Functional Specification for

TOPS20 COMMON FILE SYSTEM

1.0 PRODUCT OVERVIEW

1.1 Product Description

This project is to develop a "Common File System" for TOPS20. The Common File System capabilities are applicable to configurations of two or more 36-bit processors, each with its own main memory, interconnected by a high speed bus ("CI"). The objective of the Common File System ("CFS") is that disk structures and files within such a system are available to jobs on all processors, regardless of the physical connections of the disk devices.

1.1.1 Architectural Position -

CFS is a component of the "loosely-coupled systems" architecture and is the first piece of that architecture to be implemented. Some of the other components are:

1. DECnet/CI
2. CI-wide IPCF
3. CI-wide ENQ/DEQ
4. CI-wide GALAXY

As can be seen, the ultimate LCS product is an extensible multi-processor system. CFS is being implemented first because it is the most visible of the pieces and because it provides a useful extension to TOPS-20 even without the other LCS components.

1.1.2 Reltationship To Other CI Products -

CFS is indpendent of the other high-level CI protocols. That is, CFS can exist on a system that does not support MSCP. All that is required is the SCA layer of the CI protocol. In the following sections, mention is made of MSCP, the MSCP server and other CI applications protocols. These references are provided to explain the relationship of CFS to the other committed CI products, but CFS remains distinct and independent of them. The specifics of these other protocols, and any limitations or restrictions, are described in other documents.

1.2 Markets

The Common File System is a general operating system capability and is applicable to all present DECSYSTEM-20 markets.

1.3 Competitive Analysis

This project provides more and larger configuration alternatives than previously available. This project is not closely related to "distributed processing" in that it is only applicable to configurations on a CI and therefore within the 100 meter limit of the CI.

VAX/VMS is developing a means for multiple processors to reference files on a single disk system; however, the basic difference in filesystem architecture between TOPS20 and VMS makes the projects somewhat different.

Related capabilities include "Network File Access" or other techniques for moving files among nodes of a network. CFS is a more powerful and transparent form of file access because it implements all monitor file primitives visible to the user program and operates over a high speed bus.

"Multiple Processors" (implying shared memory as with TOPS10 SMP) is a related capability. SMP is a more powerful approach to the use of multiple processors in that it provides greater transparency and better dynamic load leveling. There are compensating advantages of CFS over SMP in the area of failsoft and isolation of failures, and in the maximum size of configurations which can be supported.

1.4 Product Audience

The principal customer for CFS is one who now has, or who needs, multiple KL-10 processors and wishes to run them as a single system. Since DEC is not offering a follow-on PDP-10 processor, most, if not all, of the LCG customers fit this description.

CFS-20 is meant as a complement to DECnet services, and in some cases sites with multiple KL10's may find that sharing via DECnet is adequate.

2.0 PRODUCT GOALS

2.1 Performance

1. All unprivileged monitor calls which affect disk files on present one-processor TOPS20 systems will work and will have the intended effect on any disk structure within the configuration.

2. The overhead associated with maintaining the common file data base on multiple processors will cause an increase of not more than 10% in execution time of file primitives and operations.

3. A processor referencing files on a disk not directly connected will incur no additional overhead in transferring data.

We expect to use MSCP (Mass Storage Control Protocol) for the data transfers to support file operations over the CI. This will exist on the CI along with other protocols supporting other functions. MSCP should achieve efficient use of the CI, low-overhead operation of the monitors, and high-bandwidth file interchange. File structure information such as directories and index blocks is passed exactly as read from disk. By passing TOPS20 file data directly, we avoid the overhead of copying and conversion incurred with other protocols.

However, a processor acting as a file server for another processor will incur overhead for this activity not relating to jobs running on it. This overhead will involve primarily instructions executed at interrupt level and main memory space to buffer data being transferred.

