40 + 4 + 20 = 64). The first 40 output bytes will consist of bytes 18 to 57 of the input record, the next 4 bytes will be input bytes 10 to 13 and the last 20 will be 61 to 80. Note that bytes 1 to 9, 14 to 17 and 58 to 60 of the input record will not appear in the output record.

The TRANSMIT Statement

The DSL deck (and DSL processing) acts as an interface between the input file and the utility that is applied to it. If record selection is used, only certain records are selected and submitted to the record re-arrangement process. These selected records are then passed on to the utility, which can perform further processing on them before they are directed to the output file.

The TRANSMIT statement allows you to either :

– Only pass the selected records to the utility (SELECTED)

or :

– Pass both the re-arranged records and the non-selected records on to the utility.

This facility enables you to select certain records for re-arrangement and then copy the whole input file, including the re-arranged records. For example, a copy of a file can be made with only certain records modified (or corrected), and the rest of the records unaltered. If the TRANSMIT statement is omitted, then only the selected records are passed on to the utility for further processing.

The format of the TRANSMIT statement is :

TRANSMIT = { SELECTED / ALL }

DSL ENTRY METHOD

The JCL statement for a utility which accepts DSL contains an input enclosure name or a source library subfile reference designated by the COMFILE keyword which describes the source of the DSL deck. If COMFILE is not present then it is assumed that there are no DSL statements to be acted upon.

The format of the keyword for DSL entry is the same for all utilities for which DSL is applicable :

COMFILE = *input-enclosure-name

or :

COMFILE = (library-name, SUBFILE = member-name [, { RESIDENT / DEVCLASS = device-name,MEDIA=volume-name }])

EXAMPLES WITH DSL USAGE

The records of an input file on tape are to be loaded into an indexed sequential file. The tape is labeled. The records are fixed length : 100 bytes. Since the file will be subject to a large number of insertions there is to be one dummy record present for every three records loaded. The format of each record is shown below :

1	5	28	78	88	100
CUSTNO UDEC	CUSTNAME CHAR	ADDRESS CHAR	SALES AREA CHAR	DEBIT UDEC	

These input records have already been sorted in ascending order on the field CUSTNO. This field will be used as the key in the resulting indexed sequential file.

The output format required for the indexed sequential file is :

1	11	15	38	88	100
SALES AREA CHAR	CUSTNO UDEC	CUSTNAME CHAR	ADDRESS CHAR	DEBIT UDEC	

In addition the user wishes to select only those records in which :

1. The first character of SALES AREA is 'A' or 'B'

   and :

2. The fourth character of SALES AREA is not 'L'.

The job deck to load the file is :

```
$JOB      PCJOB, USER = PJJC, PROJECT = DEVEL;
CREATE    INFILE =  (OLD. MASTER, DEVCLASS = MT/T9,
                     MEDIA = TLX44),
          OUTFILE  = (SALE. SAB, DEVCLASS = MS/M402,
                      MEDIA = DB4),
          OUTDEF = (DUMMY REC = 3),
          COMFILE = *CONCA;
$INPUT    CONCA;
RECORD:   INCLUDE = (78,1 EQ 'A') AND (81,1 NE 'L')
          INCLUDE = (78,1 EQ 'B') AND (81,1 NE 'L')
          ARRANGE =(78,10 1,4 5,23 28,50 88,13)
END:
$ENDINPUT;
$ENDJOB;
```

The input tape file, OLD. MASTER is on volume TLX44, the new file, SALE.SAB, on volume DB4. The SALE.SAB was allocated using $PREALLOC with the attributes KEYLOC = 11, KEYSIZE = 4, the position and length of the field CUSTNO. Note the use of the optional punctuation characters ( ) and comma to aid the understanding of the DSL statements.

Note that the INCLUDE data field specifications refer to the input record format and not to the output format. Also, in the above ARRANGE statement it would be more efficient to treat the fields CUSTNAME and ADDRESS as one single field, that is 5,23 28,50 ⇒ 5,73.

The second example (below) shows the use of DSL in conjunction with $PRINT. Selected records from a file DEPT. SKIL are to be printed in alphanumeric format after they have been re-arranged to include only three fields from the input record.

The selected records are those where the field (10,4) has a value less than or equal to ALM8 and where the field (30,6) has a value not equal to '100000'. Only one in 10 of the records selected will be printed and printing is to terminate after 50 records have been printed.

```
$JOB      LOOK, USER = JJM, PROJECT = SPEC;
PRINT     INFILE = (DEPT. SKIL, DEVCLASS = MS/M400,
                    MEDIA = VL89),
          COMFILE = *PRINX, INCR = 10,HALT = 50;
$INPUT    PRINX;
RECORD:   OMIT = (10,4 GT CHAR 'ALM8')
          OMIT = (30,6 EQ UDEC '100000')
          ARRANGE = (10,4 30,6 40,20)
END:
$ENDINPUT;
$ENDJOB;
```

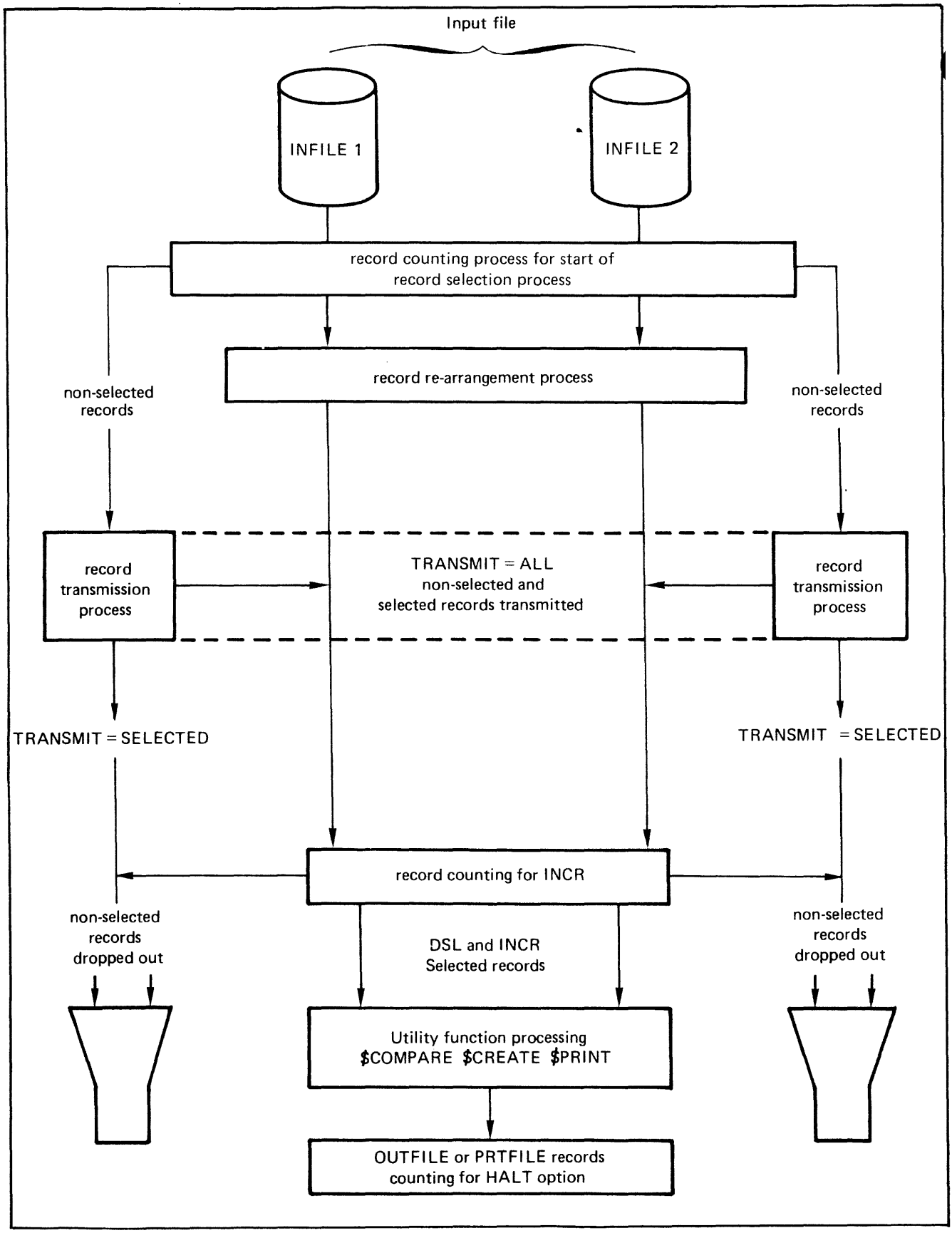


Figure 5-2. An outline of how DSL is used

6. File level utility specifications

This section contains the full specifications for the File Level utility statements. The descriptions are given in alphabetic order by statement name. Each utility specification consists of :

1. The utility function;
2. The statement form, using the notation convention defined in the Preface;
3. A description of the utility operation;
4. A description of the parameters and keywords;
5. A set of one or more examples;

A summary of these utilities is provided in Table 6-1.

Statement name	Function
$DEALLOC	Deallocate a disk or tape file
$FILCHECK	Check the integrity of a disk file
$FILDESC	List file label information
$FILDUPLI	Make a duplicate copy of a disk or tape file
$FILMODIF	Change the name and/or the expiration date of disk file
$FILPRINT	Print the physical blocks of a file
$FILREST	Restore a $FILSAVE file onto disk
$FILSAVE	Save disk file onto tape file
$PREALLOC	Allocate space for a disk or tape file
$SORTIDX	Sort and load secondary indexes of a disk file

Table 6-1. File Level Utilities

DEALLOC

– Function :

To deallocate the storage space of a disk file or cataloged tape file. If the file is cataloged, the catalog must be attached, and it is updated.

– Statement form :

```
DEALLOC external-file name , {RESIDENT
                              DEVCLASS = device class, MEDIA = media-list}
        [CATALOG = digit 1]
        [FILESTAT = { CAT   } ]  [,MOUNT = n] [,FIRSTVOL = n]
                    { UNCAT }
                    { TEMPRY}
        [,FORCE] [,BYPASS]
        [,STEPOPT = (step-parameters)] ;
```

– Statement description :

This utility removes all references to the named file from the Volume Table Of Contents (VTOC). All the space extents allocated to the file are marked as free space contents (that is, they may be used by future $PREALLOC statements). If in the deallocation process an extent is made available which is adjacent to other free extents, then the adjacent free extents are concatenated into one larger extent of free space. If the file being deallocated is multivolume, and an incomplete media-list was given, the file will only be partially deallocated, and may be left in an inconsistent state.

If the file to be deallocated is a cataloged tape file, the tape on which it resides cannot be a multifile tape. Every volume of the file is then returned to a scratch state by rewriting the labels without a "catalog flag" and with an expired expiration date.

$VOLPREP must be used to deallocate non-cataloged files.

$VOLPREP must also be used to deallocate cassette files. Passed temporary files may be deallocated, but there is no need to, as a temporary file is automatically deallocated when the file is closed and deassigned.

If the REPEAT option of STEPOPT is used, then in the case of an abort/restart,the deallocation of a multi-volume file continues, even if the first volume has been already deallocated.

At restart, every volume of the multivolume file is examined and searched for an extent of the file to deallocate. If no extent is found for that file, a warning is issued and processing continues with the following volumes.

– Statement parameters :

Apart from the two parameters described below, the others are identical to the file-description parameter group, with the following exceptions:

– FIRSTVOL may only be used with cataloged files

– MOUNT must not be used for disk files, but it can be used for tape files.

BYPASS This optional parameter causes $DEALLOC to ignore any valid expiration date on the file. If it is not specified, and the expiration date is not passed then the utility will not deallocate the file, but will abnormally terminate.

FORCE This parameter, which is optional, is used to deallocate a cataloged file on disk without consulting or modifying the catalog. It is especially useful if the catalog was destroyed or incorrectly updated. Because the catalog is not consulted, details such as the volumes and media on which the file resides must be specified, and thus FILESTAT = UNCAT must be used. The FORCE option should be used with care as a file recovery facility.

CATALOG This parameter denotes the rank within the $ATTACH statement of the catalog file that holds the file entry which is to be deallocated. When omitted, all of the attached catalogs are searched, in increasing order of rank, for the first catalog entry which matches the given file name.

Examples :

DEALLOC XQ.DAT ;

This deallocates the file named XQ.DAT which by default is cataloged. The expiration date is assumed to have been passed since BYPASS has not been coded.

DEALLOC PQ.DATP, DEVCLASS = MS/M300, MEDIA = DVA ;

The file PQ.DATP on the MSU0310 disk volume DVA is de-allocated.

DEALLOC PAY.MASTER, DEVCLASS = MS/M400,
MEDIA = (MQ64A, MP64A, MR64A) ,BYPASS ;

The multi-volume file PAY.MASTER residing on MSU0400 disk volumes MQ64A, MP64A, MR64A is de-allocated. The expiration date of PAY.MASTER will not be checked before deallocation. Note that the order of the media must be the same as that in the $PREALLOC statement.

The types of errors can be :

VTOC ERROR	:	Nonstandard VTOC
FILE NOT ON VOLUME	:	The files does not reside on the volume.
EXPIRATION DATE NOT OVER	:	The file is still valid and the parameter BYPASS has not been specified.
SYSTEM ERROR—RETURN CODE : xxxxx	:	Storage management has used a primitive giving an unexpected return code.
DEALLOCATION NOT PERFORMED FOR FILE	:	filename
ERROR DURING THE EXECUTION OF	:	system primitive (return code).

This message is displayed when a system primitive invoked by the utility has given an unexpected return code.

MISSING OR ILLEGAL PARAMETERS

Parameters of the statement are not correct for the specified request.

OPTION STRING UNAVAILABLE

The step cannot get the option string.

UNKNOWN COMMAND

In the option string, the request is neither deallocate nor allocate.

FILCHECK

– Function

To ensure that a disk file may be processed by the standard access methods.

– Statement form :

FILCHECK external-file-name

```
[,{ RESIDENT                          }]
  { DEVCLASS  = device-class          }
  {   MEDIA   = volume-name           }
  { CATALOG   = catalog-number        }

[, STEPOPT    = (step-parameters)] ;
```

– Statement description :

The utility verifies that the format of the file label, the dependent file organization label and the additional extent label in the VTOC are correct. It also verifies that control information fields within the file are correct. This last function applies only to library files and certain system files.

In checking sequential files, it scans the file from the first block and bounds the file either at the first end-of-file encountered or at the first inconsistency between the track balance and the track allocation status.

The only file organization supported by $FILCHECK are sequential and library. It does not support multivolume library files.

If an execution of $FILCHECK causes any label corrections to be made then it should be followed by an execution of $VOLCHECK to verify the updated VTOC.

The report produced by the utility consists of a header which states the name of the file being investigated followed by :

- The volume name
- The file organization
- The cylinder/track addresses of the file space
- The detected errors
- The suggested action

– Statement Parameters :

CATALOG This optional parameter gives the rank of the catalog containing the file attributes

– Messages :

The detected errors may be divided into two groups : Label Errors and File Errors.

– Label Errors

FILE ORGANIZATION DEPENDANT LABEL FAILS AND CANNOT BE CREATED VTOC EXTENT IS TOO SMALL

Self explanatory message. No correction can be made.

FILE LABEL VOLUME SERIAL NUMBER ERROR

This message is sent if in the first volume of the file, the volume serial number found within the file label is not equal to the volume serial number found in the volume label. No correction can be made.

FILE LABEL FORMAT IDENTIFICATION HAS BEEN CORRECTED

The format identification of the file label has been set to "F1"

NO SPACE WAS ALLOCATED THIS FILE BEEN DELETED

All the extents of the file label are empty. The label is deleted.

ADDITIONAL EXTENT LABEL NOT ALLOWED THIS LABEL HAS BEEN DELETED

The file label contains less than 3 extents, the additional extent label is then forbidden and is deleted.

FILE ORGANIZATION DEPENDANT LABEL FAILING A NEW LABEL HAS BEEN CREATED

Self explanatory message.

ADDITIONAL EXTENT LABEL KEY ERROR THIS LABEL HAS BEEN DELETED

The pointed label is not the expected additional extent label. The pointer to this label is deleted.

ADDITIONAL EXTENT LABEL FORMAT IDENTIFICATION HAS BEEN CORRECTED

The format identification within the additional extent label has been set to "F3"

ADDITIONAL EXTENT LABEL, NO SPACE WAS ALLOCATED, THIS LABEL HAS BEEN DELETED

Self explanatory.

EXTENTS FOUND AFTER A NON VALIDATED EXTENT HAVE BEEN DELETED

An extent has been found with an error and has been deleted, all the following extents are therefore deleted.

EXTENT SEQUENCE NUMBER ERROR EXTENTS HAVE BEEN DELETED

As the extent sequence number of an extent is wrong, this extent has been deleted.

