Honeywell

LEVEL 64

GCOS

SYSTEM MANAGEMENT GUIDE

SERIES 60 (LEVEL 64)

SYSTEM MANAGEMENT GUIDE ADDENDUM A

SUBJECT

Additions and Changes to Series 60 Level 64 System Management Guide; Includes the System Patching Facility and Introduces the File Integrity Tools Available with Level 64 GCOS

SPECIAL INSTRUCTIONS

This is the first addendum to AQ09-01, dated September 1978. Insert the attached pages according to the collating instructions provided. Pages that have been changed have an "A" after the manual's reference number at the top of the page. Change bars in the margin indicate technical changes or additions.

Note:

Insert this sheet after the manual cover to show that the manual has been updated with Addendum A.

SOFTWARE SUPPORTED

Level 64 GCOS Release 0400

ORDER NUMBER

AQ09-01A

August 1979

24490
1.2979
Printed in U.S.A.

Honeywell

Collating Instructions

To update this manual, remove old pages, where necessary, and insert new pages as follows:

Remove	**Insert**
Title page, Preface	Title page, Preface
iii through viii	iii through viii
—	ix, blank
1-3, 1-4	1-3, 1-4
6-1 through 6-5	6-1 through 6-26

SERIES 60 (LEVEL 64)
SYSTEM MANAGEMENT GUIDE

SUBJECT

Describes Installation Management Aspects of Level 64 GCOS

SPECIAL INSTRUCTIONS

For users of Release 0400, this manual replaces Rev. 0 dated July 1977 which remains valid for Release 0300 users. Because of extensive revision, change bars have not been used.

SOFTWARE SUPPORTED

Level 64 GCOS Release 0400

ORDER NUMBER

AQ09, Rev. 1

September 1978

Honeywell

Preface

This manual discusses those aspects of Series 60 Level 64 General Comprehensive Operating Supervisor (GCOS) which relate to installation management.

Section I deals with the methods by which the System is implemented and updated to include corrections to the system. Section II details the configuration facility which is available for adapting the system. Section III describes the way in which an installation may tailor the supplied system to meet installation requirements. Section IV consists of concise details relating to all the system files. Section V covers the various aspects of the GCOS accounting facility. Section VI describes how the user can make best use of the file protection facilities available with Level 64 GCOS. Section VII relates to the subject of Memory Management. Section VIII describes the use of Job Classes, Scheduling Priorities and Execution Priorities. Appendix A contains examples of the various output listings and error messages derived from the GCOS configuration procedure. Appendix B discusses the $CAT-MAINT Utility.

For further information, and in some cases as prerequisite reading, the user of this manual is referred to the following GCOS Level 64 publications:

Concepts and Characteristics (Order No. AQ08)
Functional Characteristics (Order No. AQ23)
JCL User Guide (Order No. AQ11)
Data Management Utilities (Order No. AQ20)
UFAS User Guide (Order No. AQ84)
BFAS User Guide (Order No. AQ82)
HFAS User Guide (Order No. AQ83)
System Operation, Operator Guide (Order No. AQ13)
System Operation, Console Messages (Order No. AQ14)
COBOL User Guide (Order No. AQ63)
COBOL Language Reference Manual (Order No. AQ64)
Unit Record Devices User Guide (Order No. AQ59)
Installation Guide (Order No. CQ32)

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,**
2	**Initial Capital Letters,**
3	**ALL** CAPITAL LETTERS,
4	Initial Capital Letters,
5 (lowest)	ALL CAPITAL LETTERS FOLLOWED BY COLON: Text begins on the same line.

The following notation conventions 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.
$\left\{\begin{array}{l}\text{item 1}\\\text{item 2}\\\text{item 3}\end{array}\right\}$	A column of items within braces 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. Many of the manuals may be referenced in the text.

LEVEL 64 DOCUMENT LIST

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*
AQ18	*Operator Reference Manual*
AQ20	*Data Management Utilities Manual*
AQ21	*Series 200/2000 Program Mode User Guide*
AQ22	*Series 200/2000 Program Mode Operator Guide*
AQ26	*Series 100 File Translator*
AQ27	*Series 200/2000 File Translator*
AQ28	*Library Management Manual*
AQ40	*System 3 Conversion Guide*
AQ49	*Network Control Terminal Operation Manual*
AQ50	*Terminal Operations Manual*
AQ52	*Program Checkout Facility Manual*
AQ53	*Communications Processing Facility Manual*
AQ55	*TDS/64 Standard Processor Site Manual*
AQ56	*TDS/64 User Guide*
AQ57	*TDS/64 Processor Programmer Reference Manual*
AQ59	*Unit Record Devices User Guide*
AQ63	*COBOL User Guide*
AQ60	*Interactive Operation Facility*
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*
AQ72	*Series 200/2000 COBOL to Level 64 COBOL Translator*
AQ73	*IBM COBOL Translator*
AQ77	*File Translation Manual*
AQ82	*BFAS User Guide*
AQ83	*HFAS User Guide*
AQ84	*UFAS User Guide*
AQ85	*Sort/Merge Manual*
AQ86	*Catalog Management Manual*
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 Reference Card*
AQ92	*Operator's Reference Card*
AQ93	*RPG Reference Card*
AQ94	*FORTRAN Reference Card*
AQ98	*System Overview Manual*
CQ31	*Error Messages and Return Codes Manual*
CQ35	*Remote Batch Facility*

Contents

ILLUSTRATIONS

TABLES

1. System Implementation

The General Comprehensive Operating Supervisor (GCOS) software is designed to minimize the amount of user effort required for system maintenance and implementation.

GCOS is delivered to an installation on one disk volume. This disk will contain all the basic components of GCOS plus any additional components which may have been ordered specifically for the installation. Concise details of the software installation procedure are provided in the "Installation Guide".

The system manager's first task is to perform a "Save" on the contents of the supplied disk volume. This is a security copy and should be produced by use of the $VOLDUPLI utility, and where an additional copy on tape is required, by use of the $VOLSAVE utility. The supplied master disk may be used to execute these utilities.

The name of the supplied system disk should be determined by performing an Initial System Load (ISL) from this supplied system disk.

Assuming the volume name of the supplied master is CHBSYS then the initial back-up job might be as follows :

```
$ JOB FJ,USER=OPMAN,PROJECT=SYSMAINT;
  VOLPREP OLD=(DEVCLASS=MT/T9,MEDIA=NONAME,
               LABEL=NONE),
          NEW=(MEDIA=OMRX);
  VOLSAVE VOLUME=(DEVCLASS=MS/400,MEDIA=CHBSYS),
          OUTFILE=(OMSS,GCOS,DEVCLASS=MT/T9,
                   MEDIA=OMRX);
$ ENDJOB;
```

In this instance the user first prepares a tape volume OMRX (not necessary if a standard labeled volume is available) and then saves the contents of the master disk as tape file OMSS.GCOS on OMRX.

THE CONFIGURATION AND TAILORING OPTIONS

Once the contents of the supplied master disk have been saved, GCOS may be used directly from this supplied disk without any further action. System Configuration and Tailoring are completely optional.

These Configuration and Tailoring procedures are primarily intended to be used as an aid to system software installation. Therefore, minimal use after initial installation should be anticipated.

However, careful planning by site management with consideration of the long term effects of the various options can have a significant impact on the overall efficiency of the system's operation. This impact can be both advantageous and disadvantageous, i.e. the introduction at configuration time of the optional facility $ACCOUNT will have obvious and attractive advantages, but the increase in memory cost to an already heavily loaded site could prove to be too expensive in terms of efficient system throughput.

Similarly, the facilities provided by the tailoring procedure require careful consideration. Efficient use of the tailoring facility can result in the minimum amount of disk space being required by the system files. Optional files are the more obvious areas for possible disk space saving, with the actual size of the selected files being a further important point for consideration. If the files selected are not located (System, resident or non-Resident disks) carefully, any possible saving in disk space could be heavily outweighed by the cost in the wastage of system and operator time.

The system files, which are created by GCOS for its own use, are directed to specific disc volumes by the file allocation procedures. One of the primary considerations when creating these files is to achieve an equitable balance of files across all available devices. Careful file balancing can improve the overall system efficiency by avoiding access conflicts (i.e. where two or more jobs compete for the use of a device).

Full working details of the Configuration and Tailoring procedures are given in sections II and III of this manual.

SELECTION OF OPTIONAL SYSTEM FACILITIES

As previously mentioned, the accounting feature of GCOS is an optional facility. Others include the Journal file recovery mechanism (see section VI of this manual) and the Catalog facilities feature (see section V and appendix B of this manual). Choice of use and any installation-imposed limitations (options and default values) of such features are the system manager's responsibility.

The cost in terms of additional memory requirements, any increase in operator work and the overall effect on system throughput should be considered carefully when adopting (and adapting) such facilities.

Details of the various options relating to system facilities are given in sections II and III of this manual.

USER FILES ON SYSTEM VOLUME

In transfering from one software release to another it may be necessary to transfer user files from the old system to the new.

In this case, system transition consists of the following steps :

1. Save all user files (and possibly SYS.URCINIT as well) on the old system disk using the utility $FILSAVE.
2. Build a new system disk (TAILOR, see section III of this manual).
3. Restore files to the new system disk using $FILREST (preceeded by $PREALLOC).

Warning :
Note that TAILOR first destroys (by $VOLDUPLI) all files on the output volume. It is not possible to update a system disk to a new system while retaining files on the volume.

UNIT RECORD CONTROL INFORMATION

When a new version of GCOS is installed, the user will normally have to reload any user-defined Unit Record Control (URC) definitions from the old system to the new. These user-defined elements are placed in the system file SYS.URCINIT (see Section IV, Description of System Files).

A utility program named URINIT controls storage within the SYS.URCINIT file of such parameters as character sets, belt segment images, forms, vertical format units, selection tables, etc., which are all needed by the Unit Record Drivers to initialize the Unit Record Controller.

For a new site, the release 0400 files supplied with the system should be sufficient to initiate the installation tasks and, in many cases, to start operation. If the site has only a card reader and/or printer as unit record devices, the minimum SYS.URCINIT file as contained on the release 0400 disk will be sufficient. If there is card punch, paper tape equipment, and/or a document handling unit, the UREXT file should be used to obtain the required parameters; this is done by using the editor to choose the generation commands applicable to the site, then running the URINIT utility program with these commands as input.

The URINIT utility performs a control function which enables protection of device independence. This independence allows more than one item to be associated with the same device name and is achieved through the alphabetical prefixing of the items' names.

The following list gives these specific alphabetical characters and their respective device names:

K	For Printer	Character Set.
B	For Printer	Belt.
F	For Printer	V.F.U.
P	For Printer	Print Test.
R	For Card Reader	Character Set.
U	For Card Punch	Character Set
E	For Cassette	Translation Table.

Additional device-independence facilities are also provided by subfiles containing the Specifics Tables of a given device:

CATALDIR subfile for Table of Contents of SYS.URCINIT file.
PRINTEAL subfile for the Printer.
TAPCARK7 subfile for the Card Reader.
TAPCARK7 subfile for the Card Punch.
TAPCARK7 subfile for the Cassette.

The internal structure of the SYS.URCINIT file has been modified for Release 0400, as has the URINIT utility language: the file and the utility language are therefore incompatible with the file language as used in release 0300. The minimum SYS. URCINT file can be used for system installation; if this file is sufficient for user needs, operation can then begin immediately, with no problems, If, however, the user wishes to retain the parameters of the SYS.URCINT file of release 0300 format the TURF utility must be used to translate them to 0400 format. Note that no output must be queued at the time TURF is run, and that a RESET RELOAD must be performed following TURF completion.

FIRST LOADING OF THE NEW SYSTEM

Once a new system disk is installed, the system manager may wish to perform the following actions :

1. Pre-initialize load modules by means of the PLM command; selected load modules (compilers, utilities, user applications) are permanently loaded into backing store.
   This pre-initialization should be done for those load modules that are frequently used (such as the COBOL compiler) and especially for those frequently used in more than one job stream concurrently.
   In the case of concurrent usage, the advantages of pre-initialization are twofold :
   - Avoidance of repeated loading of the module into the backing store.
   - Only one copy of code and constant data segments is present, instead of multiple copies.
2. Load shared modules; these modules (such as HIDS and BTNS) must be placed in SYS.BKST1 prior to their use. This loading is performed via the LOAD command of $LIBMAINT SM utility. Note the result of this loading may be verified by the operator command DLM SM.
3. Implement the "default" installation values for :
   - Multiprogramming Level.
   - Multiprogramming Level, Execution Priority and Scheduling Priority for each class. For further information on these see $MULTILEV statement, see Section II of this manual.
4. Define any RESIDENT volumes for use with the system disk; this is done in reply to IL01.
5. Reply RESTORE with volume-name to message IL01; this will establish, for future occasions, the name of the volume containing the file SYS.SYSTEM.
6. Establish dump options, using the DUMP-CD parameter in the IL01 reply.

For further details of the SYS.BKST1, SYS.SYSTEM and SYS.HSMLIB files refer to Section IV of this manual.

2. System Configuration

When a system is initialy built (system generation), it is of a standard format and is not dedicated to any particular hardware configuration or specific software requirement. Therefore, the running operating system should ideally be adapted to its environment, conform to site specific devices and by possible changes to the supplied default values, permit optimum installation throughput.

To adapt the standard Level 64 GCOS operating system to suit a specific environment the system configuration facility (CONFIG) can be used. This is performed on site and is designed as a quick and easy to use "tool" activated like any other standard dependant software component. The hardware configuration is not specified through any statement of configuration, this is performed automatically based on the System Resources Status Table (SRST), at each execution of CONFIG. Certain features of the supplied system are optional thus constituting some additional overhead to the site. If a particular feature, or part of one is not required, it may be adopted or deleted through use of CONFIG. The Accounting feature is one such example; it provides information, both at step and job level and on exception conditions etc. The Accounting feature, if required, and the chosen record type can be adapted by CONFIG.

In the event of a running site being supplied with a new release of system software, standard supplied default values may not suit the installation requirements; however the facitities offered may be suitable for the chosen application. By use of CONFIG the new software release can then be configurated to meet the specific requirements. For example in release 0300 there were 6 Output classes with varying default priorities; whereas under release 0400, 26 output classes exist. It may be advantageous to use the 26 output classes but with changes in the provided default priorities.

The following figure details the input requirements of CONFIG and the corresponding outputs of the completed run.

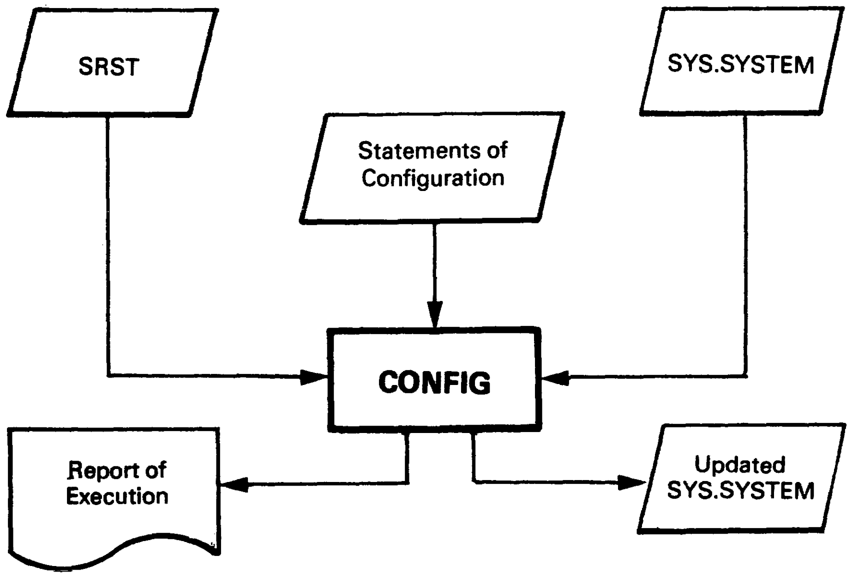


Figure 2-1. CONFIG Throughput.

The input requirements of the CONFIG component are as follows :

. The SRST of the running (existing) operating system.

. A SYS.SYSTEM file which was the output from system generation. However, it need not be the SYS.SYSTEM file which generated the backing store of the current operating system.

. A sequential file of statements of configuration.

The result of a successful execution of CONFIG is an updated SYS.SYSTEM file or as it should be retermed "A Configurated SYS.SYSTEM File".

If CONFIG is run on several occassions on the same SYS.SYSTEM file, changes will only be made in respect of the original (supplied) SYS.SYSTEM file. Consequently a second running of CONFIG with only one change specified will have the effect of cancelling all previous configuration changes to all areas of the system file, thus reverting to the supplied file format, with only the one, as specified, change being made. The changes made are against a constant reference, the status of the SYS.SYSTEM file as it was originally delivered to the site. This feature offers a fixed reference point for each execution of CONFIG; thus enabling trouble-free changes to be made, until the required (optimum) configuration is reached.

Once the SYS.SYSTEM file has been sucessfully adopted by use of CONFIG, the re-configurated system may be launched by use of the following ISL options :

– RESTORE	to restore the backing store from the configurated SYS.SYSTEM file.
– RESTART (COLD)	the type of restart specified must always be "cold".

If a RESTORE option is not issued, no change will be made to the backing store and the status of the running system after ISL will remain the same.

If an incorrect RESTART is issued, the following message is output at the operator console :

RESTART (COLD) IS MANDATORY, YES OR NO ?

If the answer given is "YES" then the restore option will be obeyed and a COLD restart performed. If the answer is "NO" then no RESTORE will be performed.

In general any incident, such as a system crash, during the execution of CONFIG will not affect the SYS.SYSTEM file upon which CONFIG is being run. This is because the SYS.SYSTEM file is either of the "OLD" or "NEW" status. If the old file is corrupted the new file will be unaffected, and if the new file is corrupted the old file will remain unaffected.

However, if an abort of CONFIG occurs, it is possible for the SYS.SYSTEM file to be left in a non-defined state (neither "OLD" nor "NEW"). A further run of CONFIG will correct this situation. If a re-run is not performed and an ISL RESTORE is attempted, (using a SYS.SYSTEM file of an undefined state), the operator will be warned that CONFIG execution could not be satisfactorily completed, and asked if a restore is required.

Console message : CONFIGURATION PARAMETERS LOST
DO YOU WANT TO RESTORE ?

If the answer is "NO" then no restore is performed. If the answer is "YES" a restore will be performed which will revert the backing store back to its state as at system generation time.
CONFIG must then be re-run

Note :
Preinitialized load modules are not valid from one run of CONFIG to another. When a configurated SYS.SYSTEM file is introduced all preinitialized load modules should be cancelled (CLM) and then re-preinitialized (PLM).

Examples of CONFIG output listings and a full list of error messages, with definitions, are given in Appendix A of this manual.

THE $CONFIG JCL STATEMENT

Function :
Enables the user to adapt SYS.SYSTEM file to suit specific needs.

Format :

```
CONFIG [ { RESIDENT                      } ]
       [ { DEVCLASS = device-class       } ]
       [ {         MEDIA = volume-name   } ]
   [, COMFILE = { *input-enclosure-name        } ]
   [            { sequential-file-description  } ]
   [, DUMP = { NO   } ]
   [         { DATA } ]
     [, PRTFILE = (print-file-description)]
     [, PRTOUT = (SYSOUT-parameters)]
     [, PRTDEF = (DEFINE-parameters)]  ;
```

Parameter Description

RESIDENT	The SYS.SYSTEM file of the RESIDENT volume will be chosen by default if no DEVCLASS and MEDIA parameters are given.
DEVCLASS	The device-class of the volume supporting the SYS.SYSTEM file to be configurated. This parameter must be specified in conjunction with the media parameter.
MEDIA	The name of the disk volume supporting the SYS.SYSTEM file to be configurated. This parameter must be specified in conjunction with the DEVCLASS parameter.
COMFILE	Command file parameter which must be given in either Standard System Format (SSF) or Standard Access Record Format (SARF).
DUMP	Standard dump parameter, the default value being NO dump.
PRTFILE	The PRTFILE parameter specifies the sequential file to which the output listing will be sent; the default is the standard SYSOUT subfile.
PRTOUT	The PRTOUT parameter is used to override and change the standard ouput parameters. Any parameter of the $SYSOUT JCL statement can be specified.
PRTDEF	The PRTDEF parameter is used to override and change the file-definition parameters used for the output file. Any parameter of PRINTER parameter group of the $DEFINE JCL statement can be specified.

Examples Of $CONFIG

```
CONFIG MEDIA = INST,DEVCLASS = MS/M400,COMFILE = *KART;
$INPUT KART;

$ENDINPUT;
```

From the above example the installation system disk (MEDIA = INST), which supports the SYS.SYSTEM file to be configurated, is located on device-class MS/M400 (DEVCLASS = MS/M400) and configurated by the input enclosure KART ($INPUT KART).

```
CONFIG COMFILE=(MY_LIB SUBFILE=MY_CONFIG,
DEVCLASS=MS/M400,MEDIA=MY_DK);
```

From the above example the RESIDENT system disk (chosen by default) supports the SYS.SYSTEM file to be configurated. The statements of configuration are located within a subfile named MY_CONFIG (SUBFILE = MY_CONFIG) of the library MY_LIB. This library is supported by a volume named MY_DK (MEDIA = MY-DK) located on device-class MS/M400 (DEVCLASS = MS/M400)

Conventions Of The CONFIGURATION Statements

1. Mandatory "$" sign before names of all configuration statements.
2. No space allowed between "$" sign and the name of a statement.
3. The end of a record acts as a delimitor.
4. Standard facilities of JCL are adopted in the following areas :
   - free format
   - no order imposed for keyword parameters
   - the optional number of blanks allowed before and after a separator
   - the use of positional parameters, and/or keyword parameters
   - the comma not a mandatory seperator between the various parameters of a statement

THE $ACCOUNT STATEMENT

Function :
To establish if the accounting facility is required, and to specify what level of detail is expected. This statement can only be modified by a further run of CONFIG.

Format :

```
$ACCOUNT ( ALL                                        )
         ) NO                                         (
         ( [STEP] [,JOB] [,EXCEPT] [,END] [,USER]     ) ;
```

If no $ACCOUNT statement is given, $ACCOUNT ALL will be assumed by default. If neither "ALL" nor "NO" is specified then at least one other parameter (STEP, JOB, EXCEPT, END or USER) must be given. Where more than one parameter is given, each parameter must be seperated by a comma, however the keyword string must not be seperated form $ACCOUNT by a comma.

Parameter Description

ALL	The ALL parameter is used to specify that every type of accounting record is required to be written to the accounting file. It also allows "USER" information to be written to the accounting file.
NO	The NO parameter is used to specify that accounting facilities are not required.
STEP	The STEP parameter specifies that step records are written to the accounting file.
JOB	The JOB parameter specifies that job records are written to the accounting file.
EXCEPT	The EXCEPT parameter specifies that exception records are written to the accounting file.
END	The END parameter specifies that a record is written to the accounting file at each shutdown.
USER	The USER parameter enables any user specific accounting information to be written to the accounting file.

Examples Of The $ACCOUNT Statement

$ACCOUNT ALL;

and

$ACCOUNT STEP,JOB,EXCEPT,END,USER;

These are equivalent statements which specify that all accounting information, including user specific information, are to be written to the accounting file.

$ACCOUNT NO;

This statement specifies that no accounting information is required.

User Guide For The $ACCOUNT Statement Memory Requirements

The parameters of this statement allow a choice of the accounting information to be written to the accounting file. Where ALL is specified, the working set memory cost at step level is about 5K (at step termination) and about 5K at job termination. The use of the USER,EXCEPT and END parameters will cost a further 5K of memory at the time when the information is written to the accounting file.

THE $ACTSIZE STATEMENT

Function :
To establish the size of the VMM (Virtual Memory Management) accounting file. This size can only be modified by a further run of CONFIG.

Format :

$ACTSIZE size;

The size expressed in this statement is given in units of allocation of backing store.The size of one unit of allocation is 1K. The system default value is 200, and the maximum value is 2000.

Example Of The $ACTSIZE Statement

$ACTSIZE 100;

This statement specifies that provision is made for 100 units of allocation for accounting information.

User Guide For $ACTSIZE Statement Memory Requirements

The accounting mechanism is based on two files, if $ACTSIZE n is specified, the cost in memory will be 2*n units of backing store. For further details see the Job Accounting Facilities section of this manual.

THE $BANNER STATEMENT

Function :
To define the default value of the number of heading banners given by Output-Writer. To override this system default value use the [N] BANNER/-BANINF options of $OUTVAL, $SYSOUT and $WRITER JCL statements.

Format :

$$\text{\$BANNER} \begin{Bmatrix} 0 \\ 1 \\ 2 \end{Bmatrix} ;$$

If any value other than 0, 1 or 2 is specified, 2 will still be assumed. The options available are : no banner (0), 1 page of banner, or 2 pages of banner, the system default value is 2 pages of banner.

User Guide For $BANNER Stationery Usage

Use of this statement depends largely on the user application. Careful consideration should be given to usage where expensive pre-printed stationery is used. Where the stationery is of a standard nature and cost per page not so inportant, consideration might be given to possible ease of use gained from larger banners which could enable quicker separation of printout and clearer identification for distribution.

THE $DEVTRACE STATEMENT

Function :
To set a default option for device trace as output at operator console. These default options may be re-set by the NOT, SDT and TDT operator commands.

Format :

$DEVTRACE [ABN] [,ATN] [,WARN] [,ALARM] ;

If no $DEVTRACE statement is specified through CONFIG, there will be no trace of devices given at the operator console. When a $DEVTRACE statement is made it must include at least one of the options, there is no default option for this statement.

Parameter Description

ABN	The ABN parameter is used to enable the display of all "abnormal" events, from SYS.ERLOG, at the operator console.
ATN	The ATN parameter is used to enable the display of all "attention" messages, from SYS.ERLOG, at the operator console.
WARN	The WARN parameter is used to enable the display of all "warning" messages, from SYS.ERLOG, at the operator console.
ALARM	The ALARM parameter enables the display of an alarm message at the operator console. This message occurs at preset intervals within a group of abnormal events. The operators attention is drawn to the fact that there is a continuing abnormal error status. Presetting of the ALARM message occurrence is performed through the operator START and TRACE commands. For further details see System Operator Operation Guide.

Example Of The $DEVTRACE Statement

$DEVTRACE WARN ;

This statement enables all "warning" messages to be displayed at the operator console.

THE $FILESHARE STATEMENT

Function :
To set the maximum number of jobs which can share the same file. This figure can be modified only by a further run of CONFIG.

Format :

$FILSHARE total-number-of-jobs ;

The total-number-of-jobs parameter specifies the maximum number of jobs which can share any one file concurrently (permissible range 2 – 32). The default value of this figure is 5. If the system assigns the file for system usage this will count as one job. If the same file is assigned more than once within the same step it will still count as one job.

Example Of The $FILSHARE Statement

$FILSHARE 3 ;

This statement specifies that not more than 3 jobs may access any one file at the same time.

THE $JOBCLASS STATEMENT

Function :

To set the system job management default values for job classes. A job class value may be modified by the MC, TC and SC operator commands. It is possible to override some default values by use of the PRIORITY parameter of the $JOB JCL statement, and by use of the XPRTY parameter of the $STEP JCL statement. For full details of the provided system job class default values, see the summary of default values and statements of this Section.

```
$JOBCLASS  job-class
           [,XPRTY = execution-priority]
           [,PRIORITY = scheduling-priority]
           [,MAXLOAD = maximum-class-load]
           [ { STARTED  } ]
           [,{ NSTARTED } ]
           [,NSC] [,NMAXPRTY]
           [,NMPRIO] [,NMLOAD] ;
```

To enable a valid $JOBCLASS CONFIG statement to be made at least one of the above optional parameters must be specified.

Parameter Description

job-class	The "job-class" parameter defines the job class to which the $JOBCLASS statement applies. This parameter can be any one from the job class range A-Z.
XPRTY	The XPRTY parameter defines the execution priority of the job class (also known as the despatching priority). This parameter may be changed by the XPRTY keyword parameter of the $STEP JCL statement. 0 is the highest priority and 9 the lowest.
PRIORITY	The PRIORITY parameter defines the scheduling priority of the job class. This parameter may be changed by the PRIORITY keyword parameter of the $JOB JCL statement. 0 is the highest priority and 7 the lowest.
MAXLOAD	The MAXLOAD parameter defines the maximum number of jobs within the specified class that can be executing simultaneously. The figure chosen should be equal to or less than the figure given by the first parameter of the CONFIG statement $MULTLEV
STARTED NSTARTED	The STARTED parameter starts (by default) the specified job class and the NSTARTED parameter inhibits the start of the job class. The NSTARTED option must not be specified when the NSC parameter is to be used for the same job class.
NSC	When the NSC parameter is specified, use of the SC and TC operator commands, to affect the related job class, is inhibited. When this parameter is not specified the operator is able to use the SC and TC commands to modify (STARTED and NSTARTED) the specified job class. The NSC parameter must not be specified when the NSTARTED option has been selected for the same job class.
NMXPRTY	The NMXPRTY parameter, when used, inhibits operator use of the MC command to modify the execution priority of the given job class.
NMPRIO	The NMPRIO parameter, when used, inhibits operator use of the MC command to modify the scheduling priority of the given job class.
NMLOAD	The NMLOAD parameter, when used, inhibits operator use of the MC command to modify the multiprogramming limit of the given job class.

Examples Of The $JOBCLASS Statement

$JOBCLASS P,XPRTY = 9,PRIORITY = 7,MAXLOAD = 3 ;

The job-class is P, with a dispatching priority of 9, a scheduling priority of 7 and a multiprogramming limit of 3. The class is started by default, and the operator has full use of the TC, SC and MC commands on this class of job.

$JOBCLASS H,XPRTY = 9,PRIORITY = 7,MAXLOAD = 0 ;

Although this class is started by default, no job of class H will be started because the multiprogramming limit is set at zero. To start jobs of class H either the FJ or MC operator command would need to be used.

Example $JOBCLASS Q Statement

$JOBCLASS Q MAXLOAD = 12, NSTARTED ;

The job class is Q with a limit of 12 (multiprogramming) IOF users being allowed to log-on. The operator is required to issue the SC Q command to start the Q job class and has the power, by use of the TC command, to stop other users from logging on.

THE $JORSIZE STATEMENT

Function :
To state the maximum number of user generated lines on the Job Occurrence Report. This figure can only be modified by a further run of CONFIG.

Format :

$JORSIZE max-number-of-lines ;

Parameter Description

max-number-of-lines

The maximum number of lines parameter sets the permissible maximum number of user written lines within the same step to be output on the JOR. If the information written exceeds the $JORSIZE statement, then no further information will be output on the JOR. The system default value for this parameter is 500 lines, the system permissible maximum is set at 2,000 lines.

Example Of The $JORSIZE Statement

$JORSIZE 50 ;

This $JORSIZE statement parameter specifies that a maximum of 50 lines per step of user generated JOR are permitted.

THE $MAXFILE STATEMENT

Function :
To set the maximum number of active files which at any point in time are within the system. The SITE.CATALOG, SYS.IN, SYS.OUT and SYS.URCINIT files are not included in this evaluation. Active files being either opened, or assigned (not opened), or not assigned but passed or those which are an attached user catalog. This figure can only be modified by a further run of CONFIG.

Format :

$MAXFILE max-number-of-files ;

The system default value for this parameter is set at 55, the permissable range is from 10 to 200.

Parameter Description

The $MAXFILE parameter should be greater than the "max-number-of-started-jobs" as given in the $MULTLEV statement.

Example Of The $MAXFILE Statement

$MAXFILE 15 ;

The maximum number of active files is set at 15.

THE $MAXJOB STATEMENT

Function :
To set the maximum number of known jobs in the system. Known jobs being those which are introduced, scheduled, executing or waiting for output. This figure can only be modified by a further run of CONFIG.

Format :

$MAXJOB max-number-of-jobs :

The system default value for this parameter is 200. The average cost of backing-store is about 3.5K per 30 Known jobs. Therefore if a reduction of the internal tables can be made, to match the true load, then a worthwhile advantage is gained. The system permissible maximum for the number of known jobs is 9999.

Example Of The $MAXJOB Statement

$MAXJOB 32 ;

The maximum number of known jobs is set at 32.

THE $MAXTAPE STATEMENT

Function :
To set the maximum number of known tapes (or cassettes) in the system at any point of time. A known tape being either assigned or passed. This figure can only be modified by a further run of CONFIG.

Format :
$MAXTAPE max-number-of-tapes ;The system default for this parameter is 64, with the range of 1-64.

Parameter Description

The maximum number of tapes parameter sets the permissable maximum number of known tapes (or cassettes) in the system at any point in time.

Example Of The $MAXTAPE Statement

$MAXTAPE 12 ;

The maximum number of known tapes is set at 12.

THE $MAXTASK STATEMENT

Function :
To set the maximum number of user tasks (user proccesses) that can simultaneously execute. This figure can only be modified by a further run of CONFIG.

Format :
$MAXTASK max-number-of-user-tasks;

The system default value is set at 50, and the system permissible maximum is set at 150.

Parameter Description

The max-number-of-user-tasks parameter defines the total user processes that may be simultaneously executing. It should be noted that the value for TDS user processes is defined at TDS generation time by the TASK parameter. One Program mode job (emulator) should be counted as two processes and one user step as one process. Little or no advantage will be gained from specifying a figure which is 15 (or greater) above that specified for the "max-number-of-started-jobs" parameter of the $MULTLEV statement.

Example Of The $MAXTASK Statement

$MAXTASK 40 ;

The maximum number of running user tasks may be 40.

THE $MULTLEV STATEMENT

Function :
To set the maximum number of : (a) started jobs including service jobs, (b) batch jobs, excluding service jobs, (c) IOF users. The values (a) and (c) can only be modified by a further run of CONFIG. The value (b) may be modified by the MS operator command.

Format :

```
$MULTLEV  max-number-of-started-jobs
          , BATCH = max-number-of-batch-jobs
          , INTERACT = max-number-of-IOF-users ;
```

The system default values for these parameters are :

```
$MULTLEV   = 14,BATCH = 5,INTERACT = 10 ;
```

The values of BATCH and INTERACT, when modified, must still be equal to or less than the values of MULTLEV. The value of MULTLEV, when modified, must be equal to, or greater than the MAXLOAD parameter of the $JOBCLASS CONFIG statement. The value of MULTLEV, when modified, must also be less than the MAXFILE parameter of the $MAXFILE CONFIG statement.

Parameter Description

max-number-of-started-jobs
The maximum number of started jobs parameter defines the total permissible number of simultaneously started jobs within the system, i.e. service jobs and user jobs.

BATCH	The BATCH parameter defines the maximum number of batch user jobs which may be simultaneously started, (excluding service jobs).
INTERACT	The INTERACT parameter defines the maximum number of IOF users.

Example Of The $MULTLEV Statement

```
$MULTLEV 15,BATCH = 6,INTERACT = 2 ;
```

The number of jobs that may be simultaneously started is 15. Batch jobs are limited to 6 and the interactive IOF users is 2.

```
$MULTLEV 12,BATCH = 10,INTERACT = 10 ;
```

The number of jobs that may be simultaneously started is 12. Batch jobs are limited to 10 and interactive IOF is also 10.

```
$MULTLEV 7 ;
```

This statement would result in an error because the default value for INTERACT (and BATCH) must always be less than the figure given for the maximum number of started jobs.

THE $OWCLASS STATEMENT

Function :
To define the default selection priority attatched to the output classes of the output Writer. This figure may be modified by the MO/MOC operator command. This figure may also be modified by use of the PRIORITY parameter of the OUTVAL, SYSOUT and WRITER JCL statements.

Format :

$OWCLASS output-class-to-be-modified

$$,\text{PRIORITY} = \left\{ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \right\} \; ;$$

The system default value for any given output class priority is as follows :

Output Class	Default Priority
A	1
B	2
C	3
D	4
E	5
F	6
G-Z	7

Parameter Description

output-class-to-be-modified
The output class to be modified parameter defines the output class whose default selection priority is to be modified.

PRIORITY The PRIORITY parameter defines the prority attached to the corresponding output class.

Example Of The $OWCLASS Statement

```
$OWCLASS A, PRIORITY=2 ;
$OWCLASS C, PRIORITY=2 ;
```

The above statements have modified both output classes A and C to priority 2.

THE $OWDEVICE STATEMENT

Function :
To define output classes activated by the Output Writer when no indication of class type is given in the SO operator command.

Format :

```
$OWDEVICE device-rank-identifier
          ,DVID = device-identifier
          ,CLASS = alphabetical-identifier ;
```

The system default value for any given output class is as follows :

The device-rank identifier is of the range 1 — 15, DVID is SRST dependent and CLASS = the full range A — Z.

Parameter Description

device-rank-identifier

The first parameter value "device-rank-identifier" has no physical meaning, it is used as a rank identifier. There can be no space (unused digit) in the ranks used i.e. if rank 6 is used then ranks 1 – 5 and 7 – 15 must also be used. When two statements have the same rank identifier, the statements must refer to the same device.

DVID The DVID parameter identifies an output device i.e. PR01, PR02, CD01, CD02 etc. When a SO operator command is issued, the output classes specified by the CLASS parameter are automatically started on the given device. The device names used must be those registered in the System Resources Status Table.

CLASS The CLASS parameter specifies the output classes which are started by default.

Example Of The $OWDEVICE Statement

The following examples assume that the output devices PR01, PR02, CD01 and CD02 are known system output devices.

EXAMPLE 1 Where no $OWDEVICE statement is made, the system supplied default values are assumed. Therefore, when the operator issues the SO command on devices PR01, PR02, CD01, and CD02, the full range of output classes (A – Z) is started on each of the devices.

EXAMPLE 2 $OWDEVICE 1,DVID = PR01,CLASS = ABCDEF ;
$OWDEVICE 2,DVID = PR02,
CLASS = GHIJKLMNOPQRSTUVWXYZ ;

From the above $OWDEVICE statements the following will occur on the operator specified devices :

Operator command	Output class Started
SO PR01	A – F
SO PR02	G – Z
SO CD01	A – Z
SO CD02	A – Z

EXAMPLE 3 $OWDEVICE 1,DVID = PR01,CLASS = ABCDEFGHIJKL ;
$OWDEVICE 2,DVID = PR02,CLASS = IJKLMNOP ;
$OWDEVICE 3,DVID = CD01,CLASS = INITIALIZATION ;

From the above $OWDEVICE statement the following actions will occur on the operator specified devices.

Operator command	Output class Started
SO PR01	A – L
SO PR02	I – P
SO CD01	A,I,L,N,O,T,Z.
SO CD02	A – Z

EXAMPLE 4 $OWDEVICE 1,DVID = PR01,CLASS = ABCD ;
$OWDEVICE 1,DVID = PR01,CLASS = EFGH ;

The second $OWDEVICE statement will override the first as both statements are for the same device (PR01). The output classes started on device PR01 will be EFGH.

