1.0 OVERVIEW

The server appears to the MSCP driver to be an intelligent disk controller much like the HSC50. The MSCP server interfaces to TOPS-20 as a device independant disk driver a la the DSKOP Jsys. It functions at a higher architectural level than PHYSIO but at a lower level than PAGEM/PAGUTL. That is the MSCP server has no knowledge of the file system format.

The only supported user of the MSCP server is the TOPS-20 MSCP driver (PHYMSC). The MSCP server implements only the subset of MSCP functions required for the operation of PHYMSC.

This document describes the structure of the MSCP server, it's major algorithms and data structures, the services it provides to the MSCP driver and the services of TOPS-20 that are utilized by the MSCP server.

2.0 REFERENCES

1. Mass Storage Control Protocol Version 1.2, 8-Apr-82 (Gardner)
2. TOPS-20 MSCP Server Functional Specification, 1.0 /date/ (Moser)
3. TOPS-20 MSCP Driver Functional Specification /TBS/ (Mclean)
4. TOPS-20 SCA Functional Specification /TBS/ (Dunn)
5. Systems Communications Architecture, 20-July-82 (Strecker)
6. TOPS-20 Coding Standard, 23-Mar-83 (Murphy)
7. PHYSIO.MAC TOPS-20 Physical IO module

3.0 GLOBAL CONCEPTS

The reader of this document is expected to be familiar with the overall concepts and interactions of MSCP disk servers and disk class drivers as described in [1]. The reader is also expected to be familiar with TOPS-20 implementation details as described in [2,3]. The following section describes global concepts, algorithms and data structures utilized by the MSCP server.

Figure 1 - MSCP server flow

In general the MSCP server performs 2 main functions. It maintains communications with MSCP drivers by means of SCA connections and it processes commands from those drivers.

3.1 Connections And Connection States

MSCP servers and drivers communicate by means of SCA connections. The MSCP server establishes connections passivly. It utilizes the services of SCA to "listen" for a connection. It listens for a connection from the name "T-20$DISK" and it gives the name "MSCP$DISK" in the listen call to SCA. Once a driver attempts to connect to a listner the server decides whether to accept or reject that connection.

A connection is accepted if the connector is identified as a KL in the confiduration data block obtained from SCA. Connections from drivers which are not KL's are not accepted.

Connections may be terminated for several reasons. They are:

1. Port error
2. Remote system dissapears
3. Driver requests connection termination
4. Server terminates connection for one of the following reasons
    1. A driver does not communicate with the server within the timeout interval
    2. A command does not complete (times out) within the timeout interval

The MSCP server maintains an internal connection status for each connection. The state of a connection is defined by the setting of 0 or more bits in the Server Connection Data Block (SCDB). The MSCP server recognizes that a connection may be in one of the following states:

1. Unused - No connection exists. This is indicated by the absence of an SCDB.
2. Listen - Waiting for a connect as an SCA listener.
3. Waiting for OK to send - A response to our listner has been accepted. This state indicates that a connection is being opened.
4. OK to send - The connection is fully open.
5. Attention enable - This indicates that the connector desires to recieve Attention messages.

6. Shutdown - This indicates that the connection has been terminated. This state has several possible substates that indicate the reason for connection termination. They are:

    1. Node offline - The other host has gone offline.

    2. Port broke connection - The connection was terminated by a port error.

    3. Forced shutdown - The connection was terminated because of an error detected by the MSCP server.

Connection state transitions are explained in detail later but the following table summarizes the major transitions between connection states.

ACTION	Previous State	New State	Comments
None	Unused	Unused	Initial connection state
SC.LIS	Unused	LISTEN	SCDB exists. Waiting for connect
Connection Accepted	LISTEN	WAIT OK	Connection accepted. Waiting for "OK to send". Need new listner
OK TO SEND	WAIT OK	OPEN	"Normal" state server and driver communicate
PORT ERROR	any	SHUTDOWN PORT ERROR	Connection closed. Messages discarded
DISCONNECT	any	SHUTDOWN	"
NODE OFFLINE	any	SHUTDOWN NODE OFFLINE	"
FATAL MSCP ERROR	any	SHUTDOWN	"