EXTENT NUMBER ERROR THIS NUMBER HAS BEEN CORRECTED

Self explanatory message.

EXTENT BOUNDS ERROR THIS EXTENT HAS BEEN DELETED

The extent bounds are not valid since there is an inversion between the extent bounds This extent is deleted.

THIS FILE IS EMPTY

There are not any records within the sequential file

THIS FILE HAS BEEN BOUNDED AT LAST SUCCESSFUL CLOSE

The last close performed on the file is validated; all the data recorded prior to this close are accessible.

THIS FILE HAS BEEN BOUNDED AT LAST AVAILABLE EXTENT END

The file check utility has closed the file at the end of the last available extent as described in the file label.

THIS FILE HAS BEEN BOUNDED AT LAST SUCCESSFUL WRITE

The file check utility has closed the file just after the last available record (at the point where data processing has been interrupted).

LAST DATA BLOCK ADDRESS HAS BEEN UPDATED IN FILE LABEL

The last data block address recorded in the file label has been set to the last available block recorded in the file.

TRACK BALANCE HAS BEEN UPDATED IN FILE LABEL

The track balance of the last available track has been modified in the label.

END FILE ADDRESS HAS BEEN UPDATED IN RECORD ZERO

The record number of the end of file record has been set in the record zero of the last available track.

TRACK BALANCE HAS BEEN UPDATED IN RECORD ZERO

The track balance of the last available track has been set in the record zero of this track.

LOGICAL TRACKS HAVE BEEN SET FREE IN THE BAM

The file salvaging has freed some logical tracks from the BAM because these logical tracks don't pertain to any valid subfile.

LOGICAL TRACKS HAVE BEEN SET BUSY THE BAM

The file salvaging has sized logical tracks in the BAM which pertain to valid subfiles : the corresponding flags are set to busy.

FREE LOGICAL TRACKS NUMBER HAS BEEN UPDATED IN THE BAM

The free logical tracks number in the header of the BAM block has been modified.

FORMATED LOGICAL TRACKS NUMBER HAS BEEN UPDATED IN THE BAM

The number of formated logical tracks in the header of the BAM block has been modified.

MAXIMUM SUBFILE IDENTIFIER HAS BEEN UPDATED IN FILE ORGANIZATION DEPENDANT FILE LABEL

Self explanatory message.

DIRECTORY OVERFLOW INDICATOR HAS BEEN UPDATED IN FILE ORGANIZATION DEPENDANT FILE LABEL.

Self explanatory message.

INVALID SUBFILE IDENTIFIER

Subfile has been bounded : one of the blocks chained to a subfile has a wrong subfile identifier. The subfile is bounded at the previous block.

POINTER ERROR A LOGICAL TRACK HAS BEEN DELETED

A prior pointer error has been found while checking a subfile. The subfile has been reduced to its shortest part.

FILE IS NOT FORMATED

The file has been allocated but not still opened (no formating).

SUBFILE BEGINNING ADDRESS HAS BEEN UPDATED

The begin address of a subfile has been updated in the subfile entry of the directory.

AN INVALID ENTRY HAS BEEN FOUND IN DIRECTORY

Self explanatory message

AN INVALID BLOCK HAS BEEN FOUND IN BAM.

Self explanatory message

DIRECTORY OVERFLOW HAS BEEN BOUNDED AT LAST AVAILABLE BLOCK

An invalid directory block has been found in directory in directory overflow, and one has been bounded in the preceding block.

SUBFILE ENDING ADDRESS HAS BEEN UPDATED

The end address of a subfile has been updated in the subfile entry of the directory
The suggested action message will be one of the following :

NO ACTION	
RESTORE FILE	Use $FILREST to restore file
RESTORE VTOC	Use $VOLREST to restore volume
DEALLOCATE FILE	Use $DEALLOC to delete file
PREALLOCATE FILE	Use $PREALLOC to re-build file
PREPARE VOLUME	Use $VOLPREP to prepare the volume

Example :

```
FILCHECK    CRU.MP,DEVCLASS = MS/M400,MEDIA = BXN ;
```

FILDESC

– Function :

To list the label and usage information for a disk, tape or cassette file whose name is known. When the external-file-name is unknown, $VOLCONTS has to be used.

– Statement form :

```
FILDESC INFILE = (file-description)
[, PRTFILE = (output-file-description)]
[, PRTDEF = (define-parameter)]
[, PRTOUT = (sysout-parameters)]
[, SHORT]
[, STEPOPT = (step-parameters)];
```

– Statement description :

This utility lists the file attributes and volume information contained in the labels of a disk or, tape or cassette file. The following information is provided in the utility report :

file name

volume name

volume sequence number

generating system
creation date

expiration date

file format family (UFAS or BFAS)

file organization

unit of allocation (disk only)

record format (RECFORM)

block length (BLKSIZE)

record length (RECSIZE)

for disk files	:	number of extents start and end address of each extent possibility of deleted records
for BFAS indexed sequential files	:	presence of master indexes presence of independent overflow presence of cylinder overflow key length and key position number of index levels number of deleted records number of active records in prime data area number of references to overflow records number of full cylinder overflow areas number of available tracks in independent overflow number of cylinder overflow tracks
for libraries	:	a list of the subfiles with creation and last modification dates and the percentage of used space.

This information is listed for the first volume. For subsequent volumes only extent information is printed and volume sequence numbers checked.

For UFAS disk files, additional information is provided.

The file status and status value information is given for software maintenance purposes only. UFAS will automatically salvage any unstable file next time it is opened in update mode.

For UFAS Sequential Relative :

file status (normal or unstable)
data control interval format
record format (RECFORM)
data control interval size (CISIZE)
record size (RECSIZE)
number of allocated control intervals
first relative track address
number of ci's per track
number of formatted control intervals

For UFAS Indexed, in addition to the above details, the following information is listed:

Data control area size (CASIZE)
key position from label
key position given in prealloc (KEYLOC)
key length (KEYSIZE)
control interval free space (CIFSP)
control area free space (CAFSP)
index control interval size
number of index levels
control interval number of root index

For the lowest level index address space:

number of allocated control intervals
first relative track address
number of ci's per track
number of formatted control intervals

This information is also listed for the higher level index address space.

Additionally, the utility provides:

number of active data control intervals
number of control interval splittings
number of control area splittings

For information on the use of the $FILDESC with UFAS I-D-S/II files see the publication *I-D-S/II User Guide.*

If the SHORT option is used, only the contents of the FL1 and FL3 disk file labels will be displayed. If it is not given, the FL2 label information is displayed, and the file can be logically opened to give further information.

When the file to be described is cataloged, the catalog entry should be accessible.

When the file is multivolume, the volume mounting requirements are those needed for a correct open of the file.

For BFAS Indexed Sequential files unless SHORT is used, additional information on file usage is displayed. This information is :

- For each prime track data :
  - . The greatest key values.
  - . The greatest key values in overflow.

  Note that the printed key value is limited to 20 characters. Leftmost characters beyond the first 20 are not printed.
  - . The number of active data records.
  - . The number of deleted records.
  - . The number of associated active records in cylinder overflow.
  - . The number of associated deleted records in cylinder overflow.
  - . The number of associated active records in general overflow.
  - . The number of associated deleted records in general overflow.
- For each cylinder, the number of available record locations and "percentage free", in the cylinder overflow.
- For the complete file, the number of available record location and "percentage free", in the general overflow.

- Statement parameters :

INFILE	This keyword introduces a parameter group which describes the file to be processed. It is mandatory, and the parameter group is described in Section 4. SHARE=DIR must be used with libraries to obtain the list of subfiles.
PRTFILE PRTDEF PRTOUT	These parameter describe the output file and how it is processed
SHORT	This optional keyword reduces the amount of information to be displayed about the given file. The absence of SHORT produces an extended description of the file and statistical information on its usage, when available. SHORT causes a less detailed description to be produced, and no statistical information is given.
STEPOPT	This parameter describes the processing options for the utilities

Example :

```
FILDESC INFILE  =  (P.XYR,DEVCLASS = MS/M400, MEDIA = (BD401,
                   BD402)) ;
```

a file description of P.XYR, a two-volume file, is requested.

- Step Completion Conditions :

The user should note that $FILDESC uses a temporary work file to store and sort the list of subfiles when the file to be described is a library.
This temporary work file is named H_SUBFLS, and is a UFAS indexed sequential file It is accessed using the name H_SUBFLS as the internal-file-name. This file will be referred to by name in some of the error messages.

If the file to be described is a multivolume file, the volume mounting depends upon the file organization requirements. For BFAS files, MOUNT = 1 can be used. For libraries and UFAS files, MOUNT should preferably not be used to avoid the risk of abnormal file openings.

The majority of errors that are encountered while accessing the file to open it are considered fatal, and result in a SEV4 or SEV3 completion code.

Errors which are encountered while editing access-method-dependent labels, or while editing the list of subfiles of a library, are considered to be of low severity, and result in warning error messages and a SEV1 completion code.

The use of the REPEAT option is possible, but because the execution time for $FILDESC is short, it is of little interest. Because $FILDESC does not modify much of the system behaviour, the job step following $FILDESC can usually execute regardless of the termination status of $FILDESC.
Thus it is recommended that the REPEAT option not be used, as its use could lead to an abort/restart cycle which would involve the operator.

For further details, see the description of the $VOLCONTS step completion conditions.

FILDUPLI

– Function :

To copy the contents of a source file into another of identical type. Decks of cards can be duplicated.

– Statement form :

```
FILDUPLI INFILE    = (file-description)
        [,INDEF    = (processing-options)]
         ,OUTFILE  = (file-description)
        [,OUTDEF   = (processing-options)]
        [,TAPEND   = nn]
        [,STEPOPT  = (step-parameters)];
```

The processing-options are the only define-parameters which have any meaning with $FILDUPLI native standard files. They are :

```
[,NBBUF  = {1|2}] [,{BSN|NBSN}] [BPB] [,ERRROPT = {SKIP|RETCODE}]
[,WRCHECK] [,CKPTLIM = {NUMBER-OF-RECORDS|NO|EOV}]
```

– Statement description :

$FILDUPLI can be used to physically duplicate files on magnetic tapes, cassettes and disks. It is the fastest way of performing file duplication.

With magnetic tape files and cassette, the duplication is done block by block or by groups of blocks if BPB is used. With disk files, the duplication is done track by track, which means that the input and output files must be on compatible volumes (that is, volumes with the same track length). Some of the features of $FILDUPLI are :

- buffer sharing in input and output
- the choice of single or double buffering (NBBUF)
- read after write verification (WRCHECK)
- the processing of read errors (ERROPT)
- the support of multi-extent mono or multivolume files
- the support of dynamic mounting of successive volumes of a multivolume file
- the support of checkpoint-restart (CKPTLIM)

Physical duplication is applicable to all disk files of all label types. In particular, it is applicable to LABEL = NONE files and to libraries, and $FILDUPLI is probably the quickest and easiest way of duplicating library files.

The output file, if it is a disk file, must have been preallocated before duplication, and must possess the same characteristics as the input file.

$FILDUPLI has control over the definition of files, and FILEORG, FILEFORM, BLKSIZE, RECSIZE, DATACODE, DATAFORM, CISIZE, CASIZE, KEYLOC, KEYSIZE, etc., must have the same values in the input and output files.

If the files are LABEL = NONE or non-standard files, no controls are applied. Their duplication is thus under the control of the user, and he must ensure the consistency and availability of the output file after duplication. This mainly concerns non-standard disk files, which includes some GCOS system files.

$FILDUPLI does not guarantee the physical duplication of non-relocatable files except BFAS Indexed Sequential files, which are logically duplicated. The user is responsible for using $FILDUPLI with non relocatable files (files having disk addresses stored as part of the data records).

If the output file is smaller than the input file, the duplication will not be successfully performed; if will be interrupted when the space deficiency is discovered. This is also the case if INCRSIZE is available on the output file. The INCRSIZE feature is not supported by $FILDUPLI.

If the output file is bigger than the input file, the duplication will occur. The additional space on the output file will not be available for further file extension in the case of BFAS Indexed Sequential and Direct files, and UFAS Relative files. For these files a warning message will be produced. For all other native files, the extra space is available for further file extension, but will not be reported as available free space by $FILDESC until an extension of the file has occurred under the access method in order to update the file control structures.

For a library file, the extra space left free in OUTFILE after file duplicaton will be used provided INFILE was preallocated with a large enough value for MAXSIZE.

To duplicate a deck of cards, use NBBUF = 2 in both INDEF and OUTDEF parameter groups. See example below. The deck used for duplication purposes can contain checkpoint cards however they are not duplicated and therefore the REPEAT option of the checkpoint facility should not be used. The deck can contain hollerith, binary, control (see JCL User Guide) and $JOB cards all of which will be duplicated.

To punch cards via a SYSOUT file use $CREATE with the SYSOUT option.

Note :

Wherever $FILDUPLI is not applicable because of discrepancies between the input and output files, duplication can be performed using $CREATE.

— Parameter description

INFILE	these parameter groups describe the input file.
INDEF	LABEL = NSTD is not supported
OUTFILE	these parameter groups describe the output file
OUTDEF	LABEL = NSTD is not supported
TAPEND	this option is only used when FILEFORM = NSTD. The processing will stop when nn tape marks have been reached in the two files.

Examples :

— Duplication of a cataloged file:

```
FILDUPLI INFILE  = (OLD. FILE, FILESTAT  =  CAT , MOUNT  =  1).
         OUTFILE = (NEW. FILE, FILESTAT  =  UNCAT, MOUNT  =  1);
```

— Duplication of a card deck :

```
FILDUPLI INFILE  = (CARDIN, DEVCLASS = CD/R/C80,
                    MEDIA = READER),

         INDEF   = (BLKSIZE = 160, RECSIZE = 160, NBBUF = 2,
                    READER = (BINARY, NJOB, NCOMMAND)),
```

```
        OUTFILE  =  (CARDOUT, DEVCLASS = CD/P/C80,
                     MEDIA = PUNCH),
        OUTDEF   =  (BLKSIZE = 160, RECSIZE = 160,
                     NBBUF 2, PUNCH = (BINARY));
```

Note that in the above example INDEF and OUTDEF options are mandatory.

— Duplication of a tape file, using the parameters that improve the execution speed of the utility :

```
FILDUPLI INFILE  =  (FIRST, DEVCLASS = MT/T9/D1600,
                     MEDIA = 1003),
          INDEF  =  (BPB = 3, NBBUF = 2),
        OUTFILE  =  (SECOND, DEVCLASS = MT/T9/D1600,
                     MEDIA = 1004),
        OUTDEF   =  (BPB = 3, NBBUF = 2);
```

— Duplication of a non-standard tape with end-of-processing after the 4th tape mark:

```
FILDUPLI INFILE  =  (FIRST, DEVCLASS = MT/T9/D1600,
                     LABEL = NONE, MEDIA = 1005),
          INDEF  =  (BLKSIZE = 2000, RECSIZE = 2000, RECFORM = U,
                     FILEFORM = NSTD),
        OUTFILE  =  (SECOND, DEVCLASS = MT/T9/D1600,
                     MEDIA = 1006, LABEL = NONE),
                     OUTDEF = (FILEFORM = NSTD),
         TAPEND  =  4 ;
```

— Step Completion conditions :

The status of the OUTFILE file at the end of $FILDUPLI step execution is as follows:

If the OUTFILE was not big enough, the step has aborted on a DATALIM condition, and its termination status is SEV3. The file is not in a consistent state.

If the OUTFILE was large enough, and no fatal malfunctions occurred, the step has completed its run normally, and the OUTFILE labels have been modified as follows :

— If OUTFILE was a preallocated disk file, its FL1 label has been updated (starting at field number 7, up to field number 25) by copying the corresponding fields of the INFILE FL1 label; the secondary space characteristics and the file indicator fields were not copied.

— If INFILE had a FL2 label or an extended label, they have been entirely copied, in order to replace the corresponding labels of OUTFILE.

— If OUTFILE was a tape file, its HDR2 label is built from the contents of the INFILE HDR2 label.

Any discrepancy between INFILE and OUTFILE characteristics will result in a SEV3 step completion status to prevent the restarting of the step if the REPEAT option was used.

Any fatal I/O error on INFILE will result in a SEV4 step completion status.

Any fatal I/O error on OUTFILE, when it is a disk file, will cause the switching of the addressed track onto an alternate track. Warning messages will be sent to the JOR, and a SEV1 step completion status will result.

Note that CKPTLIM and REPEAT options should not be used together when the input file or output file is a tape or cassette file whose fileform is NSTD. Duplication of non-standard tape or cassette files should be handled by $CREATE.

During the physical duplication of a disk file, checkpoints will not be taken unless they are specifically requested by the user via the CKPTLIM parameter. Furthermore, do not attempt to take checkpoints during duplication of a deck of cards because the checkpoints will not be supported by the card punch access method.