EXAMPLE 5 $OWDEVICE 1,DVID = PR01,CLASS = ABCD ;
$OWDEVICE 1,DVID = PR02,CLASS = EFGH ;

The above $OWDEVICE statements show an error. The two statements are of the same rank but refer to different devices. Similarly two devices of the same type could not be refered to with different ranks.

User Guide For The $OWDEVICE Statement

As previously stated, the function of the $OWDEVICE CONFIG statement is to define the output classes activated by the Output Writer when no indication of class type is given by the SO operator command. However, this statement can also be used to perform a priority output control function where only one printer is available, where two printers, and more than one stationery format are used, to perform a stationery format streaming function.

EXAMPLE OF PRIORITY OUTPUT CONTROL

Requirement :
Maximum priority output always within classes A – F, low priority output always within classes G – Z.
Assuming the installation has only one printer, PR01, the following statement will be necessary to ensure the required priorities :

$OWDEVICE 1,DVID = PR01,CLASS = ABCDEF ;

When the operator issues the SO PR01 command only the output classes of maximum priority (A – F) will be printed. Upon completion of all maximum priority output, the operator may then issue the SO PR01 G – Z command to receive all other output.
Note, this example only shows two priority divisions, the actual number may be set to meet the user specific requirements.

EXAMPLE OF STATIONERY FORMAT STREAMING

Assuming the installation has two printers PR01 and PR02, and the stationery mounted on these two is of two different types, the following statements could be given :

$OWDEVICE 1,DVID = PR01,CLASS = ABCDEF ;

$OWDEVICE 2,DVID = PR02,
CLASS = GHIJKLMNOPQRSTUVWXYZ ;

In response to the first statement, output classes A – F will be printed on paper type one at output device type PR01. In response to the second statement, output classes GHIJKLMNOPQRSTUVWXYZ will be printed on paper type two at output device type PR02.

THE $OWDFLT STATEMENT

Function :
To define the system default values for the Output Writer. These default values are observed when processing the OUTVAL,SYSOUT and WRITER JCL statements.

Format :

$$\text{\$OWDFLT} \left[\text{CLASS} = \left\{ \begin{matrix} \text{default-value} \\ \underline{\text{C}} \end{matrix} \right\} \right]$$
$$\left[\text{,DEVCLASS} = \left\{ \begin{matrix} \text{device-class-name} \\ \underline{\text{PR/H132}} \end{matrix} \right\} \right]$$
$$\left[\text{,MEDIA} = \left\{ \begin{matrix} \text{media-name} \\ \underline{\text{110000}} \end{matrix} \right\} \right]$$
$$\left[\text{,TAPE} = \left\{ \begin{matrix} \underline{\text{SYSOUT}} \\ \text{NSYSOUT} \end{matrix} \right\} \right] ;$$

Parameter Description

CLASS	The CLASS parameter defines the output value of OUTVAL, SYSOUT and WRITER JCL statements. The default value for this parameter is C.
DEVCLASS	The DEVCLASS parameter specifies the output device for Output Writer output. The default value for this parameter is PR/H132.
MEDIA	The MEDIA parameter specifies the default print belt, character set and paper identification for the printer as specified by the DEVCLASS parameter.
TAPE	When the SYSOUT option (default value) of the tape parameter is selected, the Output Writer is able to override certain user file definition parameters. The NSYSOUT option ensures compatability between releases 1C and 1D when the standard supplied utilities are not being used to read SYSOUT tapes.

Examples Of The $OWDFLT Statement

EXAMPLE 1 $OWDFLT ;

The default class for deliveries is CLASS = C, the default output device for the system is the PR/H132 printer with belt : character set/paper : I10000. If the output writer has to create a SYSOUT tape, it is allowed to override the user RECFORM and BLKSIZE parameters.

EXAMPLE 2 $OWDFLT CLASS = D, DEVCLASS = PR/H71,MEDIA = I3050 ;

The default class for deliveries is CLASS = D, the default output device for the system is a PR/H71 printer with I3 character set and paper type 050. If either the belt of paper type are wrong, i.e. they are not entered in the SYS.URCINIT file, then I10000 will be assumed.

THE $PRLOG STATEMENT

Function :
To set the threshold value at which the operator is warned to print the SYS.ERLOG. This figure can only be modified by a further run of CONFIG.

Format :

```
$PRLOG [THRESHOLD = specified-percentage]
         [,COMMAND = "operator-command"] ;
```

Parameter Description

specified-percentage
The specified percentage parameter defines the point (threshold) at which the operator is notified of the need to print SYS.ERLOG. This threshold point is expressed in percentage terms, e.g. when the SYS.ERLOG is 80% full, notification will commence. The system default value is 50%.

operator-command
The operator command parameter specifies a valid operator command which should be obeyed as soon as the threshold figure is reached. If this parameter is not specified, the system default action is to request the operator to "RUN PRLOG".

Example Of The $PRLOG Statement

$PRLOG THRESHOLD = 90 ;

This statement specifies that the point of notification to print SYS.ERLOG is when SYS.ERLOG is 90% full. By default the operator is then requested to "RUN PRLOG".

THE $ROFCLAS STATEMENT

Function :
To set the default class for a job introduced in GCOS64 using ROF. This default class can only be modified by a further run of CONFIG. The ROF user may override this default class by using the CLASS parameter in his $JOB JCL statement.

Format :

$ROFCLAS default-class ;

Parameter Description

default-class	The default class parameter specifies one of the job classes from the range A – P. The system default value is class "P".

Example Of The $ROFCLASS Statement

$ROFCLASS B ;

This statement specifies that the default job class for a job introduced using ROF is class "B".

THE $STATION STATEMENT

Function :
To define the remote station names and the corresponding protocols to be supported by the Output Writer. These values can only be changed by a further run of CONFIG.

Format :

$STATION station-rank, name + identity, PROTOCOL = $\left\{ \begin{matrix} \underline{61} \\ \text{MVIP} \end{matrix} \right\}$;

Parameter Description

station-rank	The station rank parameter defines a rank, of the range 1 – 6, which would enable a station name to be overridden when there is more than one $STATION statement for the same station. Rank numbers must start form 1 and there can be no space (unused digit) in the ranks used i.e. if rank 6 is used then ranks 1 – 5 must also be used.
NAME	The NAME parameter defines the name of the remote station. This name may comprise up to 4 characters in length, the first two characters must be alphabetical but not PR or CD.
PROTOCOL	The PROTOCOL parameter defines which protocol is to be used by the system for dialogue with the station. This protocol may be either the 61 or MVIP type. The system default value for this parameter is the 61 protocol.

Examples Of the $STATION Statement

```
$STATION 1, LYON ;
$STATION 2, XPR ;
$STATION 3, BCO ;
```

This set of statements defines three stations whose names are "LYON", "XPR" and "BCO". Therefore, any other station-name used in an Output Writer JCL statement will be rejected.

```
$STATION 3, PC09, PROTOCOL = 61 ;
$STATION 3, AG12, PROTOCOL = MVIP ;
```

The second statement overrides the first because it applies to the same station, i.e. station 3.

THE $STEPFILE STATEMENT

Function :
To establish the maximum number of file assignments made during a step. This figure can only be modified by a further run of CONFIG.

Format :

$STEPFILE $\left\{\begin{matrix}\underline{42}\\55\end{matrix}\right\}$;

Parameter Description

The parameter 42/55 defines the maximum number of internal-file-names which may be assigned during a step. The figure chosen can only be 42 or 55, with 42 being the system default value.

Examples Of The $STEPFILE Statement

$STEPFILE 42 ;

The maximum number of internal-file-names which may be assigned during a step is set at 42 ;

$STEPFILE 30 ;

This statement is an error because the value given is neither 42 nor 55.

SUMMARY OF CONFIG DEFAULT VALUES AND STATEMENTS

```
$ACCOUNT ALL ;
$ACTSIZE 200 ;
$BANNER 2 ;
$FILESHARE 5 ;
$JOBCLASS A , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;✓
$JOBCLASS B , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;✓
$JOBCLASS C , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;✓
$JOBCLASS D , XPRTY = 5 , PRIORITY = 1 , MAXLOAD = 1 , STARTED ;✓
$JOBCLASS E , XPRTY = 4 , PRIORITY = 2 , MAXLOAD = 1 , STARTED ;  5 1 5
$JOBCLASS F , XPRTY = 7 , PRIORITY = 3 , MAXLOAD = 1 , STARTED ;✓
$JOBCLASS G , XPRTY = 9 , PRIORITY = 4 , MAXLOAD = 1 , STARTED ;✓
$JOBCLASS H , XPRTY = 1 , PRIORITY = 6 , MAXLOAD = 1 , STARTED ;
$JOBCLASS I , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS J , XPRTY = 1 , PRIORITY = 6 , MAXLOAD = 1 , STARTED ;
$JOBCLASS K , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS L , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS M , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS N , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS O , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS P , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 5 , STARTED ;
$JOBCLASS Q , XPRTY = 4 , PRIORITY = 7 , MAXLOAD =10, STARTED , NMPRIO ;
$JOBCLASS R , XPRTY = 2 , PRIORITY = 0 , MAXLOAD = 6 , STARTED , NSC ;
$JOBCLASS S , XPRTY = 0 , PRIORITY = 7 , MAXLOAD = 1 , STARTED , NSC,NMPRIO,NMLOAD ;
$JOBCLASS T , XPRTY = 4 , PRIORITY = 7 , MAXLOAD = 6 , STARTED , NMPRIO ;
$JOBCLASS U , XPRTY = 2 , PRIORITY = 7 , MAXLOAD = 6 , STARTED , NMPRIO ;
$JOBCLASS V , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS W , XPRTY = 2 , PRIORITY = 0 , MAXLOAD = 8 , STARTED , NSC ;
$JOBCLASS X , XPRTY = 3 , PRIORITY = 0 , MAXLOAD = 1 , STARTED , NSC,NMPRIO,NMLOAD ;
$JOBCLASS Y , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JOBCLASS Z , XPRTY = 9 , PRIORITY = 7 , MAXLOAD = 1 , STARTED ;
$JORSIZE 500 ;
$MAXFILE 55 ;
$MAXJOB 200 ;
$MAXTAPE 64 ;
$MAXTASK 50 ;
$MULTLEV 14, BATCH = 5, INTERACT = 10 ;
```

SUMMARY OF CONFIG DEFAULT VALUES AND STATEMENTS (cont).

```
$OWCLASS A  , PRIORITY = 1
$OWCLASS B  , PRIORITY = 2
$OWCLASS C  , PRIORITY = 3
$OWCLASS D  , PRIORITY = 4
$OWCLASS E  , PRIORITY = 5
$OWCLASS F  , PRIORITY = 6
$OWCLASS G  , PRIORITY = 7

$OWCLASS H through to Z are all default PRIORITY = 7

$OWDEVICE 1 , DVID = . . . , CLASS = A through Z ;
$OWDEVICE 2 , DVID = . . . , CLASS = A through Z ;
$OWDEVICE 3 , DVID = . . . , CLASS = A through Z ;
       :    :    :    :    :    :    :    :
       :    :    :    :    :    :    :    :
       :    :    :    :    :    :    :    :
$OWDEVICE15, DVID = . . . , CLASS = A through Z ;

$OWDFLT CLASS = C, DEVCLASS = PR/H132, MEDIA = I10000, TAPE = SYSOUT ;
$PRLOG THRESHOLD = 50 ;
$ROFCLASS P ;
$STEPFILE 42 ;
```

3. System Tailoring

The system tailoring procedure allows a system disk to be built which corresponds closely to the installation's requirements. It also minimizes the amount of disk space required for system files.

Before running the TAILOR job the system manager must prepare a disk volume for receiving the GCOS components.

It is important that the volume preparation ($VOLPREP) specifies that a COMPLETE preparation is performed and that all weak tracks (as given by the list supplied with the disk pack) are specified in the Badtrack parameter group.

Example :

```
VOLPREP OLD=(DEVCLASS=MS/M400,MEDIA=WDK),
        NEW=(DEVCLASS=MS/M400,MEDIA=C020),
        COMPLETE,
        BADTRACK=(045/16,174/3,332/14);
```

For full details of the $VOLPREP statement see the Data Management Utilities manual.

PLANNING THE SYSTEM DISK CONTENTS

In the tailor operation there are two choices :

- Selection of optional files
- Selection of size values for certain files.

Optional files are those which the user need not have present if certain system features are not used. They are summarized in Table 3-1.

Table 3-1. Optional System Files

File Name	When Required	Size (CYL)
SYS.SYSTEM	System Restore at ISL	15
SYS.JRNAL	System Journal (Before)	6
SYS.JADIR	System Journal (After)	1
GMCF	Series 100 Program Mode	2
HMCF	Series 200/2000 and IBM Program Mode.	2
SITE.CATALOG	Catalog Facilities	⩾1
SYS.FTU	File Transfer thru 61/64	⩾10

These optional files will be automatically selected (or rejected) by the TAILOR job, based on the parameters supplied by the user. They need not be deleted by user-supplied $DEALLOC statements.

The following table lists the system files to which file size selection applies. The sizes shown are the default values. If the sizes of these files are to be changed, the calculation should be performed using the information provided on each file as detailed in Section IV of this manual.

Table 3-2. File Size Values

File Name	File Use	Initial Size (CYL)
SYS.HLMLIB	System Load Module Library	60
SYS.IN	System Input File	5
SYS.BKST1	System Backing Store File	60
SYS.OUT	System Output File	50
SITE.CATALOG	Catalog File For Site	1
SYS.FTU	File Transfer Utility File	10

THE TAILOR JOB

Having planned the new system, the system manager executes the TAILOR job. This job is supplied in the system and is executed by the SJ (Start Job) command, SJ TAILOR (option list). The TAILOR job may also be executed by use of the $RUN JCL statement, RUN TAILOR, SYS.HSLLIB, VALUES=(option list);. The "option list" is a group of up to 26 parameters. If any option is omitted, the appropriate default value will be assumed. (See TAILOR Option List, below). All parameter values, as used in the previous run are retained by TAILOR and refered to during the current run. This facility eliminates the need for repetitive action as only the parameters under going change need be specified. There is however one exception, the FUNC parameter, its value is always set to NORM when none of its SIX options are respecified.

TAILOR Recovery

The TAILOR job can be started from a given point (recovery phase) by use of the FUNC parameter. This parameter offers two restart options.

- FUNC=RCVYCT used for restart after abort during copy of catalog.
- FUNC=RCVTLM used for restart after abort during copy of standard load modules.

Additional Phases Of TAILOR

- The PRINT phase, which is used for the printing of the values as defined in TAILOR (i.e, FUNC=PRINT).
- The RESET phase, which is used for reseting the original default values of TAILOR. (i.e, FUNC=RESET).
- The TEST phase, which is used for rewriting and printing of the supplied parameters of TAILOR. This phase can be used before the normal (NORM) run of TAILOR, prior to processing.

THE TAILOR OPTION LIST

Notation :

CSV............Customer Supplied Value

$$\text{SJ TAILOR (INDVC=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{MS/M400}}\end{array}\right\},$$

$$\text{OUTDVC=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{MS/M400}}\end{array}\right\},$$

$$\text{INVOL= CSV ,}$$

$$\text{OUTVOL= CSV ,}$$

$$\text{FUNC=}\left\{\begin{array}{l}\underline{\text{NORM}}\\ \text{TEST}\\ \text{RCVYCT}\\ \text{RCVYLM}\\ \text{PRINT}\\ \text{RESET}\end{array}\right\},$$

$$\text{SYST=}\left\{\begin{array}{l}\text{YES}\\ \underline{\text{NO}}\end{array}\right\},$$

$$\text{JRNL=}\left\{\begin{array}{l}\underline{\text{NO}}\\ \text{BOT}\\ \text{AFT}\\ \text{BEF}\end{array}\right\},$$

$$\text{CTLG=}\left\{\begin{array}{l}\text{YES[,NBOBJ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{600}}\end{array}\right\}\text{]}\\ \underline{\text{NO}}\end{array}\right\},$$

$$\text{SYSOUT=}\left\{\begin{array}{l}\underline{\text{YES}}\\ \text{RSD,RDSVOL=CSV[,RSDDVC=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{MS/M400}}\end{array}\right\}\text{]}\end{array}\right\},$$

$$\text{OUTSZ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{50}}\end{array}\right\},$$

$$\text{RECSZ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{3128}}\end{array}\right\},$$

$$\text{BLKSZ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{3130}}\end{array}\right\},$$

$$\text{LGTKSZ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{9}}\end{array}\right\},$$

$$\text{MODE=}\left\{\begin{array}{l}\underline{\text{NATIVE}}\\ \text{PM100}\\ \text{PM200}\\ \text{PM360}\end{array}\right\},$$

[CT100],[FT100],[COBTR],[FT200],
[HUTIL],[BALTR],[RPGTR],

$$\text{ROF=}\left\{\begin{array}{l}\text{LV6}\\ \text{LV61}\\ \text{LV61R}\\ \text{BOT}\\ \text{BOTR}\\ \underline{\text{N}}\end{array}\right\},$$

$$\text{FTUSZ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{10}}\end{array}\right\},$$

$$\text{HLMSZ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{60}}\end{array}\right\},$$

$$\text{INSZ=}\left\{\begin{array}{l}\underline{\text{5}}\\ \text{CSV}\end{array}\right\},$$

$$\text{BKST1SZ=}\left\{\begin{array}{l}\text{CSV}\\ \underline{\text{60}}\end{array}\right\}\text{)}$$

OPTION LIST DEFINITIONS

INDVC		Device class of the input disk.
OUTDVC		Device class of the output disk. Possible values are : MSM350 MS/M400,MS/M402,MS/M452
INVOL		Self explanatory
OUTVOL		Self explanatory
FUNC		Execution mode of TAILOR
	PRINT	Prints default values defined in TAILOR job
	NORM	Normal use of TAILOR
	RCVYCT	Recovery, beginning at the PREALLOC of SITE.CATALOG.
	RCVYLM	Recovery, beginning at the PREALLOC of SYS.HLMLIB
	TEST	Rewrite and print option, without processing of TAILOR.
	RESET	Resets the initial default values of TAILOR.
SYST		Defines if the SYS.SYSTEM file needed on optimized disk
	YES	PREALLOC of SYS.SYSTEM
	NO	No file.
JRNL		Defines if journalization is required.
	BEF	PREALLOC of SYS.JRNAL file.
	AFT	PREALLOC of SYS.JADIR file and copy load modules
	BOT	PREALLOC of both files and copy load modules
	NO	No file.
CTLG		Defines if Catalog facilities are required
	YES	PREALLOC SITE.CATALOG and copy file and load modules.
	NO	No File.
	NBOBJ	Number of objects to be cataloged. Used for PREALLOC of SITE.CATALOG.
SYS.OUT		Defines if SYS.OUT file will be on optimized disk or another.
	YES	SYS.OUT file allocated onto OUTVOL.
	RSD	SYS.OUT file allocated onto RSDDVC/RSDVOL.
	RSDVOL	Resident disk (not system).
	RSDDVC	Device class of Resident disk (not system).
OUTSZ		Size to be given to SYS.OUT file.
RECSZ		Record size chosen for SYS.OUT file.
BLKSZ		Block size of SYS.OUT
LGTKSZ		Logical track size of SYS.OUT file.

MODE		Mode control option
	PM100	G100 file will be allocated and PM100 load module delivered
	PM200	H200 file will be allocated and PM200 load module delivered
	NATIVE	Native mode only.
CT100		Delivery of G100 program translator
FT100		Delivery of G100 file translator
COBTR		Delivery of required COBOL translator
FT200		Delivery of required file translator
HUTIL		Delivery of HFAS file & volume utilities
BALTR		Delivery of BAL to COBOL translator.
RPGTR		Delivery of RPG translator from IBM3.
ROF	LV6	ROF through level 6 (no SYS file required).
	LV61	ROF through level 61, PREALLOC of SYS.FTU onto OUTVOL and copy load modules.
	LV61R	PREALLOC of SYS.FTU onto RSDVOL and copy load modules.
	NO	No ROF required.
	BOT	PREALLOC of SYS.FTU onto OUTVOL
	BOTR	PREALLOC of SYS.FTU onto RSDVOL
FTUSZ		Size to be given to SYS.FTU.
HLMSZ		Size to be given to SYS.HLMLIB
INSZ		Size to be given to SYS.IN
BKST1SZ		Size to be given to SYS.BKST1.

EXAMPLE TAILOR JOB

```
SJ TAILOR (INVOL=CHBSYS,OUTVOL=GCOSRX,RSDVOL=GCOS,
          RSDDVC=MS/M350,FUNC=TEST,MODE=NATIVE,JRNL=BEF,
          OUTSZ=60)
```

From the above TAILOR job format a new system disk GCOSRX is created, using the GCOS disk as the resident volume with a device number of MS/350. The GCOSRX disk is created with the following characteristics :

- The phase of TAILOR used was TEST.
- All components of the Series 100,200/2000 and IBM/3 (files and load modules) are excluded. (MODE=NATIVE).
- Preallocation of SYS.JRNAL performed (JRNAL=BEF).
- SYS.HLMLIB has a size of 60 cylinders (default value assumed).
- SYS.IN has a size of 5 cylinders (default value assumed).
- SYS.FTU is not preallocated (default value assumed).
- The SYS.OUT file preallocated on GCOSRX (SYS.OUT=Y assumed) with 60 cylinders, RECSZ of 3128 bytes a BLKSZ of 3136 bytes (default values), and a logical track size of 9.

TAILOR JOB OUTPUT MESSAGES

SYSTEM CONSOLE (see sample listing)

Each stage of TAILOR activity is reported by the message :

"****SYSTEM DISK OPTIMIZATION PHASE : 1"

".. : 2"

"..: 3"

The end of a successful operation is signalled by :

"YOUR SYSTEM IS NOW READY

ONTO......output volume identify......

AND YOU HAVE AT LEAST...given number...CYL FREE"

If the SYSOUT file is to be allocated onto a resident but non-system disk, the following message will be displayed by TAILOR :

"SYSOUT file preallocation"

and the following message will be provided by the system :

"MOUNT......output volume identify...FOR...xyy"

If SYS.FTU is required on RSDVOL, the following message will be sent :

"x SYSFTU file preallocation"

LINE PRINTER OUTPUT

The line printer output details the volume contents of the optimized disk and also gives the following information.

List of delivered Compile Units ....(SYS.HCULIB).

List of delivered Sharable Modules .... (SYS.HSMLIB).

List of delivered Load Modules .... (SYS.HLMLIB).

List of delivered JCL ....(SYS.HSLLIB).

EXECUTING THE TAILOR JOB

Figure 3-1. shows a sample log report of TAILOR execution. After the command is entered ①. The options and values selected are not listed.

①

```
$JOB CROPTDSK,USER=user-name,PROJECT=project-name;
$RUN TAILOR, SYS.HSLLIB,VALUES= (INVOL=C053,OUTVOL=CS18,
                RSDVOL=C219,MODE=PM200,COBTR,BALTR,RPGTR,
                SYST=Y,JRNAL=BOT,SYSOUT=R,HLMSZ=5,INSZ=9,
                BKST1SZ=70,OUTSZ=30,RECSZ=1040,
                BLKSZ=1048,LGTSZ=33);
$ENDJOB;
```

```
19.33 X9  IN TAILOR SYSADMIN P SPR=7
19.33 X9  STARTED TAILOR SYSADMIN P
 SP06 X9  ***TAILOR UTILITY 1D809
 SP06 X9  FUNCTION REQUIRED NORM***
 SP06 X9
 SP06 X9  * SYSTEM DISK OPTIMIZATION PHASE : 1
 SP06 X9  WARNING: GPL FILES ONTO C053
 SP06 X9  NO PREALLOC PROVIDED ON C218
 SP06 X9  *                          PHASE : 2
 SP06 X9  WARNING : SITE CATALOG REQUIRED
 SP06 X9  WHEN YOU RUN TELECOM FACILITIES
 SP06 X9  SITE.CATALOG IS PREALLOCATED WITH 600 OBJECTS
 SP06 X9  *                          PHASE : 3
 SP06 X9
 SP06 X9  ******************************************************
 SP06 X9  *
 SP06 X9  *               YOUR SYSTEM IS NOW READY
 SP06 X9  *               ON C218
 SP06 X9  * AND YOU HAVE AT LEAST 20 CYL FREE
 SP06 X9  ******************************************************
 SP06 X9

 SP06 X9  * SYS.OUT FILE PREALLOCATION
*20.09 MS06  MOUNT C219 FOR X9
20.10 X9 .75 COMPLETED TAILOR SYSADMIN P
20.10 GCOS: NO MORE JOBS RUNNING
20.11 X9   OUTPUT COMPLETED TAILOR SYSADMIN
20.12 GCOS: NO MORE RUNNING
20.12 GCOS: IDLE
```

②

Figure 3.1. Sample Log Report of TAILOR Execution

The job then reports each stage of its activity allowing the user to verify that the desired result has been achieved.

To optimize the performance of TAILOR, it is recommended that the following load modules are pre-initialized before TAILOR job execution :

H_JTRA
H_LIBMAINT
H_PRE_DEALLOC
H_CATALOG

Upon successful completion of TAILOR (signalled by the message ②) the new system may be used by performing an ISL (Initial Storage Load) action from the new system disk (the output disk of TAILOR).

If an error occurs during the execution of TAILOR, the job should be repeated; however, before any attempts to repeat are made, the user should perform $VOLPREP again to clear the target volume.

If the error persists then the following actions must be taken :

. $VOLPREP the target disk
. Restore the master supplied volume

Repeat TAILOR

TAILOR JOB ERROR DIAGNOSTICS

The TAILOR job can abort for one of the following reasons :

1. Wrong option(s) given in a keyword or in a parameter.
   TAILOR will abort with the following message :

   "XXXXX: .....undefined label....."

Action required : Rerun TAILOR with the offending option(s) corrected.

Example :

```
SJ TAILOR (INVOL=MD1,OUTVOL=MD2,CTLG=YAR,
           NBOBJ=1900)
```

Abort was due to CTLG=YAR

Abort message "C YAR : undefined label".

To correct this option and continue running TAILOR, input the following message :

```
SJ.TAILOR (INVOL=MD1,OUTVOL=MD2,CTLG=Y,NBOB=1900).
```

If an incorrect keyword is given TAILOR will not abort, the default value will be assumed.

Example :

```
SJ TAILOR (......SYSTEM=Y....)
```

The keyword SYSTEM is unknown, therefore SYST=N will be assumed.

2. Error during duplication.

   Rerun TAILOR, when the error has been corrected, by use of SJ TAILOR, or if the OUTVOL is unavailable use SJ TAILOR (OUTVOL=new media).

3. Overflow on file occuring when copying load modules, and catalog files.

   a). Overflow on SYS.HLMLIB

   The following message will be given on the operator console :

```
"UNABLE TO COPY....load-module-name....LOAD MODULE
                    CHECK SIZE OF SYS.HLMLIB
                         AND RUN A RCVYLM PHASE".
```

   Action : Run RCVYLM phase with a new SYS.HLMLIB size (HLMSZ parameter).

   Example :

```
SJ TAILOR (INVOL=MD1,OUTVOL=MD2,-,HLMSZ=30)
```

   The job aborted due to insufficient size for SYS.HLMLIB.

   Therefore the following change was made :

```
SJ TAILOR (FUNC=RCVYLM,HLMSZ=40).
```

   b) Overflow on SITE.CATALOG

   An overflow may occur when the RELCAT utility is being used to copy the catalog file from the input volume.

   The following message will be given on the console :

```
"UNABLE TO COPY CATALOG FILE
                    CHECK VALUE OF NBOBJ AND
                                          RUN A RCVYCT PHASE"
```

   Action : Run RCVYCT phase with the new size for NBOBJ.

   Example :

```
SJ TAILOR (INVOL=MED1,OUTVOL=MD2,,CTLG=Y,NBOBJ=600)
```

The value given for JBOB. was not sufficient.

Therefore the following change was made :

```
SJ TAILOR (FUNC=RCVYCT,NBOBJ=1900)
```

ADDITIONAL USAGE OF TAILOR

In addition to the normal TAILOR job usage, as previously described, TAILOR can be used to perform the following functions :

CORRECTING FILE SIZE

Example :

When creating an optimized disk by running the following :

```
SJ TAILOR (INVOL=CHBSTS,OUTVOL=GCOSRX,MODE=PM200,COBTR,
          FT200,HLMSZ=44,INSZ=3,OUTSZ=30);
```

The new disk GCOSRX is created; a need then arises for a new disk (GCOSRY) to be created with an updated SYS.IN and SYS.OUT size.

To enable the change to be made the following is run :

```
SJ TAILOR (OUTVOL=GCOSRY,INSZ=5,OUTSZ=50);
```

or :

```
SJ TAILOR (INVOL=GCOSRX,OUTVOL=GCOSRY,INSZ=5,OUTSZ=50);
```

Both give the same result :

the creation of system disk GCOSRY with a SYS.IN size of 5 cylinders and a SYS.OUT size of 50 cylinders.

THE ADDITION OF A NEW PRODUCT

Example :

When it is decided to use an additional utility, like HFAS or similer, it will be required to place on the system disk the appropriate working information. Therefore the size of SYS.HLMLIB must be increased in the following way :

```
SJ TAILOR (FUNC=RCVYLM,HLMSZ=50,HUTIL);
```

4. Description Of System Files

Table 4-1 shows the layout of the GCOS master system disk as supplied to an installation. The files are listed in order of their position on the disk (the first, "VTOC" is the Volume Table of Contents found on all disks at cylinder 0).

Table 4-1. Supplied Master Disk Layout

File Name	Size (CYL)	Location	Supplied Contents
VTOC	1	ALL	
HUB	3	S	loaded
SYS.BOOT	1	S	loaded
SYS.ERLOG	2	R(S)	empty
SYS.URCINIT	2	R(S)	loaded
SYS.HSLLIB	3	R(S)	loaded
SYS.HCULIB	2	R(S)	loaded
SYS.HSMLIB	8	R(S)	loaded
SYS.SYSTEM	15	A(M)	loaded
SYS.IN	5	R(S)	empty
SYS.KNODET	1	S	empty
SITE.CATALOG	1	S	loaded
SYS.BKST	19	S	empty
SYS.BKST1	60	S	empty
SYS.OUT	50	R	empty
GMCF	2	R	loaded
SYS.HLMLIB	60	R(S)	loaded
HMCF	2	R	empty
SYS.SYSDUMP	4	S	empty
SYS.SDUMP	1	S	empty

the notation used for location in table 4-1 is :

S : must be present on the system disk.

R : must be present on a RESIDENT disk.
(note that the system disk is always RESIDENT)

M : the master supplied disk.

A : any disk.

When the required location is followed by a second choice in parentheses, the value in the parentheses is the recommended value.

The size, in cylinders, is given for disks MSU0350/400/402.

In the allocation description for each file the JCL variables &1 and &2 are the user-supplied device-class and disk volume-name.

OPTIMIZED DISK CONTENTS

The system file type and cylinder which are located on the optimized disk are identical to those of the supplied master disk with the following exceptions :

Additional SYS Files

SYS.JRNAL....cylinder size...6
SYS.JADIR.....cylinder size...1
SYS.FTU.......cylinder size...10

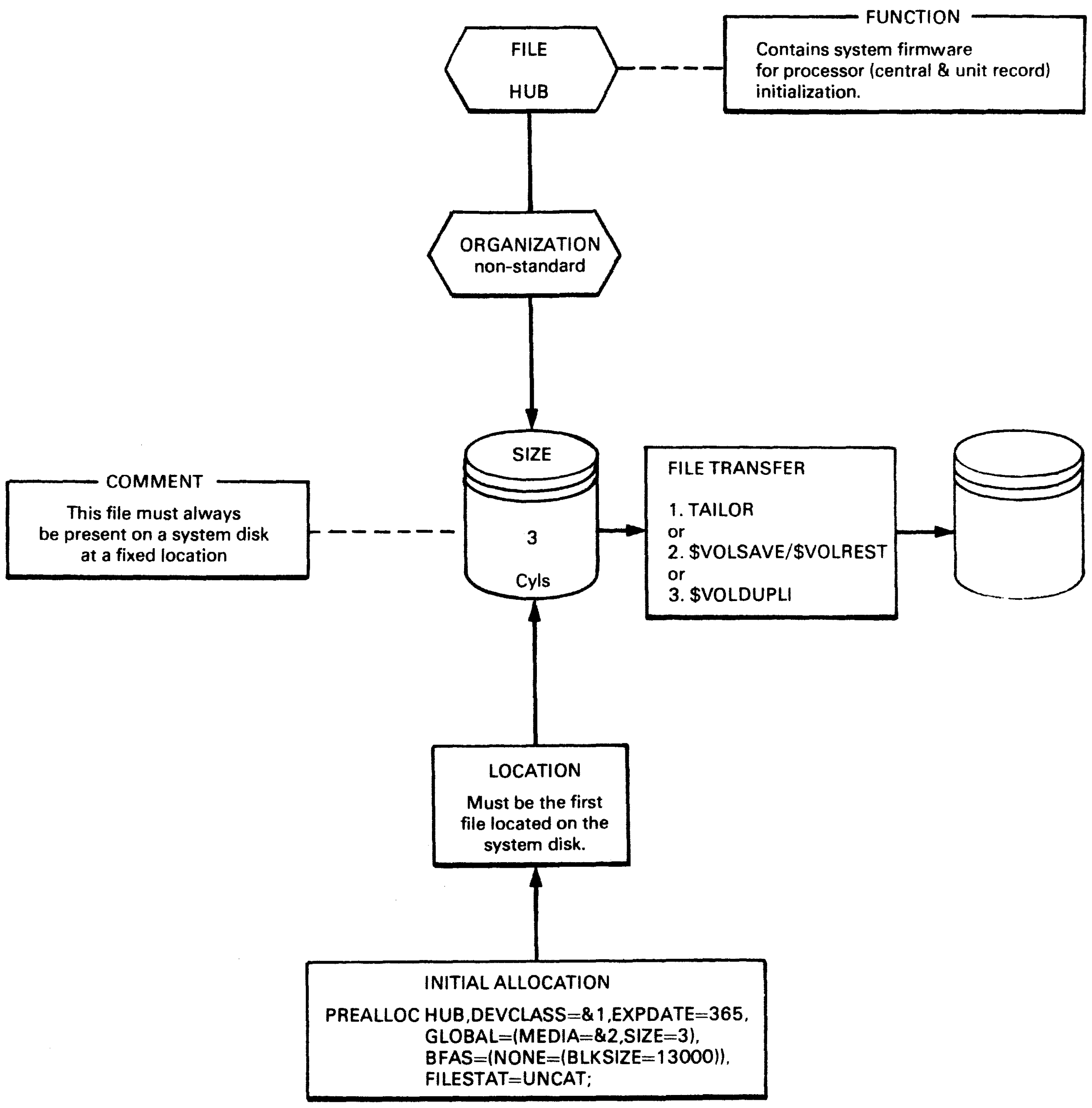
} Use TAILOR facility to include these files onto the system disk

Others

SYS files GMCF and HMCF have a **combined** cyclinder size requirement of 2.

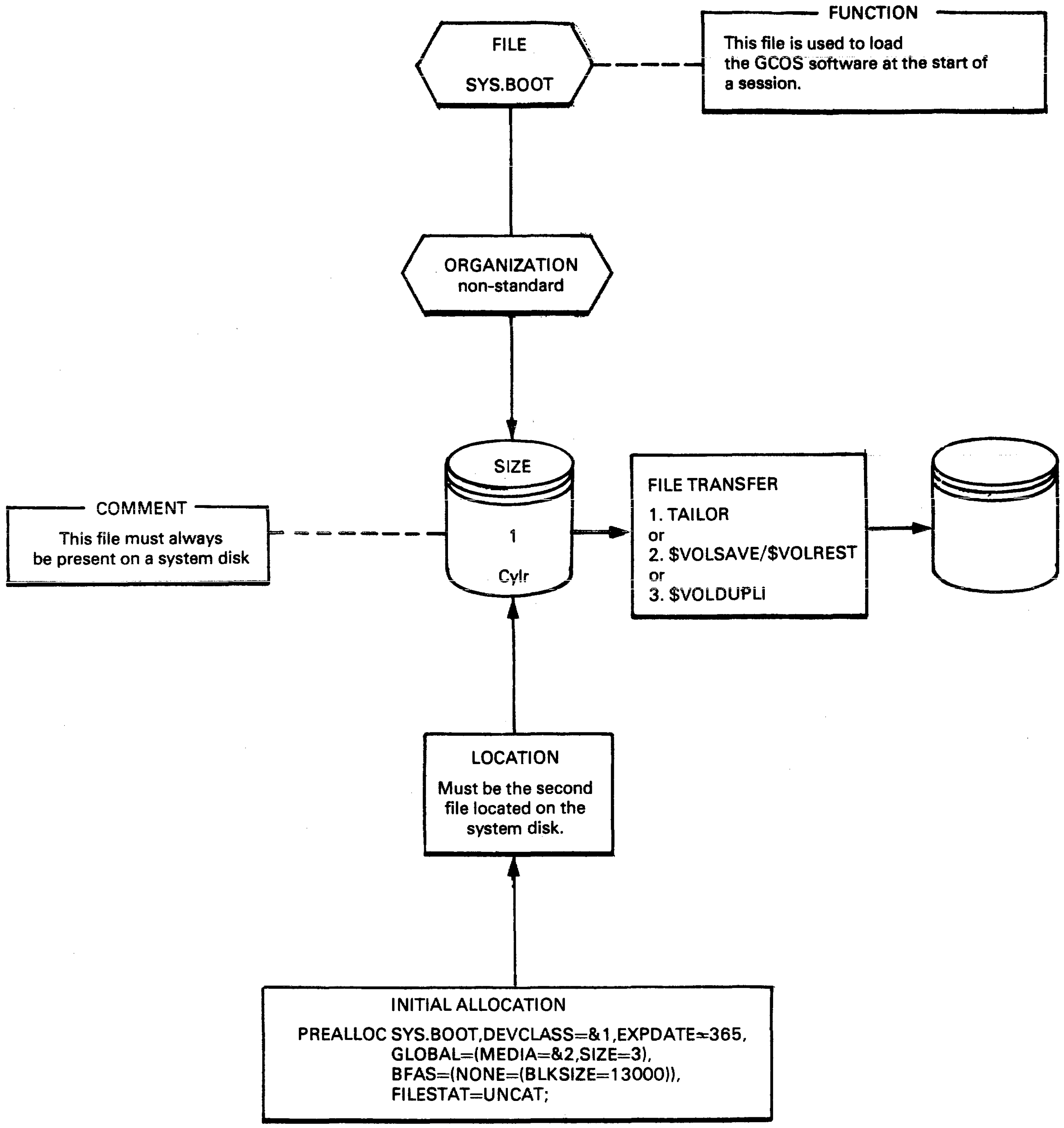
The remainder of this section gives, in simple diagramatic form, all relevant information concerning each of the system files.