CFS supports shared-writable pages (simultaneous write file access) on multiple processors. This is used for various internal mechanisms (e.g. directory lookup, disk allocation tables) as well as user program functions. This type of access generates IO activity and overhead not present on single-processor systems. Because data cannot actually be referenced simultaneously by two processors, it must be moved from one to another by the operating system. Users will be advised of this and should arrange applications so as to avoid frequent write references to the same data from different processors.

Since the monitor itself uses this facility, we conducted a study of monitor reference patterns to ensure that this activity will not be a significant bottleneck. We recorded monitor reference patterns to directories and disk allocation tables under actual and simulated loads. This was done by using the SNOOP facility to detect and record references where the job making the reference is different from the job which made the most recent previous reference.

This provides worst-case data on the frequency of moving monitor data between processors. We determined that only a few (3 or less) directories were referenced sufficiently often to be of interest. These were all common system directories (e.g. <SUBSYS>), and the frequency was not so high as to suggest a problem. This small additional overhead is greatly outweighed by the disk space savings of not having to duplicate the SYS: files for each of the CFS processors.

2.2 Environments

Minimum configuration requires two processors and a CI. Each processor must have a connection to the CI. (The question has been raised as to whether CFS might be operable over an NI connection. This will not be supported in first release. There should be no logical reason that NI couldn't be used, but additional study and experience is necessary to understand the performance implications. Additional implementation work would also be needed.)

Each processor must have its own main memory, swapping device, boot device, and console.

Each processor must have direct access to its public disk structure. In a future release, it may be possible to eliminate this requirement. However, there are other requirements for a directly connected disk (e.g. swapping) which will also have to be addressed.

The maximum configuration for first release of CFS is two processors; however there shall be no CFS-specific software limitation on a larger number. This limit is based on our current knowledge of the CI and the lack of experience with this architecture. The practical limit may be higher. A maximum of one CI will be supported for first release.

2.3 Reliability Goals

1. A customer should be able to improve net system availability of his configuration by use of multiple processors and the CFS.

2. The CFS should cause no significant decrease in the reliability of each single processor.

3. Failure of one processor will have no effect on other processors except for file data which is in the memory of the failing processor.

2.4 Non Goals

1. CFS will support only disks.

2. CFS is not intended to work with operating systems other than TOPS20 or with machine architectures other than 36-bit.

3. CFS does not provide any automatic balancing of job load among processors in a configuration as does SMP. However, users should find it convenient to login to the less loaded processor and/or to switch processors (by logging out and back in again) if the load becomes unbalanced.

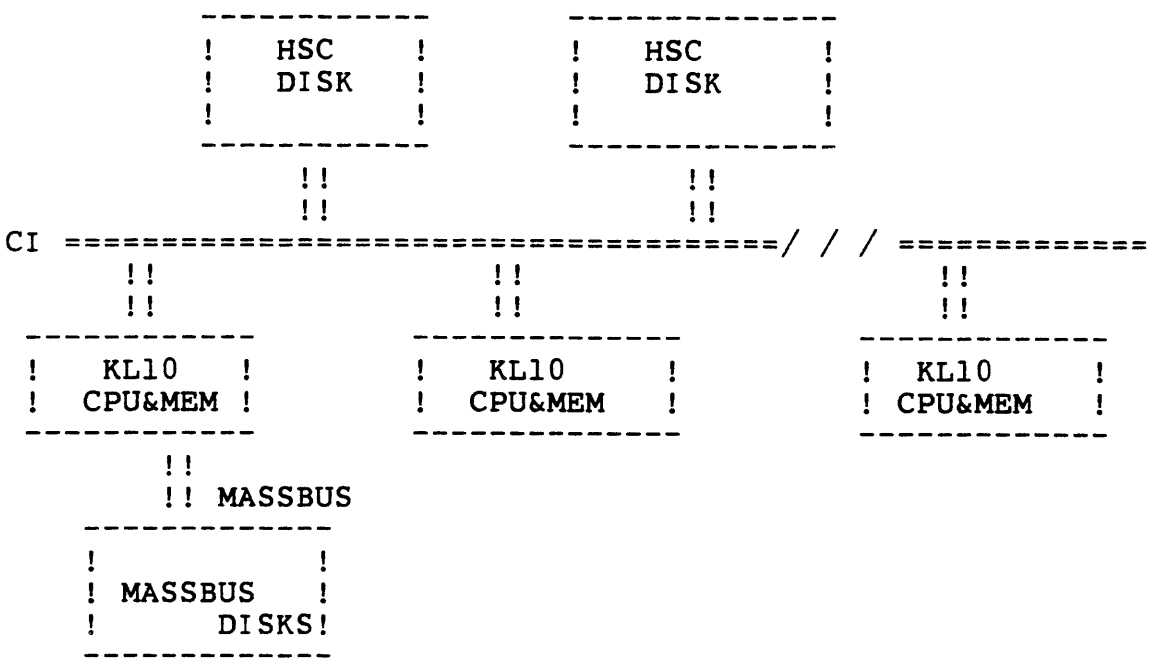
4. Applications that rely on ENQ/DEQ and OF%DUD are not supported by CFS.

