" The next IT " - Ray Johnson - HP CUPERTINO

1918 MPE RELEASE

MAJOR ENHANCEMENTS:

* MPE USER LOGGING
* CONSOLE ENHANCEMENTS/DISTRIBUTED CONSOLE
* SPOOLER ENHANCEMENTS
* STORE/RESTORE TAPE LABELS
* VIRTUAL MEMORY INCREASE
* MPE RESEGMENTATION

MPE USER LOGGING

* ALLOWS USER TO INTERFACE LOGGING TO ANY SUBSYSTEM USING NEW LOGGING INTRINSICS (LOGGING AND RECOVERY ROUTINES HAVE BEEN ADDED TO IMAGE/3000)

* LOG RECORDS MAY BE WRITTEN TO A DISC FILE OR TO A TAPE

MPE USER LOGGING

USER COMMANDS

NOTE: LG CAPABILITY REQUIRED FOR ALL BUT :SHOWLOGSTATUS

* :ALTLOG — ALTERS THE ATTRIBUTES OF AN EXISTING LOGGING IDENTIFIER

```
:ALTLOG LOGID[;LOG=LOGFILE{,DISC}][;PASS=PASSWORD]
                          {,TAPE}
```

* :GETLOG — ESTABLISHES A LOGGING IDENTIFIER ON THE SYSTEM

```
:GETLOG LOGID ;LOG=LOGFILE{,DISC} [;PASS=PASSWORD]
                          {,TAPE}
```

* :LISTLOG — LISTS LOGGING IDENTIFIERS

```
:LISTLOG [LOGID[;PASS]]
```

* :RELLOG — REMOVES A LOGGING IDENTIFIER FROM THE SYSTEM

```
:RELLOG LOGID
```

* :SHOWLOGSTATUS — DISPLAYS STATUS INFORMATION ABOUT CURRENTLY RUNNING LOG PROCESSES

```
:SHOWLOGSTATUS [LOGID]
```

MPE USER LOGGING

OPERATOR COMMANDS

* :LOG STARTS, RESTARTS, OR STOPS USER LOGGING

```
                  {START}
:LOG LOGID,       {RESTART}
                  {STOP}
```

MPE USER LOGGING

USING THE FACILITY

1. MAKE CERTAIN USER LOGGING CAPABILITY HAS BEEN ADDED TO YOUR CAPABILITY SET (LG)

2. CHECK TO SEE THAT THE REQUIRED NUMBER OF LOGGING PROCESES AND USERS PER PROCESS HAVE BEEN ALLOWED (NEW QUESTIONS IN SYSDUMP DIALOG)

ESTABLISHING A LOGGING FILE:

IF LOGGING TO DISC, YOU MUST BUILD A LOGGING FILE:

```
:BUILD NEWDATA;CODE=LOG
```

ESTABLISHING A LOGGING IDENTIFIER:

USED TO IDENTIFY THE LOGGING PROCESS THAT WRITES THE TRANSACTIONS RECORDS TO THE LOGGING FILE

TO ESTABLISH THE LOGGING IDENTIFIER FINANCE AND ASSOCIATE IT WITH A DISC LOGGING FILE NAMED NEWDATA, ENTER:

```
:GETLOG FINANCE;LOG=NEWDATA,DISC
```

MPE USER LOGGING

USING THE FACILITY

ACTIVATING USER LOGGING:

USER LOGGING IS TURNED ON WITH THE NEW CONSOLE OPERATOR COMMAND :LOG

:LOG FINANCE,START

START:	INITIATES LOGGING PROCESS.
RESTART:	RESTARTS LOGGING PROCESS
STOP:	TERMINATES LOGGING PROCESS

MODIFYING A LOGGING IDENTIFIER:

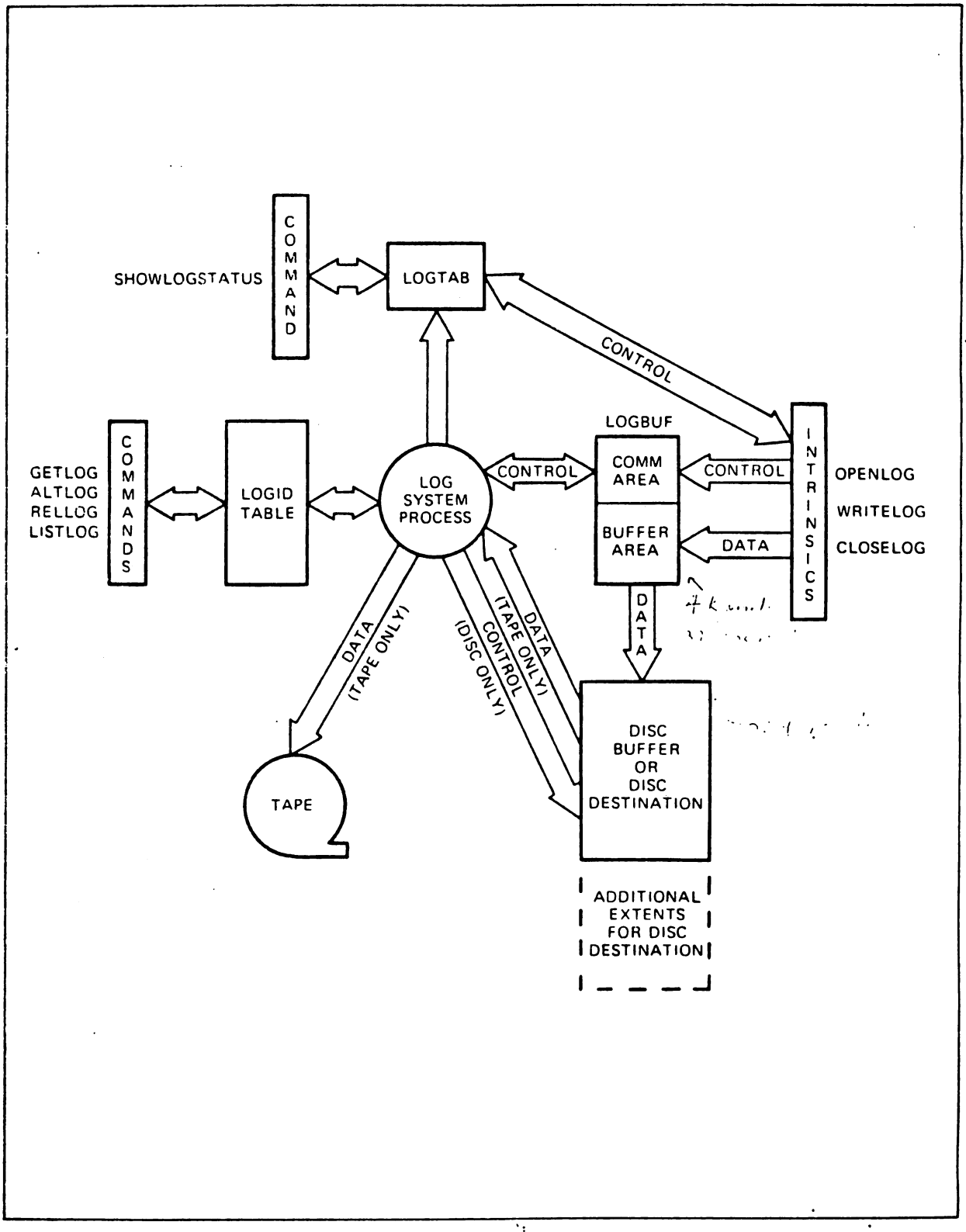
THE :ALTLOG COMMAND IS USED TO ALTER THE LOGGING FILE OR PASSWORD ASSIGNED TO AN EXISTING LOGGING IDENTIFIER