FILE NAME : HUB


FILE
HUB
FUNCTION
Contains system firmware for processor (central & unit record) initialization.
ORGANIZATION
non-standard
SIZE
3
Cyls
COMMENT
This file must always be present on a system disk at a fixed location
FILE TRANSFER
1. TAILOR
or
2. $VOLSAVE/$VOLREST
or
3. $VOLDUPLI
LOCATION
Must be the first file located on the system disk.
INITIAL ALLOCATION

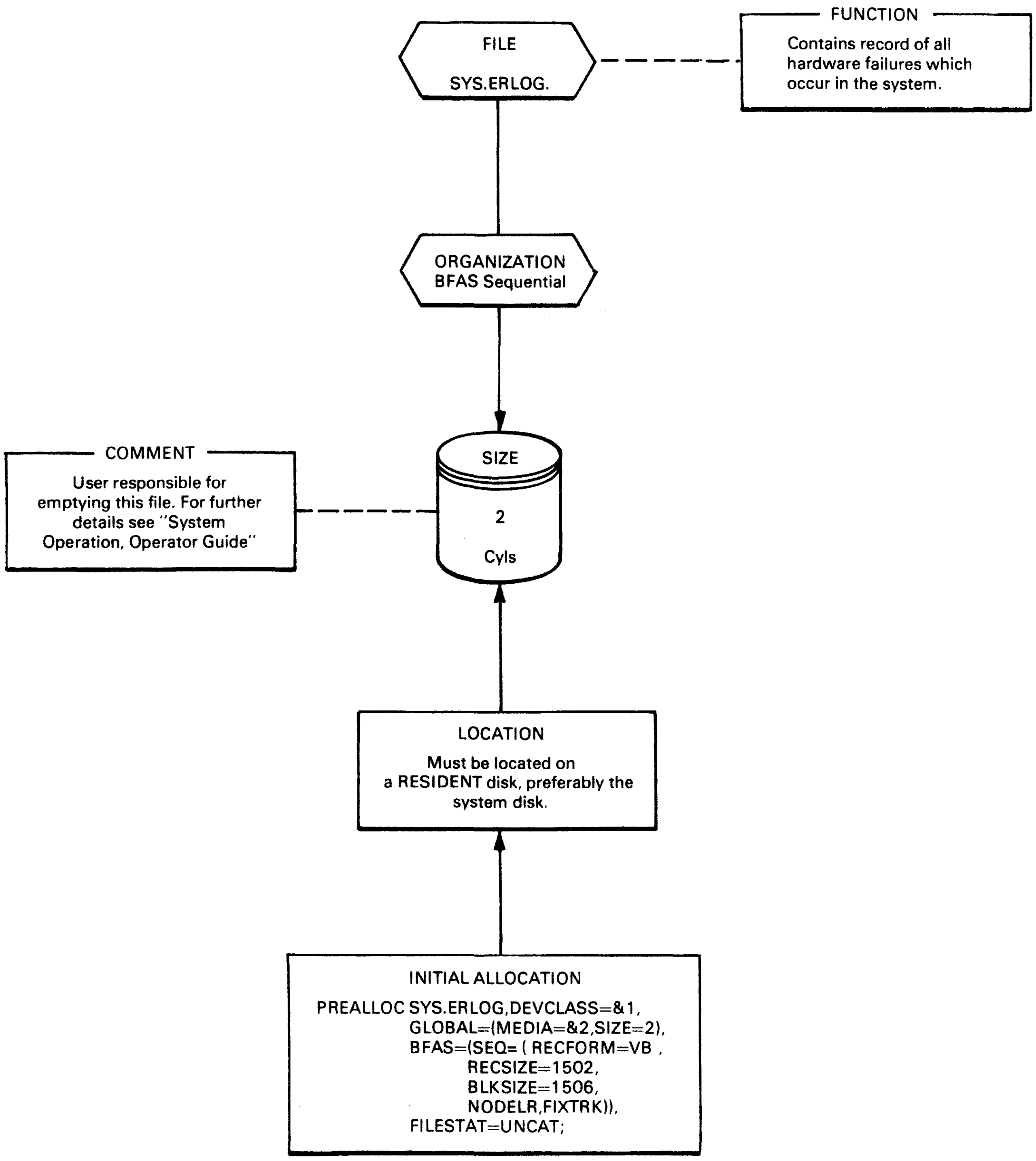
```
PREALLOC HUB,DEVCLASS=&1,EXPDATE=365,
         GLOBAL=(MEDIA=&2,SIZE=3),
         BFAS=(NONE=(BLKSIZE=13000)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS. BOOT


FILE
SYS.BOOT
FUNCTION
This file is used to load the GCOS software at the start of a session.
ORGANIZATION
non-standard
SIZE
1
Cylr
COMMENT
This file must always be present on a system disk
FILE TRANSFER
1. TAILOR
or
2. $VOLSAVE/$VOLREST
or
3. $VOLDUPLI
LOCATION
Must be the second file located on the system disk.
INITIAL ALLOCATION

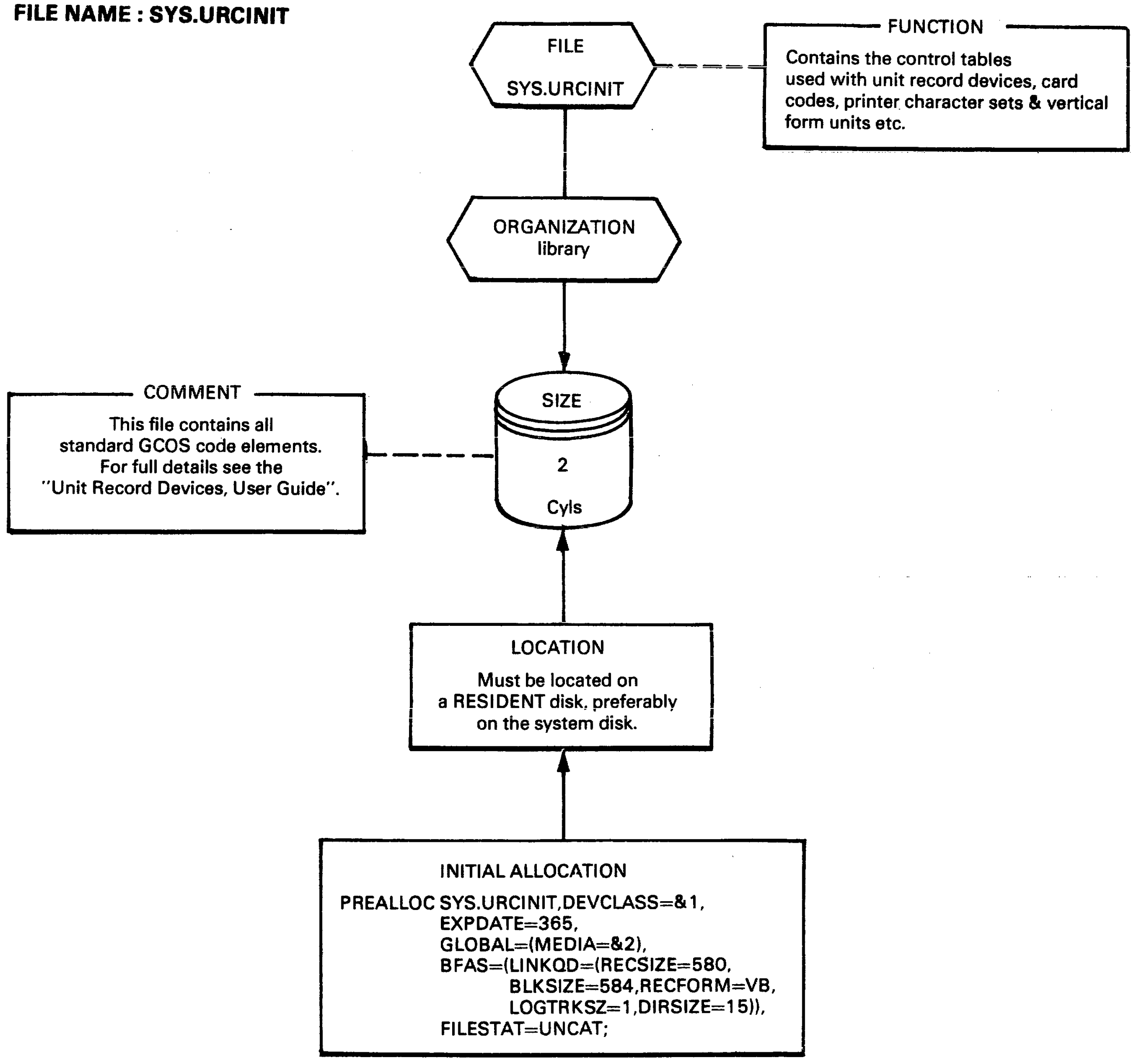
```
PREALLOC SYS.BOOT,DEVCLASS=&1,EXPDATE=365,
         GLOBAL=(MEDIA=&2,SIZE=3),
         BFAS=(NONE=(BLKSIZE=13000)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.ERLOG.


FILE
SYS.ERLOG.
FUNCTION
Contains record of all hardware failures which occur in the system.
ORGANIZATION
BFAS Sequential
SIZE
2
Cyls
COMMENT
User responsible for emptying this file. For further details see "System Operation, Operator Guide"
LOCATION
Must be located on a RESIDENT disk, preferably the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.ERLOG,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=2),
         BFAS=(SEQ= ( RECFORM=VB ,
                 RECSIZE=1502,
                 BLKSIZE=1506,
                 NODELR,FIXTRK)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.URCINIT



FUNCTION

Contains the control tables used with unit record devices, card codes, printer character sets & vertical form units etc.

COMMENT

This file contains all standard GCOS code elements. For full details see the "Unit Record Devices, User Guide".

LOCATION

Must be located on a RESIDENT disk, preferably on the system disk.

INITIAL ALLOCATION

```
PREALLOC SYS.URCINIT,DEVCLASS=&1,
         EXPDATE=365,
         GLOBAL=(MEDIA=&2),
         BFAS=(LINKQD=(RECSIZE=580,
                 BLKSIZE=584,RECFORM=VB,
                 LOGTRKSZ=1,DIRSIZE=15)),
         FILESTAT=UNCAT;
```

The user may add elements (such as forms definitions) into this library by use of the $URINIT statement.

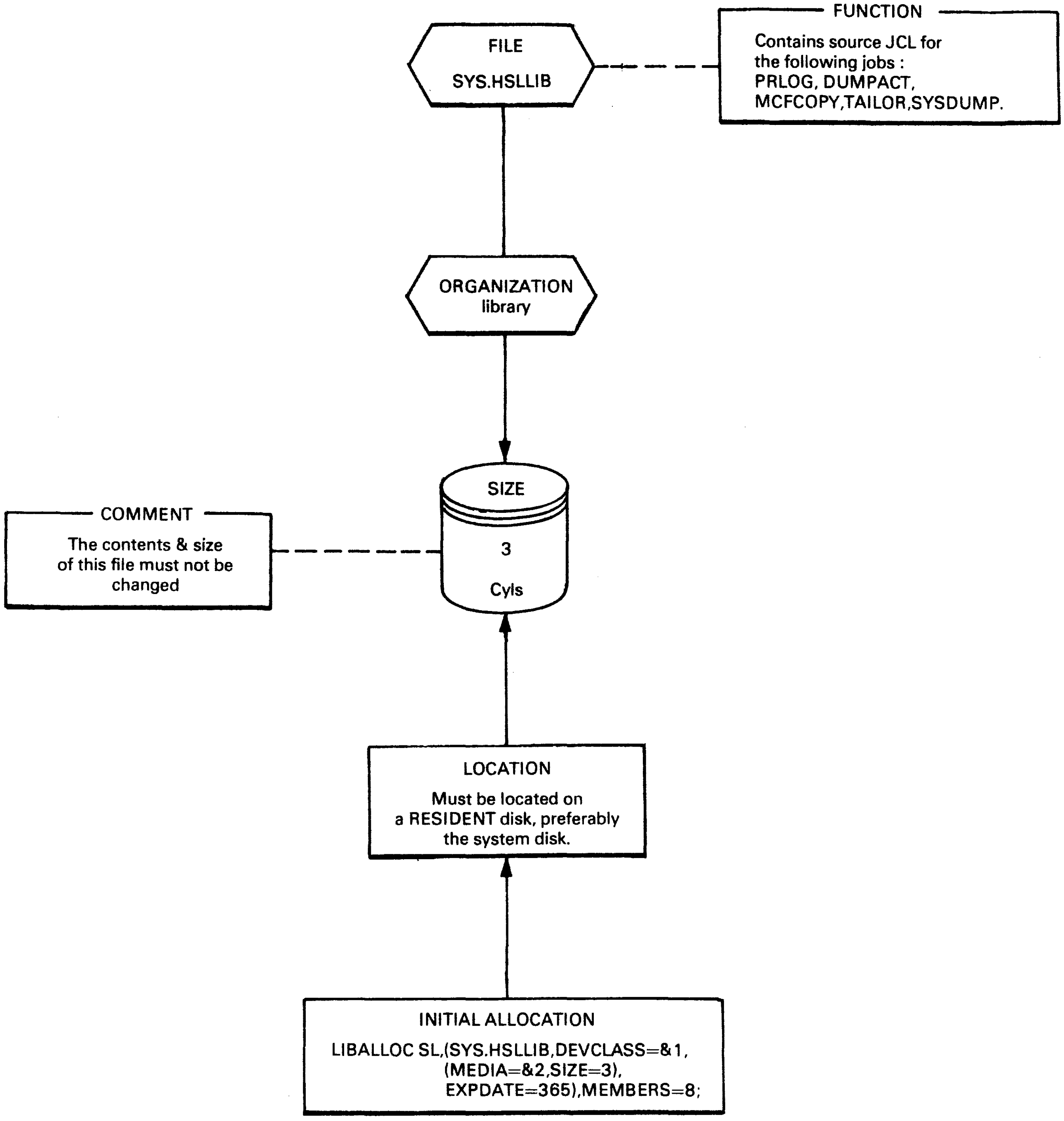
In general when moving from an old release to a new release of GCOS, the system manager will need to re-load into the SYS.URCINIT the installation defined elements which are already present in the old SYS.URCINIT file.

This transition is performed by use of the utility known as the Translator of Unit Record File (TURF). The TURF utility is so designed to overcome any in compatibility within the URINIT input language of the old and new releases. However, care must be exercized by ensuring that every installation defined element which is required to be moved into the new system disk is specified as input to the TURF utility.

This precaution is imperative as all previous contents of the SYS.URCINIT file are erased at the begining of TURF execution.

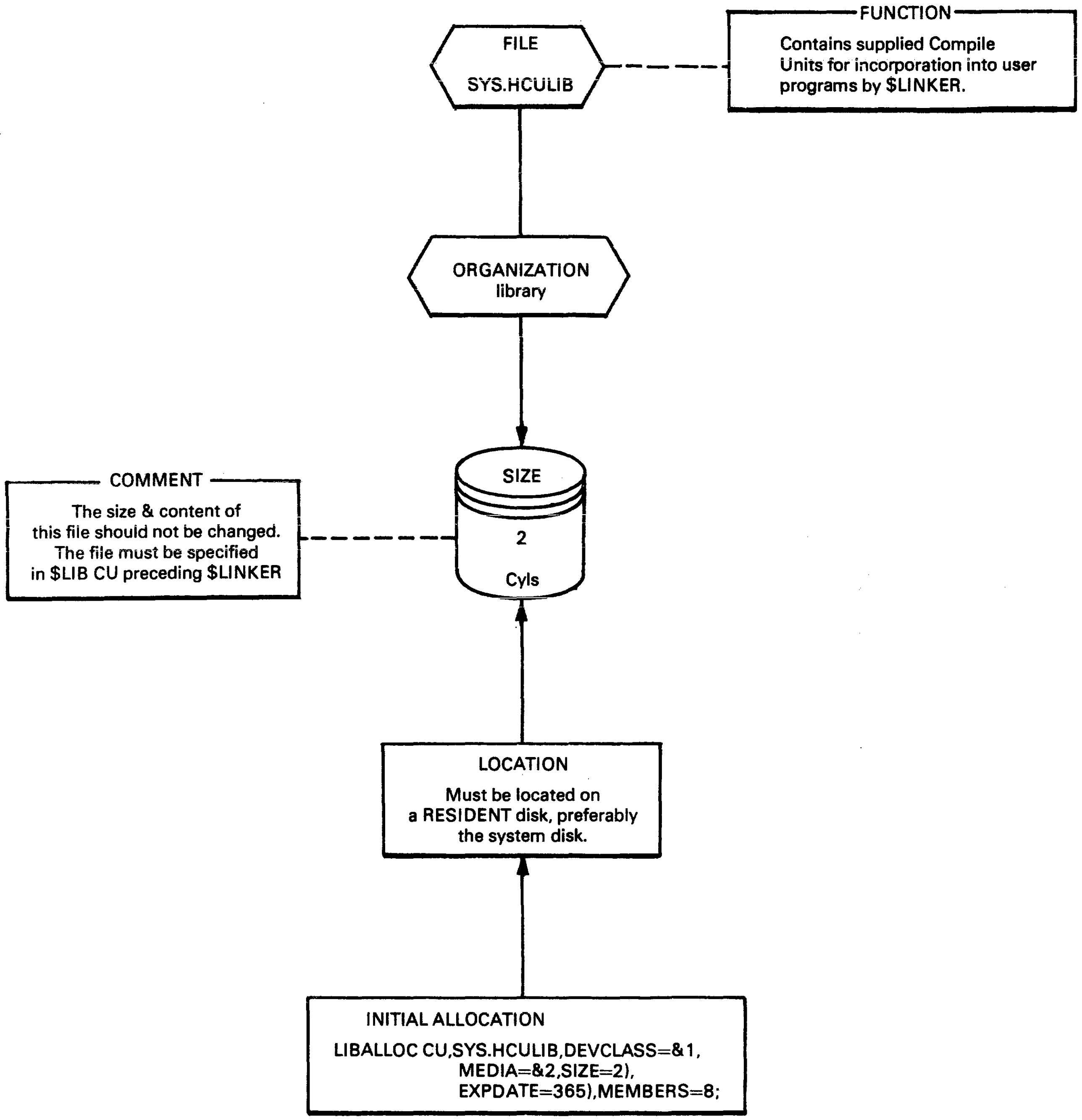
For full details of the TURF utility see the System Installation manual.

FILE NAME : SYS.HSLLIB

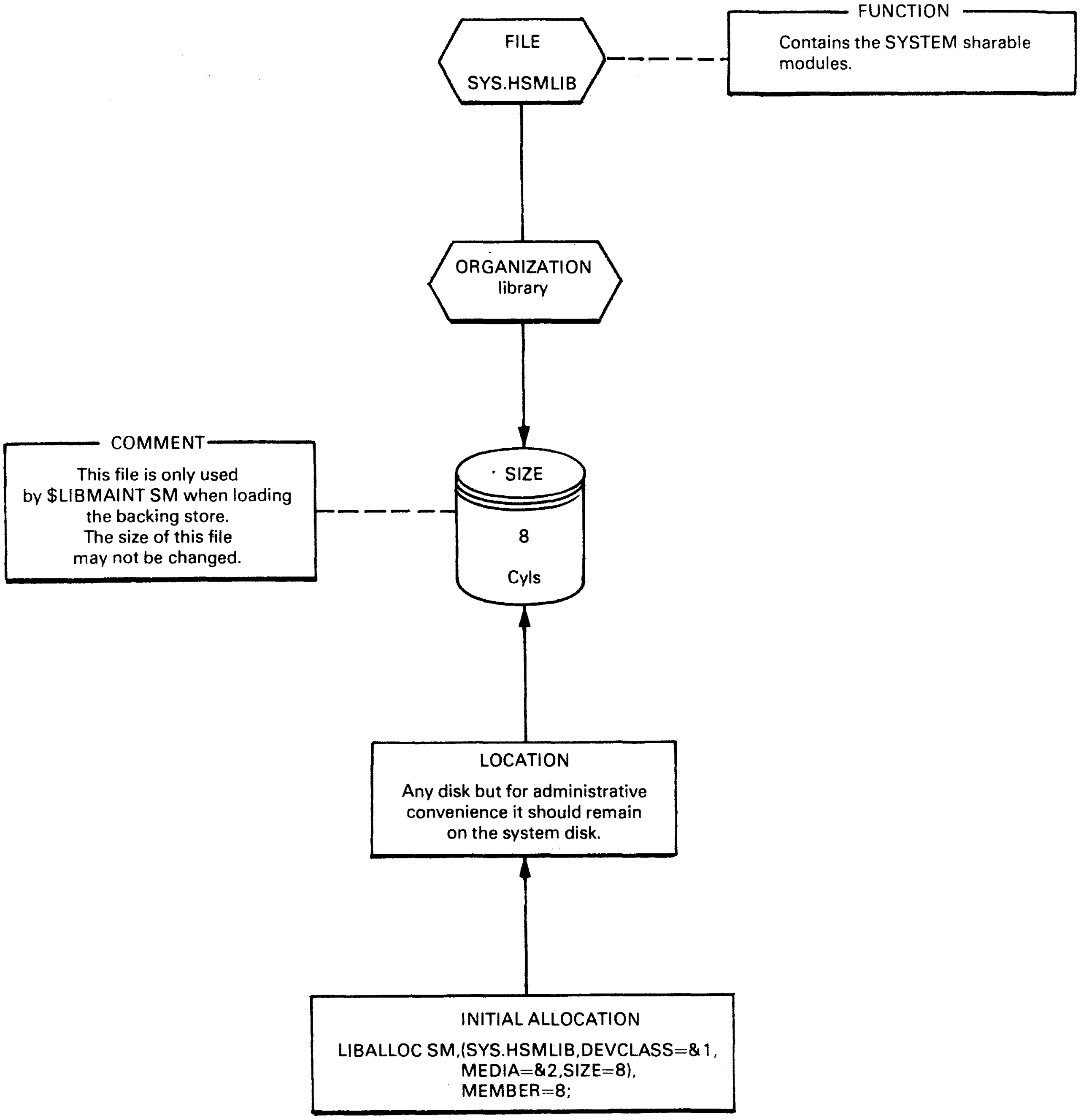

FILE
SYS.HSLLIB
FUNCTION
Contains source JCL for the following jobs : PRLOG, DUMPACT, MCFCOPY,TAILOR,SYSDUMP.
ORGANIZATION
library
SIZE
3
Cyls
COMMENT
The contents & size of this file must not be changed
LOCATION
Must be located on a RESIDENT disk, preferably the system disk.
INITIAL ALLOCATION

```
LIBALLOC SL,(SYS.HSLLIB,DEVCLASS=&1,
             (MEDIA=&2,SIZE=3),
              EXPDATE=365),MEMBERS=8;
```

FILE NAME : SYS.HCULIB

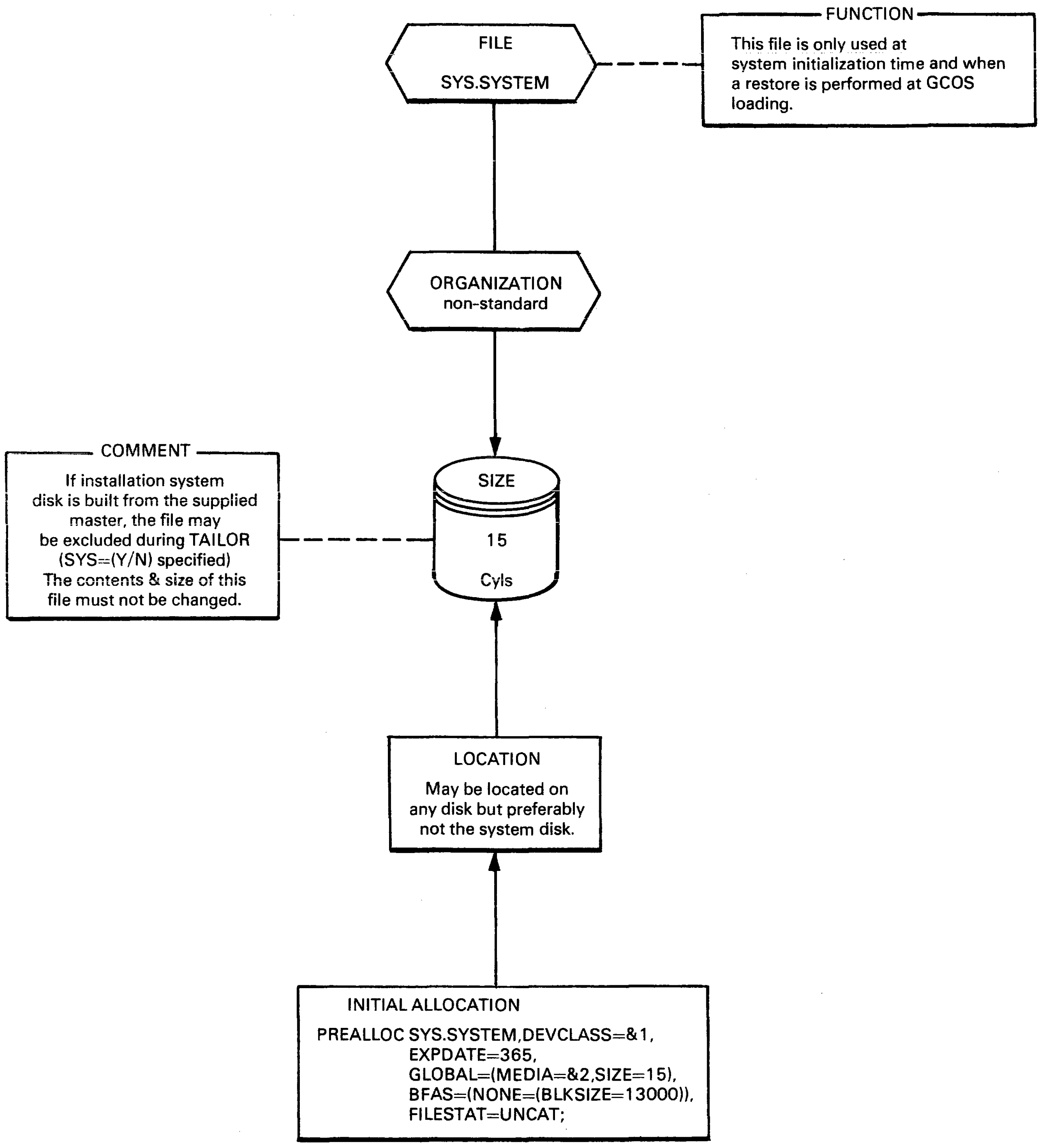

FILE
SYS.HCULIB
FUNCTION
Contains supplied Compile Units for incorporation into user programs by $LINKER.
ORGANIZATION
library
SIZE
2
Cyls
COMMENT
The size & content of this file should not be changed. The file must be specified in $LIB CU preceding $LINKER
LOCATION
Must be located on a RESIDENT disk, preferably the system disk.
INITIAL ALLOCATION
LIBALLOC CU,SYS.HCULIB,DEVCLASS=&1,
MEDIA=&2,SIZE=2),
EXPDATE=365),MEMBERS=8;

FILE NAME : SYS.HSLMLIB


FILE
SYS.HSMLIB
FUNCTION
Contains the SYSTEM sharable modules.
ORGANIZATION
library
COMMENT
This file is only used by $LIBMAINT SM when loading the backing store. The size of this file may not be changed.
SIZE
8
Cyls
LOCATION
Any disk but for administrative convenience it should remain on the system disk.
INITIAL ALLOCATION

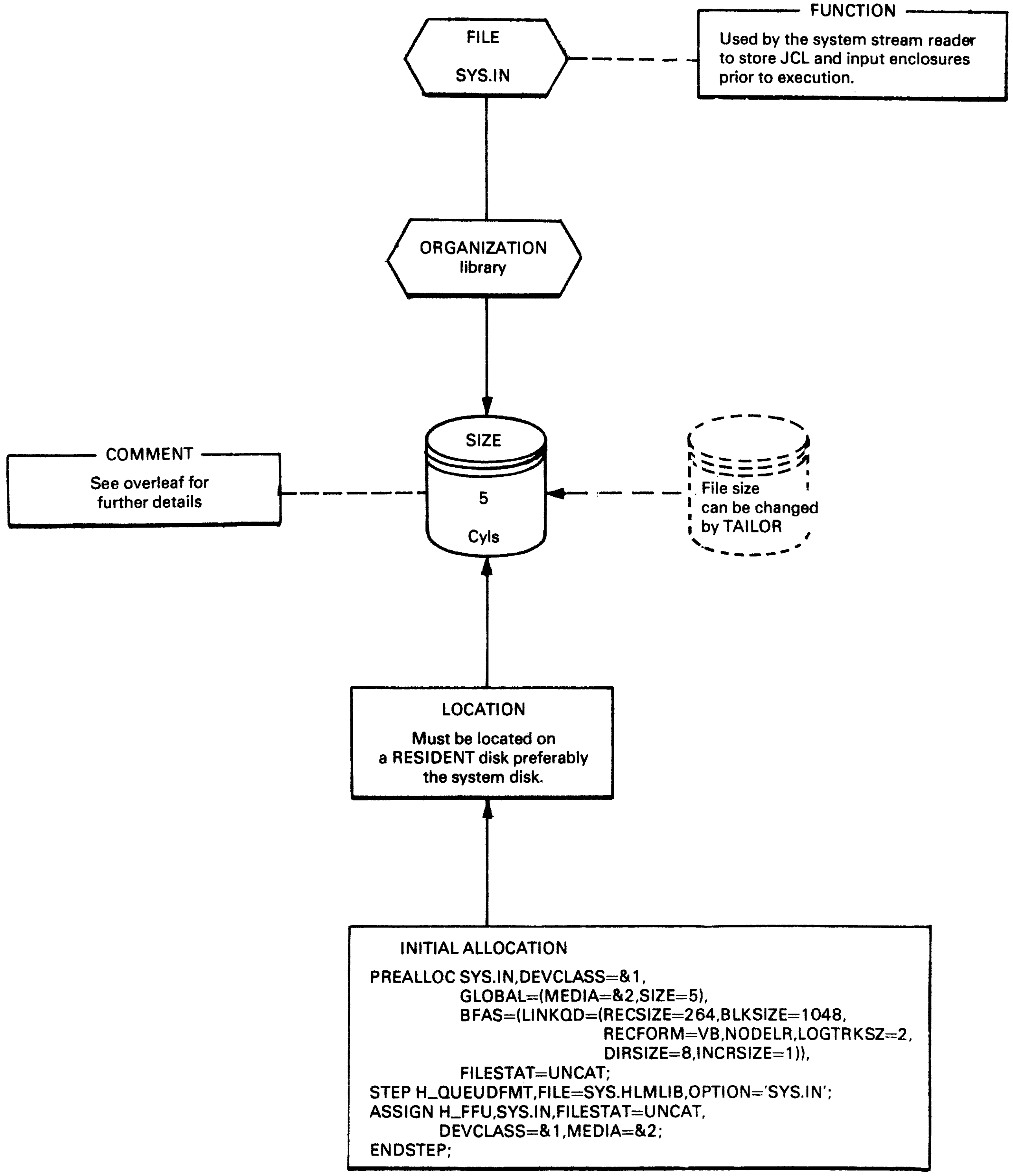
```
LIBALLOC SM,(SYS.HSMLIB,DEVCLASS=&1,
             MEDIA=&2,SIZE=8),
             MEMBER=8;
```

FILE NAME : SYS.SYSTEM


FUNCTION
This file is only used at system initialization time and when a restore is performed at GCOS loading.
FILE
SYS.SYSTEM
ORGANIZATION
non-standard
COMMENT
If installation system disk is built from the supplied master, the file may be excluded during TAILOR (SYS=(Y/N) specified)
The contents & size of this file must not be changed.
SIZE
15
Cyls
LOCATION
May be located on any disk but preferably not the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.SYSTEM,DEVCLASS=&1,
         EXPDATE=365,
         GLOBAL=(MEDIA=&2,SIZE=15),
         BFAS=(NONE=(BLKSIZE=13000)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.IN


FILE
SYS.IN
FUNCTION
Used by the system stream reader to store JCL and input enclosures prior to execution.
ORGANIZATION
library
SIZE
5
Cyls
COMMENT
See overleaf for further details
File size can be changed by TAILOR
LOCATION
Must be located on a RESIDENT disk preferably the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.IN,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=5),
         BFAS=(LINKQD=(RECSIZE=264,BLKSIZE=1048,
                       RECFORM=VB,NODELR,LOGTRKSZ=2,
                       DIRSIZE=8,INCRSIZE=1)),
         FILESTAT=UNCAT;
STEP H_QUEUDFMT,FILE=SYS.HLMLIB,OPTION='SYS.IN';
ASSIGN H_FFU,SYS.IN,FILESTAT=UNCAT,
       DEVCLASS=&1,MEDIA=&2;
ENDSTEP;
```

The size of this file may be changed by TAILOR. The size chosen depends on installation requirements. The values given below allow the calculation of a suitable size :

Record Type	Records per track	Records per cylinder
DATA	132	2508
COBOL	132	2508
DATASSF	121	2299
BINARY ($BIN)	66	1254

Therefore, to ensure capacity for 10000 DATA cards it is sufficient to have a size of (10000/2508)=4 cylinders.

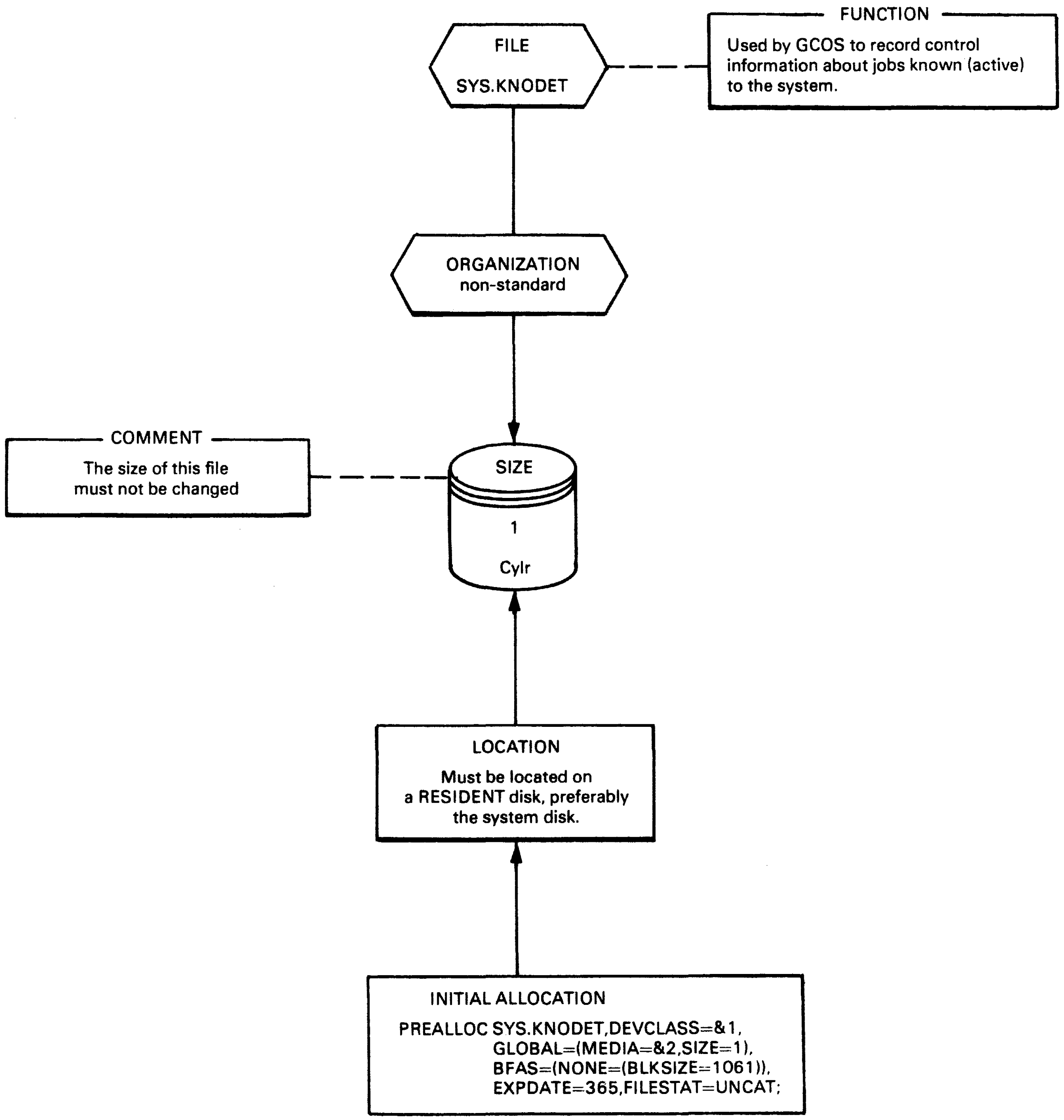
The default value supplied of 5 cylinders will accomodate 13000 DATA records.

Each input enclosure occupies an integral number of tracks. Therefore the maximum possible number of input enclosures is equal to the number of tracks available.

Note :

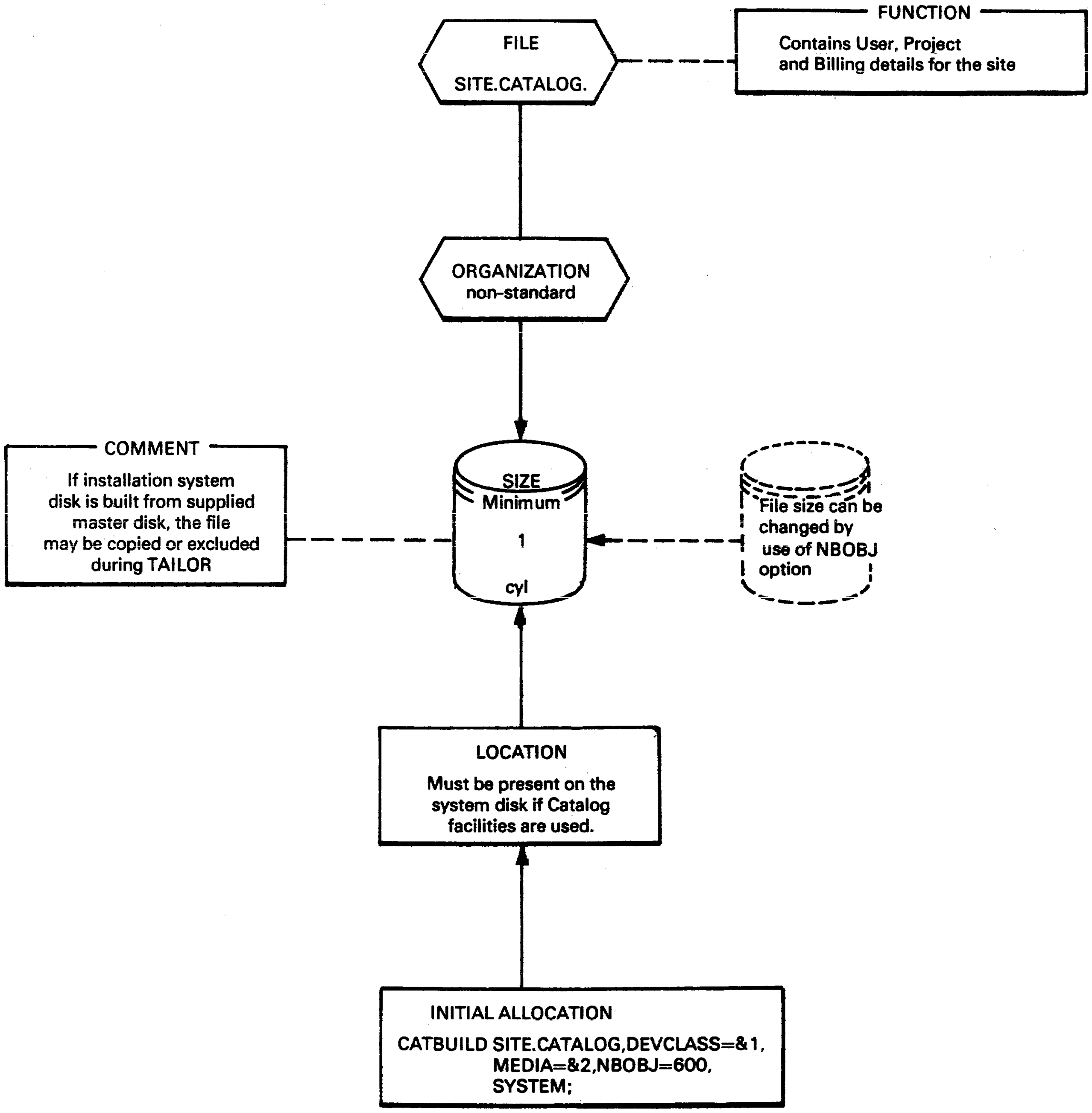
In the event of overflow, the size of SYS.IN will be increased by one cylinder.

FILE NAME : SYS.KNODET


FILE
SYS.KNODET
FUNCTION
Used by GCOS to record control information about jobs known (active) to the system.
ORGANIZATION
non-standard
COMMENT
The size of this file must not be changed
SIZE
1
Cylr
LOCATION
Must be located on a RESIDENT disk, preferably the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.KNODET,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=1),
         BFAS=(NONE=(BLKSIZE=1061)),
         EXPDATE=365,FILESTAT=UNCAT;
```