5. IPCF applications will not communicate over CFS processors.

3.0 FUNCTIONAL DEFINITION

3.1 Operational Description

Two or more processors are interconnected via a high-speed bus ("CI") having a bandwidth at least comparable to disk transfer rates. A disk which is to be used by a processor must have a direct path to that processor; the disk must be either on the CI, on a directly connected MASSBUS, or attached to another KL-10 running an MSCP server.



One or more logical structures exist on the set of disks. All of these structures are visible to jobs on all of the processors unless the system administrator specifically declares particular structures as "exclusive" to a particular processor.

In order to provide access to Massbus disks connected to a KL10, the KL10 will act as a logical disk controller on the CI for the Massbus disks. There is no visible distinction between a disk structure directly connected to a processor and one which is accessed via another processor. The usual monitor calls are used to access files and structures, and all file open modes are allowed with the exceptions listed below. Shared file access is permitted, and programs need not be aware that other jobs sharing a file are on different processors; however, it may be advisable for reasons of efficiency to avoid simultaneous modification of a file on different processors.

File facilities specifically include:

1. File naming and lookup conventions (GTJFN) - File names on the common file system include structure, directory and subdirectories, file name, extension, generation number, and attributes. Full recognition and wild-carding is available; name stepping (GNJFN); normal access to FDB.

2. Usual open and close modes (OPENF, CLOSF, CLZFF).

3. Usual data transfer primitives, both sequential and random (BIN, BOUT, SIN, SOUT, SINR, SOUTR, RIN, ROUT, DUMPI, DUMPO).

4. File-to-process mapping (PMAP) including all modes (shared read, copy-on-write, shared write, unrestricted read).

5. The device type associated with files on the common file system is the same as that presently used for disk.

6. Privileged operations MSTR (mount structure) and DSKOP.

The above includes all file system primitives relating to accessing files and transferring data but does not include other primitives which may use certain file system entities but which are considered separate and distinct facilities (e.g. ENQ/DEQ).

3.2 Restrictions

A file open with OF%DUD (don't update disk) on one processor may not be opened on any other processor. This results from the fact that processors share file data by writing any changed files to the disk before passing control to another processor. Since OF%DUD implies that the disk copy of a file may not be changed until the user process approves the change, OF%DUD cannot be supported with CFS.

Other devices such as magtapes and line printers are not part of CFS and may not be open simultaneously on multiple processors.

Use of simultaneous write access with active writing of file data by jobs on different processors requires the system to move pages among the processors and hence will be much slower than on a single processor. The write token is maintained on a per-OFN basis. This means that a program requiring write access to any one or more pages must have exclusive access to the entire OFN. Each OFN represents 256K words of the file. For large files, programs on different processors could be executing simultaneous write references with no delay if they were referencing data in different 256K sections of the file.

A structure must be "mounted" on any processor which is to access files on it. To be physically removed from a drive, a structure must be dismounted by all processors.