:ALTLOG FINANCE;LOG=FILEM,TAPE

WRITING TO THE LOG FILE:

LOGGING IS DONE PROGRAMMATICALLY BY MEANS OF THREE INTRINSICS:

OPENLOG, WRITELOG, AND CLOSELOG



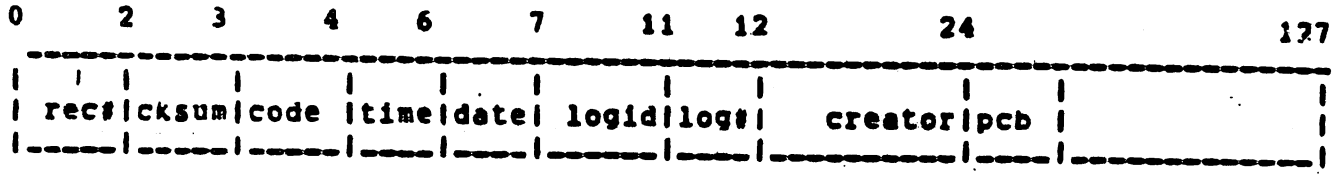
User Logging Facility

USER LOGGING RECORD FORMATS

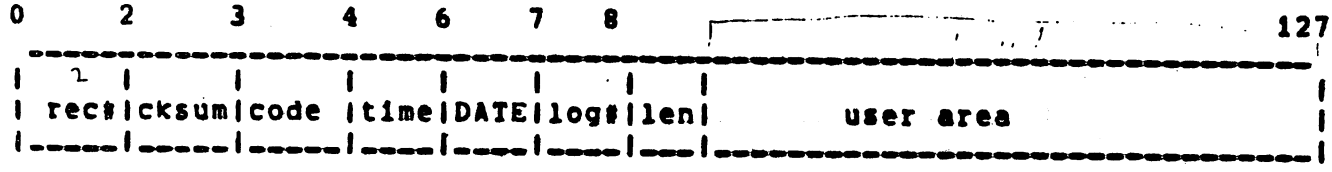
D. LOGGING RECORD FORMAT

RECORD SIZE = 128 WORDS
USER AREA = 119 WORDS

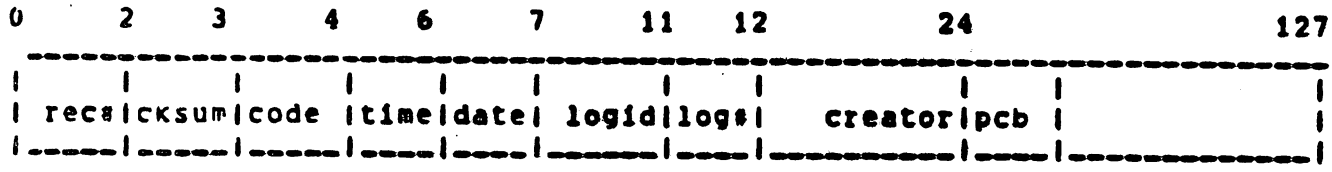
2. LOG RECORD AT OPENLOG



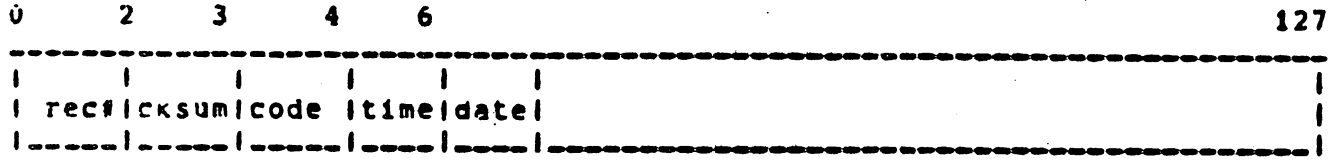
USER OR SUBSYSTEM/CONTINUATION LOG RECORD