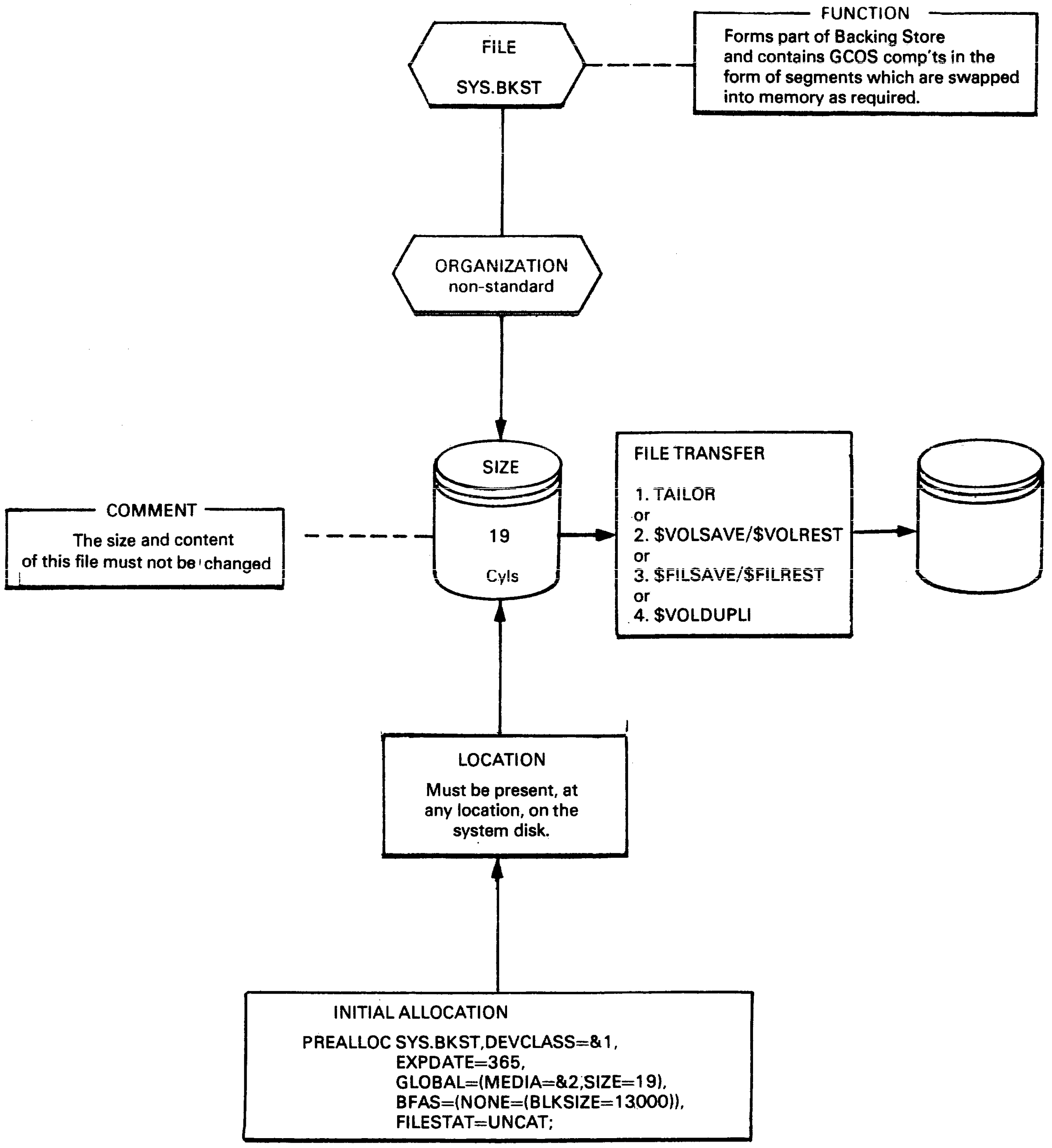
FILE NAME : SITE.CATALOG


FILE
SITE.CATALOG.
FUNCTION
Contains User, Project and Billing details for the site
ORGANIZATION
non-standard
COMMENT
If installation system disk is built from supplied master disk, the file may be copied or excluded during TAILOR
SIZE
Minimum
1
cyl
File size can be changed by use of NBOBJ option
LOCATION
Must be present on the system disk if Catalog facilities are used.
INITIAL ALLOCATION
CATBUILD SITE.CATALOG,DEVCLASS=&1,
MEDIA=&2,NBOBJ=600,
SYSTEM;

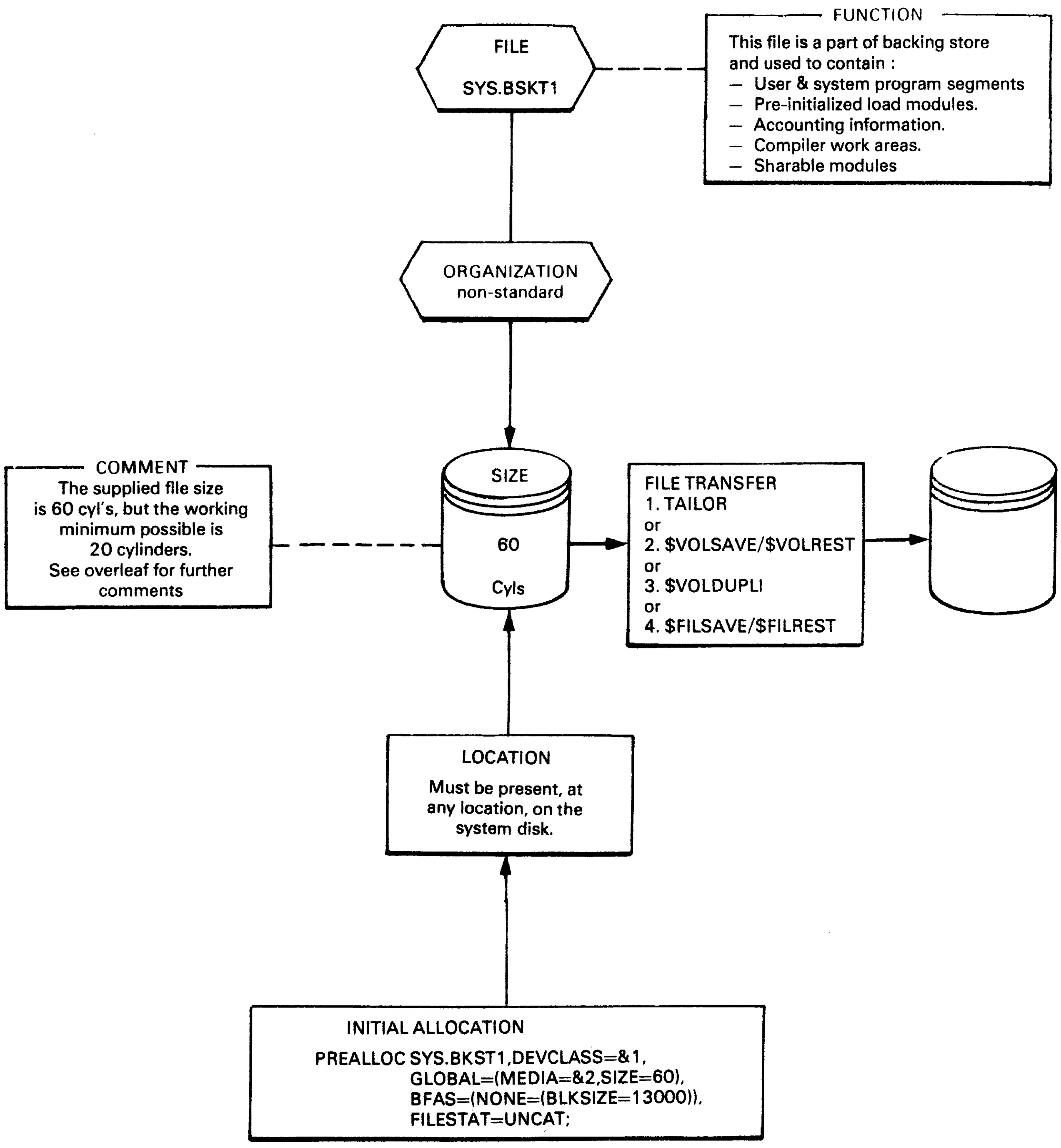
When using the TAILOR job the number of objects must correspond to the number of cylinders as shown in the following table.

No of objects as specified in NBOBJ	Corresponding number of cylinders required
Upto 600	1 cylinder
Upto 1270	2 cylinder
Upto 1900	3 cylinders
Upto 2550	4 cylinders
Upto 3190	5 cylinders
Upto 3830	6 cylinders
Upto 4480	7 cylinders
Upto 5100	8 cylinders
Upto 5770	9 cylinders
Upto 6400	10 cylinders
Upto 7050	11 cylinders
Upto 7690	12 cylinders
Upto 8330	13 cylinders
Upto 8960	14 cylinders
Upto 9600	15 cylinders

FILE NAME : SYS.BKST


FILE
SYS.BKST
FUNCTION
Forms part of Backing Store and contains GCOS comp'ts in the form of segments which are swapped into memory as required.
ORGANIZATION
non-standard
SIZE
19
Cyls
COMMENT
The size and content of this file must not be changed
FILE TRANSFER
1. TAILOR
or
2. $VOLSAVE/$VOLREST
or
3. $FILSAVE/$FILREST
or
4. $VOLDUPLI
LOCATION
Must be present, at any location, on the system disk.
INITIAL ALLOCATION
PREALLOC SYS.BKST,DEVCLASS=&1,
EXPDATE=365,
GLOBAL=(MEDIA=&2,SIZE=19),
BFAS=(NONE=(BLKSIZE=13000)),
FILESTAT=UNCAT;

FILE NAME : SYS.BSKT1


FILE
SYS.BSKT1
FUNCTION
This file is a part of backing store and used to contain :
– User & system program segments
– Pre-initialized load modules.
– Accounting information.
– Compiler work areas.
– Sharable modules
ORGANIZATION
non-standard
COMMENT
The supplied file size is 60 cyl's, but the working minimum possible is 20 cylinders.
See overleaf for further comments
SIZE
60
Cyls
FILE TRANSFER
1. TAILOR
or
2. $VOLSAVE/$VOLREST
or
3. $VOLDUPLI
or
4. $FILSAVE/$FILREST
LOCATION
Must be present, at any location, on the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.BKST1,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=60),
         BFAS=(NONE=(BLKSIZE=13000)),
         FILESTAT=UNCAT;
```

The TAILOR job allows the user to change the file size (subject to the minimum size restriction).

Calculation (in cylinders) of optimum size is : system requirement (20); plus space for each expected level of multiprogramming, plus space for pre-initialized load modules and shared modules.

The multiprogramming space requirement depends largely on the number of expected multiprogramming levels and the expected number of suspended job steps.

For each job step in EX or SUSP state it is recommended that three cylinders be reserved.

In practical terms, most job steps will only need one or two cylinders. The largest user of the backing store is the COBOL compiler, which needs space to accomodate work areas. The work area size is a factor of the number of statements to the compiled :

250 statements per cylinder.

Therefore a compilation of a 1500 line program will need six cylinders. This formula assumes that the compiler is pre-initialized.

Pre-initialized load modules are those programs which the system manager decides, for performance reasons, to maintain in the backing store. When a pre-initialized load module is executed the system does not need to load the requested program into the backing store. It is already loaded.

The user should choose which load modules to pre-initialize (operator command PLM) on the basis of frequency of use. The load modules may be either user application programs or system components such as the COBOL compiler or the utilities.

The size of user programs (i.e., the space required to accomodate it in SYS.BKST1) is found from the JOR. The value displayed in the JOR is in bytes. This value must be converted to units, where :

39090 bytes = 1 unit

A program will always occupy an integer number of units.

Table 4-2 gives the size of the system components in terms of units (6 units = 1 cylinder).

Table 4-2. System Component Size

Module Name	Function	Size (units)
H_JTRA	JCL Translation	12
H_COBOL	COBOL compiler	30
H_RPG	RPG compiler	9
H_FORTRAN	Fortran compiler	9
H_LINKER	Static linker	8
H_BTNS	Telecommunications	3
H_CNC	Telecommunications	3
H_SORT	Sort/Merge	6
H_MERGE	Sort/Merge	2
H_SORT_DISK	Sort/Merge disk	6
H_SORT_TAPE	Sort/Merge tape	6
H_SORTFMT	Sort/Merge file formatting	1
H_LIBMAINT	Library Management	3
H_FILSAVE	$FILSAVE utility	2
H_FILREST	$FILREST utility	2
H_VOLSAVEREST	$VOLSAVE & $VOLREST utilities	2
H_VOLREPARE	$VOLPREP/$VOLWORK and $VOLSCRAT utilities	2
H_CREATE	$CREATE & $PRINT utilities	2
H_PRE_DEALLOC	$PREALLOC & $DEALLOC utilities	2
H_PM100	Series 100 Program Mode	7
H_PM200	Series 200/2000 Program Mode	5

Note :

If an installation uses BTNS (Telecommunications), the IDS/II component, or the TDS component, must be placed in backing store. They are loaded under user request by the $LIBMAINT utility.

Example 1 :

An installation has the following characteristics :

- Maximum of two levels of multiprogramming.
- No pre-initialized components except the JCL translator and the COBOL compiler.

The calculation is :

```
System requirement = 20 cylinders
Multiprogramming =  6 cylinders
H_JTRA+COBOL =
12+30
42 blocks              7 cylinders
                     ____________
                      33 cylinders
```

The value 33 may be coded in TAILOR. This provides a space saving of (60-33) = 27 cylinders on the resulting system pack.

Example 2 :

An installation has the following characteristics :

- Three levels of multiprogramming (e.g., 12 cylinders)

In addition, the backing store is to contain :

- 60 units of user application programs
- The TDS and BTNS components (48 units)
- 20 units for TPRs (Transaction Processing Routines)
- All the components for COBOL program preparation (compiler, linker, $LIBMAINT,
  total 42 blocks).
- JCL translator and Sort/Merge (20 units)

Which gives a total of :

(60 + 48 + 20 + 42 + 18)/6 = 32 cylinders

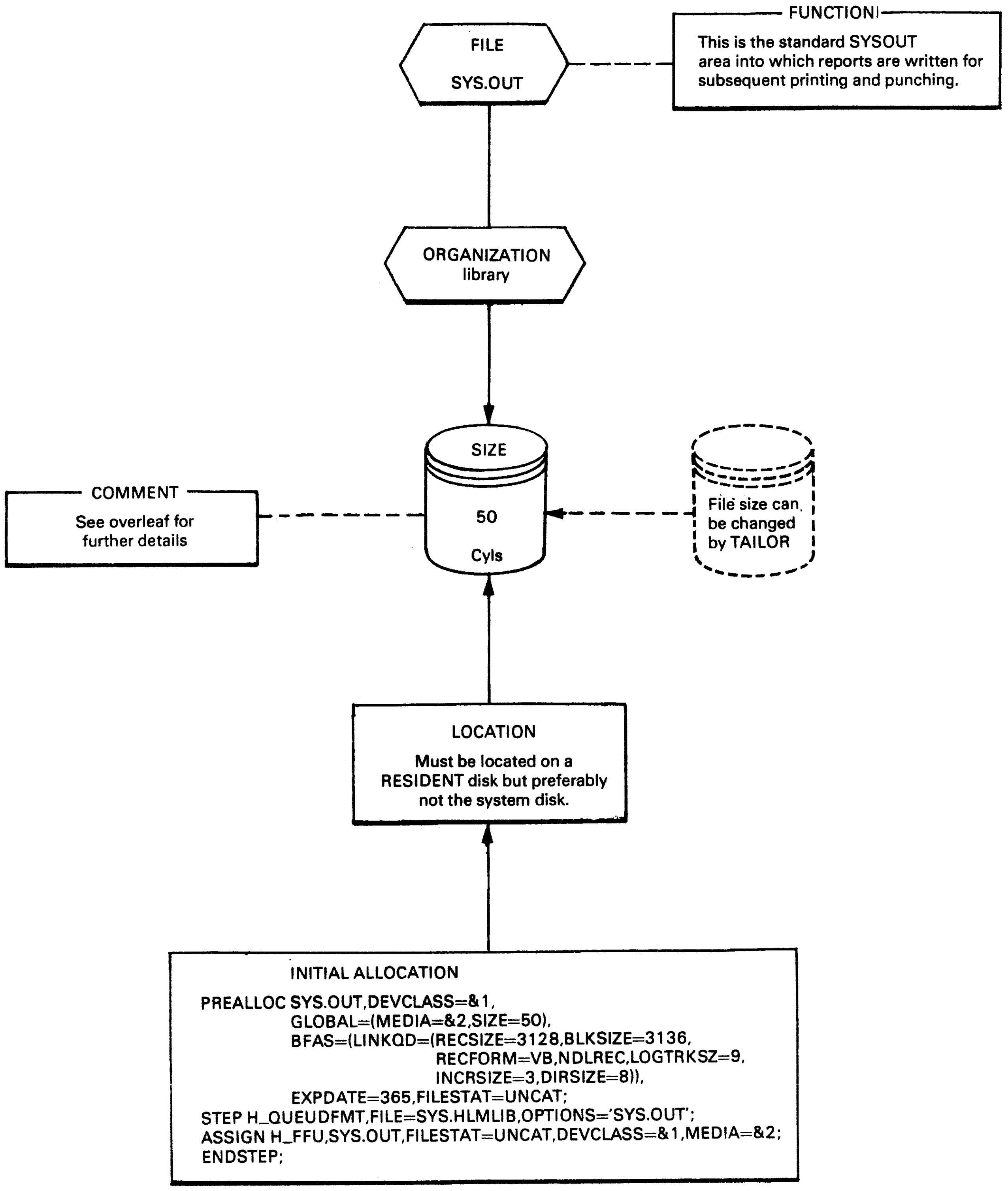
Therefore, the total size of SYS.BKST1 must be :

20 + 2 + 12 + 32 = 66 cylinders

Initial Allocation :

```
PREALLOC SYS.BKST1,DEVCLASS=&1,EXPDATE=365,
         GLOBAL=(MEDIA=&2,SIZE=60),
         BFAS=(NONE=(BLKSIZE=13000));
```

FILE NAME : SYS.OUT


FILE
SYS.OUT
FUNCTION
This is the standard SYSOUT area into which reports are written for subsequent printing and punching.
ORGANIZATION
library
SIZE
50
Cyls
COMMENT
See overleaf for further details
File size can be changed by TAILOR
LOCATION
Must be located on a RESIDENT disk but preferably not the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.OUT,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=50),
         BFAS=(LINKQD=(RECSIZE=3128,BLKSIZE=3136,
                       RECFORM=VB,NDLREC,LOGTRKSZ=9,
                       INCRSIZE=3,DIRSIZE=8)),
         EXPDATE=365,FILESTAT=UNCAT;
STEP H_QUEUDFMT,FILE=SYS.HLMLIB,OPTIONS='SYS.OUT';
ASSIGN H_FFU,SYS.OUT,FILESTAT=UNCAT,DEVCLASS=&1,MEDIA=&2;
ENDSTEP;
```

The size of this file may be changed by the TAILOR job. The information below shows the capacity of an MSU0/350/400/402 cylinder.

Table 4-3. SYS.OUT Size Calculation

Average Line Length	Number of Lines per Track	Number of Lines Per Cylinder
160	66	1254
150	66	1254
140	66	1254
136	77	1463
130	77	1463
120	77	1463
110	88	1672
100	99	1881
90	121	2090
80	132	2299
60	132	2926
50	154	3344
40	220	4180
30	264	5016
20	352	6688

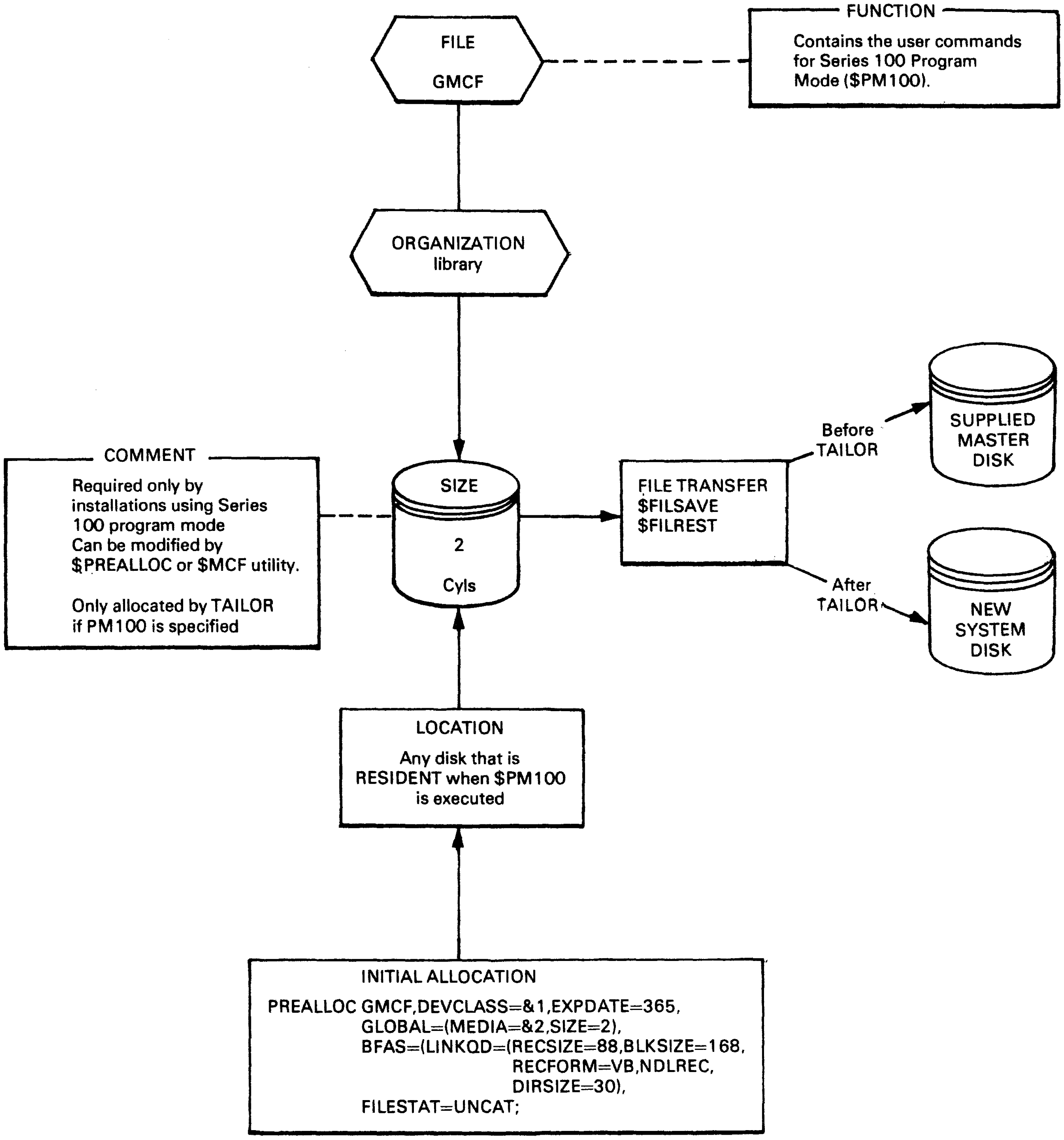
Therefore if the size is left at 50 cylinders then the capacity is, for an average length line of 80 characters :

50*2299=126 445 lines

Note :

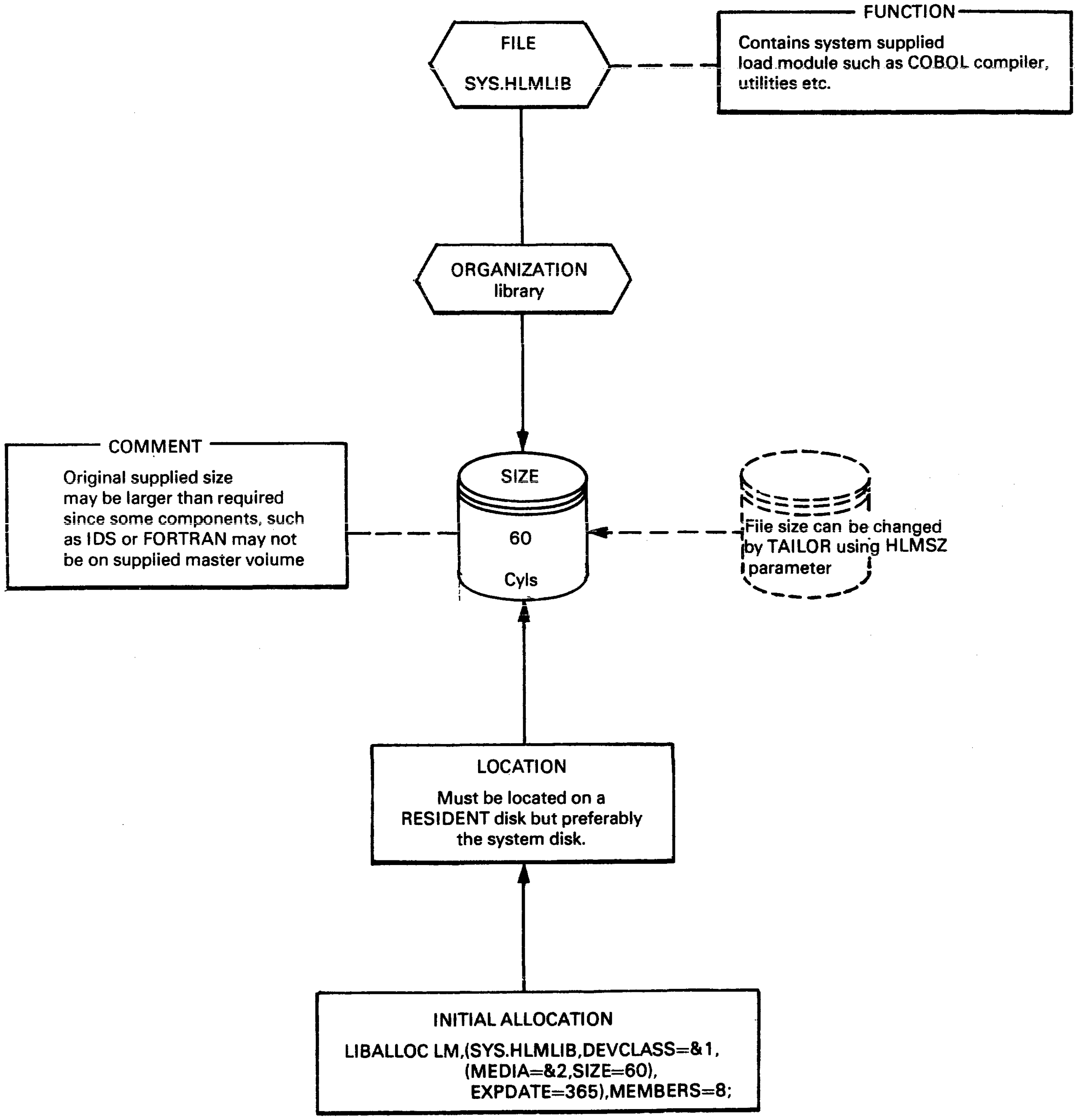
Each report will occupy a multiple of three tracks; therefore, no matter how short a report is, it will occupy at least three tracks.

FILE NAME : GMCF


FILE
GMCF
FUNCTION
Contains the user commands for Series 100 Program Mode ($PM100).
ORGANIZATION
library
COMMENT
Required only by installations using Series 100 program mode
Can be modified by $PREALLOC or $MCF utility.
Only allocated by TAILOR if PM100 is specified
SIZE
2
Cyls
FILE TRANSFER
$FILSAVE
$FILREST
Before TAILOR
SUPPLIED MASTER DISK
After TAILOR
NEW SYSTEM DISK
LOCATION
Any disk that is RESIDENT when $PM100 is executed
INITIAL ALLOCATION

```
PREALLOC GMCF,DEVCLASS=&1,EXPDATE=365,
         GLOBAL=(MEDIA=&2,SIZE=2),
         BFAS=(LINKQD=(RECSIZE=88,BLKSIZE=168,
                        RECFORM=VB,NDLREC,
                        DIRSIZE=30),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.HLMLIB


FILE
SYS.HLMLIB
FUNCTION
Contains system supplied load module such as COBOL compiler, utilities etc.
ORGANIZATION
library
COMMENT
Original supplied size may be larger than required since some components, such as IDS or FORTRAN may not be on supplied master volume
SIZE
60
Cyls
File size can be changed by TAILOR using HLMSZ parameter
LOCATION
Must be located on a RESIDENT disk but preferably the system disk.
INITIAL ALLOCATION