The relevant Galaxy components should be modified to provide mount information from processors other than the one on which they are running, but this in not planned for FCS of CFS. Hence an operator will have to query the OPR program on each processor to find out what users have the structure mounted. Each processor will know, however, which other processors have the structure mounted so that the operator can quickly determine if the structure can be removed.

Finally, it is not possible for two or more CFS processors to establish an ENQ resource for the same file. This restriction of ENQ is made to prevent malfunctioning of programs that rely on ENQ as a file semaphore and will be removed once the LCS-wide ENQ/DEQ facility is provided.

4.0 COMPATIBILITY

4.1 DEC Products

All program and user interfaces are compatible with previous versions of TOPS20.

Mountable disk structures are compatible with previous versions of TOPS20.

4.2 DEC Standards

The CFS will use the corporate SCA protocol on the CI bus and will use a private SYSAP-level protocol.

The CFS will not use DECNET.

4.3 External Standards

None applicable.

5.0 EXTERNAL INTERACTIONS AND IMPACT

5.1 Users

All users of the disk file system are potential users of CFS; however most users will not be aware of or affected by CFS. Some applications developers will rely on CFS to allow applications to exist on multiple processors and communicate through files.

5.2 Products That Use This Product

The following may use CFS: RMS, DIF, language OTS's.

5.3 Products That This Product Uses

The following hardware components are required:

KLIPA (CI20) - Interface between KL10 and CI bus.

KL10 Microcode - modifications to support "write access in CST".

The following software modules are required:

KLIPA driver

and time. Systems Communications Services (SCA/SCS).
The following are optional MSCP driver
MSCP server

5.4 Other Systems (Networks)

The CFS is not visible to other network hosts; the files in the CFS disk structures may be accessible by remote nodes as provided by other facilities (DAP, NFT, etc.) Each processor in a CFS configuration is a separate network node with its own node name.

The CFS itself does not use node names to reference files and hence is independent of any constraints or requirements of network node naming.

5.5 System Date And Time

The CFS systems guarantee that they all use the same date and time. This requirement insures that files written on one of the processors will have a creation date and time consistent with the other CFS processors. If the processors were allowed to have different date and time values, many of the file-oriented utilities would malfunction.

This is accomplished by having the systems inform each other whenever the local date and time is changed. Also, a newly loaded system will use the date and time provided by the other CFS systems. This last item implies that a CFS system loaded while at least one other CFS system is already running will not have to prompt the operator for the date and time. In order to make the start-up dialog seem the same, the system will type:

The date and time is: xxxxxxxxxx

where it now prompts for the date and time. This also serves as a check on the date and time.

5.6 Job Numbers

The CFS systems must use a mutually exclusive set of job numbers. This is because many system utilities, and user programs, include the job number in the name of "session" files and other per job data to avoid conflicts among jobs using the same file directories. CFS systems, therefore, will acquire a set of global job numbers to use and will insure that no other CFS system uses those numbers. This implies that the user-visible job number, as seen in a SYSTAT command, may not correspond to the monitor's internal representation for that job. However all JSYSes that either provide or accept a job number will be modified to account for the the new gloabl job numbers.

5.7 Interprocessor Communications

CFS provides only sharing of files. Without some ancillary capability, such as DECnet, processes on different CFS processors have now way of exchanging "events" such as interrupts. Processes on the same processor have a choice of several IPC mechanisms, such as

1. DECnet

2. IPCF

3. ENQ/DEQ

4. THIBR/TWAKE

All of these provide inter-process events (viz. interrupts) and may also "carry" some amount of data (e.g. an IPCF message). However, CFS provides a data carrying mechanism, namely shared files, but it provides no intrinsic event generator.

CI/DECnet is the ideal mechanism for a CFS IPC. However, CI/DECnet may not be available with the first release of CFS. Therefore, there will be no reliable IPC for use by "distributed" applications.

It is possible, however, to implement THIBR/TWAKE across CFS processors. This is true because CFS will guarantee that the job numbers used by the various processors are mutually exclusive of one another.