FILMODIF

– Function :

To change the name or expiration date of a disk file

– Statement form :

```
FILMODIF INFILE = (file-description)
        [, NAME =  external-file-name]
[,EXPDATE   =  { ddd       }]
               { yy/ddd    }
               { yy/mm/dd  }
[,FORCE]
[,STEPOPT   =     (step-parameters)] ;
```

– Statement description :

This utility allows you to change the recorded name or expiration date of a disk file. The file may be multivolume, in which case, the names of all the volumes must be supplied in the file description. In addition, all the volumes must be mounted. If the file is cataloged, the new name (if the name is to be changed) must be cataloged before the utility is used. If the FORCE option is used, a cataloged file will be renamed regardless of the catalog. We recommend that FORCE be used only for file recovery.

Before renaming a file, the utility checks that no file with the name given in NAME is present. This ensures that duplicate file names may not occur. The utility will abort abnormally if there is an attempt to place a duplicate file name on a volume, and will attempt to restore the already processed volumes to their original state.

This utility cannot be used with temporary files.

– Parameter description :

INFILE	Describes file whose name or expiration date is to be changed
NAME	This optional parameter gives the new name by which the file will be known after the utility has been executed.
FORCE	This optional keyword allows a cataloged file to be renamed without the new name being placed in a catalog. It should be used for file recovery purposes only.
EXPDATE	This optional key word introduces the new expiry date to be assigned to the file.

– Step Completion Conditions

If an abnormal condition occurs, the utility tries to undo what it has done, and then sets SEV3, which is a fatal status, but does not allow the system to repeat the step.

If the file modification was not done on all the volumes (for example, when a system crash occurs during renaming), the user can use the FIRSTVOL parameter of the file-description to recover that situation in a separate job.

The FORCE option can be used if the corresponding catalog entry does not match the file state; this causes the renaming of the cataloged file which is done without updating the catalog.

Examples :

```
FILMODIF INFILE=(XQ.DAT),NAME = XQ.DAT1 ;
```

In this example, the cataloged file XQ.DAT is renamed to XQ.DAT1.
The name XQ.DAT1 must have been made known to the catalog by a statement of the form CATALOG XQ.DAT1; note that the name is already cataloged as it is found for the file XQ.DAT.

```
FILMODIF INFILE=(XQ.DAT1),EXPDATE = 200;
```

In this example, the previously renamed file XQ.DAT1 is given a new expiry date of 200 days.

FILPRINT

– Function :

To print the physical blocks of a disk or tape file.

– Statement form :

```
FILPRINT    INFILE = (file-description)
          [,PRTFILE = (file-description)]
          [,PRTDEF  = (define-parameters)]
          [,PRTOUT  = (sysout-parameters)]
          [,TAPEND  = nnn]
          [,BUFFER = nnnnn]

          [,FORMAT = { BOTH  } ] [,SKIP = value]
                     { ALPHA }
                     { HEX   }

          , { ALL                                                          }
            { ITEM= {(nnnnn  [,nnnnn] . . .)                               }
            {       {(ttttt/rr  [,ttttt/rr] . . .)                         }
            { PART= {( (nnnnn,nnnnn) [, (nnnnn,nnnnn)] . . .)              }
            {       {( (ttttt/rr,ttttt/rr) [, (ttttt/rr,ttttt/rr)] . . .)  }
          [,STEPOPT = (step-parameters)] ;
```

– Statement description :

This utility will print :

– The entire file;
or
– Specified physical blocks from the file ;
or
– Specified ranges of physical blocks from the file.

The printed output of each block may be character, hexadecimal or both.

If the input file is on tape then it may be in standard labelled form (LABEL = NATIVE) or it may consist of a series of data blocks followed by a tape mark (LABEL = NONE) or it may consist of a non-standard mix of data blocks and tape marks (LABEL = NSTD). The labeling is specified in the file-description.

If LABEL = NSTD then the user may specify the number of the tape mark at which printing is to stop.

– Statement Parameters :

INFILE This parameter introduces a parameter group which describes the file to be printed.

PRTFILE
PRTDEF
PRTOUT These parameters describe the output file and how it is processed

TAPEND This parameter is only valid when LABEL = NSTD, when it should be present. The TAPEND parameter indicates when printing is to terminate.

If TAPEND = 2 the printing will finish when the second tape mark is encountered.

If TAPEND is not coded then reading terminates when the "no-record-found" condition is encountered.

BUFFER This parameter is optional for disk files but required for tape files. It specifies the number of bytes that will be printed from each block. If all the bytes of every block (tape) are to be displayed, the maximum block size must be coded.

The maximum value is 30000.

If the parameter is omitted (for a disk file) then all the bytes of each physical record will be printed.

FORMAT This parameter specifies the printing format : character (ALPHA, hexadecimal (HEX) or BOTH(default).

SKIP is a three digit decimal number which specifies how many unrecoverable read errors on the input file will be ignored.

ALL If this parameter is coded then all the physical blocks of the file are printed.

ITEM If this parameter is coded then only the blocks whose numbers are specified are printed.

For a tape the block numbers are 1,2 . . .

For a disk, the physical block number is made up of a relative address of the form :

track-number/record-within-track

Where the first data block in the file is :

0/1

If there are, say, 3 physical records on each track then the sequence is :

0/1, 0/2, 0/3, 1/1, 1/2, 1/3, 2/1 . . .

Up to 20 physical blocks can be identified in the ITEM List.

PART If this parameter is coded then up to 10 pairs of block numbers (relative addresses) may be specified. Each pair designates a range of blocks to be printed.

The range is inclusive, if

PART = ((1/1, 3/12))

the blocks 1/1, 1/2 . . . 3/11, 3/12 will be displayed.

STEPOPT This parameter describes the processing options for the utility.

Examples :

```
FILPRINT INFILE = (PX.CMS,DEVCLASS = MT/T9, MEDIA = PCT45),
                  BUFFER = 800, PART = ((1,200)) ;
```

A file PX.CMS on a standard labelled volume PCT45 is printed. The first 200 blocks are displayed in both alphanumeric and hexadecimal.

```
FILPRINT INFILE = (DLP.CMIP, DEVCLASS = MS/M400, MEDIA = DB14),
                  ITEM = (4/5, 4/6, 5/5,5/8) ;
```

Records 5 and 6 from track 4, and 5 and 8 from track 5 are printed from the disk file DLP.CMIP on volume DB14.

— Step Completion Conditions :

If the execution was successful, the only message on the JOR will be

IFN : EFN, that is, infile : file-name

If the execution was totally unsuccessful, a message is found on the JOR, indicating the error; it is usually an error in the JCL (SEV3). For example :

SKIP : illegal value

— this means that skip is probably not numeric

PART or ITEM illegal values

— Because of the large number of possible errors, only a general message is printed. Usually, the position of the parentheses is wrong, or tape format (blocks) has been used for a disk file.

In any of the cases, the JCL just needs to be corrected and the job rerun.

If the execution was partially successful, a message of severity 3 or severity 1 will be printed on the JOR. For example :

SEV1	an I/O error occurred when reading a track; a message will be printed in the sysout. As long a the number of I/O errors does not exceed the values of SKIP, the program continues.
SEV4	number of I/O errors exceeded the value of SKIP— If you wish to print more than is already printed, either change the value of SKIP, or change the range to be printed.
SEV3	printing is stopped because the track is out of the range of the file.
SEV1	the number of records to be printed on a given track is greater than actual number of records on that track.

FILREST

– Function :

To restore the contents of a disk file from a sequential tape file.

– Statement form :

```
FILREST INFILE = (file-description)
     ,OUTFILE  = (file-description)
   [ ,NAME     = external-file-name]
   [ ,SKIP     = nnn]
   [ ,NBBUF    = {1/2} ]
   [ ,STEPOPT  = (step-parameters)] ;
```

– Statement description :

This utility performs the reverse function of $FILSAVE. It restores, to a previously allocated disk file, the disk file contents previously written to tape by means of $FILSAVE. The file must be restored onto the same type of disk as that from which it was saved.

The choice of the disk file image, contained within the tape file, is decided through the keyword NAME. If NAME is not given, then if the restore is onto a disk file named AAB the utility will search the tape to find a disk file image whose name is also AAB; and this file image will be restored onto the disk.

If the disk file being restored has BFAS Indexed Sequential organization then the disk space into which it is restored must be on the same disk addresses from which it was saved. Such a file is not relocatable.

This restriction does not apply to other UFAS or BFAS or library files.

Irrespective of FILESTAT status, the original file when restored to the output disk can be either cataloged or uncataloged. If the SIZE of the output disk file is not large enough to hold the restored image, FILREST will abort even though the file may have been preallocated within the INCRSIZE attribute.

Volume mounting is user specified for the input tape file, and for the output disk file.

$FILSAVE and $FILREST are organized so that a file named A can be restored into another file named B providing A and B have identical organizations and definition parameters and reside on disks having identical track length. For details of inter-release compatability, see Appendix A.

– Statement parameters:

The meaning of the parameters for $FILREST is exactly the same as for $FILSAVE.

OUTFILE — This is followed by the parameters describing the disk file that is to be restored. The volume names, in a multi-volume situation, must be cited in the same order as in other allocation or assignment statements for the file.

Outfile can be a temporary file.

INFILE — This parameter group describes the tape file which contains the disk file image to be restored. In the case when the tape file is multi-volume, then the volume names must be cited in the same order as on the associated $FILSAVE statement.

NAME This optional parameter specifies the name of the disk file image recorded on the tape file by $FILSAVE.

SKIP This optional parameter specifies the number of unrecoverable READ errors that are to be ignored on INFILE.

NBBUF This optional parameter specifies the number of buffers for the restore file.

Examples :

```
FILREST  OUTFILE  =  (INV.TOD), INFILE = (DAL.Y,
                      DEVCLASS=MT/T9/D800,MEDIA=ST442A) ;
```

The disk file INV.TOD is restored onto its RESIDENT disk space. The restore will be from the tape file DAL.Y on tape volume ST442A.

```
FILREST  OUTFILE  =  (PAY.SET, DEVCLASS = MS/M400,
                      MEDIA = (OLA, OLB)),
         INFILE   =  (DAL.Y, DEVCLASS = MT/T9/D800,
                      MEDIA =(ST442A, ST442B) ) ;
```

The disk file PAY.SET, held on MSU0400 type volumes OLA and OLB, is restored from the tape file DAL.Y, volumes ST442A and ST442B.

```
FILREST  OUTFILE  =  (MYFILE, DEVCLASS = MS/M400,
                      MEDIA  =  MY100),
         INFILE   =  (SAVE, DEVCLASS=  MT/T9/D1600,
                      MEDIA  =  TAP1),
         NAME     =  OLDIMAGE ;
```

The disk file image OLDIMAGE is restored from the tape file SAVE into the disk file MYFILE.

The three examples given here correspond to the three examples given for $FILSAVE.

– Step Completion Conditions :

Checkpoint/Restart :

when the REPEAT option is used, checkpoints are taken at the switching of every tape volume to the next tape volume. If a Restart occurs, $FILREST asks for the first volume of the (multivolume) disk file to be mounted; it then goes to the volume that was the current one when the step aborted.

If $FILREST aborts with the message DU 09.50, NOT ENOUGH SPACE TO RESTORE FILE, OUTFILE is modified and left in its current state, which may be inconsistent.

The file attributes of the disk file (OUTFILE)are partly updated with the corresponding attributes of the restored file image. For more information on this subject, see the step completion conditions for $FILDUPLI.

Note that if the SKIP option is used, if some of the blocks from the input tape file could not be read, the step terminates with a SEV1 completion code, and the corresponding tracks from OUTFILE are left untouched. If the same thing happens when SKIP is not used, the step aborts with a SEV4 completion code.

FILSAVE

– Function :

To save the entire contents of a disk file onto a sequential tape file.

– Statement form :

```
FILSAVE  INFILE   = (file-description)
        ,OUTFILE  = (file-description)
       [,SAVEMODE = { CREATE }]
                    { APPEND }
        [,NAME    = external-file-name]
        [,SKIP    = nnn]
        [,NBBUF   = { 1 }  ]
                    { 2 }
        [,STEPOPT = (step-parameters)] ;
```

– Statement description :

This utility stores the contents of a disk file as a series of data blocks (one block per original track) in a magnetic tape file. This copy of the disk file may not be processed on the tape but may be retrieved using the utility $FILREST. Several disk files may be saved onto one $FILSAVE tape file, and both the disk file and the tape file may be multi-volume. The volume mounting is user specified for INFILE and OUTFILE.

The disk file saved by this utility may have any organization.

The file to be saved may be temporary if it is relocatable (this is not the case for BFAS indexed Sequential files). The output file may or may not be preallocated. If it is preallocated with definition parameters which do not correspond to those required, a run-time error message is sent and the execution stops.

For information on inter-release compatibility, see Appendix A.

For each save file the utility stores :

– an information block (which includes the name of the file)

– a block containing the labels of the saved file

– data blocks, one per allocated track of the disk file.

– Statement parameters :

INFILE — This keyword is followed by a group of parameters describing the disk file that is to be saved. If DEVCLASS is given then only disk device classes are permitted. All volume names given must be in the same order as when the file was created. INFILE may be a temporary file.

OUTFILE — This keyword is followed be a parameter group describing the file into which the save is performed.

The device class must be a magnetic tape.

If OUTFILE is a catalogued file, it must be preallocated with BLKSIZE = 13200, RECSIZE = 13196, and RECFORM = V.

SAVEMODE This parameter specifies whether the tape file to be used already contains other saved disk files images, and that the current operation is to add another, or whether the operation is to place a first disk file on the tape file.

If SAVEMODE = CREATE, the default, then the latter situation is assumed.

If SAVEMODE = APPEND, then the utility will not create a new tape file, but instead will ensure that an existing valid file of the name given (in the OUTFILE parameter group) is present. The program will then skip down to the current end of the file and start recording the new disk file image after the last block of data : thus the operation is one of appending to the tape file another disk file image.

If SAVEMODE = APPEND then the parameter EXPDATE must not be specified in the parameter group describing the tape file. Its presence would be inconsistent and meaningless.

NAME This optional parameter specifies the name to be given to the disk file image to be saved into the file described in OUTFILE.

SKIP This optional parameter indicates that a number of unreadable tracks are to be ignored on the input file. If a number is specified the utility will abort when this number of I/O failures has occurred.

NBBUF This optional parameter gives the number of buffers to be used for the save of the disk file. The buffer size is set to the maximum length of a data block on the disk concerned.

Programing considerations

In using $FILSAVE the following points should be borne in mind :

- No attempt should be made to store two disk file images of the same name onto a tape file using this utility. Although both save operations will be carried out successfully, it will not be possible to instruct $FILREST to restore the second, later file images from the tape file.
- Care should be taken when SAVEMODE = APPEND. The only check performed on the output file in this case is to ensure that a valid tape file of given name is present. The utility cannot check that the file contents consists of disk file images. Thus by quoting a normal data file it is possible for the utility to append to it a disk file contents. It will be impossible to retrieve the appended file using $FILREST.

Examples

```
FILSAVE  INFILE = INV.TOD, OUTFILE = (DAL.Y,
                  DEVCLASS = MT/T9/D800,
                  MEDIA= ST442A, EXPDATE = 78/9/28) ;
```

In this example the disk file named INV.TOD, which is RESIDENT, is saved into a new tape file named DAL.Y on tape volume ST442A, and the file DAL.Y will be given an expiration date of the twenty eighth (28) of September (9), 1978.

```
FILSAVE  INFILE = (PAY.SET, DEVCLASS   = MS/M400,
           MEDIA = (OLA, OLB) ),
        OUTFILE = (DAL.Y, DEVCLASS   = MT/T9/D800,
           MEDIA = (ST442A,
                    ST442B ) ), SAVEMODE = APPEND;
```

The file named PAY.SET, residing on disk volumes OLA and OLB, is saved into an existing tape "save" file named DAL.Y. Note that the user expects the file DAL.Y already to be multi-volume or to require a second volume, ST442B (by comparison with the first example above) onto which the utility can save the contents of PAY. SET.

```
FILSAVE  INFILE = (MYFILE, DEVCLASS = MS/M400,
                   MEDIA = MY100),
        OUTFILE = (SAVE, DEVCLASS   = MT/T9/D1600,
                   MEDIA = TAP1),
          NAME  = OLDIMAGE ;
```

The disk image OLDIMAGE is saved into the tape file SAVE from the disk file MYFILE.

Note that these examples correspond with the examples for $FILREST.

– Step Completion Conditions :

Checkpoint/Restart :

When REPEAT is used, checkpoints are taken at the switching of every tape volume to the next one. If there is a Restart, $FILSAVE asks for the first volume of the multivolume disk file to be mounted, and then goes to the volume which was the current one when the step aborted.

If $FILSAVE aborts with the message DU 01.51 printed in the JOR, the OUTFILE tape has invalid characteristics.

