+---------------+
| d i g i t a l ! | I N T E R O F F I C E M E M O R A N D U M
+---------------+

TO:

DATE: 8-Feb-85

FROM: Clair Grant
Ron McLean

DEPT: Large Systems
Software Engineering

LOC: MRO1-2/L10

EXT: 6877

SUBJ: TOPS-20 Multi-Access Disk Management Specification

Most of the important actions decsribed in this specification involve accessing a disk when CI communication is, or has been, disrupted. Thus, while still true for an HSC-controlled disk, most of the details are interesting only if you are referring to a dual-ported MASSBUS disk because if you can't communicate over the CI, you can't access an HSC-controlled disk anyway.

Edit History

5-Jan-85	Add symbols for UDB offsets and path info bits in the PDB picture. Remove the WIRSTS reference in the CI Wire Status section heading.
12-Jan-85	Fix SMON% arguments. Better descriptions in Disk Access Logic section. Add PHYSIO's New Homeblock Code for PDBs section.
23-Jan-85	Enhance Disk Access Logic section
31-Jan-85	Change the way RP20s get assigned serial numbers
8-Feb-85	-1 in right half of UDBNPR

1.0 GOALS

The multi-access project has 3 goals: 1) no data corruption, 2) 5.1 compatibility, and 3) minimal writing of management data (overhead) to the disk. The second goal is a bit unusual in that we are providing compatiblity for a feature we didn't support prior to 6.0, we just didn't prevent it. Some customers have come to depend on this and it is in our best interest to allow them to continue as they have in the past.

1.1 No Data Corruption

The major goal of this project is to ensure data integrity on all multi-accessed disks in the TOPS-20 file system. This is accomplished by allowing TOPS-20 to access a disk which has been accessed by other systems on the CI only when, for each of those systems:

1. the system is down, or

2. the system is up and we have a CFS connection open to it

1.2 5.1 Compatibility

Prior to Release 6.0 TOPS-20 did not provide a facility to manage access to multi-accessed disks. But, it did not prevent a customer from porting an RP06 to 2 systems and writing software to manage such a configuration. This was clearly stated as unsupported by TOPS-20.

In Release 6.0 TOPS-20 manages multi-accessed disks with CFS, yet a customer may still wish to use whatever local management scheme was used in the past on certain disks. TOPS-20 provides for this by allowing the customer to declare disk drives and disk packs as "don't-care" access, meaning they don't-care to have TOPS-20 manage multi-access and TOPS-20 should honor all access requests.

NOTE

The customer must explicitly request this "don't-care" designation, as described in later sections.

1.3 Minimal Overhead Writing To The Disk

Some amount of writing to all multi-access disks is required by this management scheme; this is kept to a minimum by reading disk only when a significant event occurs and writing the disk only when pertinent data has changed. This is the primary reason a keep-alive mechanism was rejected, to avoid constant the disk access.

2.0 RESTRICTIONS

2.1 Disk Serial Numbers

All disks in the TOPS-20 file system must have unique serial numbers in order for TOPS-20 to operate properly; serial numbers are now an integral part of TOPS-20's disk management and TOPS-20 outputs a PHYNOS BUGCHK when it discovers a disk without a serial number

Any disks without serial numbers must be fixed. Since RP20s don't have drive serial numbers, the monitor will make up one by adding the RP20 unit number to 8000 (decimal). This implies all RP20s connected to systems in a cluster must have unique unit numbers.

2.2 Configurations

The following configurations are illegal (not supported by TOPS-20) and TOPS-20 makes no predictions as to what will happen in such siutations.

1. a MASSBUS disk dual-ported between a 6.x system and a 5.x system
2. a MASSBUS disk dual-ported between two 6.x systems which are on different CIs

3.0 DEPENDENCIES ON RSX20F

Work in RSX20F was required by this project; this work provides TOPS-20 with more disk configuration information, allowing TOPS-20 to more accurrately manage multi-accessed disks.

3.1 Disk Configuration

RSX20F communicates its disk configuration to TOPS-20 at system startup by passing the drive serial numbers in a configuration packet. This helps TOPS-20 determine which disk drives are potentially being assessed by other KL10s, as opposed to those whose other port is to the Console Front End.

3.2 Stopping The CI Microcode

RSX20F stops the CI20 u-code whenever the HALT or ABORT commands are executed by the PARSER. ALso, HALT.CMD causes the CI20 u-code to halt. The instruction CONO KLP,400000 is used to halt the CI u-code.