3.2 Command States

Commands processed by the MSCP server can be in one of several states. In addition to these states a command may be "Queued" or "Aborted" or both. "Queued" and "Aborted" differ from other states in that they can be used in conjunction with the other states. Besides "queued" and "aborted" the other states are mutually exclusive. The possible states are:

1. Queued - May be used in conjunction with other states. This indicates that the command is on the command queue.

2. Aborted - May be used in conjunction with other states. This indicates that the command has been aborted.

3. Command - The origional state of an MSCP command. It indicates that no processing has been done.

4. Wait to send end - All command processing is complete but the end packet must be sent.

5. Waiting to send buffer - The command is waiting to send a named buffer to the other host.

6. Waiting to request buffer - The command is waiting to request a named buffer from the other host.

7. Waiting for send of data - The command is waiting for named buffer send to complete.

8. Waiting for request of data - The command is waiting for named buffer request to complete.

9. IORB active - The command is waiting for PHYSIO to complete an IORB.

The following diagram shows the states and state transitions for all commands.

Figure 2 - State transitions

3.3 Command Processing And Queuing

3.3.1 Command Processing - All commands are SCA messages. When a message is recieved the opcode field of the command is compared to a list of supported opcodes and if found the appropriate processing routine is called. When a command is not found in the table it is treated as if it had an illegal opcode.

3.3.2 Command Queueing - Commands which cannot be completed immediatly are queued. Command completion may be prevented by resources being unavailable or by the nature of the command. All READ and WRITE commands are queued as are commands which fail to send end packets. When a command is queued it's state indicates what action must be performed for that command to complete. Each queued command has a queued command header which is used to store information about the command as well as links to other commands in the queue. The queued command header overlaps the SCA header area of the SCA packet containing the command. The format of that header is described under the section Global Data Structures.

3.3.2.1 Queue Structure - Command queues are implemented as a doubly linked list of commands. The SCDB contains pointers to the head and tail of this list. A zero indicates that there are no commands queued. The queued command header contains the forward and backward pointers needed to implement the list.

3.3.2.2 Queueing Algorithm - A single subroutine is responsible for adding commands to a queue. It is concerned simply with list management and has no knowledge of command format beyond the structures required for list maintenance. The Command queueing subroutine relies on the status bit QUEUED to indicate that a command resides on a queue. This bit allows calling the command queueing routine whenever a command needs to be queued (regardless of whether it is already queued). The queueing subroutine simply returns sucess it it is called with a previously queued command.

Command queueing is acomplished by the following steps:

1. Point the tail of the list to the command being queued

2. Point the command being queued at the (old) tail

3. Make the tail pointer in the SCDB point at the command being queued

4. Make the command's return address in the queued command header is set to the command dequeueing routine and the SCA buffer return address is saved in the queued commands header

3.3.2.3 Dequeueing - All commands are returned when command processing is complete. This returning is acomplished by calling the subroutine whose address is stored in the queued command header word, .QCRTN, This word initially contains the buffer return address which is supplied by SCA when a command is recieved. When a command is queued this buffer return address is saved in word .QCRT2 and .QCRTN contains the address of the command dequeueing routine.

Command dequeueing is acomplished by the follwoing steps:

1. Point the previous command at the next command in the list
2. Point the next command at the previous
3. Release all command resources
4. Call the SCA buffer return routine, address in .QCRT2

3.4 Timeouts

3.4.1 Connection Timeout - The MSCP server checks for connection timeouts as required by [1]. This is probably a superfluous function since SCA implements it's own timeout mechanism. If a connection has not communicated (sent a message) within it's timeout interval then that connection is closed. The connection state becomes "Shutdown" and "Forced shutdown".

3.4.2 Command Timeout - The MSCP server times out commands. This is never necessary in theory but it is done to identify any unexpected problems which may cause a command to never complete. A command times out if it is retried more than a defined number of times. This value is determined by the parameter MSRTMX which is currently defined as 15. When a command times out a BUGCHK, SRVCTO, is issued and the connection is closed. The connection state becomes "Shutdown" and "Forced shutdown".

3.5 Register Conventions

The MSCP server uses the following register conventions.