If the option SKIP was used and a track is encountered that gives an unrecoverable read error, then $FILSAVE does not abort unless the SKIP count is already exhausted; instead, it records the address of this unreadable track in the output file. This enables $FILREST to warn the user that the restored file image was damaged, and that the restored file may be in an inconsistent state. The entire contents of the unreadable track are lost.

PREALLOC

– Function :

To allocate space and declare attributes for a UFAS or BFAS disk file ; also to extend an existing BFAS sequential disk file, or UFAS sequential or indexed file. Cataloged tape filescan also be preallocated or extended.

– Statement form 1 for disk files :

```
$PREALLOC external-file-name  [,EXPDATE = { ddd | yy/ddd | yy/mm/dd }]
        [,UNIT = { CI | CYL | TRACK | RECORD }] [,MAXEXT = value ]
      ,{ UFAS  = ( { SEQ = (sequential attributes)
                   | RELATIVE = (relative attributes)
                   | INDEXED = (indexed attributes) } )
       | BFAS  = ( { SEQ = (sequential attributes)
                   | DIRECT = (direct attributes)
                   | INDEXED = (indexed sequential attr.) } )
       | EXTEND }
    ,{ RESIDENT  = (SIZE = value)
     | DEVCLASS  = device-class
     ,{ GLOBAL  = (MEDIA = (volume-name
                            [,volume-name] . . .) ,SIZE = value )
      | SPLIT   = ( (volume-name, SIZE = value   [,CYL = address] )
                    [,(volume-name, SIZE = value   [,CYL = address] )] ...)
      | SIZE    = value }
      [,INCRSIZE=value ]
     ,FILESTAT = { TEMPRY
                 | CAT [, CATALOG = catalog-number]
                 | UNCAT }
     [,STEPOPT = (step-parameters)] ;
```

–Statement Form 2 (For Catalogued Magnetic Tape files) :

```
$PREALLOC external-file-name  [,EXPDATE = { ddd | yy/ddd | yy/mm/dd }]
      ,{ { BFAS | UFAS | ANSI } = (SEQ options for BFAS files [,NBSN] )
       | EXTEND }
        ,DEVCLASS = device-class-name
        ,GLOBAL   = (MEDIA = (volume-name [, volume-name] ...))
        ,FILESTAT = CAT
          [,MOUNT = n]
       [, STEPOPT = (step-parameters)] ;
```

The format of UFAS file attributes is :

```
                  ( SEQ = (CISIZE = value,RECSIZE = value [,RECFORM = {F|V}] )
                  |
                  | RELATIVE = (CISIZE = value ,RECSIZE = value
                  |             [,RECFORM = {F|V} ])
UFAS = (          |
                  | INDEXED  = (CISIZE = value ,RECSIZE = value
                  |             [,RECFORM = {F|V} ]
                  |             ,KEYLOC = value ,KEYSIZE = value
                  |             [,CASIZE= value] [,CIFSP = value] [,CAFSP = value])
                  |
                  | [,SECIDX  = ((KEYLOC = value, KEYSIZE = value [,DUPREC] )
                  |              [, (KEYLOC = value, KEYSIZE = value [,DUPREC])]...)] )
                  )
```

The format of BFAS file attributes is :

```
                  ( SEQ = (BLKSIZE = value  ,RECSIZE = value
                  |        [,RECFORM = {FB|F|U|V|VB}] [NDLREC] [,FIXTRACK] [,COMPACT])
                  |
                  | DIRECT = (BLKSIZE = value  , RECSIZE = value
                  |           [,RECFORM = {FB|F}] [,NDLREC] )
BFAS = (          |
                  | INDEXED = (BLKSIZE = value ,RECSIZE = value
                  |            [,RECFORM = {FB|F}] [,NDLREC][,MASTER]
                  |            ,KEYSIZE = value  ,KEYLOC = value
                  |            [,CYLOV = value] [,GENOV = value]
                  |            [,IDXSIZE = {1|index-size}] [,TRACKFAC = {1|nn}] )
                  )
```

– Statement description :

The $PREALLOC utility reserves space for a disk file and creates the necessary file labels which are set up to contain details of the file organization.

File pre-formatting occurs for BFAS Indexed Sequential files and for UFAS files.

The space reservation is done at the unit of a record, a control interval, a disk track or disk cylinder. Space is allocated to the file as a series of one or more extents. An extent is a group of one or more contiguous allocation units (tracks or cylinders). On any one volume a file may have up to 16 extents. However, for efficiency purposes, there is the facility to restrict the number of extents to one per volume.

In the case where a multi-volume file is allocated, it is possible to specify the amount of space that is to be taken on each volume and also the position (cylinder address) at which the allocation is to take place. In the latter case only one extent may be allocated per volume.

The normal allocation of extents is done as follows. A list of all the free space extents available is inspected. The smallest extent of all those which are greater or equal to the space required is chosen. Thus if the free extents were 20, 23, 25, 60 cylinders and request was for 24 cylinders then the allocation would be made on the 25 cylinders extent leaving free space extents of 1, 20, 23, 60 cylinders. If the space requested is larger than the largest free space extent then the largest extent is allocated and the remaining space still required is chosen firstly by searching for the smallest of the larger free extents or choosing the largest and then searching for space for the remainder. Thus if a request for 86 cylinders was made with the above space list then 60 and 25 extents would be allocated and one cylinder would be used from the 20 cylinders.

The FILESTAT parameter states whether the file to be preallocated is a temporary file, a permanent uncataloged file, or a permanent cataloged file. In the latter case, PREALLOC will automatically complete the catalog entry for the file. Temporary files are made available to the next step which assigns them. $ALLOCATE can also be used to allocate temporary files.

For tape files, the UFAS and BFAS file formats are the same, so you can choose either of the corresponding options. Tape files can also be in ANSI format, and a **cataloged tape file cannot be preallocated on a multifile tape.**

The second part for a $PREALLOC statement describes the UFAS/BFAS organization.

Note that the $PREALLOC form for I-D-S/II file allocation is described separately in the publication *I-D-S/II User Guide.*

The information supplied is loaded into the file label and is used when the file is accessed by program.

For details on file design and space calculation see the UFAS User Guide or BFAS User Guide.

– Statement parameters :

EXTERNAL-FILE-NAME — The external-file-name which must be present. It must conform to the rules for file names given in Section 3. The name chosen must be unique within the set of file names already present on the volume(s) to be used, unless file extension is requested.

EXPDATE — This keyword gives the expiration date for the file. If no expiration date is given then the default assumed is the current date.

UNIT — This keyword specifies the unit of allocation that is to be used by the utility. The choice is between cylinders (CYL), tracks (TRACK), records (RECORD) or control interval (CI)

If a BFAS Indexed Sequential file is being established then the allocation unit must be the cylinder. The choice CI is only valid for UFAS files. The number of CIs is transformed into a number of tracks and allocation is performed in tracks. For further details see the UFAS User Guide.

The choice of RECORD is only available for BFAS files; the given number of records is converted into cylinders, and storage allocation is performed in cylinders.

If this keyword is not given then the default assumed is CYL.

MAXEXT — This optional keyword specifies the maximum number of extents to be selected per volume. The numeric value given may not exceed 16.

If MAXEXT is not explicitly stated, the default value assumed is the maximum of five possible extents per volume; this is increased to 16 if EXTEND is used.
MAXEXT can also be used with SPLIT.

RESIDENT — If this keyword is specified then the allocation will take place on the resident volumes attached to the system.

SIZE

This keyword is followed by a numeric value, up to five digits long giving the size of the file in allocation units.

DEVCLASS — This parameter is coded when non-resident disk space is allocated. It is followed by a standard disk device class identification. If a non-resident file is being allocated, then DEVCLASS must be specified. It must be specified if a cataloged file is being allocated.

GLOBAL — This keyword specifies that allocation of extents is to be performed automatically. That is, in the parameter group that follows only the total size of the file is given. By comparison, the alternative, SPLIT, gives a discrete size parameter for each volume. In the case where only a single volume is being used, GLOBAL and SPLIT reduce to almost exactly the same form though in the case of SPLIT there is still the additional facility for specifying the disk address at which allocation occurs.

MEDIA

This parameter is followed by a list of one or more volume names to be used in the acquisition of space. The order of the list is important. The utility will allocate the maximum number of extents (see MAXEXT above) on the first volume and proceed to the second and so on. The manner in which extents are chosen is described above in "statement description".

In the case of a multi-volume allocation it is possible that the space requirement for the file will be satisfied without all the volumes being needed. If this occurs then the utility will not use the last volumes in any way, and they need not be mentioned in future references to the file.

SIZE

This keyword is followed by a numeric value, up to five digits long, giving the size of the file in allocation units.

SPLIT

This keyword specifies that the allocation is to be performed on a volume by volume basis. It is followed by a parameter group which lists each volume and the amount of space required on each. Optionally the user may specify for each volume the disk address at which allocation is to occur. It is possible to give such addresses for some of the volumes and not for others. The SPLIT feature may not be used with BFAS Indexed Sequential files, with BFAS files when UNIT = RECORD or for UFAS files when UNIT = C1. Note that cylinder number zero must not be addressed by SPLIT, even if the VTOC has been located elsewhere.

Volume-name

The volume names given must be in the desired order of usage. The first logical records will be recorded on the first volume and the last logical records on the last.

SIZE

This keyword, supplied with each volume name, gives the amount of space required on the volume. The numeric value supplied, up to five digits long, is in allocation units.

CYL

This optional keyword allows the user to specify the cylinder address within volume at which allocation is to occur. If this is specified then there is no automatic extent selection. Instead all the amount of SIZE, for the volume, is allocated starting at the supplied cylinder address. The user must ensure that there is a sufficiently large contiguous free space at the supplied address : otherwise the allocation will fail and terminate abnormally.

The numeric values giving the cylinder address must not be more than three digits long. Zero must not be used as a cylinder address. If the unit of allocation is track then a partically used cylinder can be given as the start point for allocation.

If CYL is not specified for a volume then space is reserved by automatic extent selection as in " statement description" above.

EXTEND

This optional parameter request that the current storage space of an existing file is extended. It may be given for BFAS Sequential files, UFAS Indexed Sequential files, and UFAS Sequential files. If the file is cataloged, the amount of space added to the file may be specified in the SIZE parameter without specifying any media. If the file is uncataloged, the space added is specified in the SIZE parameter of RESIDENT, GLOBAL, or SPLIT.

This parameter has no effect on the existing contents of file. If the unit of allocation given is not the same as the original unit of allocation, then the utility will re-calculate the size and continue to use the original unit.

For uncataloged, non-resident files, you must specify the last or only volume on which the file is currently held. If the extension is to be multivolume, then subsequant volumes may be specified in the volume list. In all cases, the first volume in the list must be the last volume of the file.

EXTEND can not be used with RESIDENT unless the file which is to be extended is entirely resident on the system disk.

INCRSIZE

This parameter states how much the file is to be automatically incremented by the system each time the file storage space is completely used. INCRSIZE is not available for BFAS Indexed or Direct files. The size of the increment of space is expressed in cylinders when UNIT = CYL or in tracks when UNIT does not equal CYL.

PREALLOC

FILESTAT — This mandatory parameter gives the status of the file to be preallocated. It may be a permanent uncataloged file (UNCAT), a permanent cataloged file (CAT), or a temporary file (TEMPRY). If it is a cataloged file, the number of the catalog in the $ATTACH statement may be given (CATALOG =). $PREALLOC will return the information required to complete the catalog entry for the file to the catalog containing the file attributes. Note that the catalog entry for the file must have been created previously by $CATALOG statements.

If the file is temporary, then it will only be available to the next job step which assigns it, unless END = PASS is specified in the $ASSIGN statement.

UFAS — This keyword is followed by a parameter group containing the file organization attributes. The keyword signifies that the file described is UFAS (Unified File Access System) in type. The keyword UFAS must be coded.

SEQ — This keyword signifies that the parameters which follow are describing a sequential file.

CISIZE

This keyword specifies the Control Interval size for the file. The value supplied, up to 5 digits long, is in units of bytes. The value given will be rounded up to the next multiple of 256 (unless it is already a multiple). This keyword is required. The specified value includes the CI header, record headers and (indexed files only) record descriptors.

RECSIZE

This keyword supplies the logical record length in bytes. The value supplied may be up to 5 digits long. This keyword must be supplied. The value excludes any UFAS record header. In the case of variable length records the value supplied is the maximum record length.

RECFORM

This keyword specifies the record format to be employed. There are two possible formats allowed :

Fixed : When RECFORM= F every record will have the same length.

Variable : The specification of RECFORM = V marks the file as having variable length records.

The default value of RECFORM is F.

RELATIVE — This keyword signifies that the parameters which follow describe a relative file.

The mandatory keywords are CISIZE and RECSIZE.

As can be seen from the statement form, the attributes of this organization are similiar to those for SEQ.

INDEXED — This keyword describes the organization of an Indexed Sequential file. The keywords CISIZE, RECSIZE and RECFORM are the same as those for SEQ and DIRECT above. All the other parameters are strictly concerned with Indexed file attributes.

KEYSIZE

This keyword is required. It specifies the key length in bytes. The maximum key length is 255 bytes.

KEYLOC

This keyword is required. It specifies the location of the key within the record. The value supplied is the number of the first (left-most) byte of the key field. The first byte of a record is byte 1, the second is byte 2 etc (the record header is excluded).

CASIZE

This keyword specifies the number of Control Intervals (CIs) in a Control Area.

This value will also be used as the number of index entries in an index control interval. This index CI size is rounded up to a multiple of 256. So the users CASIZE is consequently updated, to optimize index management.

If CASIZE is not specified then a default value will be chosen when the file is first opened. This default is the number of index entries that will fit in an index Control Interval of 2048 bytes.

Maximum CASIZE : number of index entries that fit in an index Control Interval of 4096 bytes.

CIFSP

This keyword specifies the free space (as a percentage) to be left in each CI when the file is sequentially loaded (opened in output).

Default value : 0; Maximum value : 100

CAFSP

This keyword specifies the free space (as a percentage) to be left in each CA when the file is sequentially loaded (opened in output).

Default value : 0; Maximum value : 100

SECIDX

This keyword specifies whether or not the file is to have secondary indexes and what keys they are related to : The DUPREC keyword specifies that duplicate keys are allowed.

BFAS — This keyword is followed by a parameter group containing the file organization attributes. The keyword signifies that the file described is BFAS (Basic File Access System) in type. The keyword BFAS must be coded.

SEQ — This keyword signifies that the parameters which follow are describing a sequential file.

BLKSIZE

This keyword specifies the block size for the file. The value supplied, up to 5 digits long, is in units of bytes. This keyword is required.

RECSIZE

This keyword supplies the logical record length in bytes. The value supplied may be up to 5 digits long. This keyword must be supplied.

RECFORM

This keyword specifies the record format to be employed. There are five possible formats allowed :

Fixed Blocked; When RECFORM = FB it is assumed that all records are of equal length and that the block size is a multiple of the record size. Each block of the file, except for the last, will contain the same number of records.

Fixed : When RECFORM = F it is assumed that there will only be one record per block and hence the block size will be the same as the record size. Every record will have the same length.

Undefined: When RECFORM = U the file is designated as having records of undefined length. The maximum record size is taken to be the block size since no attempt is made to pack records into blocks. The RECSIZE should not be specified since it is not meaningful.

Variable : The specification of RECFORM = V marks the file as having variable length records contained within variable length blocks. Each block only contains one record.

Variable Blocked : When RECFORM = VB the file is marked for use with variable length records in variable length blocks. The input/output access methods will place as many variable length records

as will fit into specified maximum block size.

With both variable and variable blocked organizations each record consists of a one word (four bytes) Record-Description-Word, followed by the data area. This RDW is not included in the maximum record length supplied through RECSIZE.

Also with variable organizations each block (whether V or VB) contains a Block-Description-Word (BDW) or four bytes. This BDW is not included in the maximum blocks length as supplied through BLKSIZE.

Thus, the maximum number of bytes taken by a variable block on the disk is four bytes greater than that specified by BLKSIZE.

The default value of RECFORM is FB.

NDLREC

This option concerns the presence or otherwise of deleted (dummy) records in the file. By default GCOS will assume that the file may contain such records. A deleted record is one which contains hexadecimal FF in the first data byte. Such records will be skipped on sequential input of the file. If NDLREC is stated then records of this type will be considered as normal data records. Therefore it is necessary to say NDLREC if the file is likely to contain valid records which have FF as the contents of the first byte.

If this parameter is omitted then it is assumed that the file may contain deleted records, according to the above convention.

FIXTRACK

This option requests that the number of physical records per track for a BFAS sequential disk file whose record format is variable (or blocked variable) be constant. The number of physical records will be derived by the system from the value of RECSIZE. This is achieved by taking into account the physical length of the track for the chosen device class.