Presently, there is no commitment to provide a CFS-wide TWAKE (THIBR needs no changes), but the work required is modest.

5.8 Data Storage, File/Data Formats, And Retrieval

The CFS requires an open file data base which is resident in each processor of a configuration. 2-4 words per OFN are required. Other resident storage requirements are one page (512 words) or less. As a side effect of allowing all processors access to all mounted structures, it may be desirable to build standard monitors with a larger number of mountable structures than at present.

The file structure will be identical with previous releases of TOPS20.

Files may be saved and restored with DUMPER without regard to which processor DUMPER is run on, except that DUMPER must be running on the processor which has direct connection to the required tape drive.

5.9 File And Data Location

CFS is unaware of the physical location of file data. That is, a shared file may be located on a CI disk, a shared Massbus disk or on a disk accessed by the MSCP server.

This latter case, that of the MSCP server, should be used only when absolutely required. That is, if the file could be located on a CI disk or a shared Massbus disk, it should be. Files accessed through the MSCP server impose a significant burden on the processor running the server, and if such files are accessed frequently, the result may well be unacceptable. Clearly, files that must reside on PS: structures and must also be shared may by shared only through an MSCP server. However, such files should not be frequently accessed by other processors.

It is, for example, entirely inappropriate to place all of the SYS: files for all of the CFS processors on disks that must be accessed be the MSCP server.

5.10 Protocols

CFS will use the corporate SCA protocol on the CI bus.

CFS will use a private protocol for control of file openings, structure mounts, file state transitions, etc. There is no present corporate protocol which supports these functions.

The CFS protocol uses only SCA messages. The general format of a CFS message is:

```
DEFSTR  CFUNQ,SCALEN,35,18      ;NUMBER OF THIS VOTE OR REQ UNIQUE CODE
DEFSTR  CFCOD,SCALEN,17,6       ;OPCODE FOR VOTING
        .CFVOT==1               ;VOTER
        .CFREP==2               ;REPLY TO VOTE
        .CFRFR==3               ;RESOURCE FREED
        .CFCEZ==4               ;SEIZE RESOURCE
        .CFBOW==5               ;Broadcast OFN change
        .CFBEF==6               ;Broadcast EOF
DEFSTR  CFFLG,SCALEN,11,12      ;Flags
 DEFSTR CFODA,SCALEN,0,1        ;Opt data present
 DEFSTR CFVUC,SCALEN,1,1        ;Vote to include HSHCOD
CFROT==SCALEN+1                 ;ROOT CODE FOR THIS VOTE
CFQAL==SCALEN+2                 ;QUALIFIER CODE FOR THIS VOTE
CFTYP==SCALEN+3                 ;Vote reply or request type
CFDAT==SCALEN+4                 ;Optional data, it present
CFDT1==SCALEN+5                 ; second word of optional data
CFDST0==CFDT1+1                 ;STR free count in bit table
CFDST1==CFDST0+1                ;Transaction count of CFDST0
```

This format is used to both request CFS resources and to reply to resource requests.

The SYSAP name for CFS is: LCS20$CFS. This name uniquely identifies the TOPS-20 CFS SYSAP for a homogeneous CI environment. Since there is no central registry of SYSAP names, configuring a CI with other processor types (e.g.

VAX) may result in confusion of names and protocols.

5.11 Protocol Operation

The CFS protocol is a "veto" protocol. That is, each request must be approved by all of the CFS processors or it is disallowed. Therefore, a single dissenting processor is sufficient to refuse a request.

Each processor is required to remember only the resources it owns. Therefore, when it "votes", it expresses only the relationship of the request to its own resources. Consequently, each processor must be polled every time a CFS resource change is to occur.

5.12 Modifications To MSTR

The MSTR JSYS has been modified to allow structures to be declared to be "shared" or "exclusive". A shared structure may be mounted by other CFS processors, whereas an exclusive structure may be mounted only on this processor.

The structure status bit, MS%EXL declares that a structure is to be mounted exlusively and is returned with the appropriate value with the structure status.