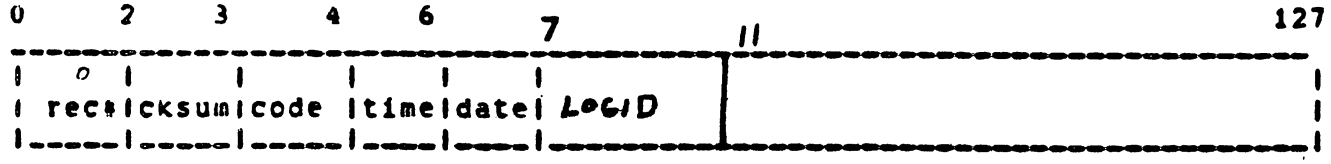
LOG RECORD AT CLOSELOG



CRASH MARK

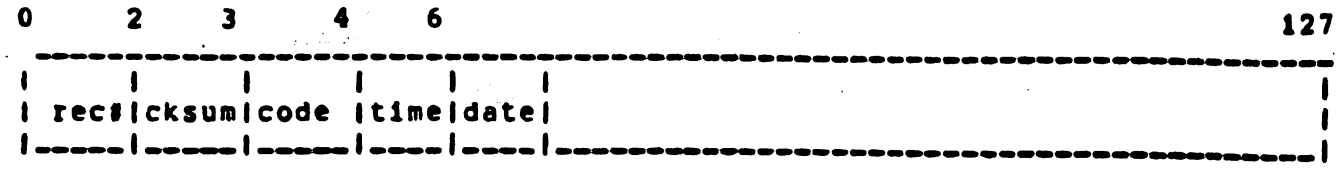


1. HEADER RECORD START/RESTART

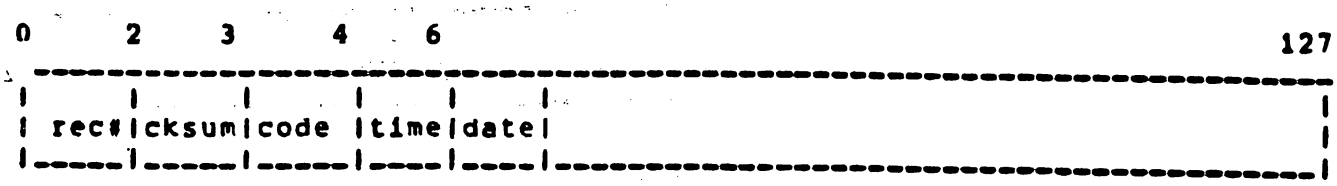


USER LOGGING RECORD FORMATS

TRAILER RECORD



NULL RECORD



CODE DEFINITION

CODE=1	OPEN LOG
2	USER/SUBSYSTEM RECORD
3	CLOSE LOG
4	HEADER
5	TRAILER
6	RESTART
7	CONTINUATION OF USER OR SUBSYSTEM RECORD
9	CRASH MARKER
SPACE	NULL RECORD (8442 or 702040)

NOTE:

1. The checksum algorithm uses the exclusive or function against a base of negative one. All records are included except the checksum.

2. Null record is used for filler.

3. The code word of the logging record can contain a subsystem code defined by the user in the first half of the word (0:8). User logging allows privileged users to pass this code in the index parameter of the Open Log intrinsic.

USER LOGGING

POWER FAIL RECOVERY

IF POWER FAIL OCCURS WHILE LOGGING TO MAG TAPE, THE FOLLOWING RECOVERY PROCEDURE MUST BE USED TO RECOVER AND CONTINUE LOGGING.

1. AFTER POWER IS BACK ON AND THE SYSTEM IS RUNNING THE CONSOLE WILL RECEIVE A NOT READY MESSAGE FOR THE TAPE

2. PRESS THE FOLLOWING BUTTONS ON THE TAPE DRIVE IN THE GIVEN ORDER:

   A. LOAD
   B. RESET
   C. ONLINE

3. AFTER THE ONLINE BUTTON IS PRESSED THE TAPE WILL MOVE BACK TO THE BEGINNING OF DATA AND MOVE FORWARD TO RECOVER THE LOG RECORDS. THEN THE SYSTEM WILL CONTINUE LOGGING.

USER LOGGING

SYSTEM FAILURE RECOVERY

* MUST WARMSTART SYSTEM

* ANY DATA IN LOGGING BUFFER AREA AT TIME OF FAILURE WILL BE LOST

* FOR DISC LOG FILES, WARMSTART RECOVERY PROCEDURE READS DISC FILE UNTIL LAST BLOCK OF DATA IS READ;

  PROCEDURE WRITES OUT CRASH RECORD AND CLOSES FILE

* FOR TAPE LOG FILES, PROCEDURE OPENS DISC BUFFER AND TAPE FILE AND COMPARES LAST RECORDS;

  PROCEDURE WRITES OUT DATA IN DISC BUFFER THAT IS NOT ON TAPE TO THE TAPE;

  PROCEDURE WRITES OUT CRASH RECORD AND CLOSES FILE

* TO RESTART LOGGING, :LOG logid, RESTART MUST BE ISSUED

MPE USER LOGGING

IMAGE: TRANSACTION LOGGING AND RECOVERY SYSTEM

IMAGE/3000 PROVIDES A LOGGING AND RECOVERY SYSTEM THAT CAN BE EXECUTED TO BRING DATA BASES BACK TO A SEMBLANCE OF THEIR STATE AT THE TIME OF A SYSTEM FAILURE.

* THE LOGGING SYSTEM PROVIDES A MECHANISM TO LOG DATA BASE TRANSACTIONS TO A LOGFILE ON TAPE OR DISC

* THE RECOVERY SYSTEM READS THIS LOGFILE TO RE-EXECUTE TRANSACTIONS AGAINST A DATA BASE BACKUP COPY IN THE EVENT OF A FAILURE

* THE LOGFILE MAY ALSO BE USED TO AUDIT MOD-IFICATIONS TO ITEMS IN THE DATA BASE

CONSOLE ENHANCEMENTS/DISTRIBUTED CONSOLE

* OPERATOR.SYS AUTOMATICALLY LOGGED ONTO SYSTEM
* CONSOLE COMMANDS ARE HANDLED BY THE COMMAND INTERPRETER
* THE CONSOLE CAN BE MOVED TO ANY TERMINAL
* THE CONSOLE OPERATOR (MASTER OPERATOR) CAN ASSIGN INDIVIDUAL USERS THE ABILITY TO EXECUTE SPECIFIC CONSOLE COMMANDS FROM THEIR TERMINALS
* USERS CAN BE ASSOCIATED WITH SPECIFIC DEVICES, ALLOWING THEM TO USE COMMANDS THAT CONTROL THE DEVICES
* USERS CAN BE GRANTED THE ABILITY TO USE JOB CONTROL COMMANDS ON THEIR OWN JOBS

OPERATOR.SYS

* USER OPERATOR.SYS AUTOMATICALLY LOGGED ON AT SYSTEM STARTUP BECAUSE MANY CONSOLE COMMANDS REQUIRE SESSION

* ALLOWS LOG ON UDC'S - STREAMS 10, VMOUNT ON,AUTO, ETC.