COMPACT

This option requests the compaction of consecutive blank characters found in a textrecord of a BFAS sequential file or in a source library member. This option can only be used when the record format is variable or blocked variable.

DIRECT

This keyword signifies that the parameters which follow describe a Direct Access file.

The mandatory keywords are BLKSIZE and RECSIZE.

As can be seen from the statement form, the attributes of this organization are a subset of those for SEQ. The main difference being that there are only two record formats available, F and FB.

The meaning of the parameters for DIRECT are exactly the same as their counterparts in SEQ above.

INDEXED

This keyword describes the organization of an Indexed Sequential file. The keywords BLKSIZE, RECSIZE and RECFORM are the same as those for SEQ and DIRECT above. Only F and FB record formats are allowed. The parameter NDLREC is also permitted for INDEXED and DIRECT files. All the other parameters are strictly concerned with Indexed Sequential file attributes.

Note that the minimum of an indexed sequential file is 2 cylinders : one for the index extent and one for the prime data extent.

MASTER

This signifies that the file is to have a master index. It is optional, and if omitted, then no such index will be maintained for the file.

KEYSIZE

This keyword is required. It specifies the key length in bytes for the file. The maximum key length is 255 bytes. For an MSU0310 the maximum length is restricted to 200 bytes.

KEYLOC

This keyword is required. It specifies the location of the key within the record. The value supplied is the number of the first (left-most) byte of the key field. The first byte of a record is byte 1, the second is byte 2 etc.

CYLOV

This keyword is optional. It allows the user to specify the number of tracks per allocated cylinder to be used as an overflow area. This overflow area is a local one, being associated only with overflow on the same cylinder and should not be confused with the independent overflow area which is specified through the GENOV keyword. This is described below. If CYLOV is not given then no local overflow space is reserved within each cylinder of the file. 18 is the maximum value allowed. If UNIT = RECORD, the number of tracks is calculated; there must be at least one track of data.

GENOV

This keyword is optional. It allows the user to specify the number of cylinders to be used as a general overflow area. These cylinders are always reserved from the last cylinders (in the last extents) of the file. There is no general overflow area reserved if this keyword is omitted. If UNIT = RECORD, the number of cylinders is calculated. There is always at least one.

IDXSIZE

This keyword gives the number of cylinders required for the index area. Default value is 1.

TRACKFAC

This keyword gives the number of tracks of cylinder index per master index entry. Default value is 1.

Example 1

```
PREALLOC   LP.PJM, UNIT = CI, RESIDENT = (SIZE = 600) ,
           UFAS = (SEQ = (CISIZE = 1000, RECSIZE = 190 ) ),
            FILESTAT = UNCAT;
```

This statement allocates a sequential file LP.PJM. By default it will take the creation date to be the expiration date. The space reserved will consist of 600 control intervals, each being 1024 (next 256 multiple above 1000) bytes. The record format by default is fixed and each record will be 190 bytes long.

Example 2

```
PREALLOC   MPTSP.DD,  EXPDATE = 300, UNIT = CYL,
           DEVCLASS = MS/M400,
           SPLIT = ( (D18A, SIZE = 10), (D18B, SIZE = 10 ) ),
           UFAS = (RELATIVE = (CISIZE = 768, RECSIZE = 52 ) ),
           FILESTAT = CAT ;
```

In this example a cataloged Relative file MPTSP.DD is allocated on two volumes, D18A and D18B. Each volume will contain 10 cylinders. The file is split evenly between the two disks, hence reducing head movement in random access. The file has a retention period of 300 days.

Example 3

```
PREALLOC   PC.UIX, UNIT = CI, DEVCLASS = MS/M400,
           GLOBAL = (MEDIA = TNDA, SIZE = 26352),
           UFAS = (INDEXED = (CISIZE = 3072, RECSIZE = 211,
           CASIZE = 60, KEYLOC = 10, KEYSIZE = 21, CAFSP = CIFSP=20)),
              FILESTAT = CAT;
```

In this example a cataloged file, PC.UIX, is allocated on an MSU0400 volume, TNDA. A total of 26352 data control intervals are requested. The space for the header track and index area will be automatically added. The records are fixed-length, 211 bytes, and contain a 21-byte key starting at position 10. The user has requested that each CA contain 60 CIs and that each CI is 3071 bytes long. When the file is opened and leaded sequentially, each CI will be left with 20% free space and each CA will maintain 20% of CI's free (12 CI's). This free space will reduce the possibility of frequent splitting to accommodate later insertions.

Example 4 :

```
ATTACH CATALOG 1   = OWN.CATALOG;
CATALOG XQ, TYPE   = DIR ;
CATALOG XQ.DAT ;
PREALLOC XQ.DAT, BFAS = (SEQ = (BLKSIZE = 2000, RECSIZE = 400 ) )
                 , DEVCLASS = MS/M400
                 , GLOBAL   = (MEDIA = D16, SIZE = 30)
                 , FILESTAT  = CAT ;
```

This example catalogs and allocates a disk file named XQ.DAT. By default, the creation date will be the expiration date, and the allocation unit will be the cylinder. Thus 30 cylinders of space will be reserved on the MSU0400 disk

called D16. XQ.DAT will be sequential in organization and will have data blocks of 2000 bytes, each block consisting of five 400-byte records. The default record format is Fixed Blocked (FB). It is assumed that the file may contain deleted records. For further details of cataloged files, see the Catalog Management Manual.

Example 5 :

```
PREALLOC  XQ.DATR, RESIDENT = (SIZE=30),FILESTAT = UNCAT,
          BFAS = (DIRECT = (BLKSIZE = 2000, RECSIZE = 400 ) ) ;
```

This statement describes a file XQ.DATR which has the same space, block and record attributes as XQ.DAT in the previous example. However the file organization is DIRECT, allowing the file to be processed randomly by the "direct" access method software.

Example 6 :

```
PREALLOC  PQ.DATP, EXPDATE = 55, UNIT = TRACK,
          DEVCLASS = MS/M402, GLOBAL = (MEDIA = DVA,
          SIZE = 438),
          MAXEXT = 3,
          BFAS = (SEQ =(BLKSIZE = 220, RECSIZE = 220,
          RECFORM = F, NDLREC) )
          ,FILESTAT = CAT;
```

This example allocates a cataloged file named PQ.DATP on an MSU0402 disk volume named DVA. The allocation unit is a track and the expiration date of the file is set to 55 days after the allocation date. The size of the file is to be 438 tracks and the maximum number of extents is 3. The blocksize is 220 bytes and since the record format is Fixed, the record size is also 220 bytes. It is a sequential file without any possibility of deleted records.

Example 7 :

```
PREALLOC  PAY.MASTER, EXPDATE = 78/8/01,
          DEVCLASS = MS/M400,
          GLOBAL = (MEDIA = (MQ64A, MP64A, MR64A ),
                    SIZE = 300) ,
          MAXEXT = 1,
          BFAS = (SEQ = (RECFORM = VB, BLKSIZE = 268,
                 RECSIZE = 264) ) ,FILESTAT = CAT;
```

In this example a cataloged multivolume file, PAY.MASTER, is allocated on the MSU0400 disk volumes, MQ64A, MP64A, and MR64A. The allocation unit, by default, is the cylinder. A space of 300 cylinders is requested and only one extent may be allocated on each volume. Thus the allocation algorithm will operate as follows :

1. The free extents on MQ64A will be checked to see if any are greater than 300 cylinders. If any exist then the smallest that is greater (or equal) to 300 will be chosen and the allocation will be completed using only MQ64A and not MP64A or MR64A. Assuming, there is no single extent of 300 cylinders or greater on MQ64A, the largest available will be chosen. The size of this extent is then subtracted from the required size to find the residual size.
2. Using the residual size as the required size a search for space is then performed on the volume MP64A. The smallest free extent which is larger or equal to the required size is chosen, or no such extents exist; then the largest single extent is chosen. Either the required space is completely accommodated or a new residual requirement results. If the latter is the case then step 3 is performed. Otherwise the allocation is complete.
3. Using the new residual size as the required space a search is made for the smallest free extent on MR64A that will accommodate it. If one is found then the utility terminates successfully. Otherwise an error condition, insufficient free space, is signalled.

If the execution was successful, a detailed message will appear on the JOR indicating the media and number of extents allocated. If the catalog has been modified, this will be stated. The message is described in detail later.

If the execution was unsuccessful, the step completion status is almost always SEV3, indicating that no file was allocated, and the file name is not available for use. With JCL errors, as many as possible will be found and indicated before the execution of the load module, but because of the involved relationship between certain variables, some JCL errors will only be detected in the load module. Errors such as duplicate name are only detected after the JCL is considered to be perfect.

When it is impossible to preallocate a file, a detailed list of the media and the reasons why preallocation was impossible will be found in the JOR.

If the file is preallocated but formatting or the writing of additional labels is impossible, $PREALLOC will try to delete any files which it cannot successfully finish processing. If this task is interrupted for any reason, $DEALLOC should be used immediately to finish the job.

If it is impossible to modify the catalog, $PREALLOC will again try to deallocate any space allocated. If this is not successful, $DEALLOC should be called immediately, using the FORCE option if necessary. $PREALLOC cannot be called again for this file. Note that catalog modification takes place at the end of the preallocation process.
The messages produced by $PREALLOC are listed below.

SPACE ALLOCATED FOR THE FILE : filename

TOTAL AMOUNT OF SPACE ALLOCATED : xxx CYLINDERS ON THE TRACKS

FOLLOWING VOLUMES :

VOLUME : volume-name EXTENTS : NOT USED no. of extents SIZE : xxx

. . .
. . .
. . .

This is the trailer message when the allocation has been correctly performed. Note that if more than one volume is specified in GLOBAL and allocation is not possible on one of the volumes, then if it is possible to allocate the file on the remaining volumes (omitting the faulty volume), the preallocation will be considered successful.

SPACE EXTENSION FOR THE FILE : filename

TOTAL AMOUNT OF ADDITIONAL SPACE ALLOCATED : xxx CYLINDERS TRACKS

ON THE FOLLOWING VOLUMES :

VOLUME : media name EXTENTS : no. of extents NOT USED SIZE : xx

. . .
. . .
. . .

This is the trailer message when the space extension has been correctly performed.

EXTENSION NOT PERFORMED FOR FILE : file-name

followed by one of the following :

ILLEGAL FILE ORGANIZATION

FILE NOT FOUND
NON STANDARD LABEL
MEDIA NOT LAST VOLUME OF CURRENT FILE
AT LEAST 1 MEDIA MUST BE SPECIFIED

This is the trailer message when a request for space extension fails.

EXTENSION ONLY PARTIALLY PERFORMED FOR THE FILE : file-name

This is the trailer message when a request for space extension is only partly fulfilled.

ALLOCATION NOT PERFORMED FOR FILE : file-name

followed by one of the following :

. SIZE : MISSING OR ILLEGAL VALUE
. RECFORM : ILLEGAL VALUE
. RECSIZE : ILLEGAL VALUE
. BLKSIZE : ILLEGAL VALUE
. KEYSIZE : ILLEGAL VALUE
. INDEXSZ : ILLEGAL VALUE
. GENOV : ILLEGAL VALUE
. CYLOV : ILLEGAL VALUE
. KEYLOC : ILLEGAL VALUE
. NUMULAB : ILLEGAL VALUE
. CIFSP : ILLEGAL VALUE
. CAFSP : ILLEGAL VALUE
. CISIZE : ILLEGAL VALUE
. CASIZE : ILLEGAL VALUE
. INCRSIZE : ILLEGAL VALUE
. TRACFAC : ILLEGAL VALUE
. KEYLOC : SECONDARY INDEX NO (index-number) : ILLEGAL VALUE
. BLKSIZE MUST EQUAL RECSIZE WITH RECFORM = F
. BLKSIZE MUST BE A MULTIPLE OF RECSIZE WITH RECFORM = FB
. SIZE : TO SMALL
. ILLEGAL RECORD FORMAT IN DIRECT
. ILLEGAL RECORD FORMAT IN INDEXED SEQUENTIAL
. ERROR IN primitive PRIMITIVE AT ADDRESS : address
. ERROR DURING EXECUTION OF PRIMITIVE CODE : . .

This is the trailer message when a request of preallocation fails before space has been allocated by storage management. These errors are all of severity 3 and will cause the program to abort.

ALLOCATION NOT PERFORMED FOR FILE : file-name

SPACE COULD NOT BE ALLOCATED ON THE FOLLOWING VOLUMES :

VOLUME : volume name ERROR : error type

. .
. .
. .

This is the trailer message when the allocation is not possible.

The types of errors can be :

VTOC ERROR : Non-standard VTOC
DUPLICATE NAME : The file-name already exists on the volume
NO ROOM IN VTOC : There is no more available file label
NO AVAILABLE SPACE : There is no room on the volume or the allocation can only be made with more than the MAXEXT number of extents.
SYSTEM ERROR RETURN CODE : xxxxx . : Storage management has used a primitive giving an unexpected return code.

If some severity 1 errors are found for one volume of a multivolume file, but allocation is possible on the others, no abort will occur.

THE FOLLOWING ERROR OCCURED AFTER PREALLOCATION OF FILE : file-name.

. I/O ERROR DURING FORMATTING
. I/O ERROR WHEN WRITTING EXTENDED LABEL
. ERROR DURING EXECUTION OF primitive RETURN CODE :
. FILE COULD NOT BE PREFORMATTED
. ERROR IN OPEN OR CLOSE FILE RETURN CODE :
. FILE LABEL 2 COULD NOT BE WRITTEN
: VTOC ERROR TRY VOLCHECK
: FILE LABEL NOT FOUND
: FILE LABEL 2 DOES NOT EXIST
: NON STANDARD LABEL

The above messages are followed by :

DEALLOCATION REQUIRED FOR ALREADY ALLOCATED FILE : file-name

This message is then followed by the same detailed report of the deallocation process as in the utility $DEALLOC.

OPTION STRING INCORRECT OR UNAVAILABLE

The step cannot get the option string.

The following messages relate to the catalog :

CATALOG WAS UPDATED

These messages follows the $PREALLOC trailer message if a cataloged file was successfully preallocated and the catalog successfully updated.

CATALOG PROBLEM : FILE NOT IN CATALOG
CATALOG PROBLEM : NAME DOES NOT FOLLOW NAMING CONVENTION
CATALOG PROBLEM : OBJECT NOT A FILE
CATALOG PROBLEM : I/O ERROR
CATALOG PROBLEM : NEW GENERATION IS REFERENCED
CATALOG PROBLEM : FILE ALREADY HAS A CATALOGUED MEDIA LIST

These messages imply that allocation was not attempted because of problems with the catalog.

UNABLE TO MODIFY CATALOG

This message occurs if preallocation was successful, but the catalog could not be modified. This message should be followed by the message and action of:

DEALLOCATION REQUIRED FOR ALREADY ALLOCATED FILE : file-name

This is to avoid having a file existing with an unstable catalog entry. If the deallocation is successful, then a new $PREALLOC can be issued. If it is not successful, the FORCE option can be used to avoid referring to the catalog.

FILE HAS NO CATALOGUED MEDIA LIST

or

DEVCLASS NOT THE SAME AS ORIGINAL DEVCLASS

These additional messages may appear if the cataloged file could not be found. The following messages apply to magnetic tape files :

PREALLOCATION OF THE FILE: file-name

ON THE FOLLOWING TAPES :

tape name

.
.
.

TAPE STILL CONTAINS VALID FILE

AN ERROR OCCURED ON THE TAPE : volume-name

EXTENSION OF FILE : file-name

ON THE FOLLOWING TAPES :

tape name

.
.

EXTERNAL FILE NAME NOT FOUND ON MEDIA LIST PROVIDED BY USER

TAPES DEMANDED ALREADY ASSIGNED TO A FILE

SORTIDX

– Function : To sort and load the secondary indexes of an UFAS Indexed Sequential File.

– Statement form :

```
SORTIDX OUTFILE = (file-description)
        [,WKTAPE  = (NBDV = n, DEVCLASS = device-class)
        ,WKDISK   = ( external-file-name | SIZE = value ) [, RESIDENT | DEVCLASS = device-class,MEDIA=volume-list ] ]
        [,STEPOPT = (step-parameters)];
```

– Statement description :

This utility must be used to sort the keys stored into the secondary index of an Indexed Sequential file into ascending order. The process of loading the secondary keys into an index is not assumed, in the general case, to include their reordering within the index. The only file organization that currently supports secondary indexes is the UFAS Indexed Sequential file organization.

– Parameter description :

OUTFILE — This parameter describes the file whose secondary indexes are to be sorted.

WKTAPE — This optional parameter declares that the sorting media are magnetic tapes.

NBDV

Allows the number of devices required to support the sort work tapes to be specified in n. The minimum value of n is 3, and the maximum value is 6.