Also, there is a new MSTR function, .MSCSM, that changes the shared/exclusive attribute of a mounted structure. The calling sequence is:

MSTR

AC1: -2,,.MSCSM
AC2: ADDR

ADDR: device designator
ADDR+1: new attribute

5.13 CFS Components

CFS is implemented throughout the TOPS-20 monitor. However, the code specific to the CFS protocol is contained in the module CFSSRV. CFSSRV is the CFS SYSAP as well as a collection of routines to interface to the preexisting TOPS-20 services. CFSSRV uses the following SCA call backs:

1.

.SSMGR message received

2.

.SSPBC port broke connection

3.

.SSCTL connect to listen

4.

.SSCRA connect response available

5.

.SSMSC message/datagram send complete

6.

.SSNCO node on-line

7.

.SSNWO node off-line

8.

.SSOSD OK to send data

9.

.SSRID remote initiated disconnect

10.

.SSCIA credit available

In addition, CFS uses the following SCAMPI routines:

1. SC.SOA
2. SC.RCD
3. SC.CON
4. SC.DIS
5. SC.SMG
6. SC.RMG

7. SC.LIS

8. SC.REJ

9. SC.ACC

SCAMPI must reliably inform CFS of any CI configuration changes, including newly established or failed port-to-port VCs.

The remaining CFS code is found in-line as part of existing TOPS-20 file system services.

CFS uses only SCA messages.

5.14 Significant Data Structures

The advent of CFS creates the following new data structures and conventions:

1. A new per-OFN word, SPTO2

2. directory locks are now CFS resource blocks

3. directory allocation entries are now CFS resource blocks

4. frozen write file openings create two resources

5. other file openings create only one resource

6. each OFN has a CFS "access token" as a CFS resource

7. each mounted structure creates two CFS resources

8. BAT block locks are CFS resource blocks

Note that in some cases the CFS resource replaces the existing lock, viz. directory locks, and in other cases the CFS resource exists as a "copy" of the information, viz. structure mounts, so that the CFS protocol service can manage the resource locally. In principle, there is no difference between these kinds of resources, and only the higher-level monitor code that creates the CFS resource knows which kind each is.

The bundled CFS, that is the release 6 monitor without CFS support, still uses CFS to manage the "changed" monitor resources. However, in many cases, as with the file resources, the CFS resource is not created as it is not needed for any internal monitor coordination.

5.15 Interfaces To CFSSRV

CFSSRV contains a number of jacket routines that interface between the TOPS-20 file system and the CFSSRV resource manager. The sigificant interface routines are:

CFSAWT/CFSAWP

T1/ OFN
T2/ access needed

Returns: +1 always

Called to manage the access token

CFSLDR/CFSRDR

T1/ Structure number
T2/ directory number

Returns: +1 always

Lock/unlock directory

CFSSMT

T1/ Structure number
T2/ access needed

Returns: +1 failed. Access invalid
+2 success

Mount strcuture

CFSSDM

T1/ Structure number

Returns: +1 always

Dismount structure

CFSSUG

T1/ Structure number
T2/ access

Returns: +1 can't change access
+2 success

Change structure access

CFSGFA

T1/ Structure number

T2/ XB address
T3/ Access type

Returns: +1 access conflicts with other system(s)
+2 success

Acquire file open locks

CFSFFL

T1/ Structure number
T2/ XB address

Returns: +1 always

Delete file open resources

CFSFWL

T1/ Structure number
T2/ XB address

Returns: +1 always

Free frozen write resource

CFSGWL

T1/ Structure number
T2/ XB address

Returns: +1 conflict with other CFS system
+2 success

Acquire frozen writer resource

As these jacket routines are really an integral part of the file system, the interfaces to these routines are really internal file system conventions and not external interfaces. Therefore, the detail of how these interfaces work is beyond the scope of this functional document.