NOTE

> The PAR>CONTINUE command does not do anything to the CI20. This means that after a HALT (which stops the CI20) the CI20 will not be restarted by the CONTINUE. The halted CI20 will be restarted when detected by the once-a-second check in PHYKLP.

4.0 USER INTERFACE

4.1 CHECKD Program

CHECKD new commands which allow a user to declare a structure as "don't-care" and "do-care". This command uses the MSTR% function .MSHOM (modify home block) to set the newly-defined word HOMDCF in the home block of each disk in the structure.

```
CHECKD>ENABLE DON'T-CARE structure-name

CHECKD>DISABLE DON'T-CARE structure-name
```

4.2 SMON%

A new SMON% function .SFDCD is used to declare to the running monitor that a disk drive is "don't-care". It's arguments are:

```
AC1/ .SFDCD
AC2/ address of argument block
```

Format of argument block:

Offset	Contents
0	Channel #
1	Controller #
2	Unit #

4.3 SETSPD Program

SETSPD has a new command which will use the new SMON% function. This command is placed in n-CONFIG.CMD and has the following format:

DONTCARE channel controller unit

4.4 BUGHLTs

The TOPS-20 BUGHLT code stops the CI20 u-code.

4.5 PAR>SHUT

If a SHUT (or PAR>DEP 20=1) causes the KL to halt, then the CI20 is also halted. If you end up at a breakpoint or nothing at all happens on the KL, nothing is done to the CI20.

5.0 DATA STRUCTURES

5.1 Processor Data Block (PDB)

The PDB resides in physical block 3 on a disk and is copied into the UDB at the following offsets:

```
        ---------------------------------------------
UDBSER  !     Current Drive Serial Number (word 1)    !
        ---------------------------------------------
        !     Current Drive Serial Number (word 2)    !
        ---------------------------------------------
UDBNPR  !Non-CI CPU Serial Number ,, -1                !
        ---------------------------------------------
UDBP00  ! Node 0 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP01  ! Node 1 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP02  ! Node 2 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP03  ! Node 3 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP04  ! Node 4 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP05  ! Node 5 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP06  ! Node 6 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP07  ! Node 7 Serial Number  ,,                !A!B!
        ---------------------------------------------
UDBP08  ! Node 8 Serial Number ,,                 !A!B!
        ---------------------------------------------
```

```
UDBP09  ! Node 9 Serial Number ,,                    !A!B!
        ---------------------------------------------
UDBP10  ! Node 10 Serial Number ,,                   !A!B!
        ---------------------------------------------
UDBP11  ! Node 11 Serial Number ,,                   !A!B!
        ---------------------------------------------
UDBP12  ! Node 12 Serial Number ,,                   !A!B!
        ---------------------------------------------
UDBP13  ! Node 13 Serial Number ,,                   !A!B!
        ---------------------------------------------
UDBP14  ! Node 14 Serial Number ,,                   !A!B!
        ---------------------------------------------
UDBP15  ! Node 15 Serial Number ,,                   !A!B!
        ---------------------------------------------


UDB%WA = 1B34
UDB%WB = 1B35

;Bit on means:
;  1) For our node - the wire is good
;  2) For a remote - the answer to request-ids is not no-response
;Bit off means:
;  1) For our node - the wire is bad
;  2) For a remote - the answer to request-ids is no-response
```

5.2 Unit Data Block (UDB)

In addition to the new UDB words for the copy of the disk's PDB, there are 2 other new UDB words which are part of the mutli-access project, namely,

UDBDCF - the don't-care flag word
UDBST1 - a second status word

The following are the bit definitions for UDBST1:

```
U1.OFS==1B0                     ;DUAL-PORTED DISK FORCED OFFLINE BY TOPS-20
U1.FED==1B1                     ;DRIVE PORTED TO FRONT-END
U1.DCD==1B2                     ;DISK HAS DON'T-CARE FLAG SET IN ITS HOMEBLOCK
U1.DCU==1B3                     ;DRIVE DECLARED DON'T-CARE
U1.HBR==1B4                     ;HOME BLOCK READ IN PROGRESS
U1.PDW==1B5                     ;PDB WRITE IN PROGRESS
U1.STC==1B6                     ;CI STATUS CHANGE WHILE READING HOME BLOCKS,
                                ; THEREFORE WE MUST DO IT AGAIN
U1.DCR==1B7                     ;DON'T-CARE ABOUT THIS DUAL-PORTED DISK (LOGICAL
U1.PHB==1B8                     ;PRIMARY HOMEBLOCK BAD
U1.SHB==1B9                     ;SECONDARY HOMEBLOCK BAD
U1.PDR==1B10                    ;PDB READ IN PROGRESS
U1.VV==1B11                     ;VOLUME VALID (TO FIX PROBLEM WITH RP07'S
                                ;AND DUAL PORTED DISKS)
```