WKDISK — This optional parameter specifies that the sorting media are disks. The work file may be permanent, in which case it must have been preallocated, or it may be temporary. If it is temporary, the space specified will be allocated when the file is opened. The file will be formatted by the sort as needed.

SIZE — Indicates that the file is temporary, and specifies that nnn cylinders should be allocated on the volume specified in the MEDIA list.

Example :

```
SORTIDX OUTFILE = XQ.DAT, WKTAPE = (NBDV = 3,
                                    DEVCLASS = MT/T9) ;
```

In this example, the secondary indexes of the file XQ.DAT will be sorted into ascending order, and 3 9-track work tapes will be the sort media.

7. Volume level utility specifications

This section contains the full specifications for the Volume Level Utilities in alphabetic order by statement name. Each utility specification consists of :

1. The utility function
2. The statement form, using the notation convention defined in the Preface
3. A description of the utility operation
4. A description of the parameters and keywords
5. A set of one or more examples
6. A list of messages and diagnostics

A summary of these utilities is provided in Table 7-1.

Table 7-1. Volume Level Utilities

Statement name	Function
$VOLCHECK	Check the integrity of a disk volume
$VOLCOMP	Compare two volumes
$VOLCONTS	List a Volume Table of Contents of a disk
$VOLDUPLI	Make a duplicate of a volume
$VOLPREP	Prepare a disk or tape volume
$VOLPRINT	Print the physical blocks of a tape volume
$VOLREST	Restore a $VOLSAVE file onto a disk volume
$VOLSAVE	Save a disk volume into a tape file

VOLCHECK

– Function : To check the integrity of a disk volume.

– Statement form :

```
$VOLCHECK   DEVCLASS  = device-class
          , MEDIA     = volume-name
          [ , DELETE ] ;
```

– Statement description :

The purpose of this utility is to verify that the format of all the VTOC labels is correct and that each track of the volume is described once and only once by the label information.

In the VTOC label the utility checks :

. The key

. The format identifier

. The VTOC extent

. The number of type-zero file labels available

. The address of the highest type-one file label

In tne file labels section of the VTOC it checks :

. The format identifiers

. The extents (extent type, extent sequence number and extent consistency).

. All the links between file labels and organization dependent file labels – and additional extent labels.

In the unallocated space label it checks :

. The format identifier

. The free extents (they are compared with those obtained after all the file labels have been checked).

. The links between unallocated space labels

The utility also verifies that there are no inconsistencies between the extents (overlapping).

Restriction : when $VOLCHECK is applied to a RESIDENT disk the user must ensure that GCOS is mono-programming, that is, no other jobs (including the input reader and output writer) are executing concurrently.

$VOLCHECK will not work on volumes with more than 16 bad tracks.

$VOLCHECK will abort with return code 1219 (TABOV) in the following circumstances :

. More than 100 extents on the volume

. Overlap condition found for more than 10 files

. More than 1024 labels in the VTOC

. More than 30 files for which label errors are detected

If the optional parameter DELETE is specified then any temporary files found will be deleted.

The report produced by the utility consists of a title line and volume identification line followed by a series of messages grouped into three sections :

1. Volume Description : This section consists of three parts :
   a. Unrecoverable errors
   b. Recovered errors
   c. Suggested action

The unrecoverable errors are those found by the utility which it cannot correct :

VTOC LABEL KEY ERROR

The key of the first label is not equal to «04». No correction is made.

VTOC LABEL EXTENT TYPE ERROR

The extent type within the VTOC is not equal to «00». No correction is made.

THE LABEL FOLLOWING VTOC LABEL IS NOT THE EXPECTED ONE

The VTOC label is not followed by the unallocated space label. No correction is made.

UNALLOCATED SPACE LABEL FAILS AND CANNOT BE CREATED VTOC EXTENT IS TOO SMALL

The creation of an unallocated space label is needed and $VOLCHECK cannot create it since there is no more VTOC space of the correct type available.

VTOC LABEL FORMAT IDENTIFICATION HAS BEEN CORRECTED

The VTOC label format identification was not equal to «F4». It is set to this value.

VTOC LABEL EXTENT SEQUENCE NUMBER HAS BEEN CORRECTED

The VTOC label extent sequence number was not equal to «00». It is set to this value.

VTOC LABEL EXTENT BOUNDS HAVE BEEN CORRECTED

The inferior extent bound within the extent field of the VTOC label was not equal to the bound within the volume label. It has been set to the volume label value.

VTOC INDICATORS HAVE BEEN UPDATED

The list «VTOC ERROR» and/or the list «NOFLS» has been reset in the VTOC indicator byte within file label 4.

VTOC LABEL HIGHEST FILE LABEL ADDRESS OR AVAILABLE LABELS NUMBER HAS BEEN CORRECTED

Self explanatory message.

UNALLOCATED SPACE LABEL NUMBER n AN EXTENT ERROR HAS BEEN CORRECTED

There was at least one extent within the unallocated space label n which was incorrect. Correction is performed.

UNALLOCATED SPACE LABEL NUMBER n KEY ERROR A NEW LABEL HAS BEEN CREATED

The unallocated space label (n-1) pointed to the label n, the key of which was not equal to «05». A new label is created and the pointer is updated.

UNALLOCATED SPACE LABEL NUMBER n FORMAT IDENTIFICATION HAS BEEN CORRECTED

The format identification of unallocated space label n has been set to «05».

The suggested action in the volume description consists of one of the following messages :

NO ACTION	:	self explanatory
RESTORE FILE	:	use $FILREST to restore file
RESTORE VTOC	:	use $VOLREST to restore the volume
DEALLOCATE FILE	:	use $DEALLOC to remove file
REALLOCATE FILE	:	use $PREALLOC to allocate file
PREPARE VOLUME	:	use $VOLPREP on the volume

2. File Description : This section of the report provides information on all the files which are present on the volume :

   - external-file-name
   - whether the file is multi-volume or not
   - the file organization
   - the number of cylinders and tracks occupied
   - detected errors
   - suggested action

The detected errors may consist of one or more of the following messages :

FILE ORGANIZATION DEPENDANT LABEL FAILS AND CANNOT BE CREATED VTOC EXTENT IS TOO SMALL

Self explanatory message : no correction can be made.

FILE LABEL VOLUME SERIAL NUMBER ERROR

This message occurs if in the first volume of a file, the volume serial number (volume name) found within the file label, is not equal to the volume serial number found in the volume label. No correction is made.

FILE LABEL FORMAT IDENTIFICATION HAS BEEN CORRECTED

The format identification of the file label has been set to «F1».

NO SPACE WAS ALLOCATED THIS FILE HAS BEEN DELETED

All the extents of the file label are empty. The label is deleted.

ADDITIONAL EXTENT LABEL NOT ALLOWED THIS LABEL HAS BEEN DELETED

The file label contains less than 3 extents, the additional extent label is then in error and is deleted.

FILE ORGANIZATION DEPENDANT LABEL FAILING A NEW LABEL HAS BEEN CREATED

Self explanatory message.

ADDITIONAL EXTENT LABEL KEY ERROR THIS LABEL HAS BEEN DELETED

The pointer label is not the expected additional extent label. The pointer to this label is deleted.

ADDITIONAL EXTENT LABEL FORMAT IDENTIFICATION HAS BEEN CORRECTED

The format identification is set to «F3».

ADDITIONAL EXTENT LABEL NO SPACE WAS ALLOCATED THIS LABEL HAS BEEN DELETED

Self explanatory message.

EXTENTS FOUND AFTER A NON VALIDATED EXTENT HAVE BEEN DELETED

An extent has been found with an error and has been deleted, all the following extents are deleted also.

EXTENT SEQUENCE NUMBER ERROR EXTENTS HAVE BEEN DELETED

The extent sequence number of an extent is wrong. This extent has been deleted.

EXTENT NUMBER ERROR THIS NUMBER HAS BEEN CORRECTED

Self explanatory message.

EXTENT BOUNDS ERROR THIS EXTENT HAS BEEN DELETED

The extent bounds are not valid since there is an inversion between the extent bounds. This extent is deleted.

The suggested action messages in the File Description section are the same as those in the Volume Description section of the utility report.

3. Overlap Information

   These messages give the names of the files and the extent addresses, cylinder/tracks, which overlap each other. The address values are printed in hexadecimal.

   When one or more of these messages occur, the following instructions are also displayed :

   SUGGESTED ACTIONS TO DO SUCCESSIVELY

   1 – TO PRINT THE FILES WHICH OVERLAP EACH OTHER

   2 – TO DEALLOCATE THE WRONG FILE

   3 – TO RUN AGAIN VOLUME CHECK UTILITY

In addition to the above message sets, the following messages may also be displayed by $VOLCHECK :

ERROR 136 NON STANDARD VOLUME RC = return-code

The volume to be checked is not a NATIVE one.

FATAL ERROR xx RC = return-code

This message will occur when an internal call to a GCOS system facility encounters an error. The xx codes are explained below :

xx values	Explanation
8, 9, 10, 11, 12	Central Processor error during OPEN/CLOSE
16, 17	Creation or deletion of buffers unsuccessful
24	Error in option string parameters, as passed to the utility from the $VOLCHECK statement
32	Open file not successful - error in one file label extent
40	Overflow has occurred in an internal software table concerned with the VTOC
48	Logical Device Number cannot be obtained
56, 64, 65, 66, 67	An I/O error has occurred during file processing
72	A non-standard label has occurred, despite previous verification.
80, 81, 136	Errors encountered in volume label handling
144	The file is not on the volume
152	The file organization is not supported in this software release
160	The file is multi-volume

Example :

VOLCHECK DEVCLASS = MS/M400, MEDIA = 15 X 14 ;

VOLCOMP

- Function : To compare two tape or disk volumes of the same type and to report on the differences encountered. Cassette volumes are not supported.
- Statement form :

```
$VOLCOMP INVOL1 = (DEVCLASS = device-class
                   , MEDIA  = volume-name
                  [ , LABEL = { NATIVE } ] )
                              { NONE   }
                              { NSTD   }
                  [ , TAPEND = nn]

        , INVOL2 = (DEVCLASS = device-class
                   , MEDIA  = volume-name
                  [ , LABEL = { NATIVE } ]  )
                              { NONE   }
                              { NSTD   }

        [, LIMIT  = value]

        [ , FORMAT = { ALPHA } ]
                     { HEX   }
                     { BOTH  }

        [ , SKIP   = nnn]

        [ , BUFFER = value]

        [ , PRTOUT = (sysout-parameters)] [ , PRTFILE = (file-description)]
        [ , PRTDEF = (define-parameters)] [ , VTOC]
        [ , STEPOPT = (step parameters)];
```

Statement description :

For disk volumes the compare operation may be divided into the following actions :

. Comparison of both volume tables of contents (VTOCS). An exception will be reported if the VTOCS do not contain the same number of files, having matching names and other attributes. Obviously the volume names will not match and this mismatch will not be reported. Similarly no exceptions will be reported when the Volume Serial Numbers (VSN) of files do not match.

. For each file, both lists of allocated extents will be compared. If these lists are different then the utility terminates abnormally.

. For each allocated extent a comparison is made between the two volumes on a track by track basis.

For tape volumes comparison is performed block by block. If a comparison is performed between labelled tapes then no exception reports will occur on mismatches of the following :

. Volume names

. Volume Serial Numbers

. Private user's labels, which may be present on one volume and not on the other.

When a comparison exception occurs the relevant block or track is printed in alphanumeric and/or hexadecimal form. It is also possible to place an upper limit on the number of exceptions to be printed.

– Statement parameters :

INVOL1	This keyword is followed by a parameter group identifying the first of the two volumes to be compared. For further details see Section III File and Volume Identification .
INVOL2	This keyword is followed by a parameter group describing the second volume.If the LABEL keyword is present in INVOL1 and is NONE or NSTD, it must be the same in INVOL2.
TAPEND	This keyword is used when LABEL = NSTD and it is then mandatory. It states the number of tape marks which are to be processed before processing stops. It is specified as a two digit decimal number. A no-record-found condition (time out on tape) also stops processing.
LIMIT	This keyword may be used to specify the maximum number of mismatches that are to be printed. When this limit is reached the comparison will terminate. The limit is expressed as a numeric value of up to 4 digits in length. If no limit is given then all exceptions found in the comparison will be reported.
FORMAT	This keyword specifies the format to be used in reporting comparison exceptions. If ALPHA is given then blocks or tracks will be displayed in alphanumeric format only. If HEX is given then the listing will be hexadecimal only. If BOTH, the default, is the FORMAT value then the listing will consist of both alphanumeric and hexadecimal formats.
SKIP	This parameter, a three digit decimal number which is optional, gives the total number of unreadable tracks that are to be skipped on the input volumes. An unreadable track is processed in the same way as a track which differs from the one it is being compared to.
BUFFER	For tapes only : this keyword supplies the maximum block size, in bytes, for tape to tape comparison. The numeric value given must not be more than five digits long and must not exceed 30,000 in value.
VTOC	For disks only : if this parameter is given then only the Volume Tables of Contents(VTOC) of the two volumes are compared. Hence only the first phase of disk comparison as described above in «Statement Description» is performed.
PRTFILE PRTDEF PRTOUT	These optional parameters introduce various parameter groups whose purpose is to define the output file and how it is to be printed (see Section 4).

Examples :

```
$VOLCOMP  INVOL1 = (DEVCLASS = MS/M400 MEDIA = SERTA),
          INVOL2 = (DEVCLASS = MS/M400 MEDIA = SERTX),
                   LIMIT = 40 ;
```

The two disk volumes SERTA and SERTX are compared. Only the first forty exceptions will be listed, and the print format will be FORMAT = BOTH, alphanumeric and hexadecimal.

```
$VOLCOMP  INVOL1= (DEVCLASS = MT/T9/D1600, MEDIA = TID4R),
          INVOL2= (DEVCLASS = MT/T9/D1600, MEDIA = TRB4R),
          BUFFER = 3000 ;
```

VOLCOMP

The two tape volumes, TID4R and TRB4R are compared. It is assumed, by default LABEL = NATIVE, that both volumes are labelled. The maximum block size is 3000 bytes and all exceptions will be listed, in both alphabetic and hexadecimal format.

VOLCONTS

- Function : To produce a printed report on the contents of a native disk, tape, or cassette volume.

- Statement form :

```
$VOLCONTS   DEVCLASS  = device-class
            , MEDIA   = volume-name  [ , LABEL = { NATIVE | COMPACT } ]
            [ , SHORT ]
            [ , PRTFILE = (print-file-description) ]
            [ , PRTDEF  = (define-parameters) ]
            [ , PRTOUT  = (sysout-parameters) ]
            [ , STEPOPT = (step-parameters) ] ;
```

- Statement description

This utility lists the following information pertaining to a disk volume :

- The volume name and VTOC (Volume Table of Contents) address.
- The list of free extents left on the volume. This information is not available for IBM DOS volumes, where free space is ignored.
- The files resident on the volume. For each file present the program reports in exactly the same way as $FILDESC does. The only exception is that BFAS Indexed Sequential file usage statistics are not printed even when SHORT is not used.
- File descriptions are printed in alphabetic order of file name.
- When the SHORT option is used, the printed report is reduced to a condensed summary table of contents of the volume.
- When SHORT is not used, a logical file opening is often performed to obtain more information. When the file is a multivolume file, the attempt to open the file may fail because other volumes are not available. This failure will be reported, and the file description will look incomplete compared to the other descriptions.

- Statement parameters

DEVCLASS	This keyword is followed by the device class of the volume.
MEDIA	This keyword is followed by the name of the volume which is to be inspected.
LABEL	Is used for COMPACT format cassettes.
SHORT	This optional parameter should be used if the free or available space is to be reported with a brief summary of the description of the files resident on the volume. The space occupied by files whose expiration date has passed is not considered as free space. For the space occupied by a file to become free, the $DEALLOC utility must be used on the file. If this parameter is omitted, then all the details of volume contents will be listed, and a complete description of every file will be given. In addition, a summary of allocated files will be printed.
PRTFILE PRTDEF PRTOUT	These optional parameters introduce various parameter groups whose purpose is to define the output file and how it is to be printed (see Section 4).

Examples :

```
VOLCONTS MEDIA  = BDXC21, DEVCLASS = MS/M400 ;
```

This statement will produce an occupancy report on MSU0400 disk volume BDXC21. All the pertinent details mentioned in the statement description above will be printed.

```
VOLCONTS MEDIA = L60D83, DEVCLASS = MS/M300, SHORT ;
```

In this example the utility is applied to an MSU0310 disk named L60D83 in order to produce a report on the available space. Details of valid files found will not be listed.