5.16 PHYSIO Services Required

CFSSRV requires a routine in PHYSIO to request that dual-ported disks not be accessed by this processor. The call is:

CALL PHYMPR

Returns: +1

Also, it requires a routine to cancel the action of PHYMPR:

CALL PHYUPR

Returns: +1

In addition to these, CFS requires that PHYSIO and its lower level drivers correctly support access to dual-ported Massbus disks. In particular, work must be completed in managing dual-ported disks and in insuring that the port is released at the proper times.

6.0 RELIABILITY/AVAILABILITY/SERVICEABILITY (RAS)

6.1 Failures Within The Product

Failures within the CFS-specific software will most likely cause a crash of one processor in a multi-processor environment. Such failures may include loss of recently modified file data. Failures which affect inactive files or file directories are possible, but should be no more frequent than at present.

6.2 Failure Of A CFS Processor

Operation of CFS should permit crash of one processor for any reason without loss of other processors in the configuration. CFS relies on SCAMPI to detect a processor failure and consequently the CFS protocol has no mecahnism for idle polling. If a processor fails, the other processors will be unaffected, except that the CFS code on each of the surviving processors must "renegotiate" any outstanding requests for file accesses.

A processor may be brought on line without restarting other processors in the configuration.

Any disks which are available only via a failed processor will be unavailable so long as that processor is inoperative. If such disks are dual-ported to a different processor, they may be mounted via that processor and remain in use although all open files must be re-opened.

With HSC50, most disk errors will not be seen by the processor(s). All recovery and logging will be handled by the HSC50. Any disk errors that are reported to the processor will be logged in the system error file for that processor. Disk errors occurring on pages that are being

"passed through" a processor (e.g. a KL10 servicing a request for a Massbus disk) will be logged on the processor to which the disk is directly connected. If a hard failure occurs such that the server processor must inform the requesting processor that the request could not be completed, then the requesting processor will also log the failure.

6.3 CI Failures

Should the CI fail, or should a processor's KLIPA fail, the CFS processors must insure that data on shared disks is not corrupted. This is accomplished as follows.

If a processor detects it is no longer connected to the CI it must refrain from referencing any sharable disks. A sharable disk is any HSC-based disk or any dual-ported MASSBUS disk. A processor is considered no longer attached to the CI if it cannot send a "loopback" message to itself.

Should a processor's KLIPA fail, and then be restarted (e.g. by reloading the microcode), the processor will not be able to continue running if there are other CFS processors on the CI. This prohibition avoids the problem of the system rejoining the CFS network having stale data about the CFS resources, or having data about a previous incarnation of the network.

Should a processor be "cut off" from the CI indefinitely, it may continue running but without being able to access sharable disks.

6.4 Testing For Errors

CFS will be run in the DVT environment so that it may be evaluated with regard to faults.

Many of the "normal" CFS errors may be tested without explicit fault insertion. The following simple procedures tests much of the CFS error recovery code

1. halt one of the CFS processors
2. bring up a new CFS processor
3. reload one of the KLIPAs

7.0 PACKAGING AND SYSTEM GENERATION

7.1 Distribution Media

CFS is an unbundled product. Each monitor has the bulk of the CFS support, but non-CFS monitors have a dummy version of the CFSSRV protocol module. CFS sites will receive a separate tape containing the proper CFSSRV.

7.2 Sysgen Procedures

Only CFSSRV differs from a bundled to an unbundled monitor. The SYSFLG switch, CFSSCA specifies the type of monitor.

7.3 Bundled And Unbundled CFS Systems

There is no protection in the monitor for running a bundled and an unbundled monitor on the same CI. This is particularly important for F-S procedures as the KLAD monitor may not be compatible with the system environment. Running a mixed configuration is potentially catastrophic as file structures may be destroyed.

8.0 REFERENCES

1. Functional Specification for Loosely Coupled Systems (LCS) - Fred Engel, 30 April 1980

2. LCG CI Port Architecture Specification, 11-July-83 (Keenan)

3. LCS and the Common File System (Memo) - Dan Murphy, 15 Jan 1980

4. CFSDOC (memo) - Arnold Miller