5.3 Request-ID Status (RIDSTS)

The table RIDSTS (indexed by CI node number) contains information about the current status of each path to each of the other nodes on the CI. This status is a result of periodically sending REQUEST-IDs to all the other nodes on the CI, alternating paths on consecutive sends to an individual node. Of interest to the disk service are the bits which indicate if the last REQUEST-ID received an answer or there was no response on that path to the node. If REQUEST-IDs are being answered we assume there is a TOPS-20 running on the remote system, and if REQUEST-IDs are not being answered we assume TOPS-20 is not currently running on the remote system.

NOTE

The previous paragraph is critical in that it is the basis upon which many of the disk management decisions are made.

5.4 CI Wire Status

The locations CIWIRA and CIWIRB indicate the results of the latest periodic loopback packets to the STAR on the 2 wires; 0 = succeeded, non-0 = failed.

6.0 DISK ACCESS LOGIC

TOPS-20 uses the following algorithm when deciding if access to a disk should be allowed.

1. If the disk is single-ported, don't-care, both ports to us, or is ported to 20F, allow access. Otherwise (the disk is dual-ported to another KL10), move to next check.

2. If the disk has never been accessed by another node, allow access. Otherwise, move to next check.

3. If we are not on a CI, set disk offline. Otherwise, move to next check.

4. If there is a non-CI processor accessing the disk, set disk offline. Otherwise, move to next check.

5. If both of our CI wires are bad, set disk offline. Otherwise, move to next check.

6. Check all other nodes

```
FOR i = 0 to 15

  IF i .ne. us, THEN
     IF node i has accessed the disk, THEN
        IF node i is answering request-ids, THEN
           IF there is a CFS connection to node i, THEN
              NEXT i
           ELSE
              set disk offline and QUIT
           END
        ELSE
           IF we don't have a wire match with node i, THEN
              set disk offline and QUIT
           END
        END
     END
  NEXT i

  Allow access to disk
```

!A flow chart of the above follows: !.pg !.require "dpflow.pic"

7.0 PHYSIO'S NEW HOMEBLOCK CODE FOR PDBS

TOPS-20 now uses 3 homeblock areas on a disk:

```
HM1BLK==:1                      ;BLOCK # OF FIRST HOME BLOCK
PDBBLK==:3                      ;ADDRESS OF PDB BLOCK
HM2BLK==:^D10                   ;BLOCK # OF SECOND HOME BLOCK
```

The UDB status bit US.CHB is set whenever reading of the homeblocks is desired; the once-a-second code UNICHK detects it, sets U1.HBR and calls RDHBLK.

7.1 When The Homeblocks Are Checked

Homeblocks are read whenever a significant CI or disk event occurs. For some events the homeblocks of all disks are read; that's called a cluster configuration check. For othter events, the homeblocks are read for those disks previously accessed by a certain CI node; that's called a node configuration check. Finally, a disk configuration check occurs for an individual disk.

A disk is in the forced offline state until its homeblock checking is comepleted. The following sub-sections describe the routines which are called to cause homeblock checking.

7.1.1 PHYFED -

```
;PHYFED - CALLED BY DTESRV AT SYSTEM STARTUP TO DECLARE DISKS TO BE
;FRONT-END DISKS
; T1/ ADDRESS OF LIST OF SERIAL NUMBERS
;       CALL PHYFED             ;(T1/)
; RETURN +1
```

PHYFED causes a cluster configuration check.

7.1.2 PHYSTC -

```
;PHYSTC - CALLED BY PHYKLP ON A WIRE STATUS CHANGE, A REQUEST-ID STATUS
;CHANGE, OR THE CI U-CODE IS STARTED
;       CALL PHYSTC             ;(/)
;RETURNS:       +1
```

PHYSTC causes a cluster configuration check. However, single-ported, don't-care, and front-end disks will not be checked.