```
LIBALLOC LM,(SYS.HLMLIB,DEVCLASS=&1,
        (MEDIA=&2,SIZE=60),
         EXPDATE=365),MEMBERS=8;
```

The basic system requirement is 2895 blocks (209 blocks = 1 cylinder). This value must be added to the total required for optional elements. Table 4-4 gives the space requirements for optional elements.

Table 4-4. Space Requirement For Optional Elements

Load Module Name	Description	TAILOR keyword	Size (blocks)
H_PM100	PM100 Control Mode	Mode= PM100	355
H_CONF100	PM100 Control Mode	Mode= PM100	41
H_PM200	PM200 Control Mode	Mode= PM200	260
H_CONF200	PM200 Control Mode	Mode= PM200	35
H_COBTRANS	COBOL Translator	COBTR	427
H_CT100	G100 Translator	CT100	78
H_FT100	G100 File Translator	FT100	185
H_FILTRANS	H200/IBM3 File Translator (includes PACKTRANS)	FT200	259
H_BALTRANS	BAL to COBOL Translator	BALTR	279
H_RPGTRANS	RPG Translator	RPGTR	200
H_HALLOC	HFAS file & volume utilities	HUTIL	57
H_HCREATE	HFAS file & volume utilities	HUTIL	64
H_HDEALLOC	HFAS file & volume utilities	HUTIL	29
H_HMAPDISK	HFAS file & volume utilities	HUTIL	46
H_HPRINT	HFAS file & volume utilities	HUTIL	90
H_HVOLPREP	HFAS file & volume utilities	HUTIL	54
H_HVOLDUMP	HFAS file & volume utilities	HUTIL	43
H_MSPLTLIB	HFAS file & volume utilities	HUTIL	30
H_FXFER	ROF through Level 61	ROF= LV61 LV61R	42
H_ROF·	ROF through Level 61	ROF= LV61 LV61R	106
H_READER	ROF through Level 6	ROF=LV6	100
H_JAGEN	For Journal After	JRNL=AFT	30
H_ROLLFORWARD	For Journal After	JRNL=AFT	23
H_CATALOG	Catalog utility	CTLG=Y	135

Table 4-4. Space Requirements For Optional Elements

Load Module Name	Description	TAILOR keyword	Size (blocks)
The following applications are managed by SYSOPT utility.			
H_DDLPROC	Integrated Data Store	-------	176
H_DMLPROC		-------	124
H_DBANALYS	Administrator Aid	-------	93
H_DBPRINT		-------	74
H_DBVALID		-------	70
H_SORT_DISK	Sort/Merge	-------	272
H_SORT_TAPE		-------	232
H_SORTFMT		-------	25
H_MERGE		-------	102
H_COBOL		-------	1107
H_BTNS		-------	134
H_CNC		-------	104
H_QMAINT		-------	42
H_RPG	RPG	-------	753
H_FORTRAN	FORTRAN	-------	362
H_TDSCTP	TDS/STP	-------	73
H_TDSGEN		-------	32
H_IOF	IOF	-------	181
H_SCANNER		-------	60

Example :

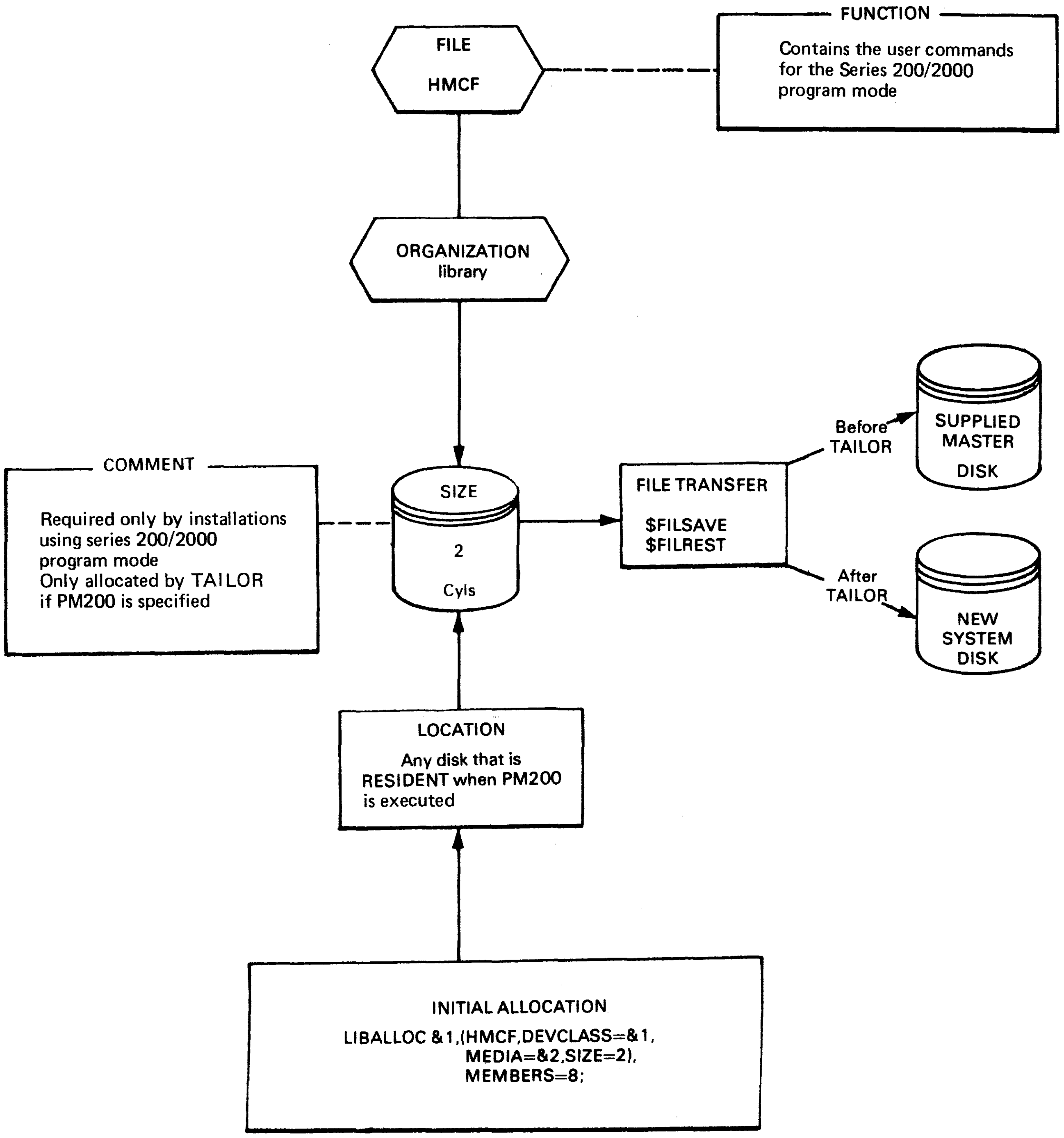
A system disk is built containing only the following optional items in SYS.HLMLIB :

COBOL	1107
SORT	631
HFAS utilities	479
PM200	235
Total :	2512
Add basic system requirement	2895
Total :	5407

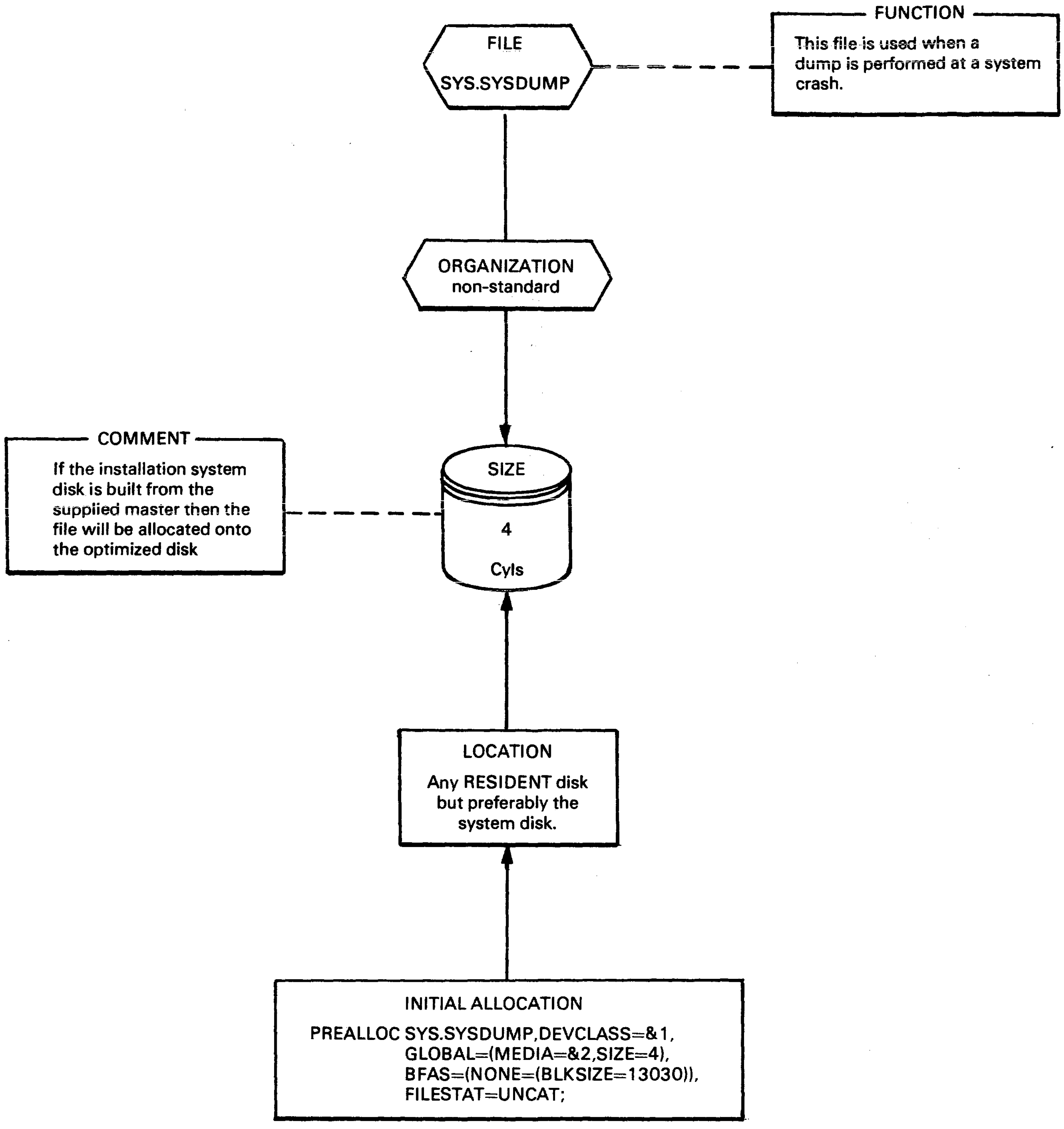
Therefore the size of SYS.HLMLIB is (5407/209)=26CYL.

This value may be specified in TAILOR, resulting in a saving of (50-35)=15 cylinders.

FILE NAME : HMCF

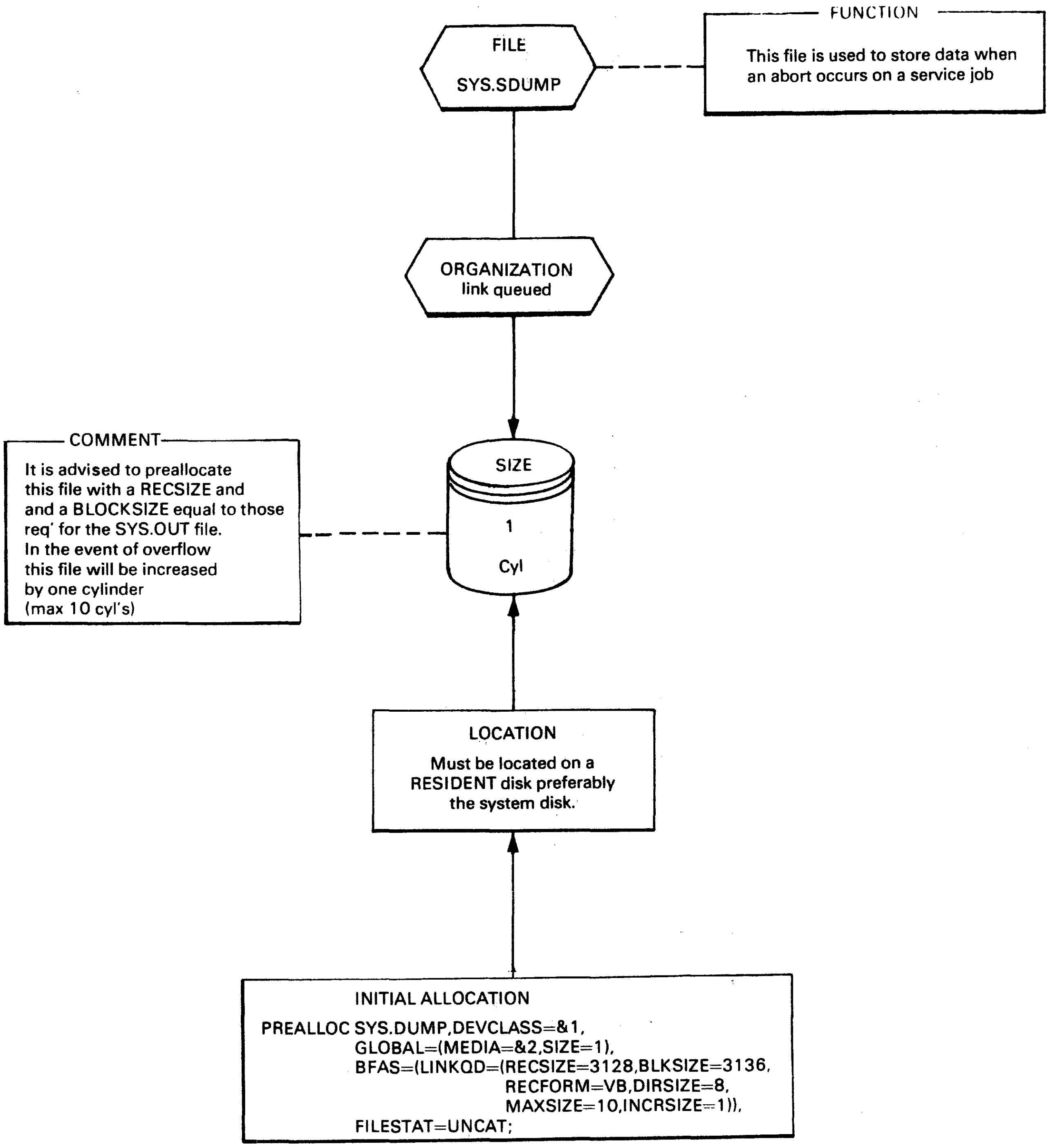

FILE
HMCF
FUNCTION
Contains the user commands for the Series 200/2000 program mode
ORGANIZATION
library
COMMENT
Required only by installations using series 200/2000 program mode
Only allocated by TAILOR if PM200 is specified
SIZE
2
Cyls
FILE TRANSFER
$FILSAVE
$FILREST
Before TAILOR
SUPPLIED MASTER DISK
After TAILOR
NEW SYSTEM DISK
LOCATION
Any disk that is RESIDENT when PM200 is executed
INITIAL ALLOCATION
LIBALLOC &1,(HMCF,DEVCLASS=&1,
MEDIA=&2,SIZE=2),
MEMBERS=8;

FILE NAME : SYS.SYSDUMP


FILE
SYS.SYSDUMP
FUNCTION
This file is used when a dump is performed at a system crash.
ORGANIZATION
non-standard
COMMENT
If the installation system disk is built from the supplied master then the file will be allocated onto the optimized disk
SIZE
4
Cyls
LOCATION
Any RESIDENT disk but preferably the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.SYSDUMP,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=4),
         BFAS=(NONE=(BLKSIZE=13030)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.SDUMP



FUNCTION

This file is used to store data when an abort occurs on a service job

COMMENT

It is advised to preallocate this file with a RECSIZE and and a BLOCKSIZE equal to those req' for the SYS.OUT file. In the event of overflow this file will be increased by one cylinder (max 10 cyl's)

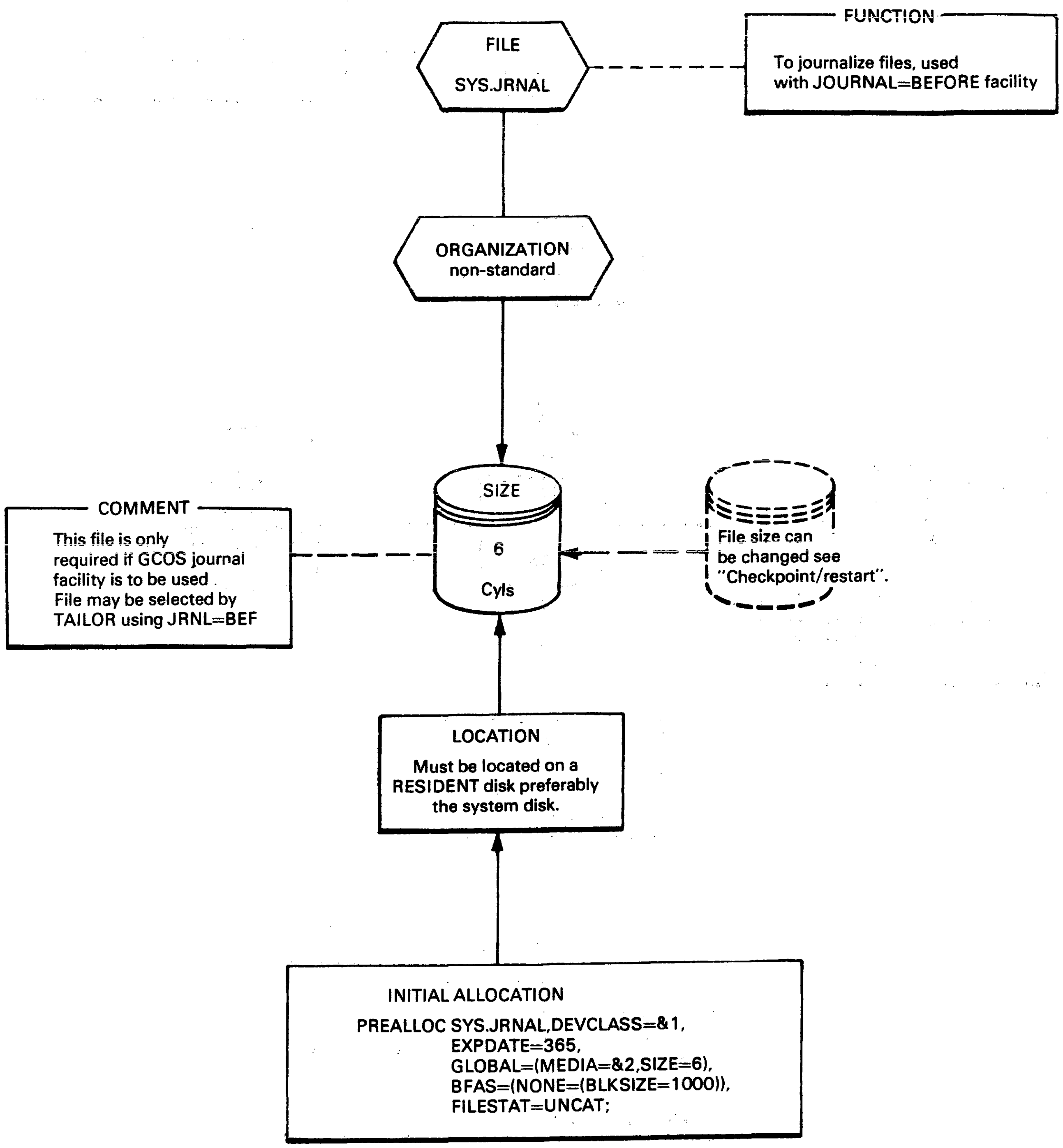
LOCATION

Must be located on a RESIDENT disk preferably the system disk.

INITIAL ALLOCATION

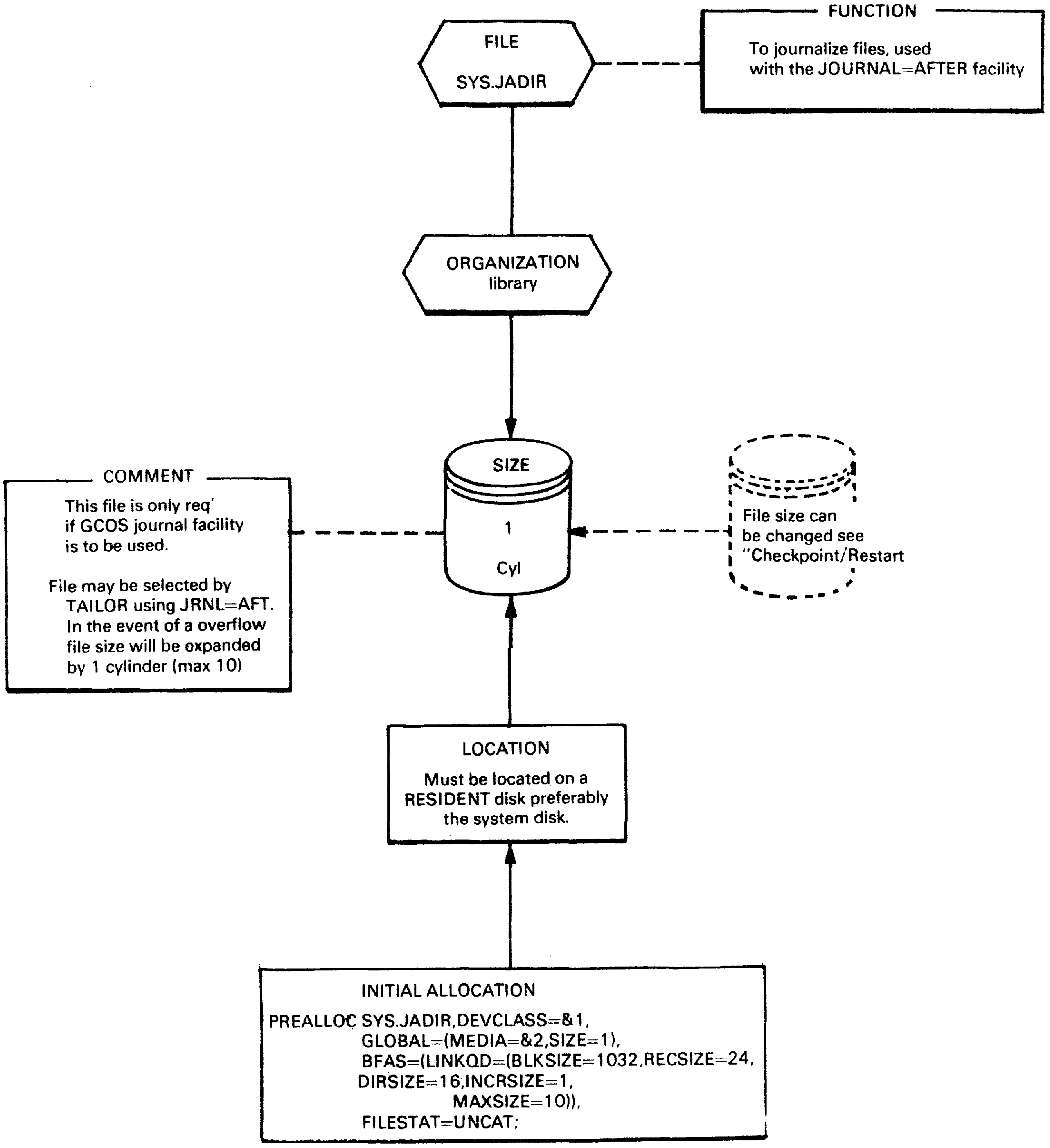
```
PREALLOC SYS.DUMP,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=1),
         BFAS=(LINKQD=(RECSIZE=3128,BLKSIZE=3136,
                       RECFORM=VB,DIRSIZE=8,
                       MAXSIZE=10,INCRSIZE=1)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.JRNAL (Optimized disk only)


FILE
SYS.JRNAL
FUNCTION
To journalize files, used with JOURNAL=BEFORE facility
ORGANIZATION
non-standard
SIZE
6
Cyls
COMMENT
This file is only required if GCOS journal facility is to be used. File may be selected by TAILOR using JRNL=BEF
File size can be changed see "Checkpoint/restart".
LOCATION
Must be located on a RESIDENT disk preferably the system disk.
INITIAL ALLOCATION

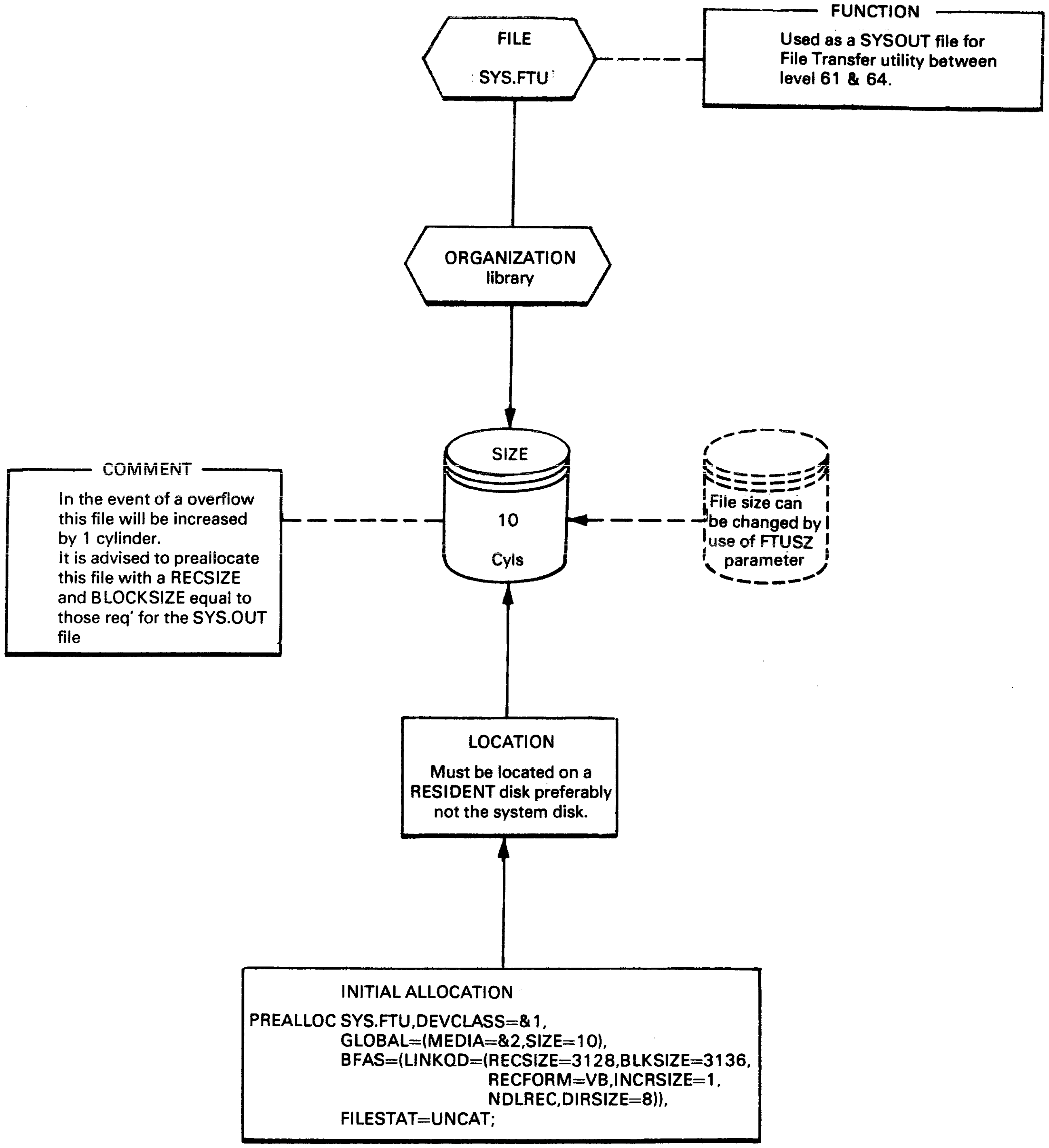
```
PREALLOC SYS.JRNAL,DEVCLASS=&1,
         EXPDATE=365,
         GLOBAL=(MEDIA=&2,SIZE=6),
         BFAS=(NONE=(BLKSIZE=1000)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.JADIR (Optimized disk only)


FILE
SYS.JADIR
FUNCTION
To journalize files, used with the JOURNAL=AFTER facility
ORGANIZATION
library
SIZE
1
Cyl
File size can be changed see "Checkpoint/Restart
COMMENT
This file is only req' if GCOS journal facility is to be used.
File may be selected by TAILOR using JRNL=AFT. In the event of a overflow file size will be expanded by 1 cylinder (max 10)
LOCATION
Must be located on a RESIDENT disk preferably the system disk.
INITIAL ALLOCATION

```
PREALLOC SYS.JADIR,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=1),
         BFAS=(LINKQD=(BLKSIZE=1032,RECSIZE=24,
        DIRSIZE=16,INCRSIZE=1,
                   MAXSIZE=10)),
         FILESTAT=UNCAT;
```

FILE NAME : SYS.FTU (Optimized disk only)



FUNCTION

Used as a SYSOUT file for File Transfer utility between level 61 & 64.

COMMENT

In the event of a overflow this file will be increased by 1 cylinder.
It is advised to preallocate this file with a RECSIZE and BLOCKSIZE equal to those req' for the SYS.OUT file

File size can be changed by use of FTUSZ parameter

LOCATION

Must be located on a RESIDENT disk preferably not the system disk.

INITIAL ALLOCATION

```
PREALLOC SYS.FTU,DEVCLASS=&1,
         GLOBAL=(MEDIA=&2,SIZE=10),
         BFAS=(LINKQD=(RECSIZE=3128,BLKSIZE=3136,
                       RECFORM=VB,INCRSIZE=1,
                       NDLREC,DIRSIZE=8)),
         FILESTAT=UNCAT;
```

5. Job accounting facilities

The accounting feature of GCOS records all relevant account information for each job and each job step. The information being gathered by the system within a user sequential file which is handled by the utility program known as BILLING.

The BILLING utility is a COBOL written program which is tailored by the user to meet his own requirements. Each job is fully identified by a user name, a project name and a billing name; any job failing on one or more of these identity checks will be refused access to the system. This facility of automatic checking and account security is provided by the Site Catalog.

The job accounting feature of GCOS can be either active or inactive depending on the system configuration parameter. Additional types of record can be stored in the account file; these must also be specified at system configuration time.

ACCOUNTING FILE DESCRIPTION

All accounting information is first gathered within two system files located within the backing store and operating in flip-flop manner. These Virtual Memory Management (VMM) files are defined as system permanent files with short blocks of 1K each, with only one of the two files being active at any point of time. These files are named ACT1 and ACT2, their size being specified at system configuration time. When the active file becomes full the operator is informed by the following message :

```
AC01*GCOS:ACCOUNTING FULL.RUN DUMPACT.
```

The system accounting will then continue, using the previously inactive file. If both files are full the older (inactive) file is erased and set to the active state. The operator then has to run the DUMPACT utility program to transfer the full file to the user accounting file. If DUMPACT is run when there is no full file, the active file is automatically closed and dumped to the user accounting file.

USER ACCOUNTING FILE DESCRIPTION

The user accounting file is a sequential tape or disk file with the following characteristics :

```
BLOCKSIZE = 4008
RECSIZE = 4004
RECFORM = VB
```

These characteristics have to be specified at file allocation time (PREALLOC) by using the following parameters.

```
BFAS = (SEQ=(BLKSIZE=4008,RECSIZE=4004,RECFORM=VB).
```

The size of this file must be chosen on the basis of how many accounting records are to be contained. This figure is dependant on how many jobs and job steps are run each day and how many days the same file is to be used to accomodate the output of DUMPACT.

If the user accounting file is to be held on disk then it should be held on a resident volume so that the operator need not load another disk specifically to fulfil the DUMPACT action.

DUMPACT UTILITY

This utility program is used to dump accounting information from the full VMM file, or the currently active file (if neither are full).

The DUMPACT utility provides various facilities. It enables the user accounting file to be opened in output or in an append mode. It can also be used as a dummy file. Furthermore it insures the compatability of user record formats between the 1C and 1D releases.

A message to the operator signifies a successful or unsuccessful DUMPACT completion. If unsuccessful, details will be given within the Job Occurrence Report.

There are two methods of DUMPACT introduction, either by operator command or by JCL use.

Introduction By Operator Command

This procedure is registered in the SYS.HSLLIB

Format :

SJ DUMPACT(efn,devclass,media,filestat,options).

efn :	user file name	
devclass :	device class name	Default : MT/T9
media :	volume name	Default : WORK
filestat :	file system option	Default : UNCAT
option :	$\left[\begin{Bmatrix}\text{AP}\\ \text{OU}\end{Bmatrix}\right]$ $\left[\text{OLD}\right]$	Default : AP and new record format (of the 1D release).

Description of Options :

AP :	User file opened in append mode.
OU :	User file opened in output mode.
OLD :	Record format of the 1C release (usefull to insure a correct compatibility).

Example Operator Command :

SJ DUMPACT (ACCFILE,MS/M400,C113,,'AP OLD")

Introduction By JCL

Format :

```
JOB=jobid, USER=userid, PROJECT=project-name, BILLING=charge-name;
STEP H_DUMPACT,FILE=SYS.HLMLIB
```

,OPTION= '$\left[\begin{Bmatrix}\text{AP}\\ \text{OU}\end{Bmatrix}\right]$ $\left[\text{OLD}\right]$' ;

```
ASSIGN USERACTF, efn, DEVCLAS=devclass,
    MEDIA=media, FILSTAT=filestat;
ENDSTEP;
ENDJOB;
```

This JCL statement can be user tailored to adjust default parameters within the SYS.HSLLIB file.

EDITACT UTILITY

This utility program is written in COBOL 64 and is used to edit in clear form the information stored in the user accounting file. It is activated by use of the following JCL.

Format:

```
JOB=jobid, USER=userid, PROJECT=project name;
STEP H_EDITACT FILE=SYS.HLMLIB;
ASSIGN H_USERACTF,efn,DEVCLASS=devclass,MEDIA=media,
        FILESTAT=filestat;
        ENDSTEP;
        ENDJOB;
```

Definition:

efn :	is the external file name of the user account file.
devclass :	device class of the volume supported
media :	volume name.

ADMINISTRATIVE FUNCTIONS OF ACCOUNTING

Administrative functions refer to the control of the user access to the system. Such functions are under the responsability of a system administrator.

The registration of users within the system, the description of their access rights and different privileges are all part of a integrated privacy protection system. This system is implemented within the GCOS catalog.

Description Of User Registration

A user is identified within the system by a user name of up to eight characters. Such a user name must be unique, i.e. all the users known by the system have different user names.

A user name is always attached to a project name of up to eight characters. A project name gathers different user names. A project name is attached to a billing name of up to eight characters. A billing name gathers several project names, see Figure 5-1.

This control structure is stored within the system catalog. Generally a user has a default project a project has a default billing, but a user may exist under different projects as well as a project may exist under different billing names.

The project is related to the protection mechanism in general.

The billing name is related to the billing aspect.

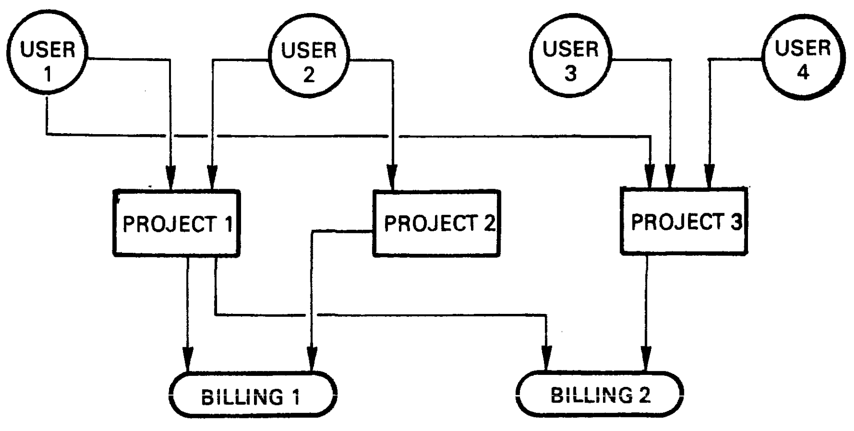


Figure 5-1. The User Registration Structure

control Of Job Submission

When the user relevant area of the SYSTEM CATALOG is active, the system automatically checks each introduced job rejecting any with invalid user, project and billing names.

This checking procedure is performed by the Stream Reader in accordance with the following rules :

USER :	is it registred?
PROJECT :	optional, it overrides the default one. Is the user registered under this project?
BILLING :	Optional, it overrides the default one. It the user registered under this billing.

User Registration

The operation that introduces a new user, a new project, a new billing or modifies file access rights and operability privilages of a project is under the responsibility of the System Administrator. Such an administrator is a particular operator that belongs to a project with specific access rights.

User registration uses the GCOS 64 Catalog Maintenance utility $CATMAINT described in Appendix A of this manual. Reference should be made to the Catalog Management Manual for all other general maintenance utilities.

DESCRIPTION OF ACCOUNTING RECORDS

Accounting information is gathered at each of the different steps of a job's life and at critical states of the system i.e. shutdown, crash, and system exception. This information consists of various types of accounting records, each one being identified by a record type. These records are stored sequentially within the user accounting file by the DUMPACT utility program.

- Job record (type=01) :
  accounting information concerning overall job execution including the print out of SYSOUT files.
- Step record (type=02/03) :
  accounting information concerning the execution of a step.
- Crash record (type=04) :
  gives a status of the system at the last crash.
- Shutdown record (type=05) :
  gives a status of the system at the last shutdown.
- Exception record (type=06) :
  gives information about an exception that occurs within a system procedure called by a user program.
- User defined record (type=user defined) :
  a record directly written by a user program by means of the $H_PUTACT primitive of the COBOL call to the H_ACT_UWRACT system procedure.

Accounting records can be selected through a system configuration parameter at system configuration time. So only the selected records are stored in the accounting file.

A system crash may imply that some accounting information is lost. However only Step record information may be missing (or not guaranteed) in the case of a step being executed at crash time. In such a case, the system will generate a Step record of a specific type in order to allow the user billing job to overcome this problem. Consequently job records may possibly be duplicated. This will occur when the system cannot retrieve a job record, of a given type. This retrieval would be necessary to store accounting information related to a sysout printout resulting from the switching of VMM files.

The 0400 release introduces new accounting information. Therefore the different record formats are not compatible with those of the 0300 release. In order to allow already existing billing jobs to run without having to be modified, the DUMPACT utility provides a choice of either 0300 or 0400 format records depending on the option used (i.e., whether or not OLD is specified).

Job Record Description

At job termination a record is added to the accounting file. This record is updated when the output writer completes its work for this job.

GPL DECLARATION OF THE JOB RECORD :

```
DCL1          JOB_RECORD,
           2  TYPE                     CHAR(2),/*TYPE="01"*/
           2  BILLING                  CHAR(8),
           2  USER_NAME                CHAR(8),
           2  JOBID                    CHAR(8),
           2  PROJECT                  CHAR(8),
           2  RON                      CHAR(4),
           2  CLASS                    CHAR(1),
           2  PRIORITY                 CHAR(1),
           2  DATE_In_CHAR             CHAR(10),
           2  DATE,
              3   MM                   bit(16),
              3   DD                   bit(16),
              3   YY                   bit(16),
           2  START                    CHAR(8),
           2  STOP                     CHAR(8),
           2  CPU                      FIXED BIN (31),
           2  ELAPSE                   FIXED BIN(31),
           2  STATUS                   FIXED BIN(31),
           2  MACHINE_ID               CHAR(21),
           2  LINES                    FIXED BIN(31),
           2  PAGES                    FIXED BIN(31),
           2  CARDP                    FIXED BIN(31),
           2  CARDR                    FIXED BIN(31),
           2  CPU_WRITER               FIXED BIN(31);
```

Total record length : 125 bytes.

COBOL DECLARATION OF THE JOB RECORD

```
01                JOB_RECORD.
           02     TYPE                 PICTURE X(2).
           02     BILLING              PICTURE X(8).
           02     USER_NAME            PICTURE X(8).
           02     JOBID                PICTURE X(8).
           02     PROJECT              PICTURE X(8).
           02     RON                  PICTURE X(4).
           02     CLASS                PICTURE X.
           02     PRIORITY             PICTURE X.
           02     DATE_IN_CHAR         PICTURE X(10).
           02     DATE
              03 MM                    USAGE IS COMP_1.
              03 DD                    USAGE IS COMP_1.
              03 YY                    USAGE IS COMP_1.
           02   START                  PICTURE X(8).
           02   STOP                   PICTURE X(8).
           02   CPU                    USAGE IS COMP_2.
           02   ELAPSE                 USAGE IS COMP_2.
           02   STATUS                 USAGE IS COMP_2.
           02   MACHINE_ID             PICTURE X(21).
           02   LINES                  USAGE IS COMP_2
           02   PAGES                  USAGE IS COMP_2
           02   CARDP                  USAGE IS COMP_2.
           02   CARDR                  USAGE IS COMP_2.
           02   CPU_WRITER             USAGE IS COMP_2.
```

Total record length : 125 bytes.

FIELD DEFINITION OF THE JOB RECORD :

TYPE	: Type of the record ("01)
BILLING	: Billing name of the user
USER_NAME	: User_Name
JOBID	: Job Identification
PROJECT	: Project Identification
RON	: Run Occurrence Number(nnnn)
CLASS	: Class of the JOB (the one at job term time)
PRIORITY	: Scheduling Priority (the one at job term_time)
DATE_IN_CHAR	: Date in Characters (YY/MM/DDbb)
DATE	: Date in numbers MM : month DD : day YY : year
START	: Start time of the JOB (hh:mm:ss)
STOP	: Stop time of the JOB (hh:mm:ss)
CPU	: CPU used by the JOB (unit: 1/1000 minute)
ELAPSE	: Elapsed time (unit: 1/1000 minute)
STATUS	: Step completion code of the last step of the job.
MACHINE_ID	: Name of the Machine as it is printed on the Job Occurrence Report.
LINES	: Number of Lines printed by the job (directly or through the Output Writer)
PAGES	: Number of Pages printed by the job (directly or through the Output Writer).
CARDP	: Number of Cards punched by the job (directly or through the Output Writer).
CARDR	: Number of Cards read by the job (directly or through the Input Reader).
CPU_WRITER	: CPU time used by the Output Writer to print lines or punch cards. (unit 1/1000 minute).

Records of job execution at crash time are fully recovered.

Time Information

The job CPU time is equal to the sum of the values of CPU time for each step. The elapsed time is counted from the initiation of the job by the Scheduler to the termination of the job. So it includes the waiting times of the different steps of the job (waiting for volume mounting, waiting for resources, etc.).

The counters LINES, PAGES, CARDP, and CPU WRITER may be separated onto two job records with one stored in each of the VMM files (ACT1 or ACT2). In such a case, the counters of both records must be added in order to get the correct values. Only the first record (date of creation) contains meaningful information about the job. The second record contains only the preceding counters plus the fields necessary to identify the job (billing name, user name, project, job name, machine id and date). The other fields are initialized to the following values :

CLASS	: blank
PRIORITY	: blank
START	: blank
STOP	: blank
CPU	: 0
STATUS	: 0

STEP Record Description

At step termination time a variable length record is added to the accounting file.

GPL DECLARATION OF THE STEP RECORD

```
DCL1
          STEP_RECORD,
        2 TYPE                    CHAR(2),/*TYPE="02" or "03"*/
        2 BILLING                 CHAR(8),
        2 USER_NAME               CHAR(8),
        2 JOBID                   CHAR(8),
        2 PROJECT                 CHAR(8),
        2 RON                     CHAR(4),
        2 REPEATED                CHAR(1),
        2 PRIORITY                CHAR(1),
        2 DATE_IN_CHAR            CHAR(10),
        2 DATE
          3  MM                   BIT(16),
          3  DD                   BIT(16),
          3  YY                   BIT(16),
        2 START                   CHAR(8),
        2 STOP                    CHAR(8),
        2 CPU                     FIXED BIN(31),
        2 WAITING                 FIXED BIN(31),
        2 READY                   FIXED BIN(31),
        2 ELAPSE                  FIXED BIN(31),
        2 H_STATUS                FIXED BIN(31),
        2 SYSOUT_WRITE            FIXED BIN(31),
        2 SYSOUT_PUNCH            FIXED BIN(31),
        2 DECLARED_WS             FIXED BIN(31),
        2 BUFFER_SIZE             FIXED BIN(31),
        2 BACKST                  FIXED BIN(31),
        2 PGMISSGNB               FIXED BIN(31),
        2 SYSMISSGNB              FIXED BIN(31),
        2 STACKOV                 FIXED BIN(31),
        2 SSN                     FIXED BIN(15),
        2 DSN                     FIXED BIN(15),
        2 CHKPT_NB                FIXED BIN(15),
        2 CHKPT_MAXSIZE           FIXED BIN(31),
        2 TEMPORARY_SIZE          FIXED BIN(31),
        2 LM_NAME                 CHAR(32),
        2 NB_OF_ENTRIES           FIXED BIN(15),
        2 ENTRY                   (NB_OF_ENTRIES),
          3  IFN                  CHAR(8),
          3  MEDIA                CHAR(6),
          3  NB_OF_LOGEVENTS      FIXED BIN(15),
          3  NB_OF_CONNECTS       FIXED BIN(31)
          3  DEV_TYPE             CHAR(2),
          3  RFU                  BIT(16);
```

Length of the record : 164 + H_NB_OF_ENTRIES *24

There is one entry within the H_ENTRY array per IFN (Internal File Name) assigned by the step with a maximum of 63 assigned IFN's per step.

COBOL DECLARATION OF THE STEP RECORD