* AFTER INSTALLATION OF 1918 IT, USER "OPERATOR" MUST BE CREATED

* OPERATOR.SYS SHOULD BE GIVEN THE FOLLOWING CAPABILITIES:
  IA, BA, SF, ND, UV, OP,LG

OPERATOR.SYS

```
HP32002B.01.00
WHICH OPTION <WARMSTART/COOLSTART>? COOl
ANY CHANGES?

DATE (M/D/Y)?7/6/79
TIME (H:M)?9:57
FRI, JUL  6, 1979,  9:57 AM? (Y/N)Y
LOG FILE NUMBER 1868 ON
*WELCOME*
:HELLO OPERATOR.SYS;HIPRI
9:57/15/SP#6/SPOOLED OUT
9:57/#S1/16/LOGON FOR: OPERATOR.SYS,PUB ON LDEV #20
HP3000 / MPE III B.01.00.  FRI, JUL  6, 1979,  9:57 AM
*** MORRIS ***
STREAMS 10
HEADOFF 6
VMOUNT ON,AUTO
ALLOCATE EDITOR
ALLOCATE SPL
```

DISTRIBUTED CONSOLE COMMANDS

:CONSOLE

:ALLOW

:DISALLOW

:ASSOCIATE

:DISASSOCIATE

:JOBSECURITY

DISTRIBUTED CONSOLE COMMANDS

CONSOLE COMMAND		EFFECT
:CONSOLE	→	CONSOLE TRANSFERRED TO SPECIFIC TERMINAL
:ALLOW	→	SPECIFIC CONSOLE COMMANDS GIVEN TO SPECIFIC USERS
:JOBSECURITY	→	ALL USERS GIVEN JOB CONTROL COMMANDS FOR THEIR OWN JOBS

USER COMMAND		EFFECT
:ASSOCIATE	→	CONSOLE COMMANDS FOR SPECIFIC DEVICES GIVEN TO SPECIFIC USERS

DISTRIBUTED CONSOLE COMMANDS

:CONSOLE

* CHANGES THE SYSTEM CONSOLE FROM ITS CURRENT DEVICE TO ANOTHER JOB-ACCEPTING (NON-DS) TERMINAL

FORMAT:

```
:CONSOLE  LDN
```

COMMAND ENVIRONMENT:

* USE THE :CONSOLE COMMAND WITH EXTREME CAUTION *

EXAMPLE:

TO CHANGE THE SYSTEM CONSOLE TO LDEV 23, ENTER:

```
:CONSOLE 23
```

NOTE: COLD LOADING THE SYSTEM CAUSES THE SYSTEM TO COME UP WITH THE CONSOLE AS ORIGINALLY CONFIGURED

DISTRIBUTED CONSOLE COMMANDS

:ALLOW

* THROUGH THE :ALLOW COMMAND, THE CONSOLE OPERATOR HAS THE ABILITY TO DISTRIBUTE CONSOLE COMMAND CAPABILITIES AMONG SYSTEM USERS BY DIRECTLY ENTERING:

```
:ALLOW { @.@          }; COMMANDS= COMMAND 1 [,COMMAND 2,..COMMANDN]
       { USER.@       }
       { @.ACCT       }
       { USER.ACCT    }
```

EXAMPLE:

```
:ALLOW USER.STUDENT;COMMANDS = REPLY,ABORTIO
```

THIS ALLOWS USER.STUDENT TO EXECUTE :REPLY AND :ABORTIO

* THE :ALLOW COMMAND CAN ALSO BE USED IN AN INDIRECT MODE. IN THE INDIRECT MODE, AN EDITOR FILE IS CREATED CONTAINING:

```
{ @.@          }; COMMANDS= COMMAND 1 [,COMMAND 2,..COMMANDN]
{ USER.@       }
{ @.ACCT       }
{ USER.ACCT    }
```

THE :ALLOW COMMAND IS THEN EXECUTED BY ENTERING:

```
:ALLOW FILE= FILENAME [;SHOW]
```

* THE :ALLOW COMMAND MAY BE USED IN SUBSYSTEM MODE AS FOLLOWS:

```
:ALLOW
>USER.STUDENT;COMMANDS = REPLY,ABORTIO
>EXIT
```

:NOTE: THE USER BEING ALLOWED MUST BE LOGGED ON, AND IS DISALLOWED AT LOGOFF.

****
* CAUTION *
****

THE :ALLOW COMMAND IS EXTREMELY POWERFUL. THE MASTER OPERATOR MUST BE CAREFUL IN ASSIGNING COMMANDS TO USERS.

FOR EXAMPLE, IF THE :CONSOLE COMMAND OR THE :ALLOW COMMAND IS ALLOWED TO A USER, THAT USER CAN "STEAL" THE CONSOLE.

NOTE: THE CAPABILITY TO "STEAL" THE CONSOLE HAS ALWAYS BEEN AVAILABLE--- PRIV. MODE DEBUG

DISTRIBUTED CONSOLE COMMANDS

:DISALLOW

* THE :DISALLOW COMMAND PROHIBITS CERTAIN USERS FROM USING THE NAMED COMMANDS

* LIKE THE :ALLOW COMMAND, THE :DISALLOW COMMAND MAY BE USED DIRECTLY, INDIRECTLY, OR IN SUB-SYSTEM MODE.

* A USER WHO HAS BEEN ALLOWED A COMMAND IS AUTOMATICALLY DISALLOWED AT LOGOFF

EXCEPTION

IF @.@ WAS SPECIFIED AS USER SUBSET IN :ALLOW COMMAND, THE LISTED COMMANDS WILL REMAIN AS SIMPLE USER COMMANDS UNTIL :DISALLOW @.@;------ IS ISSUED

DISTRIBUTED CONSOLE COMMANDS

:ASSOCIATE

* THE :ASSOCIATE COMMAND IS A USER COMMAND WHICH ALLOWS A USER TO BECOME THE OPERATOR FOR A SPECIFIC DEVICE CLASS

SYNTAX:

```
:ASSOCIATE DEVICECLASSNAME
```

FOR EXAMPLE, A SPECIFIC LINE PRINTER CAN NOW BE ASSOCIATED WITH A USER IN THE GROUP THAT ACCESSES IT. THE ASSOCIATED USER WILL GET THE OPERATOR MESSAGES FOR THE PRINTER AND WILL BE ABLE TO USE THE OPERATOR COMMANDS FOR THE DEVICE.

* THE SYSTEM MANAGER MUST FIRST CREATE A DEVICE-USER ASSOCIATION TABLE BY RUNNING THE UTILITY "ASOCTABL.PUB.SYS"

THE UTILITY WILL PROMPT THE SYSTEM MANAGER TO ENTER THE ASSOCIATION STATEMENTS:

DEVICECLASSNAME=USER.ACCT[,..., USER.ACCT]

UPON EXITING THE UTILITY, THE RESULTING TABLE WILL BE SAVED AS ASOCIATE.PUB.SYS

NOTE: AN EDITOR FILE MAY BE USED BY THE ASOCTABL UTILITY BY MEANS OF A FILE EQUATION

EXAMPLE: :FILE INPUT= FILENAME
:RUN ASOCTABL

NOTE: UNTIL 1936 A FILE EQUATION -:FILE INPUT=$STDIN- WILL BE NEEDED BEFORE ENTERING DATA INTO ASOCIATE.PUB.SYS FROM A TERMINAL.