— Step Completion Conditions

It should be noted that $VOLCONTS uses three work files, which are :

H_LABELS	accessed through the internal-file-name «h_labels». It is a UFAS indexed sequential file used to store all the file labels and reorder them according to the alphabetic sequence of external-file-names.
H_SUBFLS	accessed through the internal-file-name «h_subfls». It is a UFAS indexed sequential file used to store all of the subfile entries from a given library. This workfile is also used to reorder the description of subfiles in the alphabetic order of the subfile names.
H_SUMARY	accessed through the internal-file-name «h_sumary». This workfile is a UFAS Sequential file used to build the summary table of contents of the volume to be described. The final contents of H_SUMARY are copied into the sysout (or PRTFILE) file at the end of step execution. When SHORT is used, the produced report is restricted to that table of contents. When SHORT is not used, the table of contents is appended to the end of the extended report.

The description of every file on the volume requires that the file being described is dynamically assigned to the $VOLCONTS step (especially when SHORT is not used). In a number of cases, the file is then opened to access library members, or extended labels, or statistical information.

When a file is dynamically assigned, an abnormal return code can be obtained. In particular, one can get :

CONFLICT	the file was already assigned to the step. This happens for $VOLCONTS workfiles during the description of the system disk (or of a resident volume).
BUSY	the file is assigned to another job step.

When a attempt is made to open the dynamically assigned files, it is also possible that abnormal return codes might occur. In particular :

EXTERR	this occurs when the file to be opened is a multivolume one.
CATERR	means that a catalog should be attached.

Because there is very little user input to $VOLCONTS, most fatal malfunctions are considered to be system malfunctions, and result in a SEV3 or SEV4 termination status. Malfunctions are most likely to occur when the input volume is accessed for the collection of the file labels to edit, and when the system is asked for resources, which might not be available ; for example, secondary storage space for work files, locked memory for buffers, abnormal opens on the work files, and so on.

Once the labels have been collected, the label editing phase starts. From then until the end of the step, most malfunctions will result in SEV1 error messages (warnings). If a particular file cannot be assigned (BUSY), it will be partially described, as some of the label information will be available, and $VOLCONTS will go directly to the next file. The same thing happens if the file cannot be opened, or if some subfiles of a library cannot be accessed.

Note that when $VOLCONTS is applied to a tape or cassette volume, the entire volume is scanned, and the count of data blocks is extracted from the EOF labels. This is not done by $FILDESC.

No checkpoints are taken with $VOLCONTS.

VOLDUPLI

- Function : To copy the contents of an entire volume, disk or tape, onto another volume of the same type. The VOLDUPLI function is partly performed for cassettes by using $CREATE with FILE FORM = NSTD
- Statement form :

```
$VOLDUPLI   OLD = (DEVCLASS = device-class
                  , MEDIA   = volume-name
                 [, LABEL = = {NATIVE} ] )
                              {NONE   }
                              {NSTD   }
                  [ , TAPEND = nn ]
            , NEW = (DEVCLASS = device-class
                  , MEDIA   = volume-name
                 [, LABEL = { NATIVE } ] )
                            { NONE   }
            [ , DENSITY = density]
            [ , BUFFER = value ]
            [ , SKIP   = value] [,STEPOPT = (step-parameters)] ;
```

- Statement description :

For copying from disk to disk, the duplication is performed on a track by track basis with old alternate tracks being inserted in the new volume and any necessary new tracks being made alternate if there are defective tracks. Although only the allocated tracks of the source volume are copied, the same address space will be used to write the tracks on the target. Thus the integrity of Indexed Sequential file (where the index contains absolute addresses) is maintained. Note that $VOLDUPLI will fail if the target volume supplied contains any valid files.

For disk volumes, the following mappings are supported.

MSU0350 to MSU0400, MSU0402, MSU0452

MSU0400 to MSU0402, MSU0452

MSU0402 to MSU0400, MSU0452

Duplication from a higher capacity volume to a lower capacity volume is not supported.

Tape to tape copy operation is performed on a block by block basis. The input label organization is always maintained on output. Thus if the input volume is NATIVE then the output volume will be. If the input volume is unlabelled or non-standard, then the output volume will be the same.

If the input volume contains cataloged files, they will be copied as cataloged files onto the output volume (that is, the catalog flag will be set in the file labels). The catalog which contains the file attributes will not contain an entry for the copied files, as the media list will be out of date. If the copied files are to be cataloged, then the catalog entry must be modified by the user to contain the new media list.

The REPEAT option of the step-parameters is not supported for disks, as the system cannot guarantee the exclusive reservation of volumes between abort and restart. If REPEAT is used with disk duplication, a warning message is sent, but execution continues. If REPEAT is used for disk or tape, no checkpoints are taken.

Note that it is possible for the target tape to have previously recorded labels when the input tape is unlabelled or non-standard. In this case the output tape is checked to ensure that it does not contain a valid file. If it does then the utility will fail. If there are no files present then the copy operation takes place and the final form of the output tape is unlabelled or non-standard, like the input tape.

When an unlabelled tape is copied, the first Tape Mark encountered is taken as the end-of-volume.

When a non-standard tape is copied, the user may code that the operation be performed until a specified number of tape marks have been read.

It is not possible to duplicate a WORK volume. The output volume cannot be WORK.
It is not possible to duplicate a cassette volume.
It is not possible to duplicate an ASCII tape with the SKIP option present.

– Statement parameters :

OLD	This keyword is followed by a parameter group describing the volume to be copied. The keyword LABEL is only relevant for tape volumes. For further details see Section III, «File and Volume Identification».
TAPEND	This two digit parameter is mandatory for non-standard volumes and is only valid when LABEL = NSTD. TAPEND specifies the number of tape marks to be read before terminating. If TAPEND = 2 then duplication will finish when the second tape mark is read. Whatever TAPEND is, then reading also terminates when the no-record-found condition is encountered.
NEW	This keyword is followed by a parameter group describing the output volume for the copy operation. The device class must correspond with that for the input volume given in the OLD parameter group. In a disk-to-disk operation the device type must be compatible as indicated. The keyword LABEL, used only for tapes, describes the volume organization prior to the utility operation. The organization of the tape after duplication will be the same as that of the input volume.
DENSITY	For tapes only : this keyword enables the user to specify the new recording density to be used in writing the output tape volume. The allowed values are D200, D556, D800 (default for 7-track), and D1600 (default for 9-track).
BUFFER	For tapes only : for a tape copy operation this keyword must be supplied giving the maximum blocksize of the tape to be copied. The numeric values may be up to five digits long but must not exceed 30000.
SKIP	This parameter specifies what action is to be taken when a read error is encountered. If SKIP = nnn then the utility will only skip nnn bad blocks. If more than nnn read errors are encountered then the utility will fail. If the SKIP is omitted then the utility will fail if any read error is encountered.

Examples :

```
$VOLDUPLI  OLD = (DEVCLASS = MS/M400, MEDIA = DSKB8),
           NEW = (DEVCLASS = MS/M400, MEDIA = DSKC8) ;
```

To copy the contents of MSU0400 type disk DSKB8 onto disk DSKC8.

```
$VOLDUPLI  OLD = (DEVCLASS = MT/T9/D800, MEDIA = T2641),
           NEW = (DEVCLASS = MT/T9/D800, MEDIA = T48XQ),
           DENSITY = D800, BUFFER = 8008 ;
```

To copy the contents of the labelled tape volume T2641 onto a tape T48XQ. The density of the new output labelled volume T48XQ will be 800 bpi. The maximum block size for the copy operation is 8008 bytes.

Step Completion Conditions :

If a $VOLDUPLI step abnormally terminates when performing standard tape duplication, the output tape is rewound, and the volume and file labels are overridden ; this ensures that the output volume is left in the same state as an «empty scratch volume». The exception to this is when the output tape is not long enough to hold the entire input volume, in which case, $VOLDUPLI terminates abnormally, but the output tape is not scratched.

If a disk duplication step is restartable (REPEAT) and has aborted, the restart will probably fail because of valid files being found on the output volume ; these valid files are those already copied before the step aborted.

No checkpoints are taken with $VOLDUPLI.

VOLPREP

– Function: To label and format a disk, cassette, or tape volume for use by GCOS or to remove the existing standard labels. For disk volumes, a list of all defective tracks is produced.

– Statement form :

```
$VOLPREP  OLD = (DEVCLASS = device-class
                 MEDIA    = volume-name
                [ LABEL = { NATIVE } ] )
                          { NONE   }
                          { COMPACT}
                          { NSTD   }
                          { G100   }
, NEW =(DEVCLASS=device-class,MEDIA=volume-name [, LABEL ={COMPACT} ] )
                                                        {NATIVE  }
                                                        {NONE    }
        [ , DENSITY = { D1600 } ]
                      { D 800 }
                      { D 556 }
                      { D 200 }
        [ , { NTRKANL  } ]   [ , VTOC ADDR = { 0/1    } ]
            { NTRKPRF  }                     { ccc/tt }
            { COMPLETE }
        [ , WORK] [ , FORGET]
        [ , BYPASS] [ , FORCE]
        [ , BADTRACK = (ccc/tt [ , ccc/tt] . . . ) ]
        [ , STEPOPT = (step-parameters) ] ;
```

– Statement description :

The $VOLPREP utility, for tapes and cassettes, writes a volume label containing the new volume name and follows this with a dummy header file label and a tape mark. If there is already a valid (unexpired) file on the tape then the utility will fail unless BYPASS is specified.

For disk volumes, the following functions are performed :

- Surface analysis : each track is verified and if any is found defective then an alternate track is assigned. This validation is applied to the alternate area first.
- Track preformatting : standard home addresses are written on each track and the remainder of each is erased.
- System Load Records : space is reserved on track 0 cylinder 0 for these records.
- The VTOC (Volume Table Of Contents) is initialized. This table is recorded containing the volume name and other volume characteristics.
- User Defined Defective Tracks : These tracks are those that are found to be defective by the disk pack manufacturer. The precise addresses (cylinder/track) are supplied with each pack.
- Operator verification : a check is made on the target disk to ensure that there are no valid (or cataloged) files present. If valid files are present then the utility will fail unless BYPASS has been specified. If BYPASS is present and there are valid files on the volume then the operator is asked for permission for execution to continue.
- VOLPREP should not be considered as a means to remove cataloged files from a volume because it does not update the catalog. DEALLOC must be used to clear cataloged files from a volume which is being prepared. As a recovery tool FORCE can be used to ensure that VOLPREP ignores cataloged files.

The utility may be used to remove existing labels (for a disk the VTOC is erased) from a volume.

This feature permits the transfer of a volume from LABEL = NATIVE to LABEL = NONE.

This is the only method in GCOS of removing standard labels (with the exception of $VOLDUPLI). ★

$VOLPREP provides a means of upgrading or downgrading the volumes of a particular type to volumes of the same type but different capacities. The possible changes are :

MSU0350 volumes can be changed from/to MSU0400 volumes

MSU0402 volumes can be changed from/to MSU0452 volumes

When the volume to be upgraded does not contain files of interest to the user, the upgrade can be performed by a simple $VOLPREP. The utility recognize a device-class change for the volume to be prepared ; it asks the system to have the volume mounted on a device of the new device-class, and operates as if the FORGET option was given. The list of bad tracks must be re-specified by the user.

When the volume to be changed contains files of interest to the user, he may either perform a $VOLSAVE, a $VOLPREP, and then a $VOLREST to bring the files back onto the volume, or the $VOLSAVE and $VOLREST can be replaced by $VOLDUPLI.

DEFECTIVE AND ALTERNATE TRACKS

There are three types of defective track which may occur on a disk volume :

1. A track that does not allow the registration of a home address and a one track long record ; this is a $VOLPREP declared defective (bad) track.
2. A track that is designated defective by the manufacturer of the disk volume after volume certification. This is a Manufacturers Bad Track, and a list of these is given on the volume cover.
3. A track that is declared defective by the user when it causes frequent I/O errors. This is a user declared bad track.

The three types of defective track described above are, to a certain extent, independent, and the automatic detection of defective tracks during surface analysis by $VOLPREP cannot be assumed to be exhaustive. In particular, there is no guarantee that every manufacturer's bad track will be recognized. Therefore, we recommend that you explicitly declare the manufacturer's bad tracks the first time $VOLPREP is used on a volume or when the FORGET option is used.

The FORGET option causes every track which is flagged as defective to be unflagged.

All disk volumes contain a fixed number of tracks that are reserved for use as alternate tracks. These are used as alternates for tracks which are found to be defective when the volume is prepared, or which become defective during the life of the volume . These alternate tracks are set aside as a pool of spare tracks when the volume is prepared. The VTOC (Volume Table Of Contents) format 4 label records the total number of alternate tracks left, and the address of the next available alternate track.

An entire number of cylinders, which depends upon the volume type, is reserved for alternate tracks. These cylinders are the innermost cylinders of the volume.

In addition to the declared bad tracks, $VOLPREP performs a surface analysis of the disk volume. This analysis gives a rapid check that every bad track has been recognized. This uses the entire I/O hardware path, so if the disk drive is malfunctioning, or is not properly tuned, $VOLPREP may find problems which it incorrectly attributes to the disk volume. During the surface analysis, channel programs used by $VOLPREP are declared not retryable by software. In the case of an I/O error, $VOLPREP is notified, and checks the nature of the error. If it is a «data-error», $VOLPREP performs a few further tests and flags the track as defective.

The surface analysis is first done on the alternate tracks before declared bad tracks are processed. This means that defective alternates are withdrawn from the set of available alternates. Every abnormal I/O event is traced on the execution report.

SPACE ALLOCATION

Space is reserved on disks as follows :

- On cylinder 0/track 0 for System Load Records
- On the very innermost cylinder for the test and diagnostic needs. This space has a one cylinder extent, and did not exist with previous software releases. The absence of this space is recognized by the system, so «old» volumes are supported.

  The VTOC is allocated, preformatted, and initialized. The number of available preformatted «file label containers» on a track and in the whole table depends on the volume type. The user can supply a start address for the VTOC.This address must be different from cylinder 0/track 0, and by default, the start address of the table is cylinder 0/track 1. The size of the VTOC table is always to one cylinder or less depending upon its start address.

The length of the VTOC cannot be less than two tracks, and the extent of the VTOC lies between the start address and the end of the cylinder. The VTOC cannot be multicylinder.

– Statement parameters :

OLD — This keyword is followed by a parameter group describing the state of the volume before initialization. A description of the elements in the parameter group is given in Section 4.

A MS/M452 volume can be mounted on a MS/M402 drive. If the site has no MS/M452 configured the volume will be recognized as nonstandard and OLD must be used with LABEL = NSTD.

NEW — This mandatory parameter group describes the state of the volume once prepared.

If LABEL = NONE is coded in this parameter group then the result of the utility is a volume without labels which may only be referenced in succeeding JCL statements by LABEL = NONE or LABEL = NSTD.

DENSITY — For tape only : specifies the new recording density. An MT/T9 tape can be recorded at either 800 or 1600 bpi, but an MT/T7 tape can only be recorded at 800 bpi.

NTRKANL
NTRKPRF
COMPLETE — For disks only : if NTRKANL is specified then the utility does not perform a surface analysis and hence it is assumed that there are no defective tracks. If instead NTRKPRF is given, then neither surface analysis nor track preformatting occurs. It is assumed that the volume is already formatted. In this case only a new VTOC is written. When used with a LABEL = NONE volume NTRKPRF is ignored and a $VOLPREP NTRKANL is actually performed. If COMPLETE is given (the default) then surface analysis and track preformatting are performed.

VTOCADDR — This optional parameter, available with disks only, assigns a specified start address to the VTOC. The VTOC is always assumed to end on the last track of the same cylinder. The minimum number of tracks for the VTOC is two, between the beginning and the end of the array. The VTOC cannot reside in the alternate track area, and the address must always be in the form CCC/TT.

WORK — This parameter applies to work tapes only, and must be used to prepare a work tape. A work tape is a tape that can be used by any job step that asks the system for a work tape, regardless of the media.

FORGET — This optional keyword applies to disks only. It is used to unflag all defective tracks before the BADTRACKS option is processed and surface analysis performed. The list of manufacturers bad tracks should be supplied again in the BADTRACK option. If FORGET is not used, old bad track are preserved, and their set enlarged with the set of tracks which are listed under the BADTRACKS option.
Note that this option cannot be used in conjunction with NTRKPRF.

VOLPREP

FORCE — When this optional parameter is used, the destruction of catalogued files (if any) is allowed without any update of the catalog. The operator is asked for permission to perform the destruction when the volume to be prepared is a disk volume.

BYPASS — If this parameter is coded then utility execution will take place even if there are valid files on the volume. However, the operator will be requested to authorize this execution. If the operator answers in the negative, then the utility will terminate without preparing the volume.

BADTRACK — For disks only: this keyword allows the user to specify that certain disk tracks be assigned alternates. The keyword is followed by a list of cylinder and track addresses, each having the form ccc/hh. Leading zeros in address values may be omitted.

Note that BADTRACK operates independently of the automatic alternate track assignment feature. Assignments by both techniques may occur in the same utility execution. However, it is not valid to supply BADTRACK values in conjunction with NOTRKPRF. In this circumstance alternates are never assigned.