```
01
            STEP_RECORD
         02 TYPE                  PICTURE X(2).
         02 BILLING               PICTURE X(8).
         02 USER_NAME             PICTURE X(8).
         02 JOBID                 PICTURE X(8).
         02 PROJECT               PICTURE X(8).
         02 RON                   PICTURE X(4).
         02 REPEATED              PICTURE X.
         02 PRIORITY              PICTURE X.
         02 DATE_IN_CHAR          PICTURE X(10).
         02 DATE
           03 MM                  USAGE IS COMP-1.
           03 DD                  USAGE IS COMP-1.
           03 YY                  USAGE IS COMP-1.
         02 START                 PICTURE X(8).
         02 STOP                  PICTURE X(8).
         02 CPU                   USAGE IS COMP-2.
         02 WAITING               USAGE IS COMP-2.
         02 READY                 USAGE IS COMP-2.
         02 ELAPSE                USAGE IS COMP-2.
         02 STATUS                USAGE IS COMP-2.
         02 SYSOUT_WRITE          USAGE IS COMP-2.
         02 SYSOUT_PUNCH          USAGE IS COMP-2.
         02 DECLARED_WS           USAGE IS COMP-2.
         02 BUFFER_SIZE           USAGE IS COMP-2.
         02 BACKST                USAGE IS COMP-2.
         02 PGMISSGNB             USAGE IS COMP-2.
         02 SYSMISSGNB            USAGE IS COMP-2.
         02 STACKOV               USAGE IS COMP-2.
         02 SSN                   USAGE IS COMP-1.
         02 DSN                   USAGE IS COMP-1.
         02 CHKPT_NB              USAGE IS COMP-1.
         02 CHKPT_MAXSIZE         USAGE IS COMP-2.
         02 TEMPORARY_SIZE        USAGE IS COMP-2.
         02 LM_NAME               PICTURE X(32).
         02 NB_OF_ENTRIES         USAGE IS COMP-1.
         02 H_ENTRY OCCURS 63 TIMES
           03 IFN                 PICTURE X(8).
           03 MEDIA               PICTURE X(6).
           03 NB_OF_LOGEVENTS     USAGE IS COMP-1.
           03 NB_OF_CONNECTS      USAGE IS COMP-2.
           03 DEV_TYPE            PICTURE X(2).
           03 FILLER              USAGE COMP-1.
```

Length of the record : 164+H_NB_OF_ENTRIES X 24.

There is one entry within the H_ENTRY array per IFN (Internal File Name) assigned by the step and a maximum of 63 assigned IFN's per step.

FIELD DEFINITION OF THE STEP RECORD :

TYPE	: Type of record ("02" is the normal case and "03" refers to a record of a step that was recovered after a crash).
BILLING	: Billing name of the user
USER_NAME	: User_Name
JOBID	: Job Identification name
PROJECT	: Project identification name

RON	: Run Occurrence Number (nnnn)
REPEATED	: Blank : Step not Repeated -R : Step repeated (after an abort or a crash).
PRIORITY	: Scheduling Priority (the one at step term_time).
DATE_IN_CHAR	: Date in Characters (YY/MM/DD)
DATE	: Date in numbers MM : month DD : day YY : year
START	: Start time of the step (hh:mm:ss)
STOP	: Stop time of the step (hh:mm:ss)
CPU	: Running time of the step : CPU used (unit=1/1000 minute).
WAITING	: Waiting time of the step : waiting for I/O's completion or semaphore resources (unit=1/1000 minute).
READY	: Ready time of the step : waiting for CPU allocation (unit=1/1000 minute).
ELAPSE	: Elapsed time (unit=1/1000 minute)
STATUS	: Step completion code of the last step of the job.
SYSOUT_WRITE	: Number of lines written onto a Sysout file (to be printed).
SYSOUT_PUNCH	: Number of cards written onto a Sysout file (to be punched) access method.
DECLARED_WS	: Declared working set in bytes.
BUFFER_SIZE	: Buffer size effectively used in bytes.
BACKST	: Size of Backing Store used in bytes.
PGMISSGNB	: Number of program missing segments.
SYSMISSGNB	: System missing segments due to the program
STACKOV	: Number of Stack Overflows
SSN	: Static Step Number
DSN	: Dynamic Step Number (including job enclosure statements).
CHPT_NB	: Number of calls to Check point.
CHKPT_MAXSIZE	: Check point largest snapshot size for the step (unit= byte).
TEMPORARY_SIZE	: Size of Temporary used by the step (unit=cylinder).
LM_NAME	: Load Module Name of the step.
H_NB_OF_ENTRIES	: Number of Entries in the array H_ENTRY.
H_ENTRY	: One Entry per IFN (Internal File Name) on which I/O accounting is performed.
IFN	: Internal File Name.
MEDIA	: Volume serial number.
NB_OF_LOGEVENTS	: Number of Events detected on this file and Logged.
NB_OF_CONNECT	: Number of I/O's related to the file.
DEV_TYPE	: Type of the Device related to the file.

Records of steps in execution at crash time may not be fully recovered. Information that could be lost or possibly meaningless can be expected in the following fields :

CPU
ELAPSED
LINES
CARDS
BUFFER_SIZE
BACKST
PGMISSGNB
SYSMISSGNB
H_ENTRY array

Such records whose information can not be guaranteed are identified by a record type="03".

Time information

Time information (like CPU time, waiting time, ready time and total elapsed time of the step) is counted starting from the end of the step initiation (after having allocated all the resources and loaded the program into memory), to the end of the step termination (after having deallocted all the resources). Therefore all the time spent within the user step space is counted, but the time spent for the step within centralized tasks of the system is not counted.

The difference between the job elapsed time and the sum of the different step elapsed times gives the time spent by the job in waiting for resources and media mounting and within centalized system tasks (resource allocation and program loading).

Crash or Shutdown Record Description

After a crash or a shutdown the warm restart adds a system record to the accounting file.

GPL DECLARATION OF THE CRASH/SHUTDOWN RECORD;

```
DCL 1   SYSTEM_RECORD,
          2  TYPE                  CHAR(2),/*Type="04"*/
          2  DATE_IN_CHAR          CHAR(10),
          2  DATE,
             3  MM                 BIT(16),
             3  DD                 BIT(16),
             3  YY                 BIT(16),
          2  TIME                  CHAR(8),
          2  JOB_LIST_NUMBER       FIXED BIN(15),
          2  JOB_LIST(20),
             3  BILLING            CHAR(8),
             3  USER_NAME          CHAR(8),
             3  JOBID              CHAR(8),
             3  PROJECT            CHAR(8),
             3  RON                CHAR(4),
             3  CLASS              CHAR(1),
             3  STAGE              CHAR(1),
             3  SSN                FIXED BIN(15),
             3  DSN                FIXED BIN(15),
             3  STATUS             FIXED BIN(15),
             3  FILLER             BIT(16);
```

Length of record : 948 bytes

There is one entry within the JOB_LIST array per job in execution or suspended (EX or SUSP states). If there are more than 20 jobs, another record is created. Therefore several crash records related to the same crash may occur.

COBOL DECLARATION OF THE CRASH/SHUTDOWN RECORD

```
01      SYSTEM_RECORD.
   02      TYPE                         PICTURE X(2).
   02      DATE_IN_CHAR                 PICTURE X(10).
   02      DATE
      03   MM                           USAGE IS COMP-1.
      03   DD                           USAGE IS COMP-1.
      03   YY                           USAGE IS COMP-1.
   02      TIME                         PICTURE X (8).
   02      JOB_LIST_NUMBER              USAGE IS COMP-1.
   02      JOB_LIST OCCURS 20 TIMES
      03   BILLING                      PICTURE X(8).
      03   USER_NAME                    PICTURE X(8).
      03   JOBID                        PICTURE X(8).
      03   PROJECT                      PICTURE X(8).
      03   RON                          PICTURE X(4).
      03   CLASS                        PICTURE X.
      03   STAGE                        PICTURE X.
      03   SSN                          USAGE IS COMP-1.
      03   DSN                          USAGE IS COMP-1.
      03   STATUS                       USAGE IS COMP-1.
      03   FILLER                       USAGE IS COMP-1.
```

Length of record : 948 bytes.

FIELD DEFINITION OF THE CRASH/SHUTDOWN RECORD :

TYPE	: Type of the record ("04" in case of a crash and "05" in case of a shutdown).
DATE_IN_CHAR	: Date in literals (YY/MM/DD)
DATA	: Date in numerals : MM month DD day YY year
TIME	: Crash Time or shutdown time (hh:mm:ss)
JOB_LIST_NUMBER	: Number of entries within the JOB_LIST array (equal to 20). When it exceeds 20 another record is created.
BILLING	: Billing Name
USER_NAME	: User Name
JOBID	: Job Identification Name
PROJECT	: Project Identification Name
RON	: Run Occurrence Number ('Xnnn)
CLASS	: Class of the job
STAGE	: S if the job was suspended E if the job was in execution
SSN	: Static Step Number of the step that was in execution.
DSN	: Dynamic Step Number of the step that was in execution.
STATUS	: Step completion code of the step that was in execution

There may be several records related to the same crash or shutdown when more than 20 jobs are listed.

User Defined Record Description

A user program can write accounting records of its own by using the $H_PUTACT primitive in GPL or the external call to the system procedure "H_ACT_UWRACT" in COBOL.

GPL PRIMITIVE

```
H_PUTACT record, length;

record   : i_char(n), record to be written onto the accounting file. It must be of
           the following format :
           dcl 1 record,
                 2 type char(2),/*Record type*/
                 2 info char(2);/*Accounting information*/
length   : i_fb(15), length of the Accounting information in bytes.
```

COBOL CALL

Data description statement :

```
01      USER_RECORD.
    02    TYPE      PICTURE X(2).
    02    INFO      PICTURE X(n).

    77    LENGTH    USAGE IS COMP-1.
```

Call statement :

```
CALL "H_ACT_UWRACT" USING USER_RECORD,LENGTH.
```

where LENGTH=n (length of INFO in bytes).

Parameter Description

The record type is specified by the user program. The range 0 to 49 being reserved for system use. The INFO area is to be defined by the user. Its length is given in the LENGTH parameter, with a maximum of 800 characters.

RETURN CODES :

```
NORMAL    : DONE
ABNORMAL : RECSZERR : INFO area length exceeds 800 characters.
           RECFERR  : record type is not included within the range 50 to
                      99.
```

GPL DECLARATION OF A USER RECORD :

```
DCL 1      USER_RECORD,

           2 TYPE          CHAR(2),/*TYPE GIVEN BY THE USER*/
           2 USER_INFO     CHAR(n);
```

COBOL DECLARATION OF A USER RECORD :

```
01         USER_RECORD.

  02       TYPE                      PICTURE X(2).
  02       INFO                      PICTURE X(n).
```

The record type is specified by the user program within the range 50 to 99. The INFO area is to be defined by the user (length and contents).

SYSTEM CONFIGURATION OPTIONS OF THE JOB ACCOUNTING FEATURE

A configuration option allows the customer to specify whether he wants the job accounting facility or not, and to select what records he wants to be stored within the job accounting file. Note that the job accounting may slightly degrade some performances of functions related to job management. For further details see Section II of this manual.

Size of the Two System accounting Files

The size of the VMM files used in the flip-flop by the job accounting facility can be adjusted at system configuration time. Such an adjustement implies a CLEAN restart. It may be useful because the size of these files may be too big or too small depending on the number of records that are selected through the $ACCOUNT parameter. If it is too big, lose of valuable space will occur within the backing store. If it is too small the DUMPACT utility will be in constant use due to rapid file filling.

CONFIG File Description

```
  $ACTSIZE size;
```

Where size is the number of blocks (1K each for each VMM accounting file). Default value : 200 blocks for each VMM file.

6. File Integrity

FILE RECOVERY FACILITIES

Once a file is created, various incidents can occur during its existence that can effect the integrity of the contents of the file. The most common incidents are:

- The volumes on which the file resides are damaged, and the file is no longer accessible.
- There is a system crash, and the file is left in an unstable state.
- There is a step abort, and the file is left partially updated.
- An erroneous update is made to the file.

To reduce the effect of these incidents, there are various recovery facilities available to the GCOS 64 user. The purpose of these facilities is to bring files back to a state from which the user can run jobs again. They are designed so that the user does not lose work already done, and also does not lose large parts of jobs that were running when the incident occurred. By using these facilities the user can:

- Restart the job that was just working on the file.
- Start jobs which want to work on the file.
- Restart jobs which were running.

The facilities available are:

- The Before Journal
- The After Journal.
- The File Salvager.
- Checkpoint/Restart.

Each one of these facilities protects against some type of incident, and the speed of the restart depends upon the facility chosen. Each one is described briefly below, and a more detailed description follows later in the section.

The Before Journal

The Before Journal protects against software failures such as a system crash, a step abort, and a TPR abort (Transaction Processing Routine) in a TDS (Transaction Driven Subsystem) environment. It brings files back to their state at the last recovery point (beginning of step, last checkpoint, or beginning of TPR), and it allows the step or TPR to be repeated immediately.

Example of Use:
A user is updating a file, and the new contents of a record depend upon the previous contents. A software incident (step abort, system crash, etc.) occurs.

If the step is restarted without a Before Journal, the user program will perform the update again. As the file has already been updated before the incident occurred, and as the new contents of a record depend upon the previous contents, the final result will be incorrect (processing of the records up to the point of the crash will occur twice).

If the file was journalized with a Before Journal, after the incident occurs, the file will be reset to its state at the last recovery point. This resetting is automatic, and the repeated step will give the correct results. The Before Journal applies to all file organizations.

The After Journal

The After Journal protects files against:

- Hardware failures (the volume containing the file is damaged).
- Erroneous update by the user program.
- Software failures caused by a system crash or a step abort. In conjunction with deferred update, it protects against TPR aborts in a TDS environment (see the *TDS/64 User Guide* for details).

The operation which brings files to the state from which the user can restart jobs is called roll forward, and the After Journal has been designed to make this operation as automatic as possible.

The user has to save the contents of the files regularly in order to have a correct version of the files available. These saved versions are restored, and the After Journal is applied using rollforward to leave the files in the state they were in when the incident occurred.

File Salvager

The salvager protects files against system crashes. After a system crash, files may be in an unstable state and inaccessible to the system. For example, some control fields may have been updated and the file structure is not consistent.

At warm restart, the salvager scans all the files and volumes that were in use when the crash occurred, and makes them accessible to the system. It should be noted that it does not guarantee the integrity of the data in the files.

Example of Use:
If a system crash occurs while a user is adding new records to a file, and if the file is not closed (no EOF record is written), the salvager will find the end of the file and write an EOF record. It will not bring the file back to its original state, before updating started.

Note that the UFAS system has its own integral file salvager, and this is automatically activated when the file is re-opened.

Checkpoint/Restart

The Checkpoint/Restart facility allows a step which accesses a file in append mode to be restarted without the use of a journal. This is done by taking a checkpoint as soon as the file is opened. Note that the Checkpoint/Restart facility does not protect files being used in update mode; it only allows a restart of the step and does not bring the file back to its original status.

Summary

Figure 6-1 gives a pictorial representation of the two types of Journal.

CHOOSING A RECOVERY FACILITY

There are three factors which affect the choice of facility:

- The type of incident that the file is to be protected from
- The overhead incurred when using the facility.
- The time needed to restart the job.

Table 6 - 1 summarizes the characteristics of the four recovery facilities. Note that more than one facility can be used at a time.

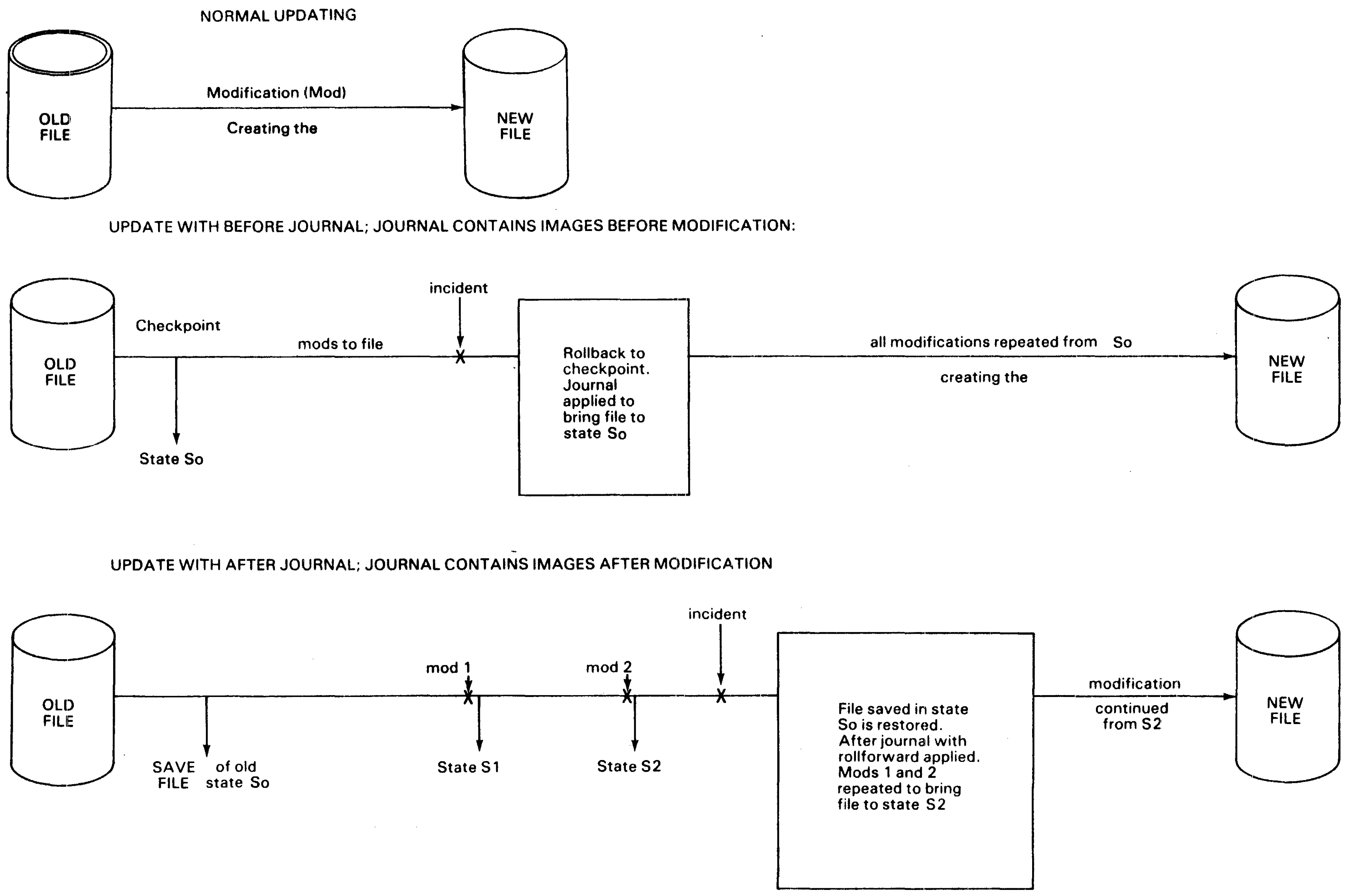

NORMAL UPDATING
OLD FILE
Modification (Mod)
Creating the
NEW FILE
UPDATE WITH BEFORE JOURNAL; JOURNAL CONTAINS IMAGES BEFORE MODIFICATION:
OLD FILE
Checkpoint
State So
mods to file
incident
Rollback to checkpoint. Journal applied to bring file to state So
all modifications repeated from So
creating the
NEW FILE
UPDATE WITH AFTER JOURNAL; JOURNAL CONTAINS IMAGES AFTER MODIFICATION
OLD FILE
SAVE FILE
of old state So
mod 1
State S1
mod 2
State S2
incident
File saved in state So is restored. After journal with rollforward applied. Mods 1 and 2 repeated to bring file to state S2
modification
continued from S2
NEW FILE

Figure 6-1 A Representation of Before and After Journal

Table 6 - 1 Criteria for Choosing a Recovery Facility

	Incident Protected Against							
Integrity Facility	TPR abort	Step abort	System crash	Volume failure or erroneous up-date	Overhead	Restart Time and Effect	Special Configuration Requirements	Comments
Salvager	NO	NO	YES	NO	Only during warm restart – none during step execution.	File guaranteed accessible, but data not guaranteed correct; restart not always possible.	NONE	
Before Journal	YES	YES	YES	NO	Double the number of I/O's needed when updating a file.	Immediate at the last recovery point and fast.	A file of 6 cylinders allocated on the system disk.	Re-used every time a recovery point is taken. Journalization at block level.
After Journal Alone	NO	YES	YES	YES	Medium	Deferred and long as it needs a file restore.	Space on the system disk for the Journal Directory + one MTH if on tape or space on disk to allocate a sequential file.	Journalization at record level. Used for UFAS files in a TDS environment.
After Journal + Deferred Update (TDS only)	YES	YES	YES	YES	Medium	Immediate at the beginning of abortal TPR's and last.		Available for TDS only.
Checkpoint Restart	NO	YES	YES	NO	Low	Immediate at the last checkpoint.		Used only in batch environment. May protect files in append mode if the checkpoint is taken immediately after the OPEN.

THE BEFORE JOURNAL

The Before Journal is an independent system facility; when requested by the user, it ensures that when an abnormal termination occurs, files will be automatically restored to their previous state. This previous state is the state of the file before the step started for both a step which is not restartable, and for a step which has not defined a new restart point.

When a step defines a new restart point (i.e. takes successful checkpoints), the previous state is the state of the file at the current restart point. Note that the original state of a journalized file is lost every time a step takes a checkpoint and cannot subsequently reach its normal termination (after any number of possible restarts).

At step restart time (after an abnormal termination, a Terminate Job or a crash) the operator may have the journalized files restored without a step repeat. Note that the files are restored to their state at the last checkpoint, if any. It is the responsibility of the user to explicitly request that files are journalized; if they are not, checkpoint will not ensure the restoring of the contents of the files before any possible restart. Files (except UFAS sequential disk files) that are accessed in append mode cannot be journalized. The program should take a checkpoint just after the opening of such a file and before any WRITE occurs on it.

The Before Journal acts in two ways:

- JOURNALIZE: The contents of user records prior to their update are recorded in system file, SYS.JRNAL.
- ROLLBACK: In the case of abnormal termination of a step (either abort or system crash), the journal may be used to rollback (to reset) the files to their contents at the last restart point; either at the beginning of the step if there are no checkpoints, or at the last checkpoint.

Note that the Before Journal does not protect the user against physical destruction of the volume on which the file is recorded.

The use of the Before Journal will produce messages on the JOR (Job Occurrence Report) and the console. These messages are fully described in the *System Error Messages and Return Codes* and the *System Operation Console Messages* manuals.

The Before Journal File

The Before Journal is a system file named SYS.JRNAL (see SYS.JRNAL, Section IV of this manual). This file is divided logically into subfiles, and the maximum number of active subfiles is site dependent. It can vary between 3 and 19, and the default is 8.

Several steps may journalize simultaneously while others, which aborted, are rolling back their files. The Before Journal file contains only the before images taken from the last restart point (either beginning of step or checkpoint). After a checkpoint or at the beginning of a step or a TPR, the new images erase the previous ones.

The Before Journal file will overflow if the space required by the Journalized records exceeds the size of the file. In this case, an automatic extension of the file (for each step that overflows from the first, shared extent) is performed, which allows up to 15 extensions of 6 cylinders of the journal for a single step, and 31 extensions for the whole system. Note that the limit of 31 extensions can be reached with three steps journalizing simultaneously, depending upon the size of the journalized records and the number of journalized records for each step.

The first extension of the file is made automatically on the system disk if there is enough room on it. Subsequent extensions are made after a question on the console asks the operator the name of the volume on which the extension should be made. Note that the volumes on which the extensions are to be made must be of the same type as the system disk.

When the next checkpoint or the normal end of the step is reached, all the extensions belonging to the step are destroyed, and the disk space is released.

In case of abnormal termination, all the extensions are destroyed and the space released after the rollback occurs.

To help you judge the appropriate size for the Before Journal file, the maximum number of tracks that have been used to journalize is indicated in the JOR.

The Before Images

These are recorded on the journal in time sequence, and are used in reverse order at Rollback.

To guarantee the integrity of the journalized files, the images are recorded before each modification of the user file.

The images belonging to all journalized files are stored in the same Journal file.

Journalized File Organizations

Table 6 - 2 below shows when the Before Journal may be used, according to the file organization and processing mode.

Table 6 - 2 Before Journal Use Details

File organization	Processing mode		
	Output	Append	Update
UFAS			
Sequential tape	NO	NO	–
Sequential disk	NO	YES	YES
Relative	YES	–	YES
Indexed	NO	NO	YES
BFAS			
Sequential tape	NO	NO	–
Sequential disk	NO	NO	YES
Direct	YES	–	YES
Indexed sequential	NO	NO	YES
HFAS			
Sequential tape	NO	NO	–
Sequential disk	NO	NO	YES
Random	YES	–	YES
Indexed sequential	NO	NO	YES

Requesting Before Journalization ($DEFINE)

To Journalize a file the user must supply in the JCL, for each file to be journalized, the $DEFINE statement :

```
DEFINE ifn, JOURNAL=BEFORE;
```

Where ifn is the internal-file-name on the $ASSIGN statement for the file to be journalized. Note that this request will be ignored if the file SYS.JRNAL is not present.

Programming Considerations

A journalized file should not be deassigned during the step ; however if it occurs before the abort :

— After an abort this file will not be rolled back.

— After a crash this file will not be rolled back except if the file was assigned with the option "PASS".

Note that, in this case, rollback may lead to unpredictable results for the file, if while the file was deassigned for a step, it was assigned and updated in another step.

Rollback Action

Roll back restores the contents of journalized files to their state at the last restart point (last checkpoint, beginning of step or beginning of TPR).

Rollback occurs either after a step abort, at Warm Restart after a system crash, or after a TPR abort. After a TPR abort, rollback is automatic.

The operator decision to perform rollback, or not, must be made when the restart question is answered. It is not possible to defer the decision to a time when, perhaps, the aborted job is no longer present. Note however that the answer to the restart question may be delayed; for further details see the paragraph "Restart Operator Response" later in this section.

All the files journalized in a step are rolled back simultaneously. If an incident occurs on one file at rollback, the rollback continues for the other files.

Either after a step abort or after a system crash a question is sent to the operator:

a) if REPEAT is not specified in the $STEP or $JOB statements, it asks if a roll-back is required.
If the answer is YES, rollback occurs immediately.
If the answer is NO, rollback is inhibited and the journal contents are deleted.

b) if REPEAT is specified in the $STEP statement it asks if a restart is to be done from the last checkpoint.
If the answer is YES, rollback occurs immediately and the step is restarted.
If the answer is NO, rollback is inhibited, the journal contents are deleted, and the step is not restarted.

If the last checkpoint number was 0 (no checkpoint has been taken) the answer may be ROLLBACK; then rollback occurs immediately but the step is not restarted.

At the end of the rollback a message is sent to the JOR. After a system crash, if an incident in the system prevents rollback for all the steps than a message is given on the Rerun Report on the JOR.

THE AFTER JOURNAL

The After Journal (currently available with TDS only) acts in several ways:

- JOURNALIZE: the contents of the user records, after they have been modified, in a system file, the After Journal file.
- JOURNALIZE: in a system file (the After Journal Directory, SYS.JADIR) the list of all the volumes of the After Journal file, the list of all aborted TPRs and the After Journal file characteristics. There is a utility (described later in this section) which will modify the characteristics and edit the contents of the After Journal Directory.
- ROLLFORWARD: When a file is left in an inconsistent state by, for example, a software failure or a volume failure, the After Journal may be used to roll the files forward from the last point of restart to the point at which the incident occurred. Currently, the last point of restart is the last non-aborted TPR.

There are two ways of using rollforward. The first is to call a utility which will bring a file back to its state at the last non-aborted TPR, and re-executes all the logical operations on the file, skipping those made by the aborted TPR and the second is dynamic rollforward, which is an entirely automatic procedure launched at TDS restart in the case of a system crash or TDS abort. The rollforward utility is described in the *TDS/64 Standard Processor Site Manual.*

Note that a file which is updated by batch programs and by TDS will not be protected against volume failure unless a save is made before each TDS session. Note that if the previous session ended in a crash or an abort, the save should be taken after the dynamic rollforward is activated.

The After Journal File

The After Journal file is a file which consists of a set of sequential files (journal files) in undefined format with a default blocksize of 2K bytes. This blocksize may be modified by the After Journal utility, JAGEN. The set of files which constitute the After Journal file are managed by the system, and there is only one After Journal for the whole system, so all steps which journalize in After mode journalize on the same journal. Currently, up to four TDS can journalize simultaneously. The journal files can be on magnetic tape or disk, the medium change being done by JAGEN.

It is up to the user to decide whether the After Journal file is to be on disk or tape, but once journalizing starts, the medium cannot be changed dynamically. See the note below for the procedure for changing the medium.

The default values for the After Journal file are:

- Device type: magnetic tape
- Device characteristics: 9-track 1600 b.p.i.
- Blocksize: 2K records

The JAGEN utility has to be run to change these values.

For an After Journal on disk, the user cannot dynamically change the blocksize, as this is equivalent to a device-class modification.

For an After Journal on tape, the blocksize can be changed dynamically without saving all of the files, and thus the After Journal file on tape can consist of files with different blocksizes. The blocksize change will take place the next time the system starts to journalize.

Note: When the medium is changed (for example, from tape to disk), all the files to be protected by the After Journal must be saved, and the After Journal Directory must be "cleaned" by the JAGEN utility, or by $LIBDELET SYS.JADIR.

The after journal on tape

The After Journal on magnetic tape requires one Magnetic Tape Handler (MTH), even if several steps are journalizing. If there is no MTH available when the first step journalizes, the step will be enqueued until an MTH is available. The resource handling is entirely automatic, and optimizes the use of MTH's for steps that are journalizing.

When the After Journal is used for the first time, the operator has to load a WORK tape, which will then be transformed to a journal tape by the system. The After Journal file can be on several volumes, in which case when the end of the current volume is reached, another work tape must be loaded. This tape, in turn, is converted to a journal tape by the system. To save time, the next work volume can be prepared on another MTH.

Each tape of the After Journal file contains one journal file. Each file will have the name: SYS.JA.tsn, where "tsn" is the sequence number of the tape, and is given by the Journal. The first journal file will be SYS. JA.00000000001, and the number increases for each tape.

When a ROLLFORWARD is launched and a step is journalizing, the current After Journal tape is closed, and a new work tape is required for the journal file. If an incident prevents the current After Journal tape being closed, a new work tape has to be loaded.

It is your responsibility to decide when an After Journal tape can be used again. Once you take this decision, run $VOLPREP to return the tape to work status (see the *Data Management Utilities Manual* for details of $VOLPREP). If a tape is no longer to be used as a journal tape, the user has to inform the system that the after images on this tape are obsolete. This is done by using the JAGEN utility with the REMOVE parameter.

After journal file on disk

To journalize in After mode on disk, the user must run the utility JAGEN to define the list of volumes on which the After Journal file can be created. This utility allocates a sequential journal file on each of the named volumes. The name of these journal files is:
SYS.JA.volume-name

Note that each disk may contain only one journal file.

The After Journal can share disks with other files, but to ensure file integrity, the journal should be on a different disk to the file which is being journalized. If they are on the same, there is no guarantee of protection against volume failure, and thus one of the advantages of using After Journal is lost.

The first step which journalizes requires the operators to load the volume on which the current journal file is allocated. A drive should be available to load this volume, otherwise the step will be queued waiting for a drive to become available.

When the first vo ume is loaded, and journalization begins, a message is sent to the operators, telling them which volume contains the current journal file, and the volume that will contain the next journal file. When the end of the current journal file is reached, a message asking for the next volume to be loaded is sent to the operators, if this volume is not already mounted. If no drive is available for this new volume, all of the steps that are journalizing in After mode are aborted.

When the system starts to use the last volume specified in the list of volumes that support the After Journal, the operator is informed, and can run the JAGEN utility with the APPEND parameter to provide a new list of disks for the journal. If this is not done, and the last disk is used, all the steps that are journalizing in After mode will abort.

Each time a journalizing step is run, the system will:

– Either request the loading of the current journal disk, used for the journal in a previous session.

- Or, if it is the first time the user has journalized, or if the current disk is not closed because of an incident (e.g. a system crash), request the loading of the first disk in the journal volume list.
- Or, if a rollforward is activated, request the loading of the next disk in the volume list, as the current disk is closed while another step journalizes.

Each time volume switching occurs, the operator is given the name of the next volume to be loaded. It is the responsibility of the user to decide when an After Journal file on disk can be erased; the contents of the file are lost as soon as the disk is entered as a new volume in the volume list maintained in the Journal Directory.

The After Journal Directory

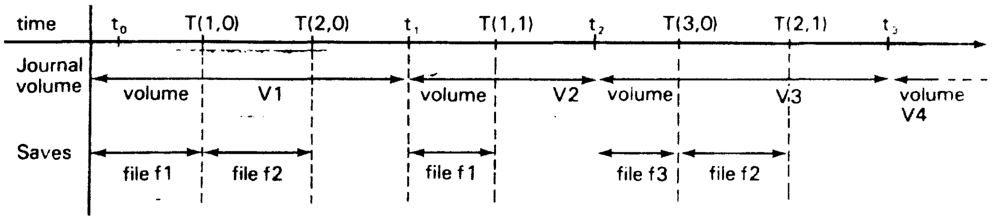
The After Journal Directory is a file which is preallocated on the system disk. If this file does not exist, steps which journalize in After mode will abort. The name of the file is SYS.JADIR, and it contains all the control information for the After Journal files, including:

- The list of all the After Journal files which can be used by rollforward, their beginning and end dates.
- For each one of the After Journal files, the list of all the aborted TPRs journalized on the file. This is done because all the records journalized by these TPRs must not be rolled forward.
- The name of the current journal file and its characteristics.
- If the After Journal file is on disk, the list of the disks on which the After Journal can be created.
- Information on the blocksize distribution if the After Journal is on tape.

The After Journal directory is handled entirely by the system, but it is essential that all restarts after a system crash are warm restarts, so that this file can be salvaged.

Recycling

It is the responsibility of the user to decide when to recycle the After Journal volumes. When to do this will depend upon the time at which saves were taken. For example,



In this example, files f1, f2 and f3 are journalized.

The volume V1 can be recycled only when file f2 is saved again at time T(2.1). The volume V2 can be recycled only when the file f1 is saved again.
If an error is made, for example, volume V2 is recycled during the interval T(3,0), T(2,1), neither file f1 or file f2 is protected. The volume V1 is now obsolete, but file f3 is still protected. If the user wishes to make files f1 and f2 protected again, they must be saved.

Journal Volume Error

If, during rollforward, there is a Read error on the After Journal, GCOS64 does not provide any automatic method of recovery.

The After Images

These are recorded on the journal in time sequence and used in this order by rollforward. The images belonging to all journalized files are recorded on the same journal file.

The images are recorded after the modification has been made to the file.

After Journal File Organizations

Table 6 - 3 shows when the After Journal may be used, according to the file organization.

Table 6 - 3. After Journal Use Details

File Organization	Processing mode		
	Output	Append	Update
BFAS HFAS	NO	NO	NO
UFAS (Batch)	NO	NO	NO
UFAS (TDS)	NO	NO	YES

Requesting After Journalization ($DEFINE)

To Journalize a file in After mode, the user must supply in the JCL, for each file to be journalized, the $DEFINE statement:

$$\text{DEFINE ifn, JOURNAL=}\left\{\begin{array}{l}\text{AFTER}\\ \text{BOTH}\end{array}\right\}$$

If AFTER is stated, the file will be journalized only on the After Journal.
If BOTH is stated, the file will be journalized on the Before and the After Journal.

Note that ifn is the internal-file-name given in the $ASSIGN statement for the file to be journalized.

If the After Directory (SYS.JADIR) does not exist, the step will abort.

If the After Journal requires a new disk, and if the volume list is empty, the step will abort.

If it is a TDS step, journalization is automatic if the required resources are available. If they are not available, the step is queued.

If it is a batch step, the step aborts at file open time.

Rollforward Action

The After Journal has been designed to allow the following:

- Several rollforwards on different files to run simultaneously.
- Rollforward of files and the journalization of other files to occur simultaneously.
- Automatic supply of the volume list required for rollforward.

There are two types of rollforward, static rollforward (called by the rollforward utility), and dynamic rollforward, automatically activated by the system at TDS warm restart time, and used in conjunction with deferred update (see the *TDS/64 User Guide* for details).

The After Journal Utility, JAGEN

Function: To define the physical characteristics of the After Journal File, or to record volumes on remove volumes from the list of volumes that support the After Journal.

Format:

```
$INVOKE JAGEN, SYS.HSLLIB
    VALUES= ,PRINT
            { APPEND, volume1 [,volume2] . . . [,volume10]               }
            { REMOVE, volume1 [,volume2] . . . [,volume10]               }
            { GEN, [DEVCLASS=device-class], [BLKSIZE=blocksize]          } ;
            {      , [SIZE=size], [INCRSIZE=increment-size]              }
```

Statement Description:
This utility is called by the $INVOKE JCL statement, and the parameters are given in the VALUES parameter of $INVOKE. See the *Job Control Language (JCL) Reference Manual* for details.

If the After Journal is on magnetic tape, the introduction or removal of volumes from the list of volumes that support the After Journal can be done automatically by the system, and the utility supplies an additional facility which can be used to clear obsolete journal tapes from the After Journal Directory, or to print the list of journal tapes.

If the After Journal is on disk, the utility must be run to supply the identification and characteristics of the supporting volumes to the system. It is also used to inform the system when a journal file on a given volume (and all journal files previous to this file) are to be released from the After Journal Directory. When a new volume is supplied with this utility, a sequential file for use by the After Journal is automatically allocated by the system.

NOTES:
- When a volume is removed from the After Journal Directory by the utility, all journal files that were created before the file on this volume are automatically cleared from the Directory. This applies for disk and tape.
- Whenever a volume identification is removed from the Directory, ensures that all entries in the Directory are chained. If an entry does not belong in the volume list, it is destroyed.

Parameter Description:

PRINT

This parameter causes the printing of:
- The current characteristics of the After Journal
- The list of disks to be used (for disk only)
- The description of and statistics of each After Journal file

APPEND

This parameter is valid only with disks. It causes the allocation of a sequential file, SYS.JA.volume-n on the disk volume-n. It also records the name of the volume in the After Journal Directory.

REMOVE

When the After Journal is on disk, the given volume names are searched for in the list of volumes to be used and amongst those already used. If the names are found, these volumes are released from the list, and all the previous volumes in the chain are also released. Note that no space deallocation occurs.

When the After Journal is on tape, the volumes given are searched for among the tapes already used. When found, they are released, along with all the previous tapes in the chain.

GEN

This parameter allows the user to choose the characteristics of the After Journal file.

It is not mandatory, because once the After Journal Directory has been allocated, the default values of MT/T9/D1600, blocksize = 2K bytes are assumed by the system. Note: SIZE⩽blocksize⩽2056 bytes.

If it is used, the utility must be run immediately after the allocation of the Journal Directory if the device-class or the device attributes are to be changed.

Any attempt to modify these characteristics will fail if journalization in After mode is current. This means that the Directory has to be cleaned using the REMOVE parameter before the volume type is changed or the characteristics are changed.

The facility to modify the default blocksize of 2K bytes applies to tapes only. Blocksize for disks can only be changed when there is no list of supporting volumes in the Directory.

The facility to modify the size and increment size applies to disk only. An increment size of 0 will cause an error.

A full description of JAGEN and its parameters is given in the *TDS/64 Site Manual.*

FILE SALVAGER

The file salvager is designed to increase the reliability of the system by reducing the effect of system crashes on the integrity of files and volumes. It is called by operator request at warm restart time, and attempts to:

- Make the data and control information recorded in the job management file SYSIN and SYS.JCF accessible. See Section IV for details of these files.
- Allow editing of job output currently recorded in the SYSOUT and SYS.REP files. See Section IV for details of these files.
- Recover the file control information for user files processed in write mode to allow an immediate or deferred rerun of the jobs.

The file salvager ensures only that files are accessible by the system, it does not ensure that the data in the files is correct. If the other integrity facilities (such as Before Journal and Checkpoint/Restart) are not used, an immediate rerun of the job may lead to incorrect results.

The action of salvager on user files depends upon the processing mode and the stage of processing reached when the crash occurred. Library files are always verified, regardless of the processing mode; sequential files processed in output mode or append mode are recovered up to the last effective write; sequential files processed in update mode are not verified. Whatever the intended activity on a file was, salvager will produce a detailed printout giving the file characteristics, the file status, and the intended action when salvaging.

To summarize, the file salvager:

- Builds a list of potentially damaged files
- Warns the operator about those files which are in an unstable state
- Verifies and recovers these files if the operator requests salvaging
- Warns their owner about their characteristics and the stage of processing reached when the system crashed, and informs the owners of the action to be taken when salvaging.

Activation and Control of File Salvaging

File salvaging is activated during a system restart following an emergency shutdown (system crash, power failure, etc.). It provides independent facilities for salvaging system files and user files. These facilities are activated in two ways:

- System file salvaging is automatically requested by job management and runs under the control of job management whenever the operator requests a warm restart. See WARM RESTART/SHUTDOWN in this section.
- User file salvaging is requested by the operator at warm restart time, and runs under his control.

After establishing the list of user files active at shutdown time, the system warns the operator about those which are potentially damaged, and asks if file salvager is to be run. If the answer is no, no salvaging will be performed. If the answer is yes, file salvaging will be activated, and the operator will receive a brief report on the status of each file and the result of the intended correctionsto it. This report is fully described in the *System Operation Console Messages* manual.

The Action of File Salvager

File salvager first informs the owners of files that the files may have been affected by a system crash, and produces a salvaging report containing the names of the owners of the files, the characteristics of the volumes containing the files and the run occurrence number (ron) of the jobs accessing the files. It then declares files as unstable if:

- They are opened in write mode
- They have been dynamically allocated by the jobs using them and the open has not been completed.

The names of these unstable files are sent to the operator with the names of the volumes supporting them and the ron of the jobs processing them.

The next stage of file salvaging is only concerned with these unstable files and then only if they are native GCOS files and monovolume (multivolume file salvaging is supported if only one volume was loaded at shutdown time).

The file organizations supported are library files and sequential disk files. Other organizations are declared as NOT SUPPORTED in the status report.

Depending upon the stage of processing, the following actions are then performed by file salvager.

- If the file is assigned but not yet opened, and the space allocation was performed in a previous step, no action is taken by the salvager.
- If the file is assigned but not yet opened and the assignment took place in the step that was running at shutdown time, the result of the allocation request is verified. The salvager verifies and corrects the space description in the VTOC of the volume, attempts to correct inconsistencies in file labels, and deletes any labels it cannot correct. If the shutdown occurred during a storage allocation procedure, the salvager resets the VTOC indicator which forbids any allocation/deallocation on the volume, but does not verify the complete VTOC. The utility $VOLCHECK should be run on the volume, before any storage allocation/deallocation on this volume, to ensure that the unallocated space is completely updated.
- If the file is opened in write mode, library files are verified regardless of the processing mode. Subfiles which have an entry in the directory are verified from the subfile entry to the last correctly chained block. Subfiles processed in append or output mode are recovered to the point of last effective I/O on the file. Subfiles processed in update mode are protected against data structure inconsistencies caused by an I/O interrupt during an insert or a delete function.

Sequential files are only verified if they were processed in output or append mode, the file is closed at the last effective write; the file is sca[illegible]om the first record and closed either at the first end of file or at the first inconsistency between track balance and effective track allocation status.

The two salvaging utilities, $FILCHECK and $VOLCHECK can be called at any time by the user; they are fully described in the *Data Management Utilities Manual.*

The UFAS File Salvager

The UFAS system has its own integral file salvager. This is called automatically when the file is reopened, and will produce JOR messages prefixed with the code DUF. These messages are fully described in the *System Error Messages and Return Codes Manual*, and the salvager is fully described in the *UFAS User Guide.*

CHECKPOINT/RESTART

The function of the Checkpoint/Restart mechanism is to save an image of the Process Group (job step) and to update the current recovery point according to this saved image. The point of activation for checkpoint, within the user program, is specified by the user.

The image (snapshot) will contain all the step status, data and code segment information which will be required for restart purposes in the event of a step abort. Snapshots saved by the checkpoint facility are stored in a system file known as the checkpoint file. Each snapshot is held in a separate subfile within the checkpoint file. The checkpoint file is shared at system level by all active jobs but will never contain more than one valid checkpoint at a time for any given job. As soon as a new checkpoint is entered in the system file by a job, the system deletes the previous checkpoint for the same job.

The Checkpoint/Restart mechanism means that aborts and restarts are automatic, and thus not visible to the user. However, the System Operator will be made aware of all abort and restart incidents and run time, and full details are given in the Job Occurrence Report.

Restart Functions

The restart facility is a centralized function of the system. When a system or step failure has occured during a restartable step, the step is aborted and returned to the beginning, or a predetermined point during step execution.

During the step termination phase the operator is asked if he wants to restart the step (see *System Operation Operator Guide*, Step Repeat); if the restart is required, a new step initiation phase is entered to re-execute the step from the specific recovery point.

Restartable Jobs and Steps

The following conditions must be satisfied for an aborted step to be restarted. Note that a system crash corresponds to a step aborted for all executing and suspended (held during execution) job steps.

The restart conditions are :

. The job is known to the system and does not have to be resubmitted.

. The step completion severity code is at least 3 and the abort did not result from a "TJ STRONG" operator command.

. All the resources allocated to the step are still allocated. They can be completely released, including the multiprogramming channel (but not the files), if the operator puts the job in the SUSPENDED state by entering a HJ command before answering the REPEAT question.

. The $STEP statement of the aborted step specifies REPEAT.

. The files held by the step to restart are closed and deassigned.

. A restartable step must keep its dynamic step number unchanged until its full termination.

. The step completion code and the job switches must be temporarily restored to the values they had during the first step initiation, thereby allowing the JCL to be reprocessed.

Note: If the $JOB statement specifies REPEAT, this will override $STEP.

Programming For Checkpoint/Restart

The system provides two entry points (routines) through which the programmer may access the Checkpoint facility. Both routines have identical parameters and return codes.

. H_CK_UCHKPT;	this routine causes a snapshot to be taken and provides, via parameters, the result of the action.
. H_CK_UMODE;	this routine reports on the current checkpoint status of the step. It does not cause a snapshot to be taken.

H_CK_UCHKPT Routine

This routine must be invoked to build up and store into the checkpoint file a new checkpoint. The current snapshot of the step is then updated by the new snapshot (provided no severe error condition was encountered). If the update of the current snapshot is successful, the checkpoint which supported the previous snapshot is deleted from the checkpoint file.

If the snapshot cannot be updated, the new checkpoint is deleted.

Consequently the checkpoint file never contains more than one valid checkpoint (containing the current snapshot) for each restartable job step which exists at any given time in the system. All journalized files (or queues of messages) are synchronized with the new snapshot.

Two static parameters (MODE and CKINF) are reset every time H_CK_UCHKPT is entered, and assigned new values when it is exited. Note that H_CK_UMODE can be called merely to read these indicators and return their current values without changing them.

MODE Parameter

This parameter is a 32-bit byte-aligned numeric variable. It informs the user of his current execution mode (normal or restarted). When MODE is set at zero, the execution mode is restarted. This will mean that the step has aborted and has been restarted using the current snapshot.

The execution mode 'restarted' will continue until a new activation of H_CK_UCHKPT is performed. When it is not zero, MODE is set to the value of the step completion code at the time of abort. Hence by checking MODE a program can take specific actions in the event of a restart of a job.

CKINF Parameter

This parameter is a 32-byte character-string variable. Every character of the returned string is set to either "0" or "1" thereby indicating the occurrence of given conditions. If one or more abnormal conditions are met, the corresponding flags are set to "1".

These conditions are listed below. The left-most byte is number 1 and the right-most is 32.

Byte	*Condition when "1"*
1 to 15	Reserved.
16	Major error or malfunction. Refer to the JOR. Corresponds to the return code FUNCNAV.
17 to 23	Reserved.
24	There were operator commands enqueued when the abort happened; the return code is ABNCCAD.
25	There were pending requests for this step in the Internal Timer queue; the return code is ABNCCAD.
26	Reserved.
27	External IDs were found pointing to this step; the return code is ABNCCAD.
28 to 31	Reserved.
32	The next call to H_CK_UCHKPT will fail; the return code is ABNCCAD.

The return code ABNCCAD is delivered with a warning message, which generally means that the checkpoint taken is valid, but it may be necessary to take special action at restart time.

The return code FUNCNAV is delivered with a fatal error message and signifies either an internal system malfunction or, more likely, a checkpoint restriction; multi-task steps (Communications only) cannot use checkpoint.

For further details see the paragraph "Checkpoint/Restart Limitations".

H_CK_UMODE Routine

This routine returns the same information to the caller as H_CK_UCHKPT, but unlike H_CK_UCHKPT it does not cause a snapshot to be taken. H_CK_UMODE is especially useful when the Checkpoint mechanism is called implicitly (as it is by the COBOL RERUN declaration).

The parameters are exactly the same as those for H_CK_UCHKPT.

Coding a Checkpoint

The calling sequences have different forms according to the programming language being used.

Checkpoint With COBOL

A COBOL program can activate H_CK_UCHKPT either directly or implicitly.

a) Implicit call: H_CK_UCHKPT is implicitly activated by the COBOL runtime package when the user program contains a RERUN statement.

The RERUN statement is:

RERUN ON CHECKPOINT-FILE EVERY integer-n RECORDS OF filename-1.

- integer-n is the count of records to be processed between two consecutive activations of H_CK_UCHKPT. In certain cases the count can be slightly exceeded because its value is checked only when an I/O operation is to be initiated;
- filename-1 is the name of the file to which the counted records belong.

b) Explicit call:

The data description statements are as follows

```
       77 mode COMP—2.
       01 ckinf.
          02 ckei PICTURE X OCCURS 32 TIMES.
```

The CALL statement is:

```
CALL "H_CK_UCHKPT" USING mode, ckinf.
```

The RERUN COBOL statement does not activate H_CK_UMODE, but this entry point can be explicitly referenced by a COBOL program which uses a RERUN clause, this action would be taken to find out whether the checkpoints are successful or not and as to what the current mode of execution is.

Checkpoint With FORTRAN

In FORTRAN the access to checkpoint is as follows :

– Data description statements :
  DIMENSION ckinf (32)
  CHARACTER ckinf *1
  INTEGER mode

– The CALL statement is :
  CALL H_CK_UCHKPT (mode, ckinf)

Calls to H_CK_UMODE are entirely similar to the H_CK_UCHKPT calls.

Submitting a Restartable Step

A step is not restartable unless it is explicitly declared as restartable. A step is declared restartable by using the REPEAT parameter in its $STEP JCL statement. Note that the REPEAT parameter of $JOB applies to every step in the job.

Description of Parameters:

NOREPEAT	is the default option ; it specifies that the step cannot be restarted after an abort. When used with a step which calls H_CK_UCHKPT, the H_CK_UCHKPT acts as a dummy procedure giving the return code DONE and the step is not granted a recovery point.
REPEAT	declares the corresponding step as restartable. If a step abort occurs the system will ask the operator whether or not to restart the step from its current recovery point. The first recovery point is automatically established at the beginning of the step. The recovery point is later updated by every call to H_CK_UCHKPT which is successfuly completed. If the DEBUG option is used together with REPEAT, any call to H_CK_UCHKPT will be unsuccessful.

The JOR of a Restartable Step

During the execution of a step certain messages are written into the Job Occurrence Report (JOR) which are related to its property of being restartable.

. Messages for a normal step execution :

```
hh:mm:ss  STEP STARTED
          ..........
          TASK J=01 P=00 COMPLETED
          START  ..........
          STOP   ..........
          CPU    ..........
          ELAPSE ..........
          CHECKPOINT ddd TIMES CALLED
          CHECKPOINT LARGEST SNAPSHOT LENGTH : dddddd
          STEP TERMINATED
```

Messages for a restarted step :

```
hh:mm:ss   STEP STARTED
           ..........
           TASK J=01 P=00 ABORTED G4=UNRECIO-H_GET
           START  ..........
           STOP   ..........
           CPU    ..........
           ELAPSE ..........
           CHECKPOINT ddd TIMES CALLED
           CHECKPOINT LARGEST SNAPSHOT LENGTH : dddddd
           STEP TERMINATED
hh:mm:ss   STEP dsn RESTART AT CHECKPOINT ccc
           ..........
           TASK J=01 P=00 COMPLETED
           START
           STOP
           CPU
           ELAPSE
           CHECKPOINT ddd TIMES CALLED
           CHECKPOINT LARGEST SNAPSHOT LENGTH : dddddd
           STEP TERMINATED
```

The user must be aware that the cost of taking a checkpoint is not negligible in CPU time nor in backing store usage. This is why the system reports the number of calls to H_CK_UCHKPT, and the maximum size (as a byte count) of the total amount of data and code saved at checkpoint. These figures are evaluated for the most recent step initiation (normal or restart), and for the actual sizes of segments (not from their maximum size).

Errors Within Checkpoint

Certain errors within checkpoint are reported as warnings to the operator and the JOR.

The form of the message in the JOR is :

```
SYSTEM ERROR CK01.06   RETURN CODE IS xxxxx FROM xxxxx (G4=xxxx)
                       WHILE IN CHKPT#12 CHECKPOINT ABORTED
```

In this example a system malfunction occurred while a checkpoint was being taken. The first line gives the message number, CK01.06 followed by further detailed status values. The second line shows that the error occurred during checkpoint number 12. Note that the job step accessing the checkpoint will be informed of the error through the CKINF parameter of H__CK__UCHKPT or H__CK__UMODE.

Checkpoint at System Shutdown

When the operator issues an END command for system shutdown, any job steps using checkpoint are automatically suspended when the next successful checkpoint is made. These job steps will then be automatically restarted if the next system session begins with a Warm Restart.

Recovery of Files

The file recovery has two aspects which at first seem to be independent but in fact are related : one can look at the contents of a file and one can look at the positioning of a file. The problem in connection with checkpoint/restart is how to synchronize files with a step to restart. In other words how to get the files back to the state they were in at the time where the restart point was established. The integrity of file contents is the responsibility of the Journal.

FILE POSITIONING

UFAS, BFAS and HFAS ensure a correct repositioning of user files at restart. Other access methods, even "library" (except for SYSOUT/SYSIN files) do not support file repositioning.

Files which were not opened at time of the restart point are not repositioned.

The system only asks for the volumes which were loaded at restart time.

UFAS and BFAS

Sequential, Indexed (Indexed Sequential) and Relative (Direct) files are repositioned to the processing point of checkpoint.

In other words, data management control structures and buffers are restored to this status and in particular, the current pointer is moved back to the checkpoint state.

Note however that the actual contents of the file are those as at abort time and not that at checkpoint, unless before images of updates to the file were taken by the Journal.

As a consequence the following rules apply to repositioning :

– Sequential

  Input, output; file is repositioned to checkpoint state.
  Update: before images should be applied to restore file contents.

– Indexed and Indexed Sequential

  Input, output; file is repositioned to the checkpoint state.
  Update: before images should be applied to restore file contents.

  Relative and Direct

  Input: file is repositioned.
  Output, Update; before image should be applied to restore file contents.

HFAS

– Sequential files: same rules as for UFAS/BFAS.

– Indexed sequential files: same rules as for UFAS/BFAS, when processing mode is Input or Update. When processing mode is Output, records loaded in the file between checkpoint and abort remain in the file but are written again if the program sequence is the same as before abort. If for some reason that sequence is changed, the remaining records which are not erased become inaccessible; so no DUPKEY return code will be received.

– Random files: same rules as for UFAS Relative/BFAS Direct files, except that DUPKEY return code is not delivered in Output processing mode.

Repositioning of Card Reader Input Stream

The card reader file control structures and buffers are restorable to a current checkpoint, but this is not an automatic repositioning facility. This repositioning is an operator responsibility which is assisted by Device Management information detailing the checkpoint card ($CKP) corresponding to the appropriate checkpoint taking.

The checkpoint name is given within a "MOUNT" request and corresponds to the name (checkpoint specific character/s) on the relevent $CKP card.

File Location

Files used at restart must have the same location on volumes as they had at recovery point. It is not possible to relocate a file or to replace it by a duplicate between the abort and the restart. If the extents of a file were modified or if the file was allocated ($ALLOCATE extension) on a volume between the recovery point and the step abort, the file is returned to the step at restart time with the location and the extents it had at the time of abort. Note however that a volume may be mounted on a different device (in the event of device failure).

COBOL MESSAGE CONTROL SYSTEM

Rollback of the Message Control System queues is provided as well as Rollback of journalized files providing the RESTART option has been indicated within the NDL QUEUE command. Such a Rollback consists of a repositioning of the message queues to the state they were in at recovery point. In addition, it keeps all the messages that have been received between the recovery and abort points.

SYSOUT

If a step is declared restartable and if it is to produce immediate SYSOUT deliveries, those deliveries are automatically changed into end of step deliveries.

If a permanent SYSOUT file was opened at recovery point, it is repositioned at restart according to its access mode.

See the *Job Control Language (JCL) Reference Manual* for details.

STEP MANAGEMENT

When a step is restarted its resources are reallocated by referring to the JCL in the step enclosure. For example, the full quota of CPU-time is reallocated regardless as to how much was used prior to the abort.

Checking procedures which were performed during the initial loading are reperformed whilst loading the checkpoint, therefore it is possible that a restart cannot be performed if a resource is dynamically requested (dynamic REQCM) at the recovery point and its requirements are in excess of the (static) JCL specification.

ACCOUNTING

Accounting records are written at step termination, whether the termination is normal or abnormal.

A step termination phase is always performed before restarting a step so that resource use is reported even during the period between the restart point and the step abort.

Every value is measured according to the period of time which starts at a step initiation (normal or restart) and lasts until the step termination (normal or abnormal). This also applies to the information which is reported on the JOR.

FILE ASSIGNMENT

Files which are dynamically deassigned before the restart point are not reassigned at restart.

Temporary files, passed files, work volumes, pools of devices and multivolume files are supported.

The system protects the integrity of files which are allocated on work tape volumes therefore the work tapes of a restartable step (which has aborted) are also reserved. These tapes cannot be assigned to any other step until the owner step is restarted.

WARM RESTART AND SHUTDOWN

The Warm Restart mechanism contributes to the system availability by minimizing the cost of a system crash due to a power failure or a hardware/software failure. It salvages different system files and tables, restores the scheduler queue and the Output Writer queue as they were at crash time and allows a restart of the jobs that were running at crash time. In other words the jobs in the states IN/SCH/HOLD/OUT are fully recovered. Jobs in the state of execution (EX) are restarted depending on the job conditions at crash time, the JCL-options and the operator decision. They may be automatically recovered at the point they were at crash time or restarted at the last checkpoint or the beginning of the current step or totally repeated.

The Warm Restart also allows a recovery of the SYSOUT listings that were being printed at crash time.

The Shutdown function provides a means of terminating the Level 64 GCOS system.

It stops the Output Writer, suspends current jobs at end of steps or first checkpoints and closes system files in order to allow an automatic restart to be actioned later.

Shutdown

The Shutdown of the Level 64 GCOS system is instigated by the operator command END. It covers all areas of the system and performs a controlled systematic termination of operations and processing.

Termination of TDS and Telecommunications jobs are also operator functions and are performed by use of the appropriate commands (M STOP,TT).

The command END is prohibited when there are jobs in the system which are held in the state of suspension (HJ). If this ruling is not observed and the END command is used the message "JOBS SUSPENDED, END NOT ALLOWED" will be conveyed to the operator. The operator will then be required to reactivate the suspended job through use of the RUN JOB (RJ) command, and then reissue the command END.

System Shutdown Actions

1. Stream Reader : the Stream Reader automatically stops after having read a $ENDJOB JCL statement, and the introduction of new jobs through the SJ command is inhibited.
2. Jobs in execution (EX state) : they are suspended at the end of the current step or the first occurrence of a checkpoint in order to be automatically restarted at the next system restart.
3. Output Writer : As soon as the command END is accepted the Output Writer is notified not to commence printing and/or punching fresh output but only to complete the current operations.
4. Jobs in other states (IN, SCH, OUT, HOLD): Scheduling is inhibited. All the job queues will be restored at system restart time in the state they were at when shutdown occurred.
5. System files and tables : system files are closed and tables saved in order to allow an accurate resart.

System Restart

This facility is fully automatic and therefore requires no operator actions. Jobs are automatically restarted at the point at which they were suspended (end of step or checkpoint). Scheduling is reactivated and the different queues are restored.

The Stream Reader and the Output Writer have to be reactivated by the operator through use of the SI, SJ and SO commands as appropriate.

Warm Restart

Warm restart is called after the Initial Storage Load (ISL), and before system ready. This facility is used when the system has not been terminated by shutdown, and neither the option "COLD" restart nor "CLEAN" restart were requested by the operator at ISL.

Warm restart uses two basic control structures, KJOB, and the JCS structure. Within these structures job states and restart points are defined. These states are described by several fields and are updated in one short move (non-interruptable instruction). In order that the system files be in a state corresponding to the job they are related to, a "SAVE" is performed on the control structures as soon as they have been updated.

Warm restart resets all the necessary files (system and user files) and control structures (KJOB, JCS, JET, KNOT, KNODET) so that when the system is ready the jobs known at crash time can resume execution at a point either implicitely defined by the system, or explicitely given by the operator.

Warm restart is called after the following actions are performed :

– addressability to system control structures (KJOB, JCS) is restored.

– salvaging of Virtual Memory files (JOR, JCFI, JCFS).

File Salvaging

Before performing any action, warm restart calls upon the Salvager to salvage the user files, and system files (SYS.IN, SYS.OUT, SYS.FTU, SYS.JADIR). The salvager bases its action on the KNODET structure to find out which files were assigned and in an unstable state at the time of the crash.

For any sequential file, or mono volume file found in a unstable state, the salvager sends a message of the following format to the operator:

```
SV01 SYSTEM : UNSTABLE efn FILE ON media FOR xn
```

When the whole KNODET structure has been scanned, the salvager then asks the operator if salvaging is to be performed by sending the following message :

SV05 *SYSTEM : FILE SALVAGING ?

If the operator answers YES, the salvager performs its action, and issues the following message for each file it salvages :

efn FILE RECOVERED

If the salvager does not find any file eligible to be salvaged it does not issue a request message to the operator.

Warm Restart Initialization

After the salvager has been called, warm restart performs the following actions :

- Writes one or more records on the accounting file,
- Rebuilds the KNOT and ifn-lists from the KNODET structure,
- Cleans the KNODET structure to J = O,
- Assigns and opens the system files in the correct processing mode,
- Restores addressability to the Output Writer structures
- Salvages the Output queue,
- Prepares a list of all JOR, JCFJ, JCFS present and checks that their state corresponds to the stage of the corresponding job,
- Opens a rerun report which gives details of current jobs,
- Sends a request to the Output Writer for the rerun report.

After these initialization actions are performed, warm restart scans the KJOB structure to process each job in turn.

Warm Restart for IN/SCH/HOLD Jobs

The action for jobs in the IN/SCH/HOLD states is taken automatically by warm restart on the basis shown in the following table and without the necessity for operator intervention.

Table 6-4. Warm Restart For IN/SCH/HOLD Jobs

OLD STATE (KJOB details)	NEW STATE (KJOB stored)	
	JOB BATCH	SERVICE JOB
in introduction	empty	empty
introduced	introduced	empty
in translation	introduced	empty
idle	———	empty
translated with error	output	output
translated + TJ	Output	Output
held + TJ	output	output
schedulable + TJ	output	output
hold	hold	empty
schedulable	schedulable	empty
output	output	output

Before updating the job state in the KJOB, warm restart completes the following actions :

- Checks the corrections of the system files related to the JOB (SYS.IN, SYS.OUT, JOR, JCFS.JCFI).
- Writes a message on the JOR if necessary.
- Sends a message to the operator if the job is deleted.
- Updates the counters in the SYSLOAD, where necessary.

Warm Restart for EX/SUSP Jobs

The service jobs JTRA, WRITER, BTNS, IOF and FTU are all aborted without any notification being sent to the operator. The action taken on jobs in the EX or SUSP state depends on the execution status of the job at the time of the crash.

The crash is only operator visible either through a message written on the JOR, or when a Xnnn JOB REACTIVATED message, or a Xnnn JOB REMAINS SUSPENDED message is sent to the operator. This visibility only applies when the execution status of the job is one of the following :

- Job being initiated
- Step being initiated or waiting for resource allocation.
- Interstep (job level statement)
- Job being terminated.
- Step being terminated (case of a normal or abnormal termination)
- Repeatable step that has aborted but the operator had time before the crash to define whether the step is to be repeated or not.
- Step being executed, and neither step or job have been declared repeatable in the corresponding JCL statement.
- Step introduction in EXEC mode.

In these cases warm restart resets all control structures, so that the job can resume its execution as if no crash has occured and without requiring any operator intervention. However the job stage will remain as it was at the time of the crash i.e. EXEC or SUSP. If the job was released, the normal processing of the scheduler will select it at an appropriate time when the system is ready.

The operator is asked what to do with the job if the level of execution of the job is one of the following:

- Step in execution, and either the step or the job have been declared repeatable in the corresponding JCL statement.
- Repeatable step that has aborted but the operator did not have time before the crash to define whether the step is to be repeated or not.

The operator interface that is used in the last and most frequent of cases consists of the message RR01 with an immediate answer which may be YES/-NO/ROLLBACK/ALL [,HOLD].

Message format :

$$\text{RR01 ron ssn} \left\{ \begin{matrix} \text{ABORTED} \\ \text{KILLED} \end{matrix} \right\} \text{load module name SEV1 [=status] G4=}$$

```
        REPEAT?
        REPEAT [FROM CHKPT sss-nnn] ?
        REPEAT WHOLE JOB ?
        ROLLBACK ?
```

The format of the question depends on the following conditions:

—Level of the job execution.

— Repeat option in $STEP.

— Journalization of at least one file.

— TJ command issued before the crash.

— Step completion code value (STATUS).

— Repeat option in $JOB.

The possible operator replies and the corresponding warms restart actions are summarized in Table 6-5 below.

Table 6-5. Warm Restart Action on Operator Response

Question / **ANSWER**	**REPEAT ?**	**REPEAT FROM CHKPT ass-nn ?**	**REPEAT WHOLE JOB ?**	**ROLLBACK ?**
YES	Repeat the step from its begining with file rollback	Repeat the step from the last chekpoint with file rollback	Repeat the job from its begining with file rollback	Rollback files and continue next step
NO	Continue next step	Continue next step	Continue next step	Continue next step
ROLLBACK	Rollback files and continue to next JCL statement	Not allowed	Rollback files and continues to next JCL statement	Rollback files and continue to next JCL statement

To avoid repetitive operator conversation, the ALL answer has the equivalence of YES and is used as a "once and for all" positive reply for all jobs within the batch which are of the EXEC or SUSP status.

When HOLD is specified, the job is suspended by warm restart as opposed to being resumed by warm restart. Where no question is asked the job remains in the state as at crash time.

File rollback includes a rollback of journalized files and a rollback of journalized telecommunication queues. A rollback applies only to the last checkpoint or the beginning of the step if there is no checkpoint. So a rollback only without a repeat of the step is forbidden if a checkpoint exists (answer ROLLBACK).

When a crash occurs during job execution, warm restart will perform the equivalent of a step termination and prepare the Internal Job Control File (JCFI) to enable either the job to be restarted at a checkpoint, or repeated from its start point, or repeated from the beginning of the current step.

The step completion code is set to 61000 if the job had a step in execution at crash time and the operator decides not to perform a step or a job repeat. It can be tested by JCL in the case of a step.

Warm Restart Action on SYSOUT

The operator is requested to specify where he wants the SYSOUT that was being printed or punched at the time of the crash to be restarted.

For further details refer to the *System Operation Console Messages* manual.

Rerun Report

Warm restart edits a report which contains information about the jobs which were known at crash time, and the salvager report.

This file is always opened in the append mode, i.e. so that if between two crashes the Output Writer has not been started, the first rerun report will not be erased by the second.

Checkpoint/Restart Limitations

It should be noted that Checkpoint/Restart does not support the following:

— Deferred restart

— Relocation of files before restart

— Immediate SYSOUT deliveries (they are deferred to the end of step)

— Unit-record device repositioning

— Several coexisting recovery points for the same step (multiple checkpoints)

— Use of the Program Checkout Facility

— Multi-task step recovery.

7. Memory Management

The memory management for GCOS Level 64 is based on the principle of virtual memory. The execution units (load modules) are composed of segments. To each segment is attached a segment descriptor. These segments and the dictionary of segment descriptors are stored either in the library or in the auxiliary memory (backing store). The address that is derived from the segment descriptor and used to reference a field in a segment is composed of the following two items :

- The first is used to reference the base of a segment.
- The second specifies a displacement in the segment.

The descriptor contains fields which indicate the access rights, the length, and the address of the segment. The descriptor also contains a "presence bit" which indicates whether the address is a central memory address (segment present) or a disk address (segment missing).

When a program which is in execution mode references a segment which is not in central memory ("presence bit"=0), the firmware automatically calls upon the Segment Fault Management (SFM) system procedure giving it the identification of the absent segment. The SFM procedure, executed in the address space of the process which referenced the missing segment, is used to fulfil the following three points :

- Search for the segment in backing store.
- Find space in central memory to load this segment.
- Restart execution of the process by re-execution of the instruction which triggered the segment fault.

SEGMENT FAULT MANAGEMENT

The SFM procedure extracts from the segment descriptor the size of the segment and its disk address, and then searches in the list of available memory areas, called the list of free memory areas (FMA), for the first area large enough to accept the segment. The appropriate area is then withdrawn from the FMA list, the segment loaded into the area, and the segment descriptor modified to point to the corresponding memory address.

The Search For The Virtual Free Memory Area

The SFM procedure may not be able to locate a zone of sufficient size from the FMA list. In such a case it is necessary to overwrite from main memory certain segments.

Within the main memory each area (both segment and free area) is proceeded by a header. This header contains chaining information for the preceding area and the following area. The header for each segment also contains the address of the descriptor and the disk address of the segment plus the current pointer of the memory address where the last search (of memory space) stopped. The search for virtual FMA consists of a circular sweep over the memory zone list. This search commences from the current pointer and is performed to locate a group of consecutive swappable segments and/or FMA which are of sufficient size to accept the requested segment. A segment is known to be swappable by reference to the indicator of segment usage (the "used bit" of the segment descriptor) which is set to 1 at each reference to the segment. When a virtual FMA satisfies the initial request the current pointer will be modified to point to the next memory zone following the last zone of the virtual FMA. The member segments of the FMA are rewritten into the backing store (if necessary) and the requested segment loaded into main memory.

The "written bit" of the segment descriptor which is set to 1 by firmware upon modification of the segment is used to avoid all unnecessary recopying in backing store.

The current pointer mechanism is used to methodically outdate segments and to integrate them, thereby reducing possible memory fragmentation. Memory fragmentation would occur if a large number of spaces arose which were too small for general usage.

Compacting

During a search for a virtual FMA, segments of high usage ratings are respected and remain untouched. This factor coupled with the possibility that a large number of segments could be temporarily non-swappable (locked) will give rise to a state where no available virtual FMA can satisfy the initial request. In such an instance the Compacting facility will move the temporarily non-swappable segments and those of a high usage rating in order to create a FMA of sufficient size to incorporate the segment initially requested.

Swappable Segments

The system is able to indicate that one or more system segments are no longer required. This segment redundancy is most likely to occur at the change of a service phase, or at the end of error processing. These segments are known as "eligible to be swapped" and will remain in memory until the system needs the space. The discharge of these segments will occur when they are found to satisfy a virtual FMA request, or when the compacting facility removes them.

Locked Segments

A certain number of segments of a load module must be locked in central memory. The system can neither discharge them into backing store nor change their place in memory, as they are pointed by absolute addresses and are therefore locked in memory.

These locked segments are essentially limited to control structures, segments containing channel programs, and to buffers. To ensure that such segments are not scattered throughout memory, causing fragmentation and affecting virtual FMA searches and compaction phases, the system will assign them to low memory addresses.

Commencing with the lowest memory address, the system searches for an adequate FMA and an area of non-locked consecutive segments large enough to contain the locked segments. When a locked segment is allocated, the system checks that the sum of the sizes of the locked segments of a job step does not exceed the memory size specified in $SIZE.

If the $SIZE specification is exceeded the job step is abnormally terminated with a return code indicating user memory overload.

Thrashing

A situation, known as thrashing, is indicated when the acceptable limit of segment fault is surpassed. An excessive number of segment faults will cause considerable time wastage and inefficiency and is generally caused by system acceptance of too many job steps in execution.

When the system detects an unacceptable level of segment faults the following message is displayed on the operator console :

"THRASHING SITUATION"

The operator should then suspend one or more jobs using the HOLD JOB command. This action will suspend the execution of the job step and give the segment an "eligible to be swapped" status.

Similarly, the system also controls the total number of input/output transactions on the system disk and will notify the operator when the number exceeds the system permissable limit by way of the following message :

"SYSTEM DISK OVERLOAD"

MEMORY SCHEDULING

To avoid memory overload situations the amount of memory required for each job step including locked and swappable segments should be specified by use of the $SIZE statement.

This memory requirement figure is known as the Declared Working Set (DWS). If at initialization time the DWS memory allocation procedure is not followed the program, by default, will be allocated 35K of main memory.

During job execution a situation can arise whereby there is insufficient memory to swap in the requested segments, whereupon the affected job step will immediately enter a WAIT state and the operator is notified by the following message :

".....job...step...identification...WAITS FOR MEMORY".

Which requires no operator action.

The system's function is to ensure that the total sum of the DWS is equal to (or less than) the schedulable memory size less all memory size reserved for the system. A total of 15% of the swappable memory is reserved for all system operations.

To obtain concise information about the memory the operator uses the following command:

DMM (DISPLAY MAIN MEMORY).

The return message is:

```
MAIN =... total swappable memory,
          RESERVED = ... total declared working set,
                    LOCKED = ... total locked segments.
```

THE MINIMUM MEMORY FACILITY

Although a residence factor (RF) may be used to influence the duration of a segment in memory, this duration is still dependent on memory load. Therefore the RF factor may not be sufficent for such programs as used in data-communication applications, which require brief response times.

The minimum memory facility enables the system to guarantee the amount of memory as requested by the communications program. It ensures that the size of memory occupied by the program segments, called the instantaneous working set (IWS), is greater than or at least equal to the amount of memory declared (DWS).

The MINMEM parameter forms part of the $STEP JCL statement in the following format:

```
STEP......,MINMEM,..........;
SIZE...declared working set...;
ENDSTEP;
```

For full working details refer to the JCL User Guide and the Communications Processing Facility manual.

RESIDENT SYSTEM FUNCTIONS

As some system functions are constantly in use by certain programs, good response times must be assured. It is therefore desirable that such procedures remain in central memory permanently.

By use of the PMM operator command specific system functions can be made resident. These specific system functions should only be segments which make up the core of the function.

The system verifies that the total DWS plus the total functions made resident by the operator are equal to (or less than) the swappable memory. When the available memory is not sufficient to load the system function the operator will receive the following message :

"TRY LATER : NOT ENOUGH MEMORY".

The space occupied by the resident system functions is accounted and declared in the locked memory field of the DMM command. This DMM system command is used to list the resident system functions. These functions remain resident until the operator requests their suppression by use of the CMM command.

In order to ensure that such segments do not disturb memory management they are packed towards high memory addresses.

The following is a list of System Functions which can be made resident by the operator using the PMM command.

BFAS	UFAS	UFASTDS
BFASI	UFASREL	QUEUED
BFASII	UFASI	IDSR
HFASST	UFASIK	IDSU
HFASSD	UFASIP	MAMM
HFASI	UFASIKP	MAMD
HFASRDM	RDMIDS	VCAM

DECLARED WORKING SET

The declared working set (DWS) is the physical memory size in units of 1K (1024 bytes), required by code/data segments, and control structures of a job step. This figure should be the optimal requirement which allows a program to execute without any significant loss in performance (or excessive memory allocation).

Assuming that degredation of performance will arise from inadequate memory, memory increase will subsequently improve performance. The DWS is therefore the ideal balance between additional memory and increased performance. Some further advantage may be gained from providing somewhat more memory than the DWS figure, however a distinct lack of advantage will be felt from providing memory greatly in excess of the DWS.

The amount of memory required can change throughout the duration of a program. At various stages of program execution the incremental working set (IWS) could be high if a large number of different segments were called upon, but as this would only be for a short time, such peaks may be ignored.

The concept of the DWS is shown in Figure 7-1. below.

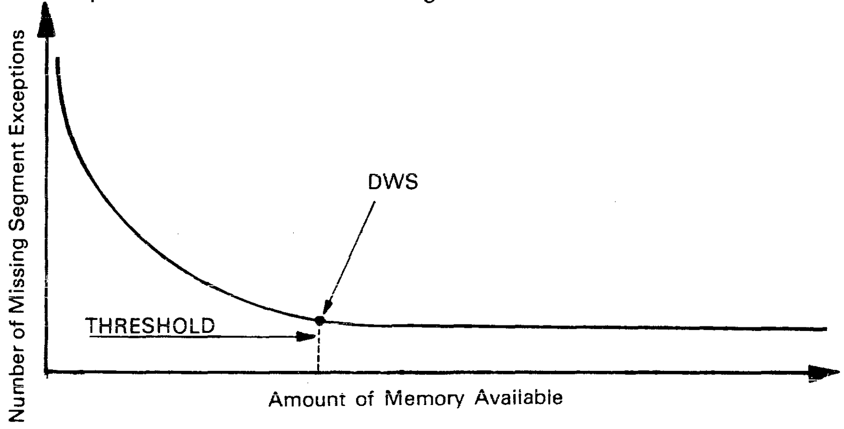


Figure 7-1. Performance Versus Memory

The curve shown depicts the increase in the total number of missing segment exceptions a program encounters in relation to the amount of physical memory available for holding pages.

This curve is a general one; each program will have its own curve depending on its own structure; but the general shape will always be the same.

From the curve, it can be seen that when a program is restricted to a small amount of memory, it encounters a large number of missing segment exceptions. When memory is increased, the number of missing segment exceptions decreases.

The threshold point where any further incremental increases in memory will give little or no advantage is the point which should be specified as the declared working set. This figure being that which is given as the DWS in the $SIZE JCL statement.

The function of this JCL statement is to define the memory requirements for a job thereby helping memory management (and memory scheduling) to avoid memory overload situations.

Format:

```
SIZE [declared working-set]
     [,CPPAGE = channel-program-page-size]
     [,NBBUF = number-of-buffers]
     [,POOLSIZE = poolsize];
```

Full working details of the $SIZE JCL statement are given in the JCL USER GUIDE.

DWS Estimation For The User Program

The first consideration when estimating a programs DWS must be given to the structure produced by the load module. If the program is organized in phases corresponding to the main sections of the program, the estimation must be made phase by phase with the programs DWS corresponding to that of the largest DWS for any one phase.

An estimation of a programs memory requirements can be ascertained from the following areas.

1. LINKER information produced after the linking of a load module, particularly code segments and data segments.

2. BUFFER SIZE information given on the JOR concerning maximum size of the buffers. (see the Storage Estimation manual). The total space occupied by buffers depends on :
   - The number of files.
   - The block, CI size each file.
   - The number of buffers used for each file.

3. FILE CONTROL TABLES
   See the Storage Estimation manual.

4. PROGRAM CONTROL STRUCTURE :
   - Stacks (6K).
   - Physical Channel Programs (PCP's) segments made up of un-swappable segments. Their sizes depend on :
     . The number, organization, and processing modes of files.
     . The number of buffers per file.
   - Process Group Control Segment (PGCS) which is usually less than 2K. The exact value is given on the LINKER listing (map).
   - Semaphore Segments usually less than 1K, the exact figure is given in the LINKER listing (map).

5. References to data management access method routines (UFAS/BFAS/HFAS). These routines are normally available to all the jobs wishing to use them so they should only be counted once (see the Storage Estimation manual).

6. Segment fault information derived from the number of user and system segment faults. The number of segment faults and the amount of memory used during a program will depend on which programs are being run at any one time and whether they are run concurrently.

Once the first run of the program has been completed, the information provided on the JOR will give the total buffer size, PCP segment size, and the number of missing segments details. This information will permit a readjustment of the DWS value to an optimal value.

However, within a multiprogramming environment where multiple programs are concurrently competing for memory, it is difficult to readjust correctly the DWS value solely from this information.

To aid an easier readjustment of the DWS value, use of the "MAXMEM" facility is recommended. When using this facility the GCOS operating system ensures that the total amount of memory devoted to the segments of a program is always equal to or less than the DWS specified in the $SIZE of the program.

Format :

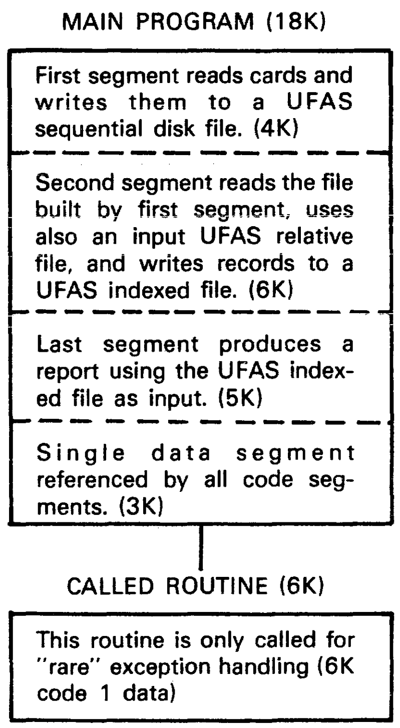
```
STEP.........,MAXMEM......................;
SIZE..........declared working set.....;
ENDSTEP;
```

For full working details refer to the JCL User Guide and the Communications Processing Facility manual.

DWS Calculations for a User Program

It should be noted that the following figures are given for example purposes only; in no way do they represent the true storage requirements of supplied components. For precise storage figures see the Storage Estimation manual.

The first consideration is the structure of the produced load module. Suppose a load module is as represented below :


MAIN PROGRAM (18K)
First segment reads cards and writes them to a UFAS sequential disk file. (4K)
Second segment reads the file built by first segment, uses also an input UFAS relative file, and writes records to a UFAS indexed file. (6K)
Last segment produces a report using the UFAS indexed file as input. (5K)
Single data segment referenced by all code segments. (3K)
CALLED ROUTINE (6K)
This routine is only called for "rare" exception handling (6K code 1 data)

SYSIN :
  BLKSIZE=1048, NBBUF=2

Sequential Disk file :
  CISIZE=1024, NBBUF=2

Relative file :
  CISIZE=512, NBBUF=5

Indexed file (file with one index level) :
  CISIZE= 2048, index CISIZE= 2048
  NBBUF=2

SYSOUT :
  BLKSIZE=1048, NBBUF=2

Notes :

1. There are three phases of execution corresponding to the three main sections of the main program. (Segmentation rules are described in the COBOL Reference Manual).

2. Since the called program of segment 1 is rarely used, it is not counted in the DWS.

3. Since there is only one phase executed at a time, the program's DWS is the requirement of the largest phase.

4. The DWS calculation (in K) for the first phase is :

Code Segment		04.096
Data Segment		03.072
SYSIN		
Control Tables	164 + 2*32	00.228
Buffer Space		02.208
UFAS Sequential Disk File (WRITE)		
Access Method	3,3 + A	11.500
Control Tables	143 + 40 + 10*2	00.203
Buffer Space	80 + 2*1024	02.128
Stack		06.560
PCPs (default value, n=2)		04.384
	Phase 1 total :	34.379 K

5. The DWS calculation for the second phase is :

Code Segment		06.144
Data Segment		03.072
UFAS Access Methods		
Relative (READ) is included in Sequential		
Sequential		12.368
Indexed (WRITE insert)		17.680
Subtotal :		30.048 K
Control Tables		
Sequential	143 + 40 + 10*2	00.203
Relative	143 + 40 + 10*5	00.233
Indexed	263 + 40 + 10*(2 + 1 + 3)	3.630
Buffer Space		
Sequential	80 + 2*1024	2.128
Relative	5* (40 + 546)	2.780
Indexed	2* [2048 + 40]	4.176
Stacks		6.560
PCPs		4.384
	Phase 2 total :	63.358 K

6. By inspection it is obvious that the DWS for the third phase will be smaller than that for the second phase. Therefore the DWS for the load module is 64K.

DWS Calculations For Buffer Space

The total space occupied by buffers depends on the number of files, the block or control interval size of each file, and the number of buffers used for each file. A good approximation of buffer space requirements may be calculated by assuming that the size of each is the same as a block. To form a precise calculation refer to the storage Estimation manual.

For library access (including SYSIN/SYSOUT) an extra 43 bytes must be added to the blocksize to get the space requirement for each buffer. Therefore, using SYSOUT with double buffering, where the block size is 1048, the buffer space in bytes is :

2* (1048 + 43 rounded to multiple of 16 (=1104)) = 2208.

8. Job Classes, Scheduling and Execution

The delivered GCOS system is pre-set with the Job Classes, Scheduling Priorites and Execution Priorities as shown below.

Table 8-1. Preset Job Classes

Job Class	Scheduling Priority	Execution Priority	Multi-program Limit	Recommended Usage
A	7	9	1	—
B	7	9	1	—
C	7	9	1	—
D	1	5	1	PROGRAM MODE
E	2	4	1	EMERGENCY JOBS
F	3	7	1	FIRST PRIORITY JOBS
G	4	9	1	"GREATER" SCHEDULING PRIORITY JOBS
H	6	1	1	TELECOM JOBS
I	5	0	1	—
J	6	1	1	TDS
K	7	9	1	—
L	7	9	1	—
M	7	9	1	—
N	7	9	1	—
O	7	9	1	—
P	7	9	5	NORMAL BATCH JOBS
Q	7	4	1	IOF
R	0	2	6	—
S	7	0	1	BTNS
T	7	4	6	—
U	7	2	6	FTU
V	7	9	1	—
W	0	2	8	WRITER
X	0	3	1	JTRA
Y	7	9	1	—
Z	7	9	1	—

Job Class A-P are user class jobs and Job Class Q-Z are service class.

An installation may modify this delivered set-up using the MC command. Additionally, the MJ command is available to the operator for modifying the class of a job.

In JCL, at job level a given job may request a different scheduling priority from that associated with the job class.

Also, at the step level, a given job step may request a different execution priority from that associated with the job class.

Note that classes K to O are not initially "started". These classes should be brought into use when there is a clear installation policy for them.

In general, it is not recommended that major or frequent changes be made to the set-up as delivered. It is best to maintain a stable set of installation conventions and defaults so that the operator's task is not unnecessarily complex.

Recommended Practice

- The classes E, F and G should be reserved for high priority (and probably, unplanned) jobs. They should be used, upon the advice of the system manager, by the operator to promote, when needed, jobs which normally use a standard job class (K to P).
- The installation rules for classes K to P (that is, batch jobs) should be defined according to the type of activity of a job.

  Job may be divided into different classes according to such characteristics as :

  . CP time versus elapsed time.

  . Device Usage.

  . File Usage.

  . Memory Requirement.

  The decision as to which class to use is a matter for the system manager to decide, based on such considerations as :

  . The desired scheduling and execution priorities between job streams.

  . The need to minimize resource queueing and contention (files, devices, memory).

  . The sequence in which jobs must be run.
- The system manager must ensure that all operators, analysts, and programmers are aware of the installation conventions and defaults.

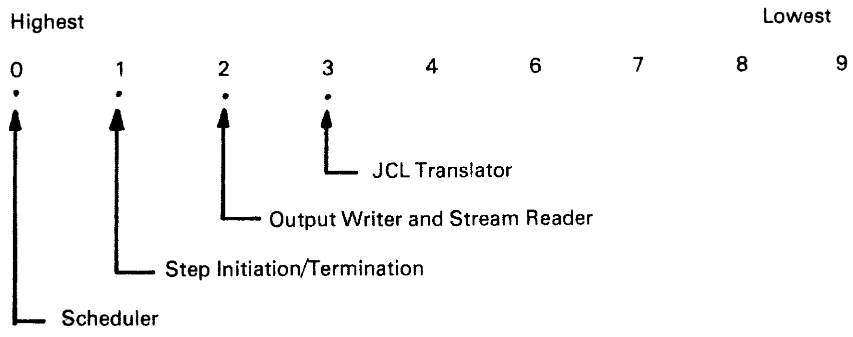
EXECUTION PRIORITY

This is the priority of a task (step) for CP allocation. There is a default execution priority associated with each job class.

This default value may be changed for a step by :

- Specifying a different priority on the $STEP statement.
- The operator changing the priority at step execution

The choice of execution priority affects system behaviour since the system itself also competes for the CP. System execution priorities are shown below :



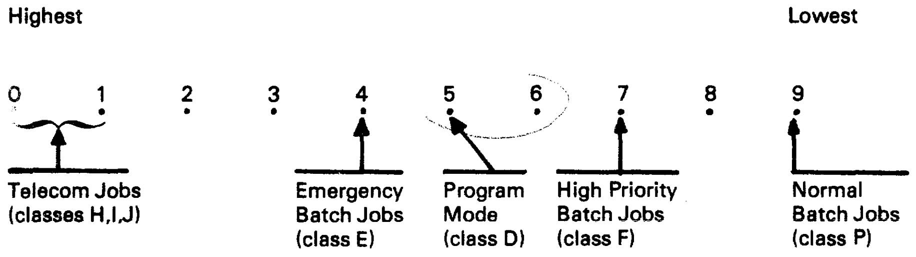
Therefore, if a user batch job executes with priority 0 or 1 it will severely impact the Output Writer, thus reducing the output rate.

Another example of erroneous management might be a batch job which is CP-bound (mostly computation, little I/O) with an execution priority of 0. This job would deny execution of the system scheduler until the batch job released the CP to perform, say, I/O.

Execution Priority Recommendations :

- Reserve priorities 0, 1, 1 and 2 (classes H, I and J) for Telecom jobs since they require rapid access to the CP in order to provide adequate response times.
- Use the priorities 4 to 9 (classes D, E, F, G, K to P) for batch jobs according to installation requirements. The choice for a job might be made on the basis :
  . I/O bound jobs having higher priorities than CP bound jobs.
  . Urgent jobs being given higher priorities than "normal" jobs.

These recommendations are summarized below :


Highest
Lowest
0
1
2
3
4
5
6
7
8
9
Telecom Jobs (classes H,I,J)
Emergency Batch Jobs (class E)
Program Mode (class D)
High Priority Batch Jobs (class F)
Normal Batch Jobs (class P)

Appendix A
CONFIG Output Listing and Error Messages

THE OUTPUT LISTING PRODUCED BY CONFIG

A standard banner is output, followed by the heading "CONFIGURATION LANGUAGE", followed by a numbered list of statements of configuration.

Example listing :

```
CONFIGURATION LANGUAGE
0001 $JOBCLASS P,NSC;
0002 $JOBCLASS A,XPRTY=7,PRIORITY=7,MAXLOAD=5,NSTARTED,NMPRIO,NMLOAD,NMXPRTY;
0003 $JOBCLASS B,XPRTY=7,PRIORITY=7,MAXLOAD=5,NSTARTED,NMPRIO,NMLOAD,NMXPRTY;
0004 $JOBCLASS C,MAXLOAD=5;
0005 $JOBCLASS D,XPRTY=7,PRIORITY=7,MAXLOAD=5,NSTARTED,NMPRIO,NMLOAD,NMXPRTY;
0006 $JOBCLASS E,XPRTY=7,PRIORITY=7,MAXLOAD=5,NSTARTED,NMPRIO,NMLOAD,NMXPRTY;
0007 $MULTLEV 20,INTERACT=12,BATCH=14;
```

Numbered error messages are given after the statements of configuration listing, these messages always refer to a statement by its number. Each of the error messages is preceded by either one, two or three stars (*), depending on the severity of the error. If no major error occurs then the configuration process proceeds.

Scheduling Parameters

Under the title "SCHEDULING PARAMETERS " the following summary is given :

Maximum number of known jobs..............	extracted from $MAXJOB statement.
Maximum number of stated jobs................	extracted from $MULTLEV statement.
Maximum number of started batch jobs...	extracted from BATCH parameter of the $MULTLEV statement.
Default class for remote jobs........................	extracted from $ROFCLASS statement.

Classes Configuration

Under the title "CLASSES CONFIGURATION" a summary of the status of the job classes is given. The summary is divided into the following categories :

CLASS	Name of the job class.
MAXLOAD	Value of the multiprogramming level of the job class.
PRIORITY	Scheduling priority attached to the job class.
XPRTY	Dispatching priority attached to the job class.
STARTED	"Y" means that the job class is started. Blank means that the job class is not started.
SC	"Y" means that use of the SC and TC operator commands is permitted. Blank means that use of the SC and TC operator commands is not permitted. (NSC parameter has been used in the corresponding $JOBCLASS statement).

MXPRTY	"Y" means that the operator may change the dispatching priority of the corresponding job class (MC command). Blank means that the operator may not change the dispatching priority of the corresponding job class (NMXPRTY parameter has been used in the corresponding $JOBCLASS statement).
MPRIO	"Y" means that the operator may change the scheduling priority of the corresponding job class (MC command). Blank means that the operator may not change the scheduling priority of the corresponding job class (NMPRIO parameter has been used in the corresponding $JOBCLASS statement).
MLOAD	"Y" means that the operator may change the multiprogramming level of the corresponding job class (MC command). Blank means that the operator may not change the multiprogramming level of the corresponding job class (NMLOAD parameter has been used in the corresponding $JOBCLASS statement).

Example Listing for "Scheduling Parameters" and "Classes Configuration"

```
SCHEDULING PARAMETERS

MAXIMUM NUMBER OF KNOWN JOBS             : 200
MAXIMUM NUMBER OF STARTED JOBS           : 20
MAXIMUM NUMBER OF STARTED BATCH JOBS     : 14
DEFAULT CLASS FOR REMOTE JOBS            : P

CLASSES CONFIGURATION :

 CLASS  MAXLOAD  PRIORITY  XPRTY  STARTED  SC  MXPRTY  MPRIO  MLOAD

   A       5        7        7              Y
   B       5        7        7              Y
   C       5        7        9       Y      Y     Y       Y      Y
   D       5        7        7              Y
   F       1        3        7       Y      Y     Y       Y      Y
   G       1        4        9       Y      Y     Y       Y      Y
   H       1        6        1       Y      Y     Y       Y      Y
   I       1        5        0       Y      Y     Y       Y      Y
   J       1        6        1       Y      Y     Y       Y      Y
   K       1        7        9       Y      Y     Y       Y      Y
   L       1        7        9       Y      Y     Y       Y      Y
   M       1        7        9       Y      Y     Y       Y      Y
   N       1        7        9       Y      Y     Y       Y      Y
   O       1        7        9       Y      Y     Y       Y      Y
   P       5        7        9       Y            Y       Y      Y
   Q      10        7        4       Y      Y     Y              Y
   R       6        0        2       Y            Y       Y      Y
   S       1        7        0       Y            Y
   T       6        7        4       Y      Y     Y              Y
   U       6        7        2       Y      Y     Y              Y
   V       1        7        9       Y      Y     Y       Y      Y
   W       8        0        2       Y            Y       Y      Y
   X       1        0        3       Y            Y
   Y       1        7        9       Y      Y     Y       Y      Y
   Z       1        7        9       Y      Y     Y       Y      Y
```

Device Configuration Information

The Device configuration information displayed on the CONFIG output listing is derived from the System Resources Status Table.

Example :

LDN	NAME	PATH	LDN	NAME	PATH
01	UC01	0200	02	TC01	0100
03	MC01	0000	04	D101	0203
05	CS01	0204	06	LN01	020D
07	LN07	0211	08	LN08	020F
09	LN09	0210	A0	LN16	0217
0B	LN02	020E	0C	CT01	0209
0D	PR02	0208	0E	PR01	0207
0F	CD01	0201	10	CD02	0202
11	MT03	0103	12	MT02	0102
13	MT01	0101	14	MS03	0003
15	MS04	0004	16	MS05	0005
17	MS06	0006	18	MS07	0007
19	MS08	0008	1A	MS02	0002
1B	MS01	0001	1C	DU03	000B
1D	DU02	000A	1E	DU01	0009

Following the device configuration information, specific error messages are displayed. These messages refer to a statement by name and not by number. If no error has been detected and no warning given, the following message is output both at the operator console and on the listing :

SUCCESSFUL SYSTEM CONFIGURATION TO RUN CONFIGURATION SYSTEM PERFORM STORAGE LOADING WITH RESTORE AND RESTART COLD OPTIONS

CONFIG ERROR MESSAGES

These error messages comprise two parts :

1) The General Message :
   these are numbered, given in numerical order and refer to a statement number.
2) Specific Error Message :
   these refer to a statement name, and are given in the alphabetical order of the statements of reference.

General Error Messages

ERROR 0000
    UNSUCCESSFUL CONFIGURATION PROCESS
(This is due to severe errors, no configuration performed).

ERROR 0001
    UNABLE TO OPEN SYS.SYSTEM
(The SYS.SYSTEM file could not be opened; check that a SYS.SYSTEM file exists on disk).

ERROR 0002
    UNABLE TO READ INPUT FILE
(I/O error while reading configuration statements file).

ERROR 0003
    PREMATURE END OF STREAM
(An EOF for a configuration statement has been encountered : correct the last statement - a ";" is missing).

ERROR 0004
DOLLAR INSIDE A STATEMENT
(A dollar sign has been encountered before reaching the final ";").

ERROR 0005
ILLEGAL SEPARATOR SEQUENCE
(An illegal set of separators has been encountered, i.e. keyword=,,).

ERROR 0006
UNABLE TO CLOSE INPUT FILE
(The configuration statement file could not be closed).

ERROR 0007
UNABLE TO READ SYS.SYSTEM
(I/O error while reading SYS.SYSTEM file).

ERROR 0008
UNABLE TO CREATE A WORK SEGMENT
(Internal CONFIG error).

ERROR 0009 ON STATEMENT XXXX
NO VALID SEPARATOR BEFORE PROTECTED STRING
(Statement XXXX contains the following :
....=ZZ'string' this should read
....='ZZ string').

ERROR 0010 ON STATEMENT XXXX
EMPTY PROTECTED STRING
(A string must not be empty : do not use COMMAND keyword in statement XXXX).

ERROR 0011 ON STATEMENT XXXX
NO VALID SEPARATOR AFTER PROTECTED STRING
(Statement XXXX contains the following :
....='string'ZZ this should read
....='string ZZ').

ERROR 0012
NO USER SUPPLIED CONFIGURATION STATEMENT
(This is a warning that no user statement has been taken into account).

ERROR 0013 ON STATEMENT XXXX
UNKNOWN STATEMENT NAME
(An unknown statement has been used in statement XXXX).

ERROR 0014 ON STATEMENT XXXX
ILLEGAL SEPARATOR AFTER FIRST POSITIONAL
(An illegal separator has been used after the first positional parameter of statement XXXX).

ERROR 0015 ON STATEMENT XXXX
UNKNOWN FIRST POSITIONAL
(The first parameter value of statement XXXX is unknown).

ERROR 0016 ON STATEMENT XXXX
KEYWORD / SIV NOT FOUND
(A mandatory keyword parameter/Self Identifying Value is missing from statement XXXX).

```
ERROR 0018 ON STATEMENT XXXX
        PRESENCE OF EXCLUSIVE SIV'S
```
(In statement XXXX conflicting Self Identifying values have been used, i.e. STARTED NSTARTED).

```
ERROR 0019 ON STATEMENT XXXX
        ILLEGAL SEPARATOR AFTER A SIV
```
(In statement XXXX an illegal separator has been used after a Self Identifying Value).

```
ERROR 0020 ON STATEMENT XXXX
        SIV DUPLICATION
```
(In statement XXXX the same SIV has been used more than once).

```
ERROR 0021 ON STATEMENT XXXX
        ILLEGAL SEPARATOR AFTER A KEYWORD
```
(In statement XXXX an illegal separator has been used after a keyword).

```
ERROR 0022 ON STATEMENT XXXX (>8 CHARACTERS)
        ILLEGAL STATEMENT/KEYWORD NAME
```
(Each keyword or statement name must have a length equal to or less than 8).

```
ERROR 0023 ON STATEMENT XXXX
        ILLEGAL PARAMETER TYPE
```
(Incorrect parameter value given).

```
ERROR 0024 ON STATEMENT XXXX
        TOO LONG PARAMETER VALUE
```
(The value of a parameter in statement XXXX exceeds the permitted size).

```
ERROR 0025 ON STATEMENT XXXX
        DUPLICATE STATEMENT
```
(An identical statement has already been used within the set configuration statement).

```
ERROR 0026 ON STATEMENT XXXX
        MANDATORY ARGUMENT MISSING
```
(A mandatory argument has been omitted from statement XXXX).

Specific error messages

```
ERROR IN STATEMENT "$ACTSIZE;"
        LIMIT FOR SIZE : 2000 BLOCKS
```
(The value given for the $ACTSIZE statement exceeds the permissible limit).

```
ERROR IN STATEMENT "$BANNER.....;"
        ERRONEOUS BANNER NUMBER : 2 ASSUMED
```
(The value of the parameter is neither 0, 1 or 2; therefore 2 is assumed).

```
ERROR IN STATEMENT "$JOBCLASS job-class,MAXLOAD";
        EXCEEDS NUMBER OF JOBS
```
(The value of MAXLOAD parameter of the "$JOBCLASS job-class" statement exceeds the permissible limit).

```
ERROR IN STATEMENT "$JOBCLASS job-class , XPRTY;"
        LIMIT FOR DISPATCHING PRIORITY : 9
```
(The value of XPRTY parameter of the "$JOBCLASS job-class" statement exceeds the range of the dispatching priorities values).

ERROR IN STATEMENT "$JOBCLASS job-class , PRIORITY;"
LIMIT FOR SCHEDULING PRIORITY : 7

(The value of PRIORITY parameter of the "$JOBCLASS job-class" statement exceeds the range of the dispatching priority values).

ERROR IN STATEMENT "$JOBCLASS job-class , NSC"
CLASS SHOULD BE STARTED OR SC ALLOWED

(Both the NSC and NSTARTED parameters have been specified in "$JOBCLASS job-class" statement).

ERROR IN STATEMENT "$JORSIZE;"
LIMIT FOR SIZE : 2,000 records

(The value given for the $JORSIZE statement exceeds the permissible limit).

ERROR IN STATEMENT "$MAXJOB POSIT--1;"
LIMIT FOR MAXIMUM JOBS : 9999

(The value given by the parameter of the $MAXJOB statement exceeds the permissible limit).

ERROR IN STATEMENT "$MAXTASK;"
LIMIT FOR NUMBER OF TASKS : 150

(The value given by $MAXTASK statement exceeds the permissible limit).

```
ERROR  IN STATEMENT "$MULTLEV  BATCH      ;"
                               INTERACT
       EXCEEDS NUMBER OF JOBS
```

(The value of BATCH or INTERACT parameters, either explicitly stated by the user, or implicitly assumed, is inconsistent with the first parameter value of $MULTLEV).

ERROR IN STATEMENT "$MULTLEV POSIT--1";
LIMIT FOR NUMBER OF JOBS : 30

(The value given by the parameter of the $MULTLEV statement exceeds the permissible limit).

ERROR IN STATEMENT "$OWCLASS;"
INVALID OUTPUT CLASS PRIORITY, NORMAL DEFAULT VALUE ASSUMED

(The value of PRIORITY parameter is outside the range 1, 2,......7. Therefore the default value is assumed).

ERROR IN STATEMENT "$OWDEVICE;"
XXXX NOT PRESENT AT INSTALLATION

(The value of the DVID parameter : XXXX is not in the installation; not found in the SRST).

ERROR IN STATEMENT "$OWDFLT;"
DEVICE CLASS ERROR. PR/H132 TAKEN AS DEFAULT CLASS

(Incorrect value given for the DEVCLASS parameter, PR/M132 assumed).

ERROR IN STATEMENT "$OWDFLT;"
UNACCEPTABLE DEFAULT MEDIA

(The value of MEDIA parameter is either incorrect, or has not been found in SYS.URCINIT).

ERROR IN STATEMENT "$OWDFLT;"
INCORRECT TAPE PARAMETER.TAPE=SYSOUT ASSUMED

(The value given for TAPE parameter was neither SYSOUT nor NSYSOUT. Default option assumed).

ERROR IN STATEMENT "$PRLOG THRESHLD;"
THRESHOLD VALUE MUST BE LESS THAN 100 AND GREATER THAN 0

(Incorrect threshold value given for the threshold parameter of the $PRLOG statement).

ERROR IN STATEMENT "$ROFCLASS POSIT--1;"
CLASSES FOR USER JOBS : A-P

(The value given by the parameter of the $ROFCLASS statement is not within the permissible range of A-P).

ERROR IN STATEMENT "$STATION;"
WARNING : TOO MANY STATIONS DECLARED. LAST XX NOT SUPPORTED

(More than 6 stations have been declared therefore only the first 6 have been accepted).

ERROR IN STATEMENT "$STATION;"
XXXXINVALID STATION NAME. NO MORE STATIONS ANALYZED

(The value given for the NAME parameter of a station is not acceptable. The following $STATION statements are not checked or acceptable).

ERROR IN STATEMENT '$STATION;"
XXXX INVALID PROTOCOL PARAMETER FOR STATION XXXX

(In the $STATION statement for station XXXX the value of the PROTOCOL parameter is XXXX and should be either 61 or MVIP).

ERROR IN STATEMENT '$STEPFILE POSIT--1;"
STEPFILE SHOULD BE 42 OR 55

(The value given by $STEPFILE statement is neither 42 nor 55).

Appendix B The Catalog Maintenance Utility

THE $CATMAINT UTILITY

Function :

To store and maintain PROJECT, USER and BILLING information in the site catalog

Statement form

```
CATMAINT COMFILE = sequential-input-file
      [,PRTFILE=(print-file] ;
```

Statement description

This utility can only be run under the project SYSADMIN.

The command file (COMFILE) may be a sequential or pseudo-sequential file (input enclosure or member of a source library) and may be temporary or permanent.

The catalog in which the project, user and billing information is stored is always the site catalog.

Parameter description

COMFILE	The file containing the commands to $CATMAINT. It may be in DATASSF or DATA format. See below for a description of the $CATMAINT commands.

THE $CATMAINT COMMANDS

The commands to $CATMAINT are contained in COMFILE.

Each command consists of :

– A mandatory operation code followed by different positional parameters and keywords, ending with a semicolon.

There are two types of command, validation commands and site description commands. These are described below.

VALIDATION COMMANDS

There are two validation commands :

VAL $\frac{\text{NBILLCHK;}}{\text{BILLCHK}}$	which validates the site catalog
NVAL ;	which causes no validation of the site catalog

After a site catalog is set up, there may be mis-specifications and so on, and if VAL is used, the project, user and billing information will be checked for these. Conversely, if NVAL is used, these checks will be ignored.

If the BILLCHK keyword of the VAL command is specified, project, user and billing information will be checked. Otherwise, only the project and user information is checked.

SITE DESCRIPTION COMMANDS

The site description commands are used to create, modify, list and delete project, user and billing information in the site catalog.

The general format of these commands is :

– Operation-code object-name, parameters;

The operation-codes available are :

CRx	Create object type x
LSx	List object type x
MDx	Modify object type x
DLx	Delete object type x

where x may be :

U	for a USER
P	for a PROJECT
B	for a BILLING

All the commands, which are described below, have a free format (for example, they may be spread over more than one card), and a string of characters may be protected by quotes ('), for example,

PASSWORD = 'A*.'

Comments may be present in commands and between commands.

Example :

```
CRB DEPT1.BILLING/*BILLING CENTRE OF DEPT1/*
```

The star convention may be used in LS and DL commands, where it means "all". For example,

DLU *.BOB	means "delete user BOB from all projects"
LSP *	means "list all projects and their dependent users and billing names".
LSU *.*	means "list all users with their projects and billing names".

THE PROJECT COMMANDS

The project commands are :

CRP project-name $\left[\text{RIFCODE} = \left\{\begin{array}{l}\text{MAIN}\\ \text{STATION}\\ \text{STATN}\end{array}\right\}\right]$

$\left[\text{,APPLIST} = (\text{application-name}\left[\text{,application-name}\right])\right]$

MDP project-name $\left[\text{,RIFCODE} = \left\{\begin{array}{l}\text{MAIN}\\ \text{STATION}\\ \text{STATN}\end{array}\right\}\right]$

$\left[\left\{\begin{array}{l}\text{,APPLIST} = (\text{application-name [,application-name]})\\ \text{,ADDLIST} = (\text{application-name [,application-name]})\end{array}\right\}\right]$

;

LSP $\left\{\begin{array}{l}*\\ \text{project-name}\end{array}\right\}$;

DLP $\left\{\begin{array}{l}*\\ \text{project-name}\end{array}\right\}$;

The parameters are :

project-name	The name of the project; a simple name of eight characters maximum length.
RIFCODE	The RIF code values. MAIN Main operator project. STATION and STATN Station operator project.
APPLIST	A list of up to four application names for TDS. The names may be up to four characters long.
ADDLIST	A list of applications to be added to the current list.

THE USER COMMANDS

The user commands are :

```
CRU project-name.user-name [,PASSWORD=password]
                           [,TDSCODE=code]
                           [, {DFLT  }]
                              {NDFLT }
                           ;

MDU project-name.user-name [,PASSWORD=password]
                           [,TDSCODE=code]
                           [, {DFLT  }]
                              {NDFLT }
                           ;

     { *.*                       }
LSU  { *.user-name               }  ;
     { project-name.*            }
     { project-name.user-name    }

     { *.*                       }
DLU  { *.user-name               }  ;
     { project-name.*            }
     { project-name.user-name    }
```

The parameters are :

project-name. user-name	Simple names of eight characters maximum.
PASSWORD	A string of up to eight characters. The length of the string is significant and no padding with blanks is allowed. The password is not used in a batch environment.
TDSCODE	A string of up to eight hexadecimal characters defining the different TPR's the user can access, each TPR being coded as a bit in TDSCODE.
DFLT	Indicates that the project is the default for the user. NDFLT indicates that it is not the default.

THE BILLING COMMANDS

The billing commands are :

```
CRP project-name.billing-name [,CREDIT=nnnnnnnn]
                              [, { DFLT  }]
                                 { NDFLT }
                              ;

MDB project-name.billing-name [,CREDIT=nnnnnnnn]
                              [, { DFLT  }]
                                 { NDFLT }
                              ;

      { *.*                         }
LSB   { *.billing-name              }  ;
      { project-name.*              }
      { project-name.billing-name   }

      { *.*                         }
DLB   { *.billing-name              }  ;
      { project-name.*              }
      { project-name.billing-name   }
```

The parameters are :

project-name. billing-name	Simple names of up to eight characters in length.
CREDIT	The maximum credit of the billing in billing units. The default value is 99999999.
DFLT	Indicates that the billing is the default billing for the project. NDFLT indicates that it is not the default.

EXAMPLES OF THE USE OF $CATMAINT

Setting up a Site Description in the Site Catalog

The following example shows the entering of project, user and billing information and their relationships in a site catalog.

```
(1)  $JOB SITE-GEN,USER=VIP,PROJECT=SYSADMIN,
     BILLING=SITE-GEN;
(2)  CATMAINT COMFILE=*deck;
(3)  $INPUT deck;
(4)  CRP SYSADMIN;
(5)  CRP FIMA;
(6)  CRU SYSADMIN.VIP;
(7)  CRB SYSADMIN.SITE-GEN;
(8)  CRU FIMA.MAN;
(9)  CRU FIMA.WOMAN;
(10) CRU FIMA. CHILD;
(11) CRB FIMA.USERACT1;
(12) CRB FIMA.USERACT2;
(13) CRU SYSADMIN.MAN;
```

```
(14) VAL;
(15) $ENDINPUT.
(16) $ENDJOB;
```

Notes :

The above job creates :

– 3 billings, SITE-GEN, USERACT 1 and USERACT 2 (7, 11 and 12)

– 2 projects, SYSADMIN and FIMA (4 and 5)

– 4 users, MAN, WOMAN, CHILD and VIP (6, 8, 9 and 10)

– 5 relationships for FIMA (8, 9, 10, 11 and 12)

making a total of 17 objects in the site catalog. All the billing attributes are set to the default (99999999). USERACT1 is the default billing for FIMA, as it is the first one listed for it.

Modifying a Site Description in the Site Catalog

The example below shows the modification of the site description created in the previous example.

```
(1)  $JOB SITE-MANLIB,USER=VIP;
(2)  CATMAINT COMFILE=*libdeck;
(3)  $INPUT libdeck;
(4)  DLU *.CHILD;
(5)  DLU FIMA.*;
(6)  DLB FIMA.*;
(7)  DLP FIMA;
(8)  $ENDINPUT;
(9)  $ENDJOB;
```

Notes :

The star convention has been used, and the project FIMA, along with all associated users and billings, deleted by statements 5, 6 and 7. The user CHILD has been deleted from all projects by statement 4.

It is also possible to modify the actual descriptions using the MD set of commands.

MESSAGES AND DIAGNOSTICS OF $CATMAINT

There are three types of message produced with $CATMAINT :

- Non-fatal warnings
- Syntax errors
- Abort messages.

Syntax Errors

```
SYNTAX ERROR  UNKNOWN OPCODE
SYNTAX ERROR  PERIOD MISSING AFTER PROJECT NAME
SYNTAX ERROR  PROJECT NAME MISSING
SYNTAX ERROR  ILLEGAL PROJECT NAME
SYNTAX ERROR  BILLING NAME MISSING
SYNTAX ERROR  ILLEGAL BILLING NAME
SYNTAX ERROR  USER NAME MISSING
SYNTAX ERROR  ILLEGAL USER NAME
SYNTAX ERROR  ILLEGAL KEYWORD
SYNTAX ERROR  PREMATURE END OF STATEMENT
SYNTAX ERROR  ILLEGAL COMMA SIGN
SYNTAX ERROR  EQUAL SIGN MISSING
SYNTAX ERROR  ILLEGAL USE OF STAR (*)
SYNTAX ERROR  DUPLICATE RIFCODE
SYNTAX ERROR  ILLEGAL RIFCODE VALUE
SYNTAX ERROR  DUPLICATE APPLICATION NAME LIST
SYNTAX ERROR  ILLEGAL APPLICATION LIST VALUE
SYNTAX ERROR  ILLEGAL APPLICATION NAME
SYNTAX ERROR  DUPLICATE DEFAULT VALUE
SYNTAX ERROR  DUPLICATE CREDIT
SYNTAX ERROR  ILLEGAL CREDIT VALUE
SYNTAX ERROR  MISPLACED DELIMITER
SYNTAX ERROR  DUPLICATE TDSCODE
SYNTAX ERROR  ILLEGAL TDSCODE VALUE
SYNTAX ERROR  DUPLICATE PASSWORD
SYNTAX ERROR  ILLEGAL PASSWORD
```

Abort Messages

*** ABORT	ILLEGAL ACCESS TO CATALOG
*** ABORT	CATALOG ERROR TO BE CHECKED
*** ABORT	NO ROOM FOR NEW APPLICATIONS
*** ABORT	COMFILE IS TOO BIG
*** ABORT	COMFILE IS NOT ASSIGNED
*** ABORT	COMFILE DOES NOT EXIST
*** ABORT	COMFILE CANNOT BE OPENED
*** ABORT	COMFILE IS DUMMY
*** ABORT	MEMORY OVERFLOW,RETRY LATER
*** ABORT	COMFILE RECORD CANNOT BE READ
*** ABORT	COMFILE CANNOT BE CLOSED
*** ABORT	

Warning Messages

* WARNING	DEFAULT VALUE IGNORED
* WARNING	PROJECT UNKNOWN,STATEMENT IGNORED
* WARNING	USER UNKNOWN,STATEMENT IGNORED
* WARNING	BILLING UNKNOWN,STATEMENT IGNORED
* WARNING	RELATION UNKNOWN,STATEMENT IGNORED
* WARNING	UPDATE INSUFFICIENT CREDIT
* WARNING	THE PROJECT ALREADY EXISTS,STATEMENT IGNORED
* WARNING	THE USER ALREADY EXISTS,STATEMENT IGNORED
* WARNING	THE BILLING ALREADY EXISTS,STATEMENT IGNORED
* WARNING	PROJECT.USER ALREADY EXISTS,STATEMENT IGNORED
* WARNING	PROJECT.BILLING ALREADY EXISTS,STATEMENT IGNORED
* WARNING	PROJECT.USER DOES NOT EXIST,STATEMENT IGNORED
* WARNING	PROJECT.BILLING DOES NOT EXIST,STATEMENT IGNORED
* WARNING	PROJECT.LINKS(USER,BILLING)EXIST,STATEMENT IGNORED
* WARNING	STATEMENT IGNORED

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE	SERIES 60 (LEVEL 64) GCOS SYSTEM MANAGEMENT GUIDE ADDENDUM A	ORDER NO.	AQ09-01A
		DATED	AUGUST 1979

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here ☐ and furnish complete mailing address below.

FROM: NAME ______________________ DATE ____________

TITLE ______________________

COMPANY ______________________

ADDRESS ______________________

____

CUT ALONG LINE

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

FOLD ALONG LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

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

FOLD ALONG LINE

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 Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

21703, 1.5978, Printed in U.S.A.

AQ09, Rev. 1