3.5.1 Internal Register Usage -

1. Q1 - Address of command being processed
2. Q2 - Address of SCA buffer to be sent to the remote system. This will be an end packet or an attention message.
3. Q3 - Address of SCDB for the connection

3.5.2 PHYSIO Register Conventions - The following registers are used when calling PHYSIO subroutines.

1. P1 - CDB address
2. P2 - KDB address
3. P3 - UDB address
4. P4 - IORB address (unused by MSCP server)

3.6 Data Structures

3.6.1 MSCP SERVER DATA -

```
!=====================================!
!               STATUS                ! SVSTSW
!-------------------------------------!
!            LISTNER INDEX            ! SVSLSX
!-------------------------------------!
!           FIRST FREE IORB           ! SVIRBH
!-------------------------------------!
!         FIRST FREE IO PAGE          ! SVIOPH
!-------------------------------------!
!           BROADCAST COUNT           ! SVBDKN
!-------------------------------------!
!        LAST COMMAND EXECUTED        ! SVLCMD
!-------------------------------------!
!                                     ! SVIORB
\                IORBS                \
!                                     !
!-------------------------------------!
!                                     ! SCDBTB
\              SCDB TABLE             \
!                                     !
!-------------------------------------!
!                                     ! SVCMDC
\          COUNT OF COMMANDS          \
!                                     !
!=====================================!
```

Data is displayed as contiguous but this is not a necessary condition. Each data section i.e. SVCMDC is contiguous but may not be contiguous with other data. All data is RS type storage unless indicated.

```
SVSTSW - 1B0 SVSINF Initialization flag. Set if initialized
         1B1 SVSILB Flag which indicates that the "Can't get listener" BUGINF
                    has been issued.
```

SVSLSX - Settable connect ID bits, index into SCDBTB, of listner. -1 if none.

SVBDKN - Number of disks which need broadcasts of AVAILABLE ATTENTION MESSAGES

SVLCMD - Address of last command routine called (for debugging)

SVIORB - IORBs (long form) used by the MSCP server

SCDBTB - Address of SCDB for each connection. 0 if none. Index to this table matches settable ID in SCA Connect ID. Alloctaed from extended resident code.

SVCMDC - Count of commands. Parrellel to command dispatch table. (for debugging - possibly SYSDPY). Alloctaed from extended resident code.

3.6.2 QUEUED COMMAND WITH HEADER -

```
!=======================================!
!            RETURN ADDRESS             !  .QCRTN
!---------------------------------------!
!             NEXT COMMAND              !  .QCNXT
!---------------------------------------!
!           PREVIOUS COMMAND            !  .QCLST
!---------------------------------------!
!        VIRTUAL IO PAGE ADDRESS        !  .QCVPA
!---------------------------------------!
!!Q!A!STATE!END LEN.!      RETRY        !  .QCSTS
!---------------------------------------!
!             IORB ADDRESS              !  .QCIOR
!---------------------------------------!
!              BUFFER NAME              !  .QCDBD
!---------------------------------------!
!        QUEUED RETURN ADDRESS          !  .QCRT2
!=======================================!
\                                       \ REMAINDER OF SCA HEADER AREA
!=======================================!
\               COMMAND                 \  .MHUDA
!=======================================!
```

Queued commands and headers are SCA message buffers. The header overlays the SCA header area.

```
.QCRTN - Buffer return address (provided by SCA for nonqueued commands or
         address of Dequeue routine for queued commands)

.QCVPA - Virtual address of locked page used for named buffer transfer
         (IO command only)

.QCSTS - 1B0 - Command is Queued
         1B1 - Command is Aborted
         STATE - Command State
         ENDLEN - Length of end packet to send (state "Wait to send end" only)
         RETRY - Command retry count (timeout counter)

.QCDBD - Buffer name used to do named buffer transfers (IO command only)

.QCRT2 - Address to return buffer if .QCRTN contains Dequeue address
         (moved here from .QCRTN at command queueing time)
```

3.6.3 SERVER CONNECTION DATA BLOCK (SCDB) -

```
!=====================================!
!               STATUS                ! .SVCIS
!-------------------------------------!
!             CONNECT ID              ! .SVCID
!-------------------------------------!
!            TIMEOUT TIME             ! .SVTMO
!-------------------------------------!
!            TIMEOUT VALUE            ! .SVTMV
!-------------------------------------!
!        HEAD OF COMMAND QUEUE        ! .SVCMD
!-------------------------------------!
!        TAIL OF COMMAND QUEUE        ! .SVCME
!-------------------------------------!
!          CONNECT ID INDEX           ! .SVCIX
!=====================================!
```

The SCDB is allocated from extended resident free space when a listner is established.

```
.SVCIS - Connection status
         1B0 - OK to send
         1B1 - Shutdown
         1B2 - Waiting for OK to send
         1B3 - Node offline
         1B4 - Port broke connection (port error)
         1B5 - Forced shutdown - Server initiated shutdown
         1B6 - Driver wants to recieve ATTENTION messages
```

.SVCIX - Settable bits of connect ID. Index into SCDB table.