The first cylinder is number 0 and the first track within a cylinder is number 0.

Examples :

```
VOLPREP   OLD = (DEVCLASS = MS/M300, MEDIA = NUPACK,
          LABEL = NONE), NEW = (MEDIA = MASPYA) ;
```

A new disk pack for an MSU0310 drive is initialized with the volume name MASPYA. Surface analysis and preformatting actions are performed on the new pack. Note that NUPACK is only a physical external name which is written onto the outside of the pack : it is not the volume name recorded prior to initialization.

```
VOLPREP   OLD = (DEVCLASS = MS/M400, MEDIA = LQA),
          NEW = (MEDIA = LQB) ;
```

An MSU0400 volume LQA is re-initialized and renamed LQB.

```
VOLPREP   OLD = (DEVCLASS = MT/T9/D800, MEDIA = MASA,
          LABEL = G100), NEW = (MEDIA = PAYC) , DENSITY =
          D800 ;
```

An old Series 100 magnetic tape, MASA, is renamed as a native volume, PAYC. The recording density on the tape reel will be 800 bits per inch.

```
VOLPREP   OLD = (DEVCLASS = MS/M350, MEDIA = LDV45),
          NEW = (MEDIA = BDV45), BYPASS,
          BADTRACK = (41/0, 231/4) ;
```

An MSU0350 disk volume LDV45 is re-initialized with a new name BVD45. If any valid files are present then operator permission will be requested (BYPASS). The user has defined two bad tracks, added to the old bad track set, for alternate assignment.

– Operator Action

When valid files have been encountered while BYPASS is specified the following message is displayed on the operator's console :

```
DU01    VOLPREP : VALID/CATALOGUED FILES ON volume-name
DESTRUCTION AUTHORIZED ?
```

The answer YES will allow preparation to continue. The answer NO will cause utility termination.

During a VOLPREP operation the volume under preparation must not be moved onto another device.

- Step Completion Conditions :

  If $VOLPREP aborts with a severity code greater than SEV3, the prepared volume is left in a non-standard state for a disk volume. If you want to execute the preparation again, it is necessary, prior to the run, to list the old bad tracks set in the BADTRACK option. Note that the old defective tracks are preserved when the volume is a native volume, and if the output format is upward compatible with the input format (that is, from MS/M400 to MS/M452, or to MS/M402 from MS/M400, or to MS/M350) , FORGET must then be omitted.
  $VOLPREP does not take checkpoints.

VOLPRINT

- Function : to print physical blocks of a tape volume. The function of $VOLPRINT is partially performed for a cassette volume by $PRINT with FILEFORM=NSTD.
- Statement form :

```
$VOLPRINT    DEVCLASS = device-class, MEDIA = media-name
             [ , LABEL = { NATIVE } ]  [ , SKIP = nnn ]
                         { NONE   }
                         { NSTD   }
                         { G100   }
             [ , TAPEND = nn ]  [ , DATACODE = ASCII ]
               , BUFFER = nnnnn
             [ , PRTFILE = (print-file-description) ]
             [ , PRTDEF = (define-parameters) ]
             [ ,PRTOUT = (sysout-parameters)]
             { ALL                                                    }
             { ITEM =  (bbbbb [, bbbbb ] . . .)                       }
             { PART =  ( ( bbbbb, bbbbb ) [ ,(bbbbb, bbbbb ) ]. . . ) }
             [ ,FORMAT = { ALPHA } ]
                         { HEX   }
                         { BOTH  }
             [ ,STEPOPT = (step-parameters) ] ;
```

- Statement description :

  This utility will print :

  . The entire volume

  or

  . Specified blocks from the volume

  or

  . Specified ranges of blocks from the volume

  The printed form of each block may be character, hexadecimal or both. If the LABEL = G100 then only hexadecimal output occurs.

  The tape may be standard in format, or unlabelled, or non-standard.

VOLPRINT

– Statement parameters

DEVCLASS
MEDIA

These parameters define the device-class and volume-name of the volume to be printed. These parameters must be present.

LABEL

This parameter specifies the tape format. If omitted then the tape is assumed to be standard GCOS/EBCDIC (that is LABEL = NATIVE).

If LABEL = NONE then the volume is unlabelled and the first tape mark encountered is taken as the end-of-input condition.

If LABEL = NSTD then the volume is non-standard in format. In these cases the user may code a TAPEND parameter to indicate when printing is to terminate.

SKIP

When this option is not present, any unretryable I/O error on the input volume will cause the utility to fail. When it is used, I/O errors will not cause failure until the specified number of failures is reached.

TAPEND

This optional parameter is only valid when LABEL = G100 or LABEL = NSTD. TAPEND specifies the number of tape marks to be read before terminating.

If TAPEND = 2 then printing will finish when the second tape mark is read.

If TAPEND is not coded then reading terminates when the no-record-found condition is encountered.

DATACODE

This optional parameter is included if the tape is in ASC11 code and can only be used when LABEL = NSTD.

BUFFER

This parameter specifies the number of bytes that will be printed from each block. If all the bytes of every block are to be displayed then the maximum blocksize (or greater) must be coded.

This parameter is mandatory for tapes. The maximum value is 30000.

FORMAT

This parameter specifies the printing format : character (ALPHA), hexadecimal (HEX) or BOTH.

The default FORMAT is BOTH.

ALL

If this parameter is coded then all the input blocks will be printed.

ITEM

If this parameter is coded then only the blocks whose numbers are specified are printed. The first physical block on the tape is number 1.

If LABEL = NATIVE then the header labels and associated tape marks are not counted. The first data block is block number 1.

If LABEL = NSTD then the labels and tape marks are counted.

Up to 20, addresses may be specified in the ITEM parameter.

PART

If the parameter is coded then up to 10 pairs of block numbers may be specified. Each pair designates a range to be printed. The range is inclusive : if

PART = ((15, 19))

the blocks 15, 16, 17, 18 and 19 will be displayed. Even if only one range is given, there must be two sets of parentheses.

PRTFILE
PRTDEF
PRTOUT

These parameters define the output file and how it is to be printed.

Examples :

```
$VOLPRINT    DEVCLASS = MT/T9, MEDIA = 11864, BUFFER = 800,
             ITEM = (10, 20, 30, 40, 50) ;
```

A standard labelled tape volume (LABEL = NATIVE by default) named 11864 is printed. The numbers of the blocks printed are: 10, 20, 30, 40 and 50.

```
$VOLPRINT    DEVCLASS = MT/T9, MEDIA = UNK, BUFFER = 10000,
             LABEL = NSTD, TAPEND = 6,    ;
```

An non-standard tape is printed until the sixth tape mark is read.

VOLREST

- Function : to restore the contents of a disk volume from a tape file.
- Statement form :

```
$VOLREST    INFILE = (file-description)
            , VOLUME = (DEVCLASS = device-class-name,
                        MEDIA = media-name)
            [, {ALL  }]
               {VTOC }
            [, NBBUF = {1}]
                       {2}
            [, SKIP = nnn]
            [, STEPOPT = (step-parameters)]
```

- Statement description :

This utility performs the reverse function to $VOLSAVE. It restores onto a disk volume a $VOLSAVE file. Before data transfer the utility checks to ensure that the volume onto which the restore is performed is empty. That is, it does not contain any valid files. If a restore is attempted onto a volume that is not empty then the program will terminate abnormally.

The target volume must be of a type compatible with the saved volume. In addition, the contents of a saved MSU0350 may be restored onto an MSU0400. For details of inter-release compatibility, see Appendix A.

Other possible mappings are :

MSU0350 to MSU0400, MSU0402, MSU0452
MSU0400 to MSU0402, MSU0452
MSU0402 to MSU0400, MSU0452

- Statement parameters :

The meaning of the parameters for $VOLREST are exactly the same as for $VOLSAVE. The following points should be noted in addition :

. In the case when the save file is multi-volume, the volume names must be specified in the same order as they appeared in the $VOLSAVE utility that was used to create the file.

. Only one tape drive will be used in execution if MOUNT = 1 is specified in the INFILE parameter, the default is MOUNT = ALL. When the restore from the first tape volume is complete, it is unloaded and the system requests that the second tape volume be mounted, and so on.

. Note that it is not necessary to restore a save file onto a volume of the same name as that from which it was saved. Thus a volume named VA might be saved onto a file named FSVA and subsequently restored onto a volume named VX. The disk named VX will still have VX as its volume name but it will contain all the files that resided on VA originally.

. The user is warned that the current version of $VOLREST works with only one buffer by default. The double buffering mode must be explicitely requested by the user. Double buffering was the default mode of operation under previous versions of this utility.

Examples :

```
VOLREST    INFILE = (FCU248, DEVCLASS = MT/T9, MEDIA = TCU248),
           VOLUME = (DEVCLASS = MS/M400, MEDIA = NCU248);
```

The contents of save file FCU248 held on tape volume TCU248 are restored onto an MSU0400 disk having volume name NCU248. The file FCU248 must have been produced by utility $VOLSAVE from an MSU0400 type disk.

```
VOLREST    INFILE = (SF54A, DEVCLASS = MT/T9,
                     MEDIA =(VS54AA, VS54AB, VS54AC) ),
           VOLUME = (DEVCLASS = MS/M400, MEDIA = ND54A) ;
```

The MSU0400 disk ND54A is restored with save file SF54A which resides on three tape volumes, VS54AA, VS54AB and VS54AC. Note that when SF54A was created using $VOLSAVE the tape volume names must have been given in exactly the same order as above.

– Step Completion Conditions :

If an abnormal condition occurs before the utility writes onto the disk volume onto which the restore is to performed, the message.

VOLUME NOT MODIFIED

is displayed.

A checkpoint is taken at each tape volume switching. Note that the step might not be repeatable if the utility was aborted after the VTOC was restored (if the VTOC contained valid or catalogued files), and before the first tape volume switching takes place.

The output disk volume is not kept reserved by the system for $VOLREST after the step has been aborted and until it is restarted. For a $VOLREST abort/restart situation to be safe, the system should be operated in monoprogramming mode.

If the SKIP option has been used and the save file contains unreadable blocks, the number of skipped blocks is not counted by the utility ; the address of the last track which has been restored before encountering the unreadable block(s) is displayed. The step completion code is then SEV1.

On the output disk volume every track which corresponds to a skipped block from the tape is left untouched.

VOLSAVE

- Function : to save the contents of a disk volume into a native labelled tape file.

- Statement form :

```
$VOLSAVE    VOLUME = (DEVCLASS = device-class, MEDIA = volume-name)
            , OUTFILE = (file-description)
            [, SKIP = nnn]
            [, { ALL  }]
               { VTOC }
            [, NBBUF = {1}]
                       {2}
            [, STEPOPT = (step-parameters) ] ;
```

- Statement description

  The contents of the nominated disk volume are written track by track onto the designated tape file. Only those tracks which are allocated are written. Any alternate tracks on the volume are re-inserted into their correct logical position.

  The output file used may be multi-volume if required. The recording density used is that already on the tapes as recognized by AVR (Automatic Volume Recognition).

  The file format used to save the volume is such that no user defined processing should be attempted on it. The only other programs which handle this format are $FILDUPLI and $VOLREST – volume restore.

  The utility will fail if an attempt is made to save onto a tape which already contains a valid file.

  If the tape file is catalogued, it must have been preallocated before performing the save ; BLKSIZE should be 13200, RECSIZE should be 13196, and RECFORM should be V. For an MSU0300, BLKSIZE should be 7400, and RECSIZE should be 7396. If there are incorrect values, $VOLSAVE will fail.

  For details of inter-release compatibility, see Appendix A.

- Statement parameters :

  VOLUME — This keyword is followed by a parameter group defining the disk volume to be saved. DEVCLASS may only specify a disk.

  OUTFILE — This parameter is followed by a parameter group describing the file to be used in saving the disk volume. The group defines the external-file-name, the device-class (which must be tape), and the volume identification.

  When a multi-volume save is required, the volume mounting is performed according to the user's instructions. The order In which the volumes are used is taken to be the order in which they are specified in the $VOLSAVE statement. If an insufficient number of volumes is provided, additional WORK volumes will be requested. The produced tape file can be used as the input file to $VOLREST via the INFILE option of this utility.

ALL VTOC	These optional keywords specify whether all the extents are to be saved (ALL, the default), or just the VTOC is to be saved.
SKIP	When this option, a three digit number, is not present, any irrecoverable 1/0 error on the input disk volume will cause the utility step to abort. When this option is used, the same kind of failure will not cause the utility step to abort unless the failures exceed a specified limit, but instead, the track that was read is not saved.
NBBUF	This statement describes the number of buffers to be used for input and output by $VOLSAVE.

Note that $VOLSAVE operates by default with only one buffer. Double buffering must be explicitely requested by the user. Previous versions of this utility were working in double buffering mode by default.

Examples :

```
$VOLSAVE    VOLUME = (DEVCLASS = MS/M300, MEDIA = VCU248)
            , OUTFILE = (FCU248, DEVCLASS = MT,
                         MEDIA = TCU248) ;
```

In this example the contents of MSU0310 disk named VCU248 are saved as file FCU248 on tape volume TCU248.

```
$VOLSAVE    VOLUME = (DEVCLASS = MS/M400, MEDIA = BD54A),
            OUTFILE = (SF54A, DEVCLASS = MT,
                       MEDIA = (VS54AA, VS54AB, VS54AC) ) ;
```

In this example the MSU0400 disk BD54A is saved as file SF54A on the tape volumes VS54AA, VS54AB and VS54AC. Note that the utility will use VS54AA first, followed by VS54AB and finally VS54AC.

– Step Completion Conditions :

If an abnormal condition occurs before the utility writes onto the output tape(s), the message :

OUTFILE NOT MODIFIED

is displayed.

If the SKIP option has been used and the input disk volume contains unreadable tracks, these tracks will not be saved at all, but their addresses are recorded in order to warn the user about the tracks when the volume image is reloaded by $VOLREST. The step terminates with a SEV1 completion code.

When an error occurs during the writing to the save file, the file will be closed at the point that has been reached.

A checkpoint is taken at each tape volume switching. If an abort/restart situation occurs, the disk volume must not be used by other jobs before the step is restarted. The user and operator should take care, because the system does not keep the disk volume reserved for the $VOLSAVE step during the time this step is aborted and not yet restarted.

When REPEAT is used, the $VOLSAVE utility should be executed in monoprogramming mode if the abort/restart is to be safe.

Appendix A
Inter-release Compatability of Saved Files and Volumes

The following points should be taken into account when dealing with files and volumes saved with Release 0300 of GCOS.

1) Tapes saved by Release 0300 are in undefined format (U format); only one buffer was used when saving files, and double buffering was used when saving volumes. U format is always read with one buffer.

2) Save tapes from Release 0300 will be processed by 0400 with one buffer; this will mean that $VOLREST will be less efficient when processing these tapes.

3) Saves for 0400 are in variable format (V format), and only one buffer will be used by default. Double buffering is allowed, but must be explicitly requested by the user. The use of double buffers will improve the efficiency of $VOLREST and $VOLSAVE.

4) Saves made with Release 0400 which are processed by Release 0300 must be changed from V format to U format. This can be done by $CREATE.

Table A – 1. The Relationship Between Formats and Functions

	Release 0300		Release 0400	
	RECFORM	**NBBUF**	**RECFORM**	**NBBUF**
FILSAVE	U_f	1	U_f	1
			V_f	1 or 2
VOLSAVE	U_v	2	U_v	1
			V_v	1 or 2
FILREST	U_f	1	U_f	1
			V_f	1 or 2
VOLREST	U_v	2	U_v	1
			V_v	1 or 2

In Table A-1, U_v designates the U format generated by $VOLSAVE. It is processed with double buffers under Release 0300 and with one buffer under Release 0400.

U_f designates the U format generated by $FILSAVE. It is processed with one buffer in Release 0300 and Release 0400.

V_v designates the V format generated by $VOLSAVE, and V_f designates the V format generated by $FILSAVE.

The device upgrades available with $VOLREST are :

Release 0300: MSU/350 to MSU/400/402, MSU/400 to MSU/402

Release 0400: MSU/350 to MSU/400/402/452,MSU/400 to MSU/402/452, MSU/402 to MSU/452

With Release 0400, all Release 0300 saves are accepted directly, but processing will be slower than it is for 0400 saves.

With Release 0300, all U_f and U_v 0400 saves are accepted directly, but 0400 V_f and V_v saves must be converted to U format using $CREATE.

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE: LEVEL 64
DATA MANAGEMENT UTILITIES

ORDER NO. AQ20-03

DATED JULY 1980

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATE

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In the U.K.: Great WestRoad, Brentford, Middlesex TW8 9DH
In Australia: 124 Walker Street, North Sydney, N.S.W. 2060
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

27908, 7.5C880, Printed in U.S.A.

AQ20-03