:ASSOCIATE (CONT)

* ANY USERS LISTED IN ASOCIATE.PUB.SYS MAY NOW USE THE :ASSOCIATE COMMAND FOR THE APPROPRIATE DEVICE CLASS. ALL MESSAGES FOR THAT DEVICE CLASS WILL COME TO THE FIRST USER ISSUING THE :ASSOCIATE COMMAND, AND THE USER WILL BE THE ONLY USER ON THE SYSTEM ABLE TO ISSUE CONSOLE COMMANDS FOR THE DEVICE CLASS. (WITH THE EXCEPTION OF THE MASTER OPERATOR, OR ALLOW.SYS WHO WILL BE ASKED IF HE WANTS TO OVERRIDE THE ASSOCIATED USER)

NOTE: ONLY ONE USER MAY BE ASSOCIATED WITH A DEVICE AT A TIME. AT USER LOG-OFF, DEVICE CONTROL RETURNS TO THE CONSOLE.

DISTRIBUTED CONSOLE COMMANDS

:ASSOCIATE (CONT)

* THE OPERATOR COMMANDS WHICH ARE MADE AVAILABLE TO USERS THROUGH THE :ASSOCIATE COMMAND ARE:

ABORTIO
ACCEPT
ALTSPOOLFILE
DELETESPOOLFILE
DOWNLOAD
GIVE
TAKE
SUSPENDSPOOL
HEADOFF
REFUSE
REPLY
RESUMESPOOL
STARTSPOOL
STOPSPOOL
HEADON

:ASSOCIATE

EXAMPLE

```
:HELLO BOB.STAMPS

:FILE INPUT=$STDIN
:RUN ASOCTABL.PUB.SYS

>LP=BOB.STAMPS
>EXIT

+-F-I-L-E---I-N-F-O-P-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  FILE NAME IS ASOCIATE.PUB.SYS                  !
!  FOPTIONS: NEW,B,*FORMAL*,F,N,FEQ,T             !
!  AOPTIONS:  IN/OUT,SREC,NOLOCK,EXC,BUFFER       !
!  DEVICE TYPE: 0      DEVICE SUBTYPE: 4          !
!  LDEV: 2        DRT: 4         UNIT: 1          !
!  RECORD SIZE: 13     BLOCK SIZE: 117   (WORDS)  !
!  EXTENT SIZE: 15     MAX EXTENTS: 8             !
!  RECPTR: 7           RECLIMIT: 1023             !
!  LOGCOUNT: 258          PHYSCOUNT: 31           !
!  EOF AT: 256         LABEL ADDR: %002000000215  !
!  FILE CODE: 0      ID IS BOB       ULABELS: 0   !
!  PHYSICAL STATUS: 1000000000000001              !
!  ERROR NUMBER: 93    RESIDUE: 117      (WORDS)  !
!  BLOCK NUMBER: 31           NUMREC: 9           !
+-----------------------------------------------+
ASOCIATE.PUB.SYS- SECURITY VIOLATION  (FSERR 93
```

ASSOCIATE - EXAMPLE

```
:HELLO MANAGER.SYS

:FILE INPUT=$STDIN
:RUN ASOCTABL

>LP=BOB.STAMPS
>EXIT

END OF PROGRAM
:LISTF ASOCIATE,2
ACCOUNT=  SYS          GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD----------  ----SPACE----
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

ASOCIATE           13W  FB         256       1023   9       30  2  8


:HELLO BOB.STAMPS

:ASSOCIATE LP
:ASSOCIATE TERM

ASSOCIATING THIS DEVICE REQUIRES SYSTEM MANAGER'S PERMISSION. (CIERR 3059
```

:ASSOCIATE - EXAMPLE

```
:SHOWDEV
LDEV   AVAIL      OWNERSHIP          VOLID          ASSOCIATION

  1    DISC       10 FILES
  2    DISC       2 FILES
  3    DISC       0 FILES
  4    DISC       0 FILES
  6    SPOOLED    SPOOLER OUT                       #S13-LP
  7    AVAIL
  8    AVAIL
  9    AVAIL
 10  A SPOOLED    SPOOLER IN
 15    AVAIL
 16    AVAIL
 20  A UNAVAIL    #S2: 2 FILES
 21  A AVAIL
 22  A AVAIL
 23  A AVAIL
 24  A AVAIL
 25  A AVAIL
 26  A UNAVAIL    SYS
 27  A AVAIL
 28  A AVAIL
 29  A AVAIL
 30  A AVAIL
 31  A AVAIL
 32  A AVAIL
 33  A AVAIL
 34  A AVAIL
 35  A AVAIL
 36  A AVAIL
 37  A AVAIL
 38  A AVAIL
 39  A UNAVAIL    #S14: 2 FILES
 40  A AVAIL
 41  A AVAIL
 42  A UNAVAIL    #S6: 2 FILES
 43  A UNAVAIL    #S7: 2 FILES
 44  A UNAVAIL    #S13: 2 FILES
 45  A AVAIL
 46  A AVAIL
 47  A AVAIL
```

DISTRIBUTED CONSOLE COMMANDS

:DISASSOCIATE

* REMOVES A DEVICE CLASS FROM THE CONTROL OF A USER

SYNTAX

:DISASSOCIATE DEVICECLASSNAME

* WHEN A USER DISASSOCIATES A DEVICE, THE OPERATOR IS NOTIFIED

NOTE: THE MASTER OPERATOR MAY REMOVE CONTROL FROM THE USER BY ISSUING THE DISASSOCIATE COMMAND HIMSELF.

AT LOG OFF, THE USER IS AUTOMATICALLY DISASSOCIATED.

DISTRIBUTED CONSOLE COMMANDS

:JOBSECURITY

* CONTROLS CAPABILITY OF USERS TO USE JOB-RELATED CONSOLE COMMANDS ON THEIR OWN JOBS

* SYNTAX:

:JOBSECURITY {HIGH / LOW}

- WHEN SET LOW, ALL USERS MAY USE THE COMMANDS ABORTJOB, ALTJOB, BREAKJOB, AND RESUMEJOB ON THEIR OWN JOBS

  ALSO, ACCOUNT MANAGERS WILL BE ABLE TO USE THESE COMMANDS ON ANY JOB IN THEIR ACCOUNT

- WHEN SET HIGH, THESE COMMANDS MAY ONLY BE EXECUTED FROM THE CONSOLE (UNLESS INDIVIDUALLY ALLOWED)

- DEFAULT AT SYSTEM GENERATION TIME IS HIGH, UNLESS A WARMSTART WAS USED, IN WHICH CASE JOBSECURITY WILL REMAIN AS IT WAS PRIOR TO THE WARMSTART.