The MSCP server also uses word IRBLEN of the long form IORBS for it's own use. This field contains the address of the queued command which "owns" the IORB and the Settable connect ID bits for the connection. These bits are used to find the SCDB at IORB done time.

4.0 ALGORITHM DESCRIPTIONS

4.1 Initilization

The MSCP server is an SCA SYSAP, as such it is initialized at system startup by SCA calling a routine whose address is in the SCA SYSAP initialization table. This rotuine does the following:

1. Links all long IORBs used by the MSCP server into a linked list where the first word of each IORB points at the next IORB. The list is terminated by a zero link. The head of the list is stored in SVIRBH.

2. Aquires a linked list of locked pages for IO requests using system routine PGRSKD.

3. Marks that no SCA connection listner exists.

4. Tries to aquire an SCA listner. See description of "LISTEN" for details.

5. Calls the periodic check routine.

4.1.1 Possible Errors - Initialization may fail if no pages are available for IO. If less than the desired number of pages are aquired a BUGINF, SRVNOP, is issued.

4.2 Periodic Check

The MSCP server performs a periodic check which is invoked through the normal CLK2 scheduler cycle checks. This CHECK does the following:

1. Set the time for the next check to be SRVCHI in the future. Currently the value is 10 seconds.

2. Check to see that all active connections have sent a message within the connectors timeout interval. If the connector has not sent a message the connection is closed. The connection status becomes "shutdown" and "forced shutdown".

3. Retry all queued commands. For details see the description of CREDIT AVAILABLE since it uses this same routine.

4. If no listner exists attempt to get one.

5. Send available messages to nodes which may need to know the status of new units. This function is described under SMON INTERFACE.

4.3 SCA Interface

SCA calls the MSCP server at the address specified in the "listen" call. This address dispatches through a table to various service routines. Some SCA callbacks are usused and dispatch to the monitor routine R which simply returns. The functions which have service routines are:

1. Function 1 - Message recieved

2. Function 2 - Port broke connection

3. Function 3 - Response to listen

4. Function 7 - Little credit remains (unused a label exists for debugging)

5. Function 11 - OK to send data

6. Function 12 - Credit available

7. Function 13 - Node going offline

8. Function 14 - Named buffer transfer complete

4.3.1 Function 1 - Message Recieved - Messages are MSCP commands. When a message is recieved the following actions are performed:

1. The queued command header area of the message is zeroed. (This area is coincident with the SCA header area)

2. The address to return the message buffer is stored in the command header

3. The current time is saved as the time of the last message recieved.

4. The commands OPCODE is compared to the list of supported command OPCODEs

5. If a matching OPCODE is found dispatch to the corresponding command processing subroutine.

4.3.1.1 Errors - If the connection is not in a legal state for recieving a message (shutdown or not "OK to send") the server BUGHLTs with SRVICS (Illegal connect state).

If no matching OPCODE is found in the table of legal OPCODEs then an invalid command end message is sent to the MSCP driver.

4.3.2 Function 2 - Port Broke Connection (SHUTDOWN) - Port broke connection is simply one case of shutting down a connection. It differs from the other cases only in that the status bit MC.PBC is set in the SCDB.

In general shutting down a connection involves the following steps:

1. Set the generic shutdown bit, MC.SHT, and any special shutdown bits in the SCDB.

2. A BUGCHK SRVFSN is issued.

3. The connection is aborted using SCA call SC.ABT

4. All disk units are marked "offline" to the connector by turning off the appropriate bit in the UDB

5. All queued commands (except those awating IORB completion) are released.

4.3.3 Function 3 - Response To Listen - This callback indicates that some connector wants to establish a connection with us. This rotine checks to insure that the connector is identified as a KL in the SCA configuration data. If it is not the connection is rejected. If it is a KL the connection is accepted and the state of the connection becomes MC.WOK (Waiting for OK to send).