7.1.3 PYCOFF -

```
;PYCOFF - CALLED BY CFSSRV WHEN A CONNECTION IS BROKEN TO INSURE WE THROW ALL
;DISKS OFFLINE THAT ARE POTENTIALLY ACCESSED BY THE NODE
; T2/ CI NODE NUMBER
;       CALL PYCOFF             ;(T2/)
; RETURN +1
```

PYCOFF causes a node configuration check. All disks ever accessed by the node will be checked unless they are now front-end or don't-care disks.

7.1.4 PYCON -

```
;PYCON - CALLED BY CFSSRV WHEN A CONNECTION IS ESTABLISHED TO INSURE WE CHECK
;ALL DISKS THAT ARE POTENTIALLY ACCESSED BY THE NODE
; T2/ CI NODE NUMBER
; T3/ CPU SERIAL NUMBER
;       CALL PYCON              ;(T2,T3/)
; RETURN +1
```

PYCON causes a node configuration check. All disks ever accessed by the node will be checked.

7.2 Reading/Writing The PDB

The actual reading of the homeblocks takes place in stages. First, the primary homeblock is read and if there there were no errors with the read, the PDB is read. If there were errors, the secondary homeblock is read.

When a IORB is queued, you can specify pre and/or post processing routine(s) by placing the routine address(s) in the LH and RH, respectively, of offset IRBIVA of the IORB. The homeblock checking makes use of this facility, declaring the routines CHBSTR and CHBDON.

7.2.1 CHBSTR -

```
;CHBSTR - SETS IS.ERR IF THE DRIVE HAS GONE OFFLINE.  THIS WILL CAUSE THE
;HOMEBLOCK CHECKER TO TRY AGAIN LATER.  THIS ROUTINE WILL BE CALLED FROM
;STRTIO AS THE ENTRY ROUTINE BEFORE ACTUALLY SCHEDULING THE HOMEBLOCK IORB.
; P3/ UDB
; P4/ IORB
;       CALL CHBSTR             ;(P3,P4/)
; RETURN +1: ERROR BIT SET IN IORB
;        +2: READY TO TRANSFER
```

7.2.2 CHBDON -

```
;CHBDON - FIGURES OUT THE STATE OF HOMEBLOCK CHECKING AND INITIATES THE
;NEXT STEP.  CHBDON IS CALLED BY DONIRB AS THE EXIT ROUTINE FOR THE HOMEBLOCK
;IORB. THIS ALLOWS US TO PROCESS THE COMPLETED IORB BEFORE THE POLLER CAN SEE
;THE COMPLETED REQUEST.  THIS WILL NOT BE CALLED IF THE DRIVE IS OFFLINE SO
;IS.ERR IMPLIES REAL ERROR HERE.
; P1/ CDB
; P3/ UDB
; P4/ IORB
;       CALL CHBDON             ;(P1,P3,P4/)
; RETURN +1:
```

7.2.3 RDPDB -

```
;RDPDB - READ A PDB FROM DISK INTO THE UDB
; P3/ UDB
; P4/ IORB
;       CALL RDPDB              ;(P3,P4/)
; RETURN +1
```

7.2.4 WRTPDB -

```
;WRTPDB - WRITE UDB'S PDB TO THE DISK
; P1/ CDB
; P3/ UDB
;       CALL WRTPDB             ;(P1,P3/)
; RETURN +1
```

7.3 Checking/Updating The PDB

7.3.1 CHKPDB -

```
;CHKPDB - CHECK THE CURRENT STATE OF THE UDB'S PDB.  IF IT DIFFERS FROM
;REALITY, WE MUST FIX IT, WRITE IT BACK TO THE DISK, AND START ALL OVER.
; T1/ STATUS BITS FOR UDBST1
; P3/ UDB
;       CALL CHKPDB             ;(T1,P3/)
; RETURN +1: NEED TO WRITE PDB TO DISK
;        +2: ALL OK, NO RE-WRITE NECESSARY
```

7.3.2 UPDPDB -

```
;UPDPDB - UPDATE THE UDB'S PDB WITH CURRENT STATUS
; P3/ UDB
;       CALL UPDPDB              ;(P3/)
; RETURN +1: UDB'S PDB UPDATED
;        +2: NO CHANGES MADE
```

7.3.3 CLRPDB -

```
;CLRPDB - CLEAR THE UDB'S PDB LEAVING NOTHING BUT OUR OWN NODE'S STATUS AND THE
;DISK'S SERIAL NUMBER.
; P3/ UDB
;       CALL CLRPDB              ;(P3/)
; RETURN +1: UDB'S PDB CLEARED
;        +2: NO CHANGES MADE, DISK NOW OFFLINE
```