COMMAND INTERPRETER CONSOLE COMMANDS

* THE FOLLOWING COMMANDS ARE STILL "CNTL A" COMMANDS:

  =LOGOFF
  =LOGON
  =SHUTDOWN
  =DSLINE
  =MPLINE
  =MRJE

* ALL OTHER CONSOLE COMMANDS ARE NOW HANDLED BY THE COMMAND INTERPRETER. ACCORDINGLY, BETTER ERROR MESSAGES AND THE HELP SUBSYSTEM WILL BE AVAILABLE FOR CONSOLE COMMANDS.

REDUCED CODE IN PROGEN – BANK Ø

* =MOUNT AND =DISMOUNT HAVE BEEN REPLACED WITH :LMOUNT AND :LDISMOUNT BECAUSE OF EXISTING :MOUNT AND :DISMOUNT USER COMMANDS.

* :DSTAT AND :VSUSER ARE NOW USER COMMANDS ONLY.

NOTE: REPLY, ABORTIO, AND ABORTJOB MAY BE ISSUED AS COMMAND INTERPRETER OR "CNTL A" COMMANDS. THIS ALLOWS, FOR EXAMPLE, THE OPERATOR TO ISSUE A =REPLY WITHOUT HAVING TO BREAK FROM AN EXECUTING PROGRAM TO ISSUE A :REPLY. (BREAK DURING STORE AT CONSOLE ABORTS THE STORE)

SPOOLER ENHANCEMENTS

* DEVICE CLASS CAPABILITY

* NEW CONSOLE COMMANDS

SPOOLING ENHANCEMENTS

NEW CONSOLE COMMANDS

* =SPOOL, =DELETE, AND =ALTFILE HAVE BEEN REPLACED BY A NUMBER OF NEW, LESS CONFUSING CONSOLE COMMANDS

:STARTSPOOL

:STOPSPOOL

:SUSPENDSPOOL

:RESUMESPOOL

:ALTSPOOLFILE

:DELETESPOOLFILE

:STARTSPOOL

* CAUSES THE SPOOLER TO OWN A SPECIFIED DEVICE
* THE SPOOLER DETERMINES WHETHER THE DEVICE IS AN INPUT OR OUTPUT DEVICE

SYNTAX

```
:STARTSPOOL {LDEV       }
            {DEVICECLASS}
```

NOTE: IF DEVICECLASS IS SPECIFIED, A DEVICE IN THE CLASS MAY BE USED UNSPOOLED BY REFERRING TO THE LDEV WHEN OPENING THE DEVICEFILE

THIS WILL BE DISCUSSED FURTHER IN THE SECTION ON DEVICE CLASS CAPABILITY

EXAMPLE: :STARTSPOOL 6

THE SPOOLER NOW OWNS LDEV 6

:STOPSPOOL

CAUSES THE DEVICE TO STOP AFTER PRINTING THE ACTIVE PRINT JOB (IF ANY) AND BECOME AVAILABLE.

SYNTAX

```
:STOPSPOOL {LDEV       }
           {DEVICECLASS}
```

:SUSPENDSPOOL

* CAUSES THE SPOOLED DEVICE TO STOP

SYNTAX

:SUSPENDSPOOL LDEV [;FINISH]

* IF FINISH OPTION IS SPECIFIED, PRINTER WILL CONTINUE TO PRINT AND IT WILL STOP UPON COMPLETION OF THE CURRENTLY ACTIVE PRINTJOB.

:SUSPENDSPOOL 6;FINISH IS EQUIVALENT TO THE OLD

=SPOOL 6,WAIT

:RESUMESPOOL

* CAUSES THE DEVICE STOPPED BY THE SUSPENDSPOOL COMMAND TO RESUME

SYNTAX

```
:RESUMESPOOL LDEV
```

EQUIVALENT TO OLD =SPOOL LDEV,RESUME

:DELETESPOOLFILE

* DELETES THE SPECIFIED READY OR ACTIVE SPOOLFILE

SYNTAX

```
:DELETESPOOLFILE { #ONNN }
                 { #INNN }
                 { LDEV  }
```

EXAMPLE:

TO DELETE AN OUTPUT SPOOLFILE BEING PRINTED ON LDEV 6 (LP), STOP THE PRINTER AND ENTER:

```
:DELETESPOOLFILE 6
```

:ALTSPOOLFILE

* ALLOWS USER/OPERATOR TO ALTER A READY OR LOCKED SPOOLFILE

SYNTAX

```
                                    [;PRI=OUTPUTPRIORITY]
                                    [;COPIES=NUMCOPIES]
:ALTSPOOLFILE #ONNN                 [;DEV= {LDEV    }]
                                           {DEVCLASS}
                                    [;DEFER]
```

* THE OUTPUTPRIORITY, NUMBER OF COPIES, OR DEVICE OF SPOOLFILES MAY BE ALTERED

* THE DEFER OPTION RESULTS IN THE OUTPUTPRIORITY OF THE SPOOLFILE BEING CHANGED TO 0 IMMEDIATELY AND THE FILE BEING PLACED IN THE READY STATE.

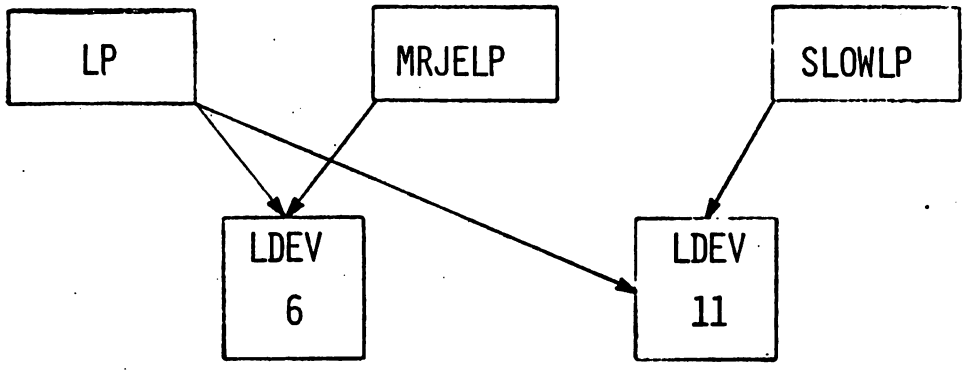
SPOOLER ENHANCEMENTS

DEVICE CLASS CAPABILITY

* ALLOWS USE OF A DEVICE AS UNSPOOLED WHILE STILL RETAINING THE ABILITY TO CREATE SPOOLED FILES FOR LATER OUTPUT TO THE DEVICE WHEN IT IS AGAIN SPOOLED