4.3.3.1 Errors - If the settable ID bits in the connect ID do not match the number stored in SVSLSX (the number of the known listner) then the server BUGHLTs with MSSLNM (Listner no match)

4.3.4 Function 7 - Little Credit Remains - This routine simply returns immediatly.

4.3.5 Function 11 - OK To Send Data - This SCA callback indicates that a connection is fully open and ready to send and recieve messages. The conection state is set to MC.OKS (ok to send) and the timeout value is set to 60 seconds as required by [1]. The current time is used as the last time a message was recieved from the connector.

4.3.6 Function 12 - Credit Available - This routine retries all queued commands for a connection. If a command has been retried more than MSRTMX (the timeout value currently 15) times the server assumes that the command will never complete and it closes the connection (see description of Port broke connection). The following table describes the retry routine based on command state.

STATE	ACTION
Command	Treat like new message recieved
Waiting to send end	Send end packet
Wait for send data	No action

Wait for recv data	No action
Wait for IORB done	No action
Wait to send data	Send data
Wait to request data	Request data
Any other state	BUGHLT SVILST

4.3.6.1 Errors - A BUGCHK, SRVCTO (Command TimeOut) is issued if a command times out and the connection is closed.

If an illegal command state is found the system BUGHLTs with SVILST (ILlegal STate).

4.3.7 Function 13 - Node Going Offline - Node going offline is simply one case of shutting down a connection. It differs from the other cases only in that the status bit MC.NOF is set in the SCDB.

4.3.8 Function 14 - Named Buffer Transfer Complete - This function is used to notify the MSCP server that a transfer of data has completed. These data transfers are used to obtain and send the object of READ and WRITE commands, namely pages of data. The service routine searches the queued commands for a command whose buffer name matches the one provided by SCA. When the matching command is found it's state is checked against all legal states. The following table summarizes the action taken based on state.

STATE	ACTION
Aborted	Send aborted end packet
Waiting for send data	Send end packet (new state "wait to send end")
Waiting for recv data	Give IORB to PHYSIO (new state "wait for IORB")
Any other state	BUGHLT SVILST

4.3.8.1 Errors - If no command is found which has a matching buffer name the system BUGHLTs with SRVDNQ (buffer Done command Not Queued).

If an illegal command state is found the system BUGHLTs with SVILST (ILlegal STate).

4.4 MSCP Commands

All MSCP commands are SCA messages. All commands except READ and WRITE are handled in a similar manner; the command specifies an action to be performed and the driver which issued the command is returned an "end packet" which is also an SCA message. The commands which are handled in this manner are:

1. Set Controller Characteristics
2. Abort
3. Get Command Status
4. Get Unit Status
5. Available
6. Online

All of these commands fail only if the end packet buffer cannot be obtained from SCA or if the send of the end packet fails. If the buffer is unavailable the command is queued and it's state remains the origional "Command" state.

4.4.0.1 End Packet Transmission - If the end packet cannot be sent the command is queued in state "wait to send end". The image of the end packet will be BLTed to the message area of the queued command and the length of the end packet saved in the queued command header. When credit becomes available this end packet image is used to build the end packet which is resent at this time.

The following describes any features of the implementation which differ from the description in [1] or are specific to TOPS-20.

4.4.1 Set Controller Characteristics - This command functions as described in [1]. It sets the bit in the connection status word to enable ATTENTION messages if the driver requests it. It also sets the driver timeout value and returns the server timeout value.

4.4.2 Abort - This function aborts the command specified. If the command is queued and not waiting for an IO operation to complete (IO states are "Wait IORB", "Wait for send data" and "Wait for recieve data") then the server simply sends an aborted command end packet. If the command is waiting for IO then it cannot be aborted until the IO completes. The command is flagged as aborted in the queued command header. This flag is checked during IO completion processing and the aborted end packet is sent then.