EXAMPLE:

CONSIDER TWO DEVICES AND THREE DEVICE CLASSES:



LDEV 6 IS INCLUDED IN DEVICE CLASSES "LP" AND "MRJELP"
LDEV 11 IS INCLUDED IN DEVICE CLASSES "LP" AND "SLOWLP"

ASSUME THE FOLLOWING COMMANDS ARE ISSUED:

:STARTSPOOL 6
:STOPSPOOL 11

SPOOLER ENHANCEMENTS

- DEVICE CLASS CAPABILITY EXAMPLE (CONT)

OUTPUT REFERENCED TO "LP","MRJELP", OR LDEV 6 WILL BE PRINTED VIA SPOOLFILE

OUTPUT REFERENCED TO "SLOWLP" OR LDEV 11 WILL BE PRINTED "HOT"

IF WE NOW ISSUE THE COMMANDS

```
:STOPSPOOL 6
:STARTSPOOL LP
:STOPSPOOL MRJELP
```

NOW, REFERENCES TO "MRJELP" WILL GO HOT TO LDEV 6
REFERENCES TO LDEV 6 WILL GO HOT TO LDEV 6
REFERENCES TO "LP" WILL PRODUCE SPOOLFILES WHICH WILL NOT BE PRINTED UNTIL :STARTSPOOL 6 IS ISSUED

STORE/RESTORE TAPE LABELS

* STORE TAPES MAY NOW BE LABELLED

* LOCKWORDS CAN NOW BE PLACED ON STORE TAPES

* STORE TAPES CAN NOW BE DATED (EXPIRATION DATES)

NOTE: NEW LABELLED STORE TAPES WILL NOT BE READABLE BY PRE-1918 VERSIONS OF STORE/RESTORE

PRE-1918 UNLABELLED TAPES WILL STILL BE READABLE BY RESTORE

STORE/RESTORE TAPE LABELS

EXAMPLE

```
:FILE T=T/XXX;DEV=TAPE;LABEL=BOBLAB
:STORE AAA;*T;SHOW
?9:57/#S6/16/MOUNT TAPE VOLUME BOBLAB (MAX CHARS.=2)?
=REPLY 16,7
9:57/#S6/16/VOL RFS    (ANSI) MOUNTED ON LDEV# 7
THU, JUL  5, 1979,  9:57 AM

  FILES STORED = 1

    FILE      .GROUP    .ACCOUNT    LDN   ADDRESS   VOLUME

    AAA       .PUB      .STAMPS     2     %363364      1




  FILES NOT STORED = 0


:FILE T;DEV=TAPE;LABEL=BOBLAB
:RESTORE *T;AAA;KEEP;SHOW
?10:03/#S6/16/MOUNT TAPE VOLUME BOBLAB (MAX CHARS.=2)?
10:03/12/VOL BOBLAB(ANSI) MOUNTED ON LDEV# 7
TAPE LABEL LOCKWORD VIOLATION  (FSERR 121)
ERROR ACCESSING TAPE - RESTORE STOPPED.  (CIERR 1026)
:FILE T=T/XXX;DEV=TAPE;LABEL=BOBLAB
:RESTORE *T;AAA;KEEP;SHOW
?10:04/#S6/16/MOUNT TAPE VOLUME BOBLAB (MAX CHARS.=2)?
10:04/12/VOL BOBLAB(ANSI) MOUNTED ON LDEV# 7
THU, JUL  5, 1979, 10:04 AM

  FILES RESTORED = 0

  FILES NOT RESTORED = 1

    FILE      .GROUP    .ACCOUNT FILESET      REASON

    AAA       .PUB      .STAMPS      1     ALREADY EXISTS
```

STORE/RESTORE TAPE LABELS

EXAMPLE

```
:FILE T=T/XXX;DEV=TAPE;LABEL=BOBLA
:STORE AAA;*T;SHOW
?10:08/#S6/16/MOUNT TAPE VOLUME BOBLA  (MAX CHARS.=2)?
10:08/12/VOL UNLABELLED   MOUNTED ON LDEV# 7
=REPLY 16,7
10:08/#S6/16/VOL UNLABELLED   MOUNTED ON LDEV# 7
?10:08/#S6/16/MOUNT TAPE VOLUME BOBLA  (MAX CHARS.=2)?
=REPLY 16,7
10:08/#S6/16/VOL BOBLA (ANSI) MOUNTED ON LDEV# 7
?10:08/#S6/16/MOUNT TAPE VOLUME BOBLA  (MAX CHARS.=2)?
=REPLY 16,7
10:08/#S6/16/VOL BOBLA (ANSI) MOUNTED ON LDEV# 7
THU, JUL  5, 1979, 10:08 AM

  FILES STORED = 1

    FILE     .GROUP    .ACCOUNT    LDN   ADDRESS   VOLUME

    AAA      .PUB      .STAMPS     2     %363364      1


  FILES NOT STORED = 0

:RESTORE *T;AAA;KEEP;SHOW
?10:09/#S6/16/MOUNT TAPE VOLUME BOBLA  (MAX CHARS.=2)?
10:09/12/VOL BOBLA (ANSI) MOUNTED ON LDEV# 7
TAPE APPARENTLY NOT A STORE TAPE.  (CIERR 1035)
T H E R E F O R E , EXACTLY 6 CHAR. REQUIRED FOR ID
```

VIRTUAL MEMORY INCREASE

* THE LIMIT ON THE AMOUNT OF CONFIGURED VIRTUAL MEMORY ON THE SYSTEM HAS BEEN RAISED FROM 32767 SECTORS TO 65535 SECTORS.

MPE RESEGMENTATION

* AN OVERALL RESEGMENTATION OF MPE HAS RESULTED IN 9 FEWER CST ENTRIES, LEAVING MORE CST SPACE FOR USER SEGMENTS.

RESEGMENTATION OF MPE

$FIVE MODULES NO LONGER EXIST

* Module 61 - formerly CLOCKIO
* Module 65 - formerly PROCMAIL
* Module 66 - formerly PINT
* Module 68 - formerly CRIO
* Module 77 - formerly STKDUMP

$FOUR MODULES ARE NEW

* Module 55 - formerly DISKSPC
  - now HARDRES (CRIO & CLOCKIO combined)
* Module 56 - formerly MMCORER
  - now SOFTRES (MMCORER & CROUTINE combined)
* Module 58 - formerly ABORTRAP
  - now ABORTDUMP (ABORTRAP & STKDUMP combined)
* Module 60 - formerly CROUTINE
  - now PROCSEG (PROCMAIL & PINT combined).

$ ONE MODULE HAS CHANGED

* Module 54 - ALLOCATE
  - formerly ALLOCATE & ALLOCUTIL code segments
  - now, the code segment names are the same, but ALLOCUTIL is ALLOCUTIL & DISKSPC combined.

MPE MODULE SOURCE

SYSTEM PROGRAMS

Module Name	Mod#
INITIAL	00
SYSDUMP	01
SEGPROC	02
SEGDVR	03
DISPATCH	04
LOAD	05
UCOP	07
DEVREC	08
PROGEN	09
ININ	10
MEMLOGP	11
LOG	12
IOPTRDØ	13
IOPTPNØ	14
IOPLOTØ	15
IOMDISCØ	16
IOFDISCØ	17
IOTAPEØ	18
IOLPRTØ	19
IOCDRDØ	20
IOTERMØ	22
IOPRPNØ	24
IOREMØ	25
IOMDISCI	27
PFAIL	30
PVPROC	31
VINIT	32

SYSTEM SL SEGMENTS

Module Name	Mod#
MAKECAT	40
FILESYS	50
COMM'INT	51
STORE/RESTORE	52
DIRC	53
ALLOCATE	54
HARDRES	55
SOFTRES	56
MMDISKR	57
ABORTDUMP	58
MESSAGE	59
PROCSEG	60
NRIO	62
PCREATE	63
MORGUE	64
DATASEG	67
CHECKER	69
UTILITY	70
SEGUTIL	71
LOADER1	72
RINS	73
JOBTABLE	74
DEBUG	75
NURSERY	76
FIRMWARESIM	78
SPOOLING	79
SPOOLCOMS	80
PVSYS	81
UDC	82
USER	83
HELPUSER	84
OPCOMMAND	85
LABSEG	86
SDISC	87
MEASIO	88
LOGSEGo	90
LOGSEG1	91

APPENDIX A

USER LOGGING

```
PROCEDURE OPENLOG(INDEX,LOGID,PASS,MODE,STATUS);
DOUBLE INDEX;
INTEGER MODE,STATUS;
LOGICAL ARRAY LOGID,PASS;
OPTION EXTERNAL;
```

PARAMETERS:

INDEX
A double word returned to the user to identify the logging access. The first word of the parameter contain the logging buffer DST number, the second word contains the entry number for the user in the logging buffer. This information is used to check the validity of subsequent calls to writelog and closelog.

LOGID
An array in which the user supplies his logging identification. It can be up to eight characters long and must match a logging identifier previously established using the :GETLOG command. The logging identifier can be used to identify ownership of logging records for auditing or recovery. If the logging identifier is less than eight characters long, it must be terminated with a blank.

PASS
An array in which the user supplies the pass word associated with the logging identifier. To obtain access, the password must match the password established with the logging identifier using the :GETLOG command. The password can be up to eight characters long. If the password is less than eight characters, it must be terminated with a blank.

MODE
An integer in which the user indicates whether or not his process should be suspended in the event that his access to the logging facility cannot be obtained immediately. This can occur when the user is sharing a logging buffer which is currently full and the logging process has been too busy to empty it. If the user wants to wait for his access, he should enter a zero in the mode parameter, if the user does not want to wait, he should enter one.

STATUS
An integer that the logging system uses to return status information to the user.

DESCRIPTION:

USER LOGGING PROCEDURES

The OPENLOG intrinsic is used to obtain accesss to a user logging file. The intrinsic insures that the caller has access to the logging file by checking the validity of the logging identifier and pass word. If the access is found to be valid, a double word is passed back to the user to be used in calls to writelog or closelog. A record is also written in the User Logging File that identifies the access to the file.

```
PROCEDURE WRITELOG(INDEX,DATA,LEN,MODE,STATUS);
DOUBLE INDEX;
INTEGER LEN,MODE,STATUS;
LOGICAL ARRAY DATA;
OPTION EXTERNAL;
```

PARAMETERS:

INDEX
The parameter returned from OPENLOG that identifies the user's access to the logging file. For privileged users, the first word of the index parameter can be used to pass a subsystem code. This code is placed in the top half of the code word on the logging record and can be used by subsystem or privileged user recovery systems.

DATA
An array in which the user passes the actural information to be logged. A log record contains 128 words of which 119 is available to the user. .The most efficient use of log file space is a multiple of 119 words.

LEN
The length of the data in DATA. A positive count indicates words a negative count indicates bytes. If the length is greater than 119 words, the information in data will be divided into two or more physical log records.

MODE
An integer in which the user indicates whether or not his process should be suspended in the event that his access to the logging facility cannot be obtained immediately. This can occur when the user is sharing a logging buffer which is currently full and the logging process has been too busy to empty it. If the user wants to wait for his access, he should enter a zero in the mode parameter, If the user does not want to wait, he should enter one.

STATUS
An integer that the logging system uses to return error information to the user.

USER LOGGING PROCEDURES

DESCRIPTION:

The #RITELOG intrinsic is used to write a physical record to a logging file. When the user calls the intrinsic it, provided there is room, places the request into the buffer area of the logging data segment. If the request fills the buffer, the intrinsic will write the contents of the buffer to the disc logging buffer for a tape log file, or the user's disc logging file for a disc log file. After writing the buffer, the intrinsic will awake the logging process using the PCB number stored in the communications area of the logging data segment.

In the event that the disc buffer used for logging to tape becomes full, the intrinsic will sleep the caller unless a nowait mode has been requested in which case the intrinsic will return to the caller without processing the request. In this case, it is up to the caller to re-issue the request at a later time.

```
PROCEDURE CLOSELOG(INDEX,MODE,STATUS);
DOUBLE INDEX;
INTEGER MODE,STATUS;
OPTION EXTERNAL;
```

PARAMETERS:

INDEX
The parameter returned from OPENLOG that identifies the user's access to the logging file.

MODE
An integer in which the user indicates whether or not his process should be suspended in the event that his access to the logging facility cannot be obtained immediately. This can occur when the user is sharing a logging buffer which is currently full and the logging process has been too busy to empty it. If the user wants to wait for his access, he should enter a zero in the mode parameter, if the user does not want to wait, he should enter one.

STATUS
An integer that the logging system uses to return error information to the user.

DESCRIPTION:

The CLOSELOG intrinsic is used to close a user log file. When the intrinsic is called, the logging entry in the logging data segment is deleted and any information remaining in the buffer portion of the logging data segment is flushed to the destination file. A record identifying the close of the log file is also written.