4.4.3 Get Command Status - This returns a commands status if the command is found (queued). A commands status is determined by the following formula:

STATUS = (MAX TIMEOUT VALUE + 1) - CURRENT VALUE

If command is waiting for IO then STATUS = STATUS * 2

4.4.4 Available - This command marks a unit MSCP available. The states AVAILABLE and ONLINE are determined by the setting of a bit in the UDBCHR word of the disk units UDB. There is 1 bit for each driver (CI host) in this word. Collectivly these bits are defined as UC.OLB (OnLine Bits). The bit number for a driver is (35 - MAXDRIVERS) + DRIVERNUMBER. DRIVERNUMBER is the index into the SCDB table and is stored as the settable bits in the connect ID. The AVAILABLE command clears the bit.

4.4.5 Online - The ONLINE command sets the bit describes inder the AVAILABEL command.

4.5 IO Commands - READ And WRITE

The IO commands, READ and WRITE, are handled somewhat differently than other MSCP commands since they require more resource management and do not normally complete in a single step. The 2 commands differ in the order of execution of their steps as shown in figures 3 and 4 but the steps themselves are common. Both commands must:

1. Find the correct disk unit for IO operations.
2. Aquire a physical memory page for data transfer
3. Build a long form IORB to send to PHYSIO to initiate the actual data trantfer from/to memory
4. Initiate the data transfer to/from memory form/to disk using PHYSIO
5. Initiate a named buffer transfer
6. Send a command end packet to the driver

Figure 3 - READ Command

Figure 4 - WRITE command

4.6 PHYSIO Interface

The MSCP server uses PHYSIO as a resource. It uses various Global PHYSIO subroutines in the same manner as the rest of TOPS-20 (for example "find a unit", "check legality of a unit", "step to the next unit" ...). It also calls the subroutine PHYSIO to schedule IORBs for disk transfers. Drivers which call PHYSIO can be notified at IORB start time and IORB completion time by specifying subroutine addresses in word IRBIVA of a long form IORB.

The MSCP server is called only at IORB completion time. It checks to see that the command has not been aborted. If it has been it sends the aborted command end packet and releases the command and it's resources.

If the command has not been aborted the MSCP server proceeds with processing the command as shown in figures 3 and 4.

4.7 SMON / TMON / SETSPD Interface - AVAILABLE ATTENTION MESSAGE

The MSCP server requires a small user interface so that a subset of massbus disks may be selected for the MSCP server's use. This interface is provided by the SMON and TMON jsyses and through the SETSPD commands ALLOW and RESTRICT.

A description of the SETSPD commands and JSYS interface may be found in [TBS]. The following describes the implementation of these commands and the handling of AVAILABLE ATTENTION MESSAGES.

4.7.1 RESTRICT / ALLOW - The SETSPD RESTRICT command causes a disk to stop being serviced by the MSCP server. The ALLOW command has the oppsoite effect. The MSCP server determines whether a disk is RESTRICTed or ALLOWED through the use of a bit, US.CIA, in the UDBSTS word of the disk units UDB. US.CIA (CI Available) is cleared when a drive is RESTRICTed and set when the drive is ALLOWed. When a drive is allowed the MSCP server must notify the MSCP drivers that have open connections to the server that there is a new disk online. This must also happen when an ALLOWed drive comes online to PHYSIO.

4.7.2 AVAILABLE ATTENTION MESSAGE - When The server must inform a driver that a new disk is online (either because the drive was ALLOWed or because a previously ALLOWed comes online to TOPS-20) it sends all drivers with connections an AVAILABLE ATTENTION MESSAGE. Each drive that must have an attention message sent for it has the bit US.BDK (Broadcast) set in the UDBSTS word of the UDB. The MSCP server maintains a count of the number of drives which must have attention messages sent.

The MSCP server sends this attention message to all drivers who have enabled attention messages and who have not issued an ONLINE command for that unit. When all drivers are successfully sent the attention message the bit, US.BDK, is cleared and the count of needed attention messages is decremented.

If the server fails to send any driver an attention message it leaves the bit, US.BDK, set in the UDB and does not decrement the count. If the count is non-zero during a periodic check the UDBs are scanned and all those having US.BDK set have attention messages sent regarding them. A driver may recieve multiple attention messages for the same unit (allowed by [1]) if the following occurs:

1. A send of an attention message succeeds to this driver and fails for some other driver.

2. The driver who recieved the attention message did not ONLINE the drive before the